Implementation of a streaming server

Master Thesis Report

Albert Llongueras Clotet

June 23, 2011
To my parents and sister.
Acknowledgements

I would like to thank the University of Maribor, in particular the faculty of Electrical Engineering and Computer Science, for giving me this opportunity. In particular, I would like to thank my project tutor, Zdravko Kacic, who deposited since the first day all his trust in me. Thanks to him, I discovered another way of understand telecommunications which has motivated me to pursue my career on the field.

I must admit, and I’m very proud do it, that without the help of my colleagues from university: Dani Alonso, Dani, Alvaro, Didac, Carles, Anxo, Miki, Juacks, Nuria, Pau, the development of this project wouldn’t have been possible without them, as I would have probably not made to it to the end of my studies. I not only want to be grateful for the constant support received, but also for all the moments we have shared during the time spent in university or in the trips we have done. Referring also to the University, I want to thanks Lino, father of my IT knowledge, Alex and Oriol for all the help they have provided me. Also ClickerPau, for correcting my project and always being the light which guides me to success in the mist of doubt and fear. I don’t want to forget in my mind the experience lived during the second half of my studies: the Telecogresca. This organization has made me grow as a person not only for being able to prepare the biggest event in Europe, but also for preparing me for anything that happens during my life.

A person to whom I owe a lot is Nuria, my girlfriend. During all the time, she has been next to me specially in the difficult moments, always helping me with the personal or university problems. Thank you for all the advices you have always given me and which that have provided guidance into the future.

Finally, I really thank my parents, Jan and Rosa, and my sister Maria. Without their support, which they have always given me during my whole studies, my degree wouldn’t be a reality. I’m very proud of them for all the efforts they have made for me and for never “throwing the towel”.

Finally, a big hug to my grandma Marta because she has always provided me with experienced advice for life.
Abstract

This project concerns the implementation of a streaming server, Darwin Streaming Server. Moreover, we have also developed certain pieces of software to allow the final user to upload multimedia files and create playlists. The application is used to provide the University with new services, some of which are already running, and other will be implemented in the future.

The document also involves the XBMC software as it is where the application should be adapted. Finally, the project provides improvements to develop in the future that will provide functionalities for the user.

Aquest projecte tracta sobre la implementació d’un servidor streaming, el Darwin Streaming Server. Apart d’ell, també es devenenupen uns programaris perquè l’usuari final pugui pujar fitxers multimedia i crear llistes de reproducció. L’aplicació és utilitzada per donar nous serveis a la universitat, algun d’ells funcionant i d’altre que funcionaran en un futur.

També fa referència al programari XBMC ja que és el programari final on s’ha d’adaptar el projecte descrit anteriorment. Finalment es donen possibles millors per a fer en un futur que aportaran funcionalitats de cara a l’usuari.
## Contents

<table>
<thead>
<tr>
<th>Acknowledgements</th>
<th>iii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>v</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2 Global aims</td>
<td>3</td>
</tr>
<tr>
<td>3 Bases of the Streaming</td>
<td>5</td>
</tr>
<tr>
<td>3.1 Technology used for the streaming servers</td>
<td>5</td>
</tr>
<tr>
<td>3.1.1 Introduction</td>
<td>5</td>
</tr>
<tr>
<td>3.1.2 Communication Protocols</td>
<td>5</td>
</tr>
<tr>
<td>3.1.2.1 Real-Time Transport Protocol and Real-Time Transport Control Protocol</td>
<td>6</td>
</tr>
<tr>
<td>3.1.2.2 Real Time Streaming Protocol</td>
<td>11</td>
</tr>
<tr>
<td>3.1.2.3 Session Description Protocol</td>
<td>12</td>
</tr>
<tr>
<td>3.1.3 3GPP and 3GPP2 Groups</td>
<td>15</td>
</tr>
<tr>
<td>3.1.3.1 Packet Switched Streaming Service</td>
<td>17</td>
</tr>
<tr>
<td>3.1.3.2 Multimedia Streaming Services</td>
<td>19</td>
</tr>
<tr>
<td>3.2 Connectivity with the streaming server</td>
<td>21</td>
</tr>
<tr>
<td>3.2.1 Access to the web server</td>
<td>21</td>
</tr>
<tr>
<td>3.2.2 Dialog between server and client</td>
<td>22</td>
</tr>
<tr>
<td>3.3 Streaming server used</td>
<td>23</td>
</tr>
<tr>
<td>3.3.1 Introduction</td>
<td>23</td>
</tr>
<tr>
<td>3.3.2 Multicast and unicast</td>
<td>25</td>
</tr>
<tr>
<td>3.3.3 Election for a server</td>
<td>26</td>
</tr>
<tr>
<td>3.3.3.1 History of the streaming servers</td>
<td>26</td>
</tr>
<tr>
<td>3.3.3.2 Compression and standards</td>
<td>27</td>
</tr>
<tr>
<td>3.3.3.3 Service model</td>
<td>31</td>
</tr>
<tr>
<td>3.3.3.4 Differences between the others</td>
<td>32</td>
</tr>
<tr>
<td>3.4 XBMC</td>
<td>34</td>
</tr>
</tbody>
</table>
### CONTENTS

3.4.1 Introduction .................................................. 34  
3.4.2 XBMC database ............................................. 35  
3.4.3 Development on Linux .................................... 35  

4 Implementation part ............................................... 39  
- 4.1 Darwin Streaming Server .................................. 39  
  - 4.1.1 Installation and function .............................. 39  
    - 4.1.1.1 Installation .................................... 39  
    - 4.1.1.2 Interface ..................................... 41  
    - 4.1.1.3 Playlist .................................... 42  
  - 4.2 Software for converting files ......................... 47  
    - 4.2.1 Converting software .............................. 48  
  - 4.3 Interface for upload files .............................. 51  
    - 4.3.1 Features and functionality ..................... 51  

5 Future functionalities ............................................ 55  
- 5.1 Introduction ................................................ 55  
- 5.2 Converting software ...................................... 55  
- 5.3 Website Interface ......................................... 56  
- 5.4 XBMC ......................................................... 56  

6 Conclusions ......................................................... 61
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>RTP can be viewed as a sublayer of the transport layer</td>
<td>6</td>
</tr>
<tr>
<td>3.2</td>
<td>Both senders and receivers send RTCP messages</td>
<td>7</td>
</tr>
<tr>
<td>3.3</td>
<td>Operation of the SSRC, CSRC and a RTP mixer</td>
<td>9</td>
</tr>
<tr>
<td>3.4</td>
<td>Structure of a RTP header</td>
<td>9</td>
</tr>
<tr>
<td>3.5</td>
<td>Example of a sender/receiver RTCP operation</td>
<td>11</td>
</tr>
<tr>
<td>3.6</td>
<td>Example of a may configuration of a SDP session description</td>
<td>12</td>
</tr>
<tr>
<td>3.7</td>
<td>Offer and Answer operation</td>
<td>14</td>
</tr>
<tr>
<td>3.8</td>
<td>Protocol stack</td>
<td>15</td>
</tr>
<tr>
<td>3.9</td>
<td>Evolution of the 3GPP Releases</td>
<td>16</td>
</tr>
<tr>
<td>3.10</td>
<td>Elements involved in a network in 3GPP PSS</td>
<td>17</td>
</tr>
<tr>
<td>3.11</td>
<td>Multimedia Streaming Service</td>
<td>19</td>
</tr>
<tr>
<td>3.12</td>
<td>Multimedia Streaming Services Terminal Function</td>
<td>20</td>
</tr>
<tr>
<td>3.13</td>
<td>Protocol Stack for Multimedia Streaming Service</td>
<td>21</td>
</tr>
<tr>
<td>3.14</td>
<td>Instruction for access to the stream file using the VLC player</td>
<td>21</td>
</tr>
<tr>
<td>3.15</td>
<td>Dialog between server and client</td>
<td>22</td>
</tr>
<tr>
<td>3.16</td>
<td>Video and audio live in a streaming operation</td>
<td>24</td>
</tr>
<tr>
<td>3.17</td>
<td>Example of a unicast stream</td>
<td>25</td>
</tr>
<tr>
<td>3.18</td>
<td>Example of a multicast stream</td>
<td>26</td>
</tr>
<tr>
<td>3.19</td>
<td>TCP transmission</td>
<td>27</td>
</tr>
<tr>
<td>3.20</td>
<td>Encoder and Decoder MPEG</td>
<td>29</td>
</tr>
<tr>
<td>3.21</td>
<td>Complete schema about delivering content</td>
<td>31</td>
</tr>
<tr>
<td>4.1</td>
<td>URL for access into the website interface</td>
<td>40</td>
</tr>
<tr>
<td>4.2</td>
<td>Configuration of the .playlist file</td>
<td>45</td>
</tr>
<tr>
<td>4.3</td>
<td>Configuration of the .config file</td>
<td>45</td>
</tr>
<tr>
<td>4.4</td>
<td>Command for start a video playlist</td>
<td>45</td>
</tr>
<tr>
<td>4.5</td>
<td>Check of the PID number of a playlist</td>
<td>46</td>
</tr>
<tr>
<td>4.6</td>
<td>Kill the playlist using the PID found</td>
<td>46</td>
</tr>
<tr>
<td>4.7</td>
<td>Actions available using the command PlaylistBroadcaster</td>
<td>46</td>
</tr>
<tr>
<td>4.8</td>
<td>Configuration of the .config audio file</td>
<td>47</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>4.9</td>
<td>URL for the video playlists</td>
<td>47</td>
</tr>
<tr>
<td>4.10</td>
<td>URL for the audio playlists</td>
<td>47</td>
</tr>
<tr>
<td>4.11</td>
<td>Instruction that shows the hint option of a file</td>
<td>47</td>
</tr>
<tr>
<td>4.12</td>
<td>Example of how to add the setting <code>hint</code></td>
<td>49</td>
</tr>
<tr>
<td>4.13</td>
<td>Command for convert the avi file into mp4</td>
<td>50</td>
</tr>
<tr>
<td>4.14</td>
<td>Command for convert the avi file into 3gp</td>
<td>50</td>
</tr>
<tr>
<td>4.15</td>
<td>URL for the mobile playlists</td>
<td>50</td>
</tr>
<tr>
<td>4.16</td>
<td>URL for access to the upload web page</td>
<td>51</td>
</tr>
<tr>
<td>4.17</td>
<td>View of the website</td>
<td>52</td>
</tr>
<tr>
<td>4.18</td>
<td>View of the video website</td>
<td>53</td>
</tr>
<tr>
<td>5.1</td>
<td>Example of the XBMC settings</td>
<td>58</td>
</tr>
<tr>
<td>5.2</td>
<td>Example of the XBMC interface</td>
<td>59</td>
</tr>
</tbody>
</table>
List of Tables

3.1 SDP types . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
3.2 Streaming related protocols of PSS . . . . . . . . . . . . . . . . . . 18
3.3 Codec for audio and video that use 3GPP PSS . . . . . . . . . . . . . 18
3.4 XBMC features, functions and supported audio/video/picture formats 37

5.1 Current available features if the web interface option is enabled . . . 58
Chapter 1

Introduction

During the last two decades, technology has improved in such a way that it has lead the whole area to a set of new services which are widely demanded. It has also given us an important progress in many fields, for example the way of listening and seeing videos on the Internet. These new services have created in many companies, higher developed systems with the aim of offering to the clients best and most comfortable ways of accessing to this kind of multimedia.

Thanks to this great leap forward that technology has done, not only the use of audio and video has increased, but also the appearance of new devices, such as, computers, laptops or mobiles. Apart from being an important feature for customers, it is a way of access to videos and audio files.

The name of this technology is streaming. Streaming is referred to as the capability of watching "live" videos on the Internet instead of downloading them. This is not the only feature that provides this new technology, but also, the possibility to broadcast the files to multiples clients.

This last feature, is quite important because of the applications which have been developed based on it. Referring to this field of the streaming technology, some new applications of communication have emerged, such as: video conferences or live streaming. A current example that begins to be in our lives, could be the television or radio over IP.

This Master Thesis, is a part of a project that is based in the streaming technology. Its main goal is to have a whole IP Multimedia Subsystem based over IP according to the necessities of the customers. The part which done in this project, caters the new way of looking videos and listening audio using the broadcast streaming technology.
Chapter 2

Global aims

The way of implementing this kind of system, is setting a server that allows to reproduce this multimedia in a broadcast way. This server, has to have the capability to supply the necessities of the customers, not only playing the multimedia files, but also, enabling the possibility to upload and create playlists with the multimedia files that are already in the server or that will be uploaded in the future.

The deployment of the server consists in a set of programs where each one has its own responsibility, and all together, work in making the server suitable for clients’ needs. Thus, the server will have a software for uploading files, another one for converting them and one to create a playlist. Finally, a last one is necessary for playing the multimedia files.

One of the problems that we will find during the implementation of the server, will be the protocols that the server works with, and the file formats accepted by the server. Therefore, at each moment in time we must take this into account to check if the files are in the right format, and if not, convert these new files into an acceptable format.

A future goal of this project, is the possibility of adapting it to the whole IP project, that includes VoIP or Television over IP for example. The big project, has an open source interface that is installed in a multimedia hard drive and connected to a television. It is important that all software used is open source, necessary in the future to unify the other subprojects into the big one.

As a culmination to the project, there is a chapter that explains the new features that the project has to have eventually, no matter when, to complete the project.

The structure of the project is detailed in the following lines:
The first chapter is an introduction and details why and how our particular implementation has been selected.

The following chapter explains the aims that the project wants to achieve.

The third block talks about theory, giving details about the protocols used, the election of the server chosen, the format files used for making possible the broadcast streaming technology, the databases created for storing the information of the files uploaded...

The fourth chapter, is about the implementation of the server. Here it is explained all the installation processes and the realization of the specific software developed for making possible the streaming transmission.

As an extended project, it is necessary to devote a section to the developments that will be interesting in a future: chapter fifth.

The last chapter is about the conclusions that we can extract once we have tested the system, and tries to expose the benefits and inconveniences of the project developed.
Chapter 3

Bases of the Streaming

3.1 Technology used for the streaming servers

3.1.1 Introduction

In this chapter we explain the technology implemented in the project. When speaking about technology, we refer to protocols, format of files and codification used for the transmission of video and audio files in real time.

This section is divided in two parts, the first one is related to the protocols used for the transmission and the second one, the format of files that accept this kind of transmission, the real time transmission.

3.1.2 Communication Protocols

The big breakthrough that enabled the streaming revolution was the combination of the UDP protocol and the new encoding techniques for compressing audio files into extremely small packets of data. UDP protocol made streaming media feasible by transmitting data more efficiently than previous protocols from the server to the client.

Recently, new protocols appeared such as the Real-Time Streaming Protocol who’s goal is to make the transmission even more efficient than in older protocols.

Actually, the combination of RTP and UDP protocols is used for streaming of continuous media whereas HTTP and TCP is used for the transport of discrete media such
as text or images. In the first case, there is a necessity to establish a session set-up and control protocol while in the second case this is unnecessary. The protocol which sets-up and controls a streaming media is the RTSP protocol, explained below:

![Diagram of transport layer protocols]

There are two ways of transmitting media: transmitting without monitoring if a packet drops out, and taking into account the transmission status.

The first one is implemented with UDP and RTSP and the second one with TCP and HTTP. For streaming media files, first option is used because the speed of transmission is greater than in the other option. A feature of the transmission over UDP is that the server is continuously sending information even when a packet gets dropped out. This causes a small gap in the transmission instead of a big silence. TCP, on the other hand, keeps trying to resend the lost packet before sending anything further, causing greater delays and breakups in the broadcast.

Another important protocol is the SDP. This is in charge of describing the initialization parameters of multimedia fluxes. SDP maintains a communication with the final application and is the responsible to communicate the information on the file, like format, content or other parameters.

3.1.2.1 Real-Time Transport Protocol and Real-Time Transport Control Protocol

The main goal of this project is to create a system that can play streaming files to clients. These files, after being converted into a correct format to be played, must
flow over the network in order to supply the file to the client. Assuming that all the
process before the transmissions are done successfully, what lacks to known is: how
do we take the data from the application done, which can be in a real-time or not,
and send it through a network that cannot guarantee reliable delivery? The answer
can be found in the use of the Real-time Transport Protocol.

This protocol is defined in RFC 3550 \[3\]. Works with RTP Control Protocol \[4\] and
is in charge of monitoring the quality of service and conveying information about the
participants in the communication session.

What RTP does, is to provide an end-to-end delivery service for data that requires
real-time support. This kind of data can be a live audio or video, a video-conference...
RTP uses as a transport protocol UDP and talking about UDP means that there is
no quality of service or guarantee. It is important to note the functionalities that RTP does not provide such as mechanisms to ensure timely delivery or preventing out-of-order packet delivery. Each packet sent, has a number that helps the receiver to reconstruct the packet sequence.[1] [6] [7] [8]

In the RFC 3550, the RTP definition is divided in two parts:

- **Real-time transport protocol** is the charge to carry data that has real-time properties.
- **RTP control protocol** monitors the quality of service and conveys information about the participants that are in the session.

Figure 3.2 shows its operation.

The most important features about RTP:

- **Structure of packet:** RTP packet can transport data such as audio or compressed video data. This packet consists of a fixed RTP header, a possible empty list of contributing sources, and the data. An RTCP packet also has a fixed header that has almost the same structure than RTP. The next part of the RTCP packet is structured differently than RTP. It can be one of 5 types of packet. For more information about them, there is a detailed explanation in RFC 3550 section 6.

- **Transport address:** For delivery of a packet to the correct destination, the combination of the IP address and the port number is used, i.e a IP address and a UDP port.

- **Session:** Is a set of participants that communicate with RTP. For each participant, the session is defined by a pair of destination transport addresses. RTP and RTCP each one have one port. In the multicast case, the destination transport address pair is common for all participants but in the case of unicast, it is different for each one. In a multimedia session, each stream is carried in a different RTP session with its own RTCP packets.

- **Synchronization source (SSRC):** Indicate the synchronization source for the client. It is unique and is identified in the packet during the transmission. The number used, is a random number with the intention that no two synchronization sources have the same SSRC within the same session. SSRC is the last one which has made modifications on RTP content. Normally who does this is the server but sometimes, it can be mixed. Also, it is possible to find CSRC, Contributing Source. Is each element that has contributed to the combined stream produced by an RTP mixer. CSRC has a list, where are all the identifiers of the sources.
that contributed to the generation of a particular packet into the RTP header of the packet. A clear example would be a videoconference where CSRC indicates all the participants that are taking part of this outgoing packet.

- **End system**: Is an application that generates the content that will be sent for an RTP packet.

- **Mixer**: Is a system that is in the middle and receives RTP packets from one or more sources. It can changes the data format. Then combines the packets and forwards a new RTP packet with a new SSRC as the Figure 3.3.

- **Monitor**: Application that estimates, among other things, the current quality of service or long-term statistics.

- **Non-RTP**: Protocols and mechanisms that are needed to make the service usable. For example: a multimedia conference, has a conference control application that distributes the multicast address and keys for the encryption. Also, it can negotiate the encryption algorithm to be used.

![Figure 3.3: Operation of the SSRC, CSRC and a RTP mixer](image)

Taking a look to an RTP message header, we can see that the first twelve octets are present in every RTP packet. During the follow lines, we explain, using Figure 3.4 each part of a RTP message header.

![Figure 3.4: Structure of a RTP header](image)
• **Version:** Identify the current version of RTP.

• **Padding:** Indicates if the padding is set or not. Due to an RTP packet has to have a multiple of 32 bits, in case this is not possible, it is added into the packet as numbers of white bits as necessary. The last byte, indicates the number of octets filled.

• **Extension:** If the extension bit is set, the fixed header is followed by exactly one header extension, 32 bits.

• **CSRC count:** Is the counter of CSRC and indicates the number of identifiers that follow the fixed header. The number can be between 0 and 15.

• **Marker:** Bit marker is used for specific applications.

• **Payload type:** Kinds of data that the packet contain. Identify the format of the RTP payload.

• **Number of sequence:** Let to know if a packet is in or out of a sequence. The number of sequence is indicated with a random number and incremented by one. Knowing the number of packets lost, it is possible to make some analysis of the packets.

• **Timestamp:** Reflects the sampling instant of the first octet in the RTP data packet. Increments monotonically and linearly and allow synchronization and jitter calculations.

• **SSRC:** Identifies the synchronization source. The identifier is chosen randomly due to no other synchronization, in the same session, has the same SSRC.

As can be seen, the function of RTP is to transport, apart from data, the useful information to perform a communication control. However one problem that appear, is that do not has a mechanism for inform about the information carried. For this reason, RTP works together with Real-Time Control Protocol. This generates a set of reports for controlling how the transfers are going on. Furthermore, these reports have information about packets loss, jitter... Thanks to the reports, it is possible to know how the buffer of a client, that is in the same session, is. RTCP has five different packets formats, but the two most important are the Sender Report or SR and the Receiver Report or RR.

The Figure 3.5 shows how the operation between them is.

• **Sender Report:** transmission and reception statistics from participants that are active senders.
3.1. TECHNOLOGY USED FOR THE STREAMING SERVERS

3.1.2.2 Real Time Streaming Protocol

The Real-Time Streaming Protocol is defined in RFC 2326 [2]. It is an application layer protocol and is in charge of controlling the delivery of data that has properties of real-time such as live video or audio.

This protocol is in charge of the establishment and control of the session. Thanks to this protocol, the client has the possibility to play, pause and stop the stream. RTSP works with other protocols such as RTP/RTCP, as RTSP cannot perform the control and transport of the stream. RTSP acts as a network remote control for multimedia servers. Figure 3.15 shows how is an RTSP scenario is.

RTSP messages can be sent over either TCP or UDP and use the 554 port number. In order to perform an RTSP connection, the communication must be active from the moment the client sends the request to the server, until it has sent the last packet of video or audio. [1] [9]

Operations that RTSP supports:

- The client can request a presentation description via HTTP or other methods. The presentation contains the multicast address and the port if the presentation is being multicast, and contains the destination provided by the client, if the presentation is unicast.
- A media server can join to an existing conference.

Figure 3.5: Example of a sender/receiver RTCP operation

- Receiver Report: reception statistics from participants that are not active senders.
An interaction between client and server using RTSP, RTP and RTCP is described in the section "dialog between server and client".

3.1.2.3 Session Description Protocol

This protocol contains enough information for a remote user to join a session. This information contains included the IP address and the port number where the multi-media needs to be sent and the codecs used to encode the voice and the images of the participants.

Let’s walk through an SDP configuration file as depicted in Figure 3.6. It is an example that shows how an SDP session description is. A user (A) sent multimedia to user (B). The SDP contains, among other information, the (A) IP (192.168.1.2), the port number where (A) wants to receive audio (40000), the other port for receiving video (40001) and the audio and video codec that (A) supports. There is a list that contains all the codecs and each one has a numeration. The numeration is used for not having the necessity to write the exact full name of the codec.[6][7][8]

Here is an example of an SDP configuration file:

```
v=0
o=(A) 2790844657 2867892807 IN IP4 192.168.1.2
s=This is a SDP example
c=IN IP4 192.168.1.2
t=0 0
m=audio 40000 RTP/AVP 0
a=recvonly
m=audio 40001 RTP/AVP 31
a=recvonly
```

Figure 3.6: Example of a may configuration of a SDP session description

Looking at the Figure 3.6, the SDP file is divided in two parts. The first one is about session-level, and the second one, media-level information.

- **Session-level information**: Referring to the file, it is the lines before the line m=. In the first two lines, the version is explained (v=) and the user identifiers (o=). Then, it mentions the subject of the session (s=), the (A) IP address (c) and finally the time of the session (t=).

- **Media-level information**: In the following lines, specify the media stream information. The m’s lines are necessary, whereas a’s lines are not and add information about the media stream. It is necessary to have a m line for each media
stream has. In this example the line \textit{a=sendrecv} means that the streams are bidirectional.

There is a special format for making an SDP file. This format consists in a command \textit{type=format} where the type always is a character. The list of all the types available is in Table 3.1. We should take into account the use of capital letters as SDP files are case sensitive.

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>Protocol version</td>
</tr>
<tr>
<td>b</td>
<td>Bandwidth information</td>
</tr>
<tr>
<td>o</td>
<td>Owner of the session and session identifier</td>
</tr>
<tr>
<td>z</td>
<td>Time zone adjustments</td>
</tr>
<tr>
<td>s</td>
<td>Name of the session</td>
</tr>
<tr>
<td>k</td>
<td>Encryption key</td>
</tr>
<tr>
<td>i</td>
<td>Information about the session</td>
</tr>
<tr>
<td>a</td>
<td>Attribute lines</td>
</tr>
<tr>
<td>u</td>
<td>URL containing a description of the session</td>
</tr>
<tr>
<td>t</td>
<td>Time when the session is active</td>
</tr>
<tr>
<td>e</td>
<td>Email address to obtain information about the session</td>
</tr>
<tr>
<td>t</td>
<td>Times when the session will be repeated</td>
</tr>
<tr>
<td>p</td>
<td>Phone number to obtain information about the session</td>
</tr>
<tr>
<td>m</td>
<td>Media line</td>
</tr>
<tr>
<td>c</td>
<td>Connection information</td>
</tr>
<tr>
<td>i</td>
<td>Information about the media line</td>
</tr>
</tbody>
</table>

Table 3.1: SDP types

In an SDP file, we can find some further information that may be useful to implement or know more about the stream, like the feature \textit{hint} or the cryptographic keys. As we will see in the following chapters, \textit{hint} is an important feature for creating a stream file.

The SDP protocol is used when a user accesses an RTSP link. The first action is a DESCRIBE request and the server answers showing the available information. This information, is based on the SDP protocol. In the SDP file, the server shows the different flows available, such as codification used, assigned ports to each flow or the bandwidth that has each work-flow.
In the figure 3.6, (A) sent a session description to (B). In this description, there is
the IP and the port numbers. Another session description is necessary for establish
a session between them. (B) has to inform (A) his transport addresses.

To create a new session between two users, is used the offer/answer method. In the
Figure 3.7, shows how works. The user that wants to start a session (the offer),
sends a session description to the remote user (the answerer) and this generates a
new session description and sends it to the offer. For a detailed explanation, see the
RFC 3264 [5]. In the RFC, there are the rules for the offer and answer generation.

![Sending SDP session description generated to (B)](image)

![Sending SDP session description answer generated to (A)](image)

Once the offer and the answer of their session description are exchanged, the session
is established and both have a common view of the new session created.
A summary of how all the protocols described before work is in Figure 3.8.

![Protocol stack](image)

**Figure 3.8: Protocol stack**

### 3.1.3 3GPP and 3GPP2 Groups

When talking about streaming media files, it means that the audio or video files are distributed by Internet continuously, without interruptions. But to get it, it is necessary that the files are in a specific format for being reproduced. The list of these accepted formats is not closed because of the variety of existence of streaming servers.

However, it is necessary to talk about a project that brings together almost all the accepted file formats for being reproduced in a streaming server. This project is the 3GPP (Third Generation Partnership Project).

The 3GPP is a collaboration agreement reached in 1998. This cooperation is between groups of different nationalities: ETSI (Europe), ARIB/TTC (Japan), CCSA (China), ATIS (North America) and TTA (South Korea). The original scope of 3GPP was to produce Technical Specifications and Technical Reports for a 3G Mobile System based on evolved GSM core networks and the radio access technologies that they support (for example, Universal Terrestrial Radio Access (UTRA) both Frequency Division Duplex (FDD) and Time Division Duplex (TDD) modes). This generation, works with GSM networks using WCDMA radio technology. Another collaboration that also exists is 3GPP2. It is like a “sister project” of 3GPP and has similar goals as 3GPP but is based on evolving North American and Asian cellular networks using CDMA2000@radio access technology into third-generation system. The list of 3GPP2 organizational partners is the same as 3GPP. Like 3GPP, 3GPP2 does not produce standards but, instead, Technical Specifications and Technical Reports.

3GPP is redefined periodically. The standards are structured as Releases. Each release, means a new version with new technical specifications and incorporates hum-
dreds of individual standards documents, each of which may have been through many revisions.[17]

3GPP created the IMS which is part of Release 5 of the 3GPP specifications. Although the IMS is independent of the access network, 3GPP has focused on access proceeded by WCDMA networks and the GPRS packet core network.

Currently 3GPP standards incorporate the latest revision of the GSM standards. 3GPP’s plans for the future beyond Release 7 are in the development under the title Long Term Evolution (“LTE”).

In the next Figure 3.9 there is a mark that explains the evolution about the releases made until the last one.

The current version is Rel-10, although, the Rel-11 is almost done but will appear at the end of 2011.

![Figure 3.9: Evolution of the 3GPP Releases](image)

In 3GPP there is a specification that defines framework for an interoperability stream-
3.1. TECHNOLOGY USED FOR THE STREAMING SERVERS

A streaming service that is called Packet Switched Streaming Service (PSS). Appear in the Release 4. Also exist with the 3GPP2 under the term Multimedia Streaming Service (MSS).

3.1.3.1 Packet Switched Streaming Service

PSS is an application level service. This specification deals with the server and the client. The Figure 3.10 explains how works. One important feature about PSS, is that can work over different bearers and this provides that can adapt to the network increasing the QoS of the service.

![Figure 3.10: Elements involved in a network in 3GPP PSS](image)

In Figure 3.10 there is the stack of the protocols that 3GPP PSS uses and in the Table 3.2 a summary of the streaming related protocols.

Furthermore, the 3GPP PSS defines audio and video codecs. These codecs are very important because is the format that an streaming file has to be reproduced in an streaming way. The codecs are summarized in the Table 3.3

The MP4 file format that originally derived from the QuickTime file format, is a referent in the 3GPP file format. In the Release 4 of PSS, the main file format is MP4. This file format is flexible and supports both local playback and streaming delivery. The 3GPP media file has specific media types such as H.263 for video and AMR for audio. After the Release 4 appeared the Release 5 and bring a new functionality that is the User-Agent Profile (UAProf).

UAProf is a file that has the extension .xml or .rdf. It is a configuration file plus an API that access the data in real-time environments for wireless devices. The “database” of mobile devices description is done for the telecommunication companies and also with the users from different countries and can be used in any applications. This allow the user device to adapt with the server according to the physical capabilities of the client.

---
1 Extensible proprietary multimedia framework developed by Apple Inc.
2 The configuration file is rdf if is a document schema that is extensible.
3. BASES OF THE STREAMING

RTP | Real-Time Transport Protocol [RFC 1889, RFC 1890] | Transport layer protocol used with voice and video transmission. Operates with UDP and provides information about packet sequence in order to help to detect the delay, and packet loss.

RTCP | Real-Time Control Protocol [RFC 1889] | Is a protocol that can work in conjunction with RTP in a networked multimedia application. The primary function is to provide feedback on the quality of data distributed. In a multicast scenario, RTCP packets are transmitted by each participant in an RTP session to all other participants into the session using an IP multicast.

RTSP | Real-Time Streaming Protocol [RFC 2326] | Is the protocol that allows the user to control the playback of continuous media by pause stop or start a playback. Also allows to forward or rewind the playback but only if the player is compatible.

SDP | Session Description Protocol [RFC 2327, RFC 2326] | Is in charge of making the description of the session to be established. Contains information for the remote user to join the session.

<table>
<thead>
<tr>
<th>Type</th>
<th>Codec (Decoder)</th>
<th>Support</th>
<th>Max. bit rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speech</td>
<td>AMR-NB</td>
<td>Required</td>
<td>12.2 kbps</td>
</tr>
<tr>
<td>Speech</td>
<td>AMR-WB</td>
<td>Required</td>
<td>23.85 kbps</td>
</tr>
<tr>
<td>Audio</td>
<td>MPEG-4 AAC-LC</td>
<td>Recommended</td>
<td>N/A</td>
</tr>
<tr>
<td>Audio</td>
<td>MPEG-4 AAC-LTP</td>
<td>Optional</td>
<td>N/A</td>
</tr>
<tr>
<td>Video</td>
<td>H.263 profile 0 level 10</td>
<td>Required</td>
<td>64 kbps</td>
</tr>
<tr>
<td>Video</td>
<td>H.263 profile 3 level 10</td>
<td>Recommended</td>
<td>64 kbps</td>
</tr>
<tr>
<td>Video</td>
<td>MPEG-4 Simple Visual Profile Level 0</td>
<td>Recommended</td>
<td>64 kbps</td>
</tr>
</tbody>
</table>

Table 3.2: Streaming related protocols of PSS

The majority of the capabilities of a device, are defined in a user profile or UAProf. These profiles, which are stored in the configuration files, specify the device model and version. This information can be useful for the streaming servers.

3GPP2 also has an specification that defines framework for an interoperability streaming service that is called Multimedia Streaming Services (MSS).
3.1.3.2 Multimedia Streaming Services

The scope is to standardize the functionality of MSS that can be incorporated into the operations of wireless telecommunications networks [11]. Audio and video streaming are a part of MSS.

MSS is a structure point-to-point service. It is not a symmetric connection between sender (mostly MSS server) and client because the actions that take place on the server are more usual than the actions in the client, such as creation, storage, packetization and the transmission, whereas in the client there is the reception. The streaming service supports real-time streaming and pseudo-streaming retrieval of pre-encoded content and real-time encoded content from both parts.

The operation of real-time and pseudo streaming have some differences such as real-time the encoded information is sent after being packetized to the MSS server. Then the MSS client has to de-packetizes the data using the appropriate multimedia decoders. At this time, is when the data is ready for playback. Apart from this operation, MSS includes system control protocols for setting up connections between the clients and the server. These protocols make actions like negotiating various options, configurations and capabilities. Furthermore they control the various source codecs that the MSS uses. These are the main procedures although it includes advanced ones

---

3. Is the transfer of dynamic media content such as video, speech, audio, timed text, to the MSS terminal accomplished via file download using TCP protocol

4. Who realize the packetization operation are the media components that know how to break media into packets for RTP transmission
like maintaining QoS and monitoring under dynamic conditions. One of the goals is to be interoperable with 3GPP PSS.

A global scheme of how MSS works is in Figure 3.11 and also Figure 3.12 depicting the function of a Multimedia Streaming Services Terminal. [7][11]

![Multimedia Streaming Service Terminal Functions](image)

Figure 3.12: Multimedia Streaming Services Terminal Function

Once the protocols have been explained, what they are and how they works it is possible to understand Figure 3.13 that is a summary of the protocols that are involved in a Multimedia Streaming Service.
3.2 Connectivity with the streaming server

This section attempts to explain the operation between a server and a client: from when the client sets the address of the server until it finishes sending more packets to the client. It is assumed that there is a web server and offers to the clients the RTSP resources.

3.2.1 Access to the web server

In the server there is the stream files. To access them, there are three basic requirements: the address where the server is located, the port and the name of the resource. In the Figure 3.14 the instruction that shall be followed for access to the stream file is shown.

```
Server@Server~$ vlc rtsp://ipservlet/playlist.sdp
```

Figure 3.14: Instruction for access to the stream file using the VLC player
In this example the VLC player is used because it is a player that allows the RTSP protocol operation. In the next section its operation is explained.

The streaming server has a dedicated port specially for the RTSP protocol. This port is 554. The name of the resource needs to be a file with an .sdp extension. This extension is used for the playlist. The section Streaming server used talks about this type of file nevertheless, a little summary of what it is, is a file that is created from multimedia files. In the following lines is explained how is the dialog between server and client.

### 3.2.2 Dialog between server and client

Once the client issues the command rtsp, the dialog starts. An example is shown in Figure 3.15. In the lines below, the steps that this operation follows are explained:

- For access to the media it is supposed that the client knows the location of it. This location will be a RTSP URL.

- Once added the RTSP URL into the streaming player, it sends an RTSP DESCRIBE command.

---

5Open source player that allow the utility of RTSP/RTP protocols. Can be downloaded from http://www.videolan.org/vlc/
• The server responds with an SDP description which includes information about the stream, such as media type, number of streams or bandwidth required.

• After the SDP command, the server issues an RTSP SETUP for each stream in the session.

• Now it is the time to assign the port that the user will use for the stream. The command SETUP, tells the server which ports the client uses to receive the media.

• Once the session and the stream have been set up, the client asks for the command PLAY and it is when the client starts sending the media streams with an RTP packet over UDP.

• When the client wants to finish the stream, issues a TEARDOWN command.

3.3 Streaming server used

3.3.1 Introduction

What does the streaming word mean?

The Streaming process delivers media from a server over a network to a client without the necessity to download the media file before. Normally over Internet or a LAN. The files delivered never are downloaded to a viewer’s hard drive. The streaming file is offered while the client is playing it.

When someone needs to send an email, an email server is required, and the same happens when someone wants to send real-time streams, as he needs a streaming server.

The streaming server transmits video and audio streams when a client performs a request using client software such as Video Lan (VLC). This requests are handled using RTSP and the streams are sent using RTP. This method is used for transmitting real-time multimedia content over networks.

When talking about delivery media, we should distinguish two categories: live and on demand. Most of the servers allow to serve both categories.

• Live events such as videoconferences, concerts, speeches...are commonly streamed over the Internet while they are happening. There is specialized software dedi-
3. BASES OF THE STREAMING

cated to broadcast and encode the live source such as a video from a real-time camera. Once the live source is encoded, the server, delivers the live stream to clients. It is not important when the clients connect to the server because each of them will see the same point at the same time in the stream. It is not necessary that the content should be live source, but it also can be recorded content. The way of transmitting the media file will be the same.

- On demand delivery, each client initiates the streaming from the beginning of the file. So, there is no possibility that any customer arrive late to the stream. In this case, is not necessary a broadcasting software.

Figure 3.16 shows how to set-up a streaming media system with live video and audio.

![Figure 3.16: Video and audio live in a streaming operation](image)

At a first glance, we can say that the transmission over Television or Internet are the same, that the only difference is that one is over a cable or electromagnetic wavelengths and the other over the wired Internet. Thoroughly analyzing both types of transmission, it is possible to detect a lot of differences that make the two types of transmissions far away one from the other.

On the one hand, talking about the transmissions over the cable or electromagnetic wavelengths we can say that most of them are uncompressed and consume large amounts of transmission bandwidth. On the other hand, talking about the transmissions over Internet, means speaking about encoding and compressing.

Referring to the first kind of transmission, it is normal not to use the technology of
encoding and compressing due to each transmission having its own frequency and this means that they do not have to share or compete with other transmissions.

Totally different is the situation of the second type of transmission. These, have to compete and share with other transmissions and therefore must be encoded and compressed.

### 3.3.2 Multicast and unicast

Unicast means a connection between 1 to 1. In this connection, a server and a client take part. Each client gets a separate stream and only if they request it. Unicast streaming works either for live streaming or on demand streaming. Normally, unicast streaming is used when we want to encode with multiple-bit-rate, when the multicast is not enabled or when the server calculates the streaming media according to the network capabilities of the client.

![Figure 3.17: Example of a unicast stream](image)

Multicast streaming is the opposite technology for delivering data and means that a single stream is shared among the clients. All of them, have the same stream but this does not signify that the higher the number of clients, the slower speed transmission to each client. This happens because most of the streaming servers, have a special work-flow for each client connected, making the transmission speed not be a function respect to the number of customers.

The operation of multicast streaming, is that the sever uses a single IP address to deliver the content and then, clients are able to access it by requesting the IP. One feature important in this kind of transmission, is that all the customers, receive at the same time the same content and cannot control the playback. This is an important property as at a glance, it seams that the customers should control the playback, but
if referring to TV it is the same situation. No one, can control the TV except the people who manage it. So, the same happens with the streaming playlist.

![Multicast stream diagram](image)

**Figure 3.18: Example of a multicast stream**

Multicast streaming, has a lot of applications in our lives, such as the TV or radio. Also can find the videoconferences, the football matches over Internet...

Basically this type of connection reduces the network bandwidth needed to stream over the network.

### 3.3.3 Election for a server

#### 3.3.3.1 History of the streaming servers

The first step to create a streaming system is the selection of the server. Nowadays there are many options about servers that allows to create this kind of systems.

Back in the 90’s, Apple open the multimedia field in 1991, but since the appearance of RealNetworks in 1995, the multimedia transmission over Internet was not an option available.

Referring to streaming as a technology, it arises in 1995 for the transmission of audio, and a little bit later, the transmission of video. This technology opened possibilities over the Internet, to allow access to the content without the need to download it.

Some of the most important characteristics that allowed this progress, are the new codecs and the types of compression. The main function of these features, is to optimize the quality being possible to see or listen the file in real-time.
With the growth of broadband users, the streaming technology is not at all the services it should be. As always, it has formed a business vision over the streaming servers. It seemed at the beginning of this technology, that the bandwidth should be a restriction, but one of the problems that appeared, was that of copyright. Because of this problem, excluding others, the growth of this technology is still slow.

Increasingly, this technology is used more frequently even though the streaming servers are still expensive.

### 3.3.3.2 Compression and standards

As told before, stream means a transmission where everyone can connect to it at the time the customer wants. It is not necessary for the client to connect at the beginning of the transmission, but he can make the request connection in the middle of it.

What characterizes the live broadcast, is that a user can see almost in real-time the content. With the reception of a small part, the client is able to see the content received while he is receiving another flow of data that corresponds to the next part of the file.

The stored part is like a gap between the TCP transmission, shown in Figure 3.19 and the transmission that streaming services need: a continuous transmission.

![Figure 3.19: TCP transmission](image)

Remember that the transmission can be, either a file that is stored into a server and created in a streaming way, or a live media streaming and that means that the content
is created at the same moment as the deliver. Both cases, the audio and video are
distributed with a codification format that is the same in the encoder and decoder.

Encoding is strongly recommended, but not absolutely necessary for playback of
high-quality streaming media. Encoding is the compression of normal sound or video
files. This process reduces files and allows more efficient streaming. Different codecs
optimize the encoding for different bandwidths.

The schema of compression, deletes data for saving space in the disc through math-
ematical formulas. Most of the video compression is lossy, that is a data encoding
method which compresses data by discarding some of it. This method is based on
the fact that there is some data that are not necessary to achieve good quality.

The operation of the method is to make groups of four pixels and compare them with
the following group. The encode or decode will send only the differences within the
two blocks of four pixels.

This procedure is divided in two different operations: spatial and temporal com-
pression. The first one is on two dimensions of the image whereas the second one,
represents the time domain:

- Spatial compression deletes all the areas that have similar features such as bright-
  ness or color. To summarize, it discards the features that human eyes cannot
  appreciate.
- Temporal compression encodes the frames that will be almost the same in a
  series of following frames.

Referring to the standards, Motion Picture Expert Group (MPEG) developed stan-
dard file formats and compression algorithms that the users can use for their video
and audio applications.

MPEG has a methodology of compression asymmetric as the encoder is more complex
than the decoder. The reason for the compression being asymmetric, is because the
encoder needs to be algorithmic or adaptive, and the decoder doesn’t. An example
of the operation is in Figure 3.20. The asymmetric method could be advantageous in
applications such as broadcasting where there is a few number of expensive complex
encoders but a lot of simple inexpensive decoders.

What MPEG did, is not to standardize the encoder, but the way that decoder inter-
prets the bitstream. This is an advantage because not only one type of decoder is
compliant, but all the decoders that can interpret the bitstream. Another advantage
is that over time encoding algorithms can improve.

The MPEG standards did not give a lot of information about the structure and the operation of the encoder. The objective of this, is to let a big scope for the designers and to create a competition on who does the better design according to the standardization. This is good for the users because they have greater choice. The user can choose different levels of cost and complexity, ensure that all are compliant decoders.

A list of the standards that MPEG created are below:

- **MPEG-1**: Done in 1993. It is the first MPEG compression standard for audio and video. It is used on VIDEO CD, SVCD and low quality on DVD VIDEO. Also was used in digital satellite/cable TV. It downsamples the images according to the requirements. It includes the MPEG-1 Audio Layer III, popularly named MP3.

- **MPEG-2**: Released in 1995. Was created a coding of moving pictures and associated audio information. Standards of video and audio use for broadcast-quality television. High-Definition appears. It is considered important because a lot of services such as DVB, digital television signals or SVCD and DVD-Video have been chosen the MPEG-2 compression scheme. Digital cable TV has been stitched from MPEG-1 to MPEG-2. This method of compression, is also used on Blu-ray Discs although they now use MPEG-4 Part 10

- **MPEG-3**: It is the first to have a multi-resolution and scalable compression.
The intention of MPEG-3 was to be the compression of HDTV but was found redundant and was merged with MPEG-2. As a result of failure, there is no standard working.

- **MPEG-4**: Appeared in 1998. Is one of the most popular standard. Characterized for being the coding of audio-visual objects, distribution in CD, bidirectional transmission by videophone and TV emission. MPEG-4 take a lot of features of MPEG-1 and MPEG-2, like support to the VRML (Virtual Reality Modeling Language), support for the Digital External and Varied Right.

  Use further coding tools to achieve higher compression than MPEG-2. Is a standard closer to computer graphics applications.

  The standard includes concepts such as profile and level. These concepts allow to define specific set of capabilities that can be implemented to meet with particulars objectives:

  - **MPEG-4 Part 3**: Audio: A set of codec compression for the flow codification of audio. Includes some variants like AAC (Advanced Audio Coding).
  - **MPEG-4 Part 8**: Transport over IP networks. This part, specify a method for transport MPEG-4 content over IP networks.
  - **MPEG-4 Part 12**: Explains the format of the media based in ISO. A format of files for store multimedia content.
  - **MPEG-4 Part 17**: Format subtitles.

- **MPEG-7**: Once finalized the MPEG-4 standard, appears MPEG-7 with the intention of, linking the audiovisual content elements, finding and selecting the information that the user needs and identifying and protecting the rights of the content. MPEG-7 arises at the moment that the necessity appears to describe the audiovisual contents due to the growing quantity of information.

  MPEG-7 offers a mechanism to describe the audiovisual information, so it is possible to develop a system capable of indexing large multimedia databases and search in these databases, manually or automatically.

- **MPEG-21**: The MPEG-21 standard, presents a framework for exchanging the multimedia content, respecting the rights of the author and distribution, and during all the time, adjusting the capabilities according to the users.
This standard attempts to solve most of the currently existing problems with the distribution of the digital contents. Since the ad-hoc networks endanger the digital contents, MPEG-21 attempts to reduce this problem by giving the participants of a transmission.

In summary, MPEG-21 tries to develop a framework where the main object is the digital content for preventing the misuse as will be the commercial objects in some time.

3.3.3.3 Service model

The conventional scheme for the installation of a video streaming service has two main well distinct activities: the elaboration of the contents in a digital format using the compression and distribution procedures of the contents over the network to clients and final users.

For the elaboration of the contents, it generally exists a first part that consist of a capture of video and audio, when referring to live events or videoconferences. The next step is the compression of this content. In this operation, audio and video are separated.

The distribution of the contents includes the diffusion by URLs. A streaming server stores the content and takes charge to distribute it to the clients. The servers can provide two types of contents:

- VoD (Video on demand): characterized when an individual client, requests a file saved in the server. When this happens, the customer has the same control as in a home video, like play, pause, stop, rewind...
• Diffusion (broadcast): several clients have the same content. Does not matter if is a live content or a file stored previously in the server. The customers do not have control about playback.

3.3.3.4 Differences between the others

The most appropriate option is to install a specialized server, as will be analyzed in the following lines. On one hand, we are able to offer broadcast services that are not available in the conventional web servers, and also, try to save bandwidth as much as possible. On the other hand, the servers take care about the bandwidth available in every moment. Another important feature, is that the clients cannot make copies of the multimedia files that are playing.

In the next lines, there is a list with the most currently streaming servers. About each server, is expose the main features, the benefits and inconvenient.

• **Helix DNA**: The Helix DNA Server is a universal delivery engine supporting the real time packetization and network transmission of any media type to any device. The distribution of it is under the General Public Source License. The list of the supported files are:
  - MP3 audio (mp3)
  - Real Audio and Real Video (.rm, .ra, .rv)

The supported protocols are:
  - RTSP/RTP
  - SDP
  - HTTP delivery support

Looking to the supported files, it is easy to see that it is a server specially for the own formats of Real. Apart from them, it only allows MP3 format. Due to the project often working with format files that are not part of Real, we rule out this server option.

• **QuickTime**: Apple has two very similar products; one is QuickTime Streaming Server (QTSS) and the other one very similar as the first one, Darwin Streaming

---

[8]https://community.helixcommunity.org/content/rpsl
[9]Own Real formats
Server (DSS). The difference between both, is that the second one is an open source server and the code is based on QTSS. Also has the Apple Public Source License\(^\text{[10]}\) that is the license that Apple has for open source applications.

From now, we will make reference to the Darwin Streaming Server. The version when the project started was the 5.5.5, although version 6 is now available. Is a server that unlike the others, is open source and also free to download and use without restrictions of years used.

The main feature already mentioned, is that it is an open source code and this allows us to program or adapt the server as we want. In this project, this feature will not be used for the moment due to the priority of it, being to set a streaming server that allows the users to play media-files without having the necessity to download them before. DSS allows to stream media-files with a concrete format.

All of these files, except the MP3 format, have to be hinted. This is an option that able the file to be transmitted in a stream way. A detailed manual of how to get this formats, is in the chapter Implementation part section Darwin Streaming Server.

DSS use the RTSP, RTP, RTCP and SDP protocols. These protocols are detailed in their operation in the sections above.

Another important feature, is that it has the possibility to create playlists with the files stored in the server or with the live streaming. The playlist is the file that a user requests together with the RTSP protocol. These playlists can be created with a web interface or using the console. They also allow to use a relay system in order to get or give flows from or for another server.

The server, provides a web interface useful for controlling and setting new configurations in it. The website allows actions such as creating playlists, changing the main media-file directory, doing some monitoring tasks like checking how many users are connected in the server and in a playlist or for example, the use of the CPU. Furthermore, it also allows to change settings on configuration: set the utmost number of connections at the same moment, allow the log files or set the throughput. The logs are useful for checking, in case of error, which was the problem that has caused it.

Darwin Streaming Server is a server that currently has benefits comparing to the others free and open source servers. It has an easy installation and operation. It only needs to store the files into the server in the correct format and create playlists that will allow the clients to connect to it and enjoy the media-file.

\(^{10}\)http://www.opensource.apple.com/license/apsl/
Another feature, as told before, is that it is able to program new modules. There is a detailed manual here [12]. It actually is a server that supports almost all the formats and protocols that the users currently use, like PSS and MSS.

Due to it being a free server it has an inconvenient: it is not easy to modify the code or create a new module because of the complexity of the files and codes.

As can be seen, in this project we elected the Darwin Streaming Server as the main server for having a streaming system. In the next chapter, its installation and functioning are detailed.

3.4 XBMC

3.4.1 Introduction

This project, as told in the introduction, is a part of a complete project based on IP Multimedia Subsystem (IMS). Once finished this project, a software is necessary to unify all the “little” projects done. This software is XBMC.

XBMC is an open source software for digital media and is used for media player and entertainment functions. It is a non-profit project that is available for all the main platforms such as Linux, Windows and OSX. Created in 2003 by a programmer and currently is developed and maintained by volunteers located around the world. To expand the software over the world, around 50 translators have been needed to achieve 30 different languages and also around one hundred developers have contributed in order to expand their functionalities.

One of the most important features of XBMC, is that it has an easy operability software that makes it feel natural and comfortable to the client while he is using it. It also has functionalities such as remote control[1] or a beautiful interface and powerful skinning engine. It can be used in many situations and places like in a company for doing presentations or at a domestic house for simply looking the TV or media files.

Another feature which is quite important is the file formats it accepts. Almost all the popular video and audio formats can be played over XBMC software. At the beginning, the software was created for network playback. This means that the user can stream multimedia from his XBMC software using practically any protocol available. The formats accepted for video and audio are detailed in the Table ??.
3.4. XBMC

i.e CD’s or DVD’s, almost all the currently file formats around and even compressed files such as ZIP or RAR.

The idea of this software, is not to make a multimedia software but also to have a multimedia software with almost all the functionalities that a TV and a personal computer have. One of the most popular features is the creation of an automatic library using the files stored in the same hard drive where XBMC software is installed or the possibility to create playlist or slideshow with the media stored. It also has a function that allows the user to check the weather. All this function can be used once the installation is completed. [13]

Another important characteristic is that it is an open source software. This is very important as it can be expanded and new functionalities can be created in the future. In our project, referring to the whole one, it is a main requirement because of the unification of the other different projects.

The table 3.4 is shown a summary of the main XBMC supports.

3.4.2 XBMC database

XBMC has a database that is created when installing the XBMC software. It is a database that is created using SQL[12]. This database has the goal of storing all the information about media-files stored in the hard-disk drive. It is divided in two parts: the first one is called ”xbmc_video” and is used to store the information related to the videos. For example, in this project there is a web interface for helping the users to upload files from their computer to the server. All the information inserted in the website should store in the database. The second part is for the music. The name is ”xbmc_music”. It is important to have a complete database, as XBMC has, for having a complete library, because when a user wants to check if there is a concrete video or music in the server, it only requires a query on the database.

3.4.3 Development on Linux

Two years ago, XBMC Media Center did not exist for Linux platforms. Currently there is a stable version but still not having all the features that other platforms do. The long term goal is to adapt all the features that the other platforms have into the Linux version. As told before, it is an open source software and the language used for programming new features is C/C++. The license used is GPL/LGPL[13].

[12] Structure Query Language
For developing with Linux, the development platform is 32-bit Ubuntu Desktop.

It is possible to add new features or functions, or to improve the existing ones. One option to start with, is to check through the source code how it is structured and try to understand it.

For development, XBMC provides a virtual machine called VMware Player\(^\text{14}\) that help the developer to see how modifications of the code affect the software.

In the main XBMC website there is a detailed manual about how to compile and build the code.\(^\text{14}\)

\(^{14}\)http://www.vmware.com/products/player/
### 3.4. XBMC

**Hardware platforms:**
- Linux on computers of x86 processor architecture. Ubuntu as reference distribution
- Windows versions XP, Vista or 7. Also for computers of x86 processor architecture.
- MAC versions Tiger and Leopard. Also in Apple TV.

**Controller peripherals:**
- Remote control depends on platform and remote
- Mouse, Touchscreen and keyboard via USB

**Output media formats and devices:**
- Widescreen (16:9) and normal-TV (4:3)
- HDTV and Standard Definition output resolution such as 480p, 576p, 720p, 1080i, 1080p and higher
- NTSC and PAL for America/Asia and Europa/Africa respectively
- NTSC playback on PAL TV and vice versa

**Physical input media formats and devices:**
- CD’s, DVD’s, DVD-VIDEO, VCD/SVCD and Audio-CD/CDDA
- Play media-files from any built-in hard-disk drive
- Streaming from a computer/server over a network via UPnP, SMB/SAMBA/CIFS, FTP, HTTP or XBMSP
- Stream supported media-files from the Internet
- Stream media stored

**File/container formats:**
- Audio-CD (CDDA)
- SVCD, VCD DVD-Video from Xbox, hard-disk-drive or also network
- MPEG, MPG, M2V formats
- MP4 (MPEG-4 video) container, AAC and QT/MOV
- MPA, MP”, MP3 and WAV containers

**Video codecs:**
- MPEG-1 and MPEG-2
- MPEG-4 ASP (H.263) and MPEG-4 Advanced Video Coding (H.264)

**Audio codecs:**
- AAC Advanced Audio Coding
- AVS Audio Video Standard

**Picture/image formats:**
- JPEG, JPG, GIF, PNG, BMP, TIF and RAW

Table 3.4: XBMC features, functions and supported audio/video/picture formats
Chapter 4

Implementation part

4.1 Darwin Streaming Server

4.1.1 Installation and function

4.1.1.1 Installation

Darwin Streaming Server can be found in its official website. The software can be downloaded in two different versions, source code or precompiled. The first one is used to modify the code before installing, in order to acquire new modules that bring new features. The second version, is prepared to install directly with the default features.

To install the server, it is necessary to compile the source code in case the first option has been chosen. Once compiled, it is necessary to create a new user in the server that must be called “qtss”. This new user, is to ensure that DSS is executed in a safe mode due to the new user not having root privileges.

After creating the user and downloading the original packet, the next step is go to the folder where the packet is stored and execute the file “Install”. During the installation, it will ask for a “username” and a “password”. These parameters will be needed in a future.

DSS, after its installation, creates a website to check and control parameters such as general settings, connected users or playlist. To access the web page it is necessary to insert into a browser the URL that appears in Figure 4.1.
The website will ask for the “username” and “password” input during the installation.

The default folder that DSS has for the files is /usr/local/movies/ but it is possible to change it for whatever folder desired. To change this setting, go to the website and in General Settings change the path.

Automatically, the installation creates different folders and files that are necessary to develop new modules, change the website, add new files in the default folder of video and audio, or for example, check the logs.

These folders and files are:

```
/usr/local/sbin/DarwinStreamingServer
```

This is the file that starts the streaming server. It might be possible that when the server starts, Darwin Streaming Server does not start. If this happens, it means that the streaming server does not initiate with the server. To solve this, we should place the file in the `/etc/init.d` directory in order to initiate each time the server starts. Also it is possible to start manually and in background using the option `-d`.

```
/usr/local/bin/PlaylistBroadcaster
/usr/local/bin/MP3Broadcaster
```

Both files are used to start the playlists. The first one is used for the video playlists and the second one for the MP3. These files are only used if the playlist is configured using the console. It is necessary to have the right files and then execute one of these files to start the playlist in background.

```
/etc/streaming
```

In this directory we can find the main configuration files. Inside the folder, there is the streamingserver.xml file. This is the file where all the web interface configuration
is. Furthermore, inside the folder there are some information files about groups and users that have access to the server.

/etc/streaming/streamingserver.xml

In the streamingserver.xml file, there is the possibility to modify and add new features in the web page. To modify the file, root privileges should be held. The language used for the web is xml.\(^1\)

/var/streaming/

Here is where the folders and files that contain the configuration of the playlists available and used are. Also stores the log files.

/usr/local/movies

Is the directory where the streaming files are stored. As has been told before, it can be changed. The files stored must be configured as streaming files with the right format and if they are video files, with the hint option.

### 4.1.1.2 Interface

Darwin Streaming Server has an interface that allows the user to perform some control actions on the streaming server. To access it, use the command in Figure 4.1.

Once inside, is possible to change the password.

The list of these actions available in the interface is:

- **Connected users:**
- **Relay status:**
- **General settings:**
- **Port settings:**

\(^1\)Extensible Markup Language
4. IMPLEMENTATION PART

• **Relay settings:**

• **Log settings:**

• **Playlist:**

• **Error log:**

• **Access history:**

The two first options are about statistics. They are useful to see how many clients are connected in a playlist and in a relay, respectively. Furthermore, information about the *IP of the client, bit rate, bytes sent, % packets loss, time connected and connected to* is shown. By default, this page will not refresh automatically, but it is possible to change the option in the refresh interval at the top of the page.

After these options, there is the general settings. They allow to change the media directory. If the directory has been modified, the new one must have the right permissions but, it will be impossible to access the new one. In the same window, it is possible to change the number of users that can access the streaming server at the same moment. Also the throughput can be changed. Finally, it is possible to change the type of the authentication and the passwords.

“Streaming on Port 80” allows to deliver streams over HTTP port 80. If the server also runs a web server, enabling the streaming on port 80, can cause a port conflict that results in one or both of the servers not behaving properly.

The window playlist is, as the name indicates, to create and manage playlist with the files stored in the default directory. In the next section we talk about how to create these playlists.

Going to the last window, Access History, shows the number of times that users access a given playlist.

For more information take a look at [15]

4.1.1.3  Playlist

The playlists is one of the main features DSS has. A playlist is a set of files that have the same format. After being created, it is started, and the clients can connect and play. So, a playlist is the file that allows the client to connect to the streaming server and reproduce the playlists that are being played.
One limitation found during the creation of a playlist, is that not all the formats (audio and video) are accepted. This is because, as has been said before, not all the formats are valid to reproduce in a streaming way.

DSS lets the user to control the server and create playlists into two different ways. The first one is using the GUI and the second one, using the console.

At the time of creating a new playlist, the server lets us to modify different options:

**Name:** is the name given to the playlist and appears in the Available Playlists pane.

**Mount point:** is the path where the clients connect. All the video playlists have the ".sdp" extension even when the audio playlists do not have extension.

One important thing, is that the name and the mount point must be unique. No other playlist can have the same name or mount point as another one.

**Play mode:** there are three options that can be chosen:

- **Sequential:** broadcasts the media in the order in which it appears in the playlist file. When the last media file has ended, the broadcast stops.

- **Sequential Looped:** broadcasts the media in the order in which it appears in the playlist file. When the last media file has ended, the playlist repeats in the same order.

- **Weighted Random:** broadcasts the media in random order using the specified weights to determine how often an item plays. The higher the weight, the more often the item is played. The media plays until a client stops the broadcast.

**Repetition:** lets set the number of items that must play before an item can repeat. If set a value other than zero for repeated items, the number must be less than the number of media files in the playlist.

**Genre:** available only for MP3 playlists and lets choose a category to display in MP3 players.

The streaming server plays the playlist in the same server and this allows the clients to see all the playlists available. When a client interacts with a playlist, the server returns the information of the playlist using the RTP protocol, informing on the number of sequences, temporals marks, etc.
The server gives to each client the same throughput independently of the numbers of clients connected. This is because every client has an independent workflow respect the others.

The clients when are connected to a playlist, receive the information but do not have interaction with the server.

When the user that controls the server starts a playlist with the interface, the server executes the PlaylistBroadcaster process automatically, however if is started with the console, the PlaylistBroadcaster process, must start manually.

Take a look to the two different ways of creating a playlist:

- **Interface**: Creating a playlist using the GUI is much easier than using the console. But this method has a problem: it is not possible to start a playlist automatically. It is necessary to enter the GUI and do it manually.

  With a few steps, the playlist is created:

  The first one is to access to the playlist menu. There, the list of the playlists available for being reproduced appears. Playlists that are still playing or stopped may appear. In this interfaces it is possible to delete, edit and create playlists.

  To access a new playlist, it is only necessary to press the ”New Media/MP3 Playlist”. A window with the options of name, mount point, played mode... will appeared as detailed previously. After putting the files that are located in the DSS directory, the final step is to start the playlist. If all the parameters are right and the media files are in a correct format and hinted, the playlist will start and will be ready to reproduce in a player.

  One important thing, is that all the files that are in a playlist, should have the same codification, otherwise this might cause problems during the reproduction.

- **Console**: The option to create playlists using the console offers advantages such as creating and starting a playlist automatically, using an script or program. This is an important feature because if some client wants to put a video or audio in the server and reproduce, it is useful that the client does not have access to the interface neither the server.

  Thus, this project implements a functionality that allows the client to upload some media files and automatically create a new playlist and then start it. This functionality is developed by using scripts that call the PlaylistBroadcaster process.
For creating a video playlist using the console it is necessary to follow the following steps:

– Create the folder that will contain the .sdp, .config and .playlist files. This folder should be created in the directory /etc/streaming/playlists/ and has to have the name of the playlist.

– Go to the folder created and make a new file called nameplaylist.playlist. This file should contain *PLAY-LIST* followed by the path of the files that will be in the playlist. The path should be written in between ” ” and has to have a number at the end of the line. This number is the weight of the file, and only it is used with the weighted random option.

```
*PLAY-LIST*
'complete_path_of_the_file' 5
```

Figure 4.2: Configuration of the .playlist file

– After the playlist file lacks another file, Figure 4.3 that contains the configuration of the playlist. This file should have the lines below:

```
playlist file /var/streaming/playlists/picasso/picasso.playlist
play mode sequential looped
destination ip address 127.0.0.1
#broadcast name "picasso"
sdp file " /var/streaming/playlists/picasso/picasso.sdp"
destination sdp file " picasso.sdp"
broadcast SDP is dynamic enabled
logging enabled
log file /var/streaming/playlists/picasso/picasso.log
pic file /var/streaming/playlists/picasso/picasso.picd"
```

Figure 4.3: Configuration of the .config file

– Once creating these two files, two last steps are only missing: the startup of the playlist and the copy of the created files.

– To startup a playlist we only need to call the PlaylistBroadcaster command.

```
Server@Server-47$ PlaylistBroadcaster nameplaylist.config
```

Figure 4.4: Command for start a video playlist
– After the command in Figure 4.4 go to the directory created for storing the files .playlist and .config, and we will see that 5 more files have appeared. Those are .current, .pid, .log, .sdp and .upcoming. Therefore, we are only left with copying the .sdp file to the directory /usr/local/movies/. After this step, the playlist will be created and will be playing in background.

– For checking that the playlist has begun in background, one way is to search into processes running with the command of the Figure 4.5:

```
Server@Server-$ ps aux | grep .sdp
```

Figure 4.5: Check of the PID number of a playlist

– If we want to kill the process, we need to search the PID number and kill the process using:

```
Server@Server-$ kill -9 PID_number
```

Figure 4.6: Kill the playlist using the PID found

– With the command PlaylistBroadcaster it is possible to do some more actions other than starting a playlist. Using the command:

```
Server@Server-$ PlaylistBroadcaster -h
```

Figure 4.7: Actions available using the command PlaylistBroadcaster

shows all the possibilities of this command. For example, stopping a current playlist, listing of the currently running broadcasts...

Advice: For creating an MP3 playlist all the steps are the same excepting the .config file. A .config MP3 file has to have the same structure as the Figure 4.8

For create the .sdp file, the command is not PlaylistBroadcaster but rather MP3Broadcaster -c followed by the path of .sdp file.

Independently on how the playlist has been created, to reproduce a starting playlist, it is necessary to have a player compatible with the RTSP protocol. One of the best,
4.2 Software for converting files

Darwin Streaming Server is a server, as the name indicates, very useful because it is easy to install, not difficult to work on and is open source. But one of the problems that the streaming servers have, is that the files have to be in an accepted format. The biggest problem has been getting the formats that the streaming server requires.

```bash
play mode sequential
destination ip address 127.0.0.1
destination base port 554
broadcast mount point ""/playlist""/file"
broadcast name ""file"
broadcast sample rate -1
broadcast genre Pop
working dir "/var/streaming/playlists/file/file"
broadcast password ""videoms"
pid file "/var/streaming/playlists/file/file.pid"*/var/streaming/playlists/file/file.config
```

Figure 4.8: Configuration of the .config audio file

and it is open source, is VLC. Another one that is free is QuickTime. To reproduce a playlist it is just necessary to put the correct URL in the player and wait for a few second and enjoy the streaming media file.

Reproducing a video media file:

```bash
Server@Server-$ vlc rtsp://ips/playlist.sdp
```

Figure 4.9: URL for the video playlists

Reproducing an audio media file:

```bash
Server@Server-$ vlc http://ips/8000/playlist
```

Figure 4.10: URL for the audio playlists

Note that the last command, does not have the extension sdp.

4.2 Software for converting files
At the beginning of the project, this was a problem due to a restriction. The restriction was that only converting the files using the console was valid. This project wanted to do an automatic system that detected a file in a wrong format and then converted it into the right one. The only possible option is using the console.

There are a lot of programs that convert files, but the majority of them are only available with graphic interface. What I was looking for, was a program available for Linux that could operate using the console. Also this program had to have the option to convert the files into MP4 for the video part and AAC for the audio part.

The program elected is the ffmpeg. This program is open source and can be downloaded from the official website

http://www.ffmpeg.org/

Installing the ffmpeg software is not as easy as DSS. New libraries and packets are probably required. An extended manual can be found here [16].

For converting the files that are uploaded into the server, a packet of scripts that do this job automatically is programmed. The function of the whole program is detailed:

### 4.2.1 Converting software

A website interface is done for helping the user to upload video and audio files. The functionalities of this web page are explained on the next section. In this chapter, the process is explained, from once uploaded the media file until the playlist is ready to reproduce.

When a file is uploaded by the website, the “php” file executes a script that starts the conversion process. This script, prepares the file for being converted and calls the software ffmpeg with all its parameters; bit rate, codecs for audio and video, size of the output video... The time of the conversion depends on the size of the original file.

When the conversion ends, the script is responsible for modifying the output file, converting it into a streaming file. This setting that the script performs, is probably the most important because without it, DSS will not allow us to start the playlist. This option is called hint.

Hint is a parameter that is used in all the streaming files. The function of this parameter is to add into the file a set of labels that the server uses to forward or rewind in a faster way. The program that makes this setting is the mp4creator

http://mp4creator.sourceforge.net/

The instructions to follow to get the right configuration are explained in the lines below:
4.2. SOFTWARE FOR CONVERTING FILES

First of all it is necessary to see if the file has setup the option. The command for checking it is:

```
Server@Server:~$ mp4info namefile.mp4
```

Figure 4.11: Instruction that shows the hint option of a file

If in the “type” column there is any row with the hint option, it means that the hint parameter has not been used before. So, the next step is to add the hint setting:

```
Server@Server:~$ mp4creator -hint=1 namefile.mp4
Server@Server:~$ mp4creator -hint=2 namefile.mp4
```

Figure 4.12: Example of how to add the setting hint

It is necessary to call these two commands because one is for the video part and the other for the audio.

Now the file is prepared for being played with the Darwin Streaming Server.

The next step that the script does is to prepare the playlist with the file converted and hinted. As has been explained before, for creating a playlist it is necessary a list of files that contains the configuration and the files included in the playlist. The script prepares all the necessary files and finally start the playlist.

After the video conversion, the playlist was tested in a computer such as a client computer. The playlist works perfectly but one problem found, was the high bandwidth that the computer needed. It is no problem if the computer is in the same network as the University. The problem appears when the network is not the same as the University and the speed of the provider network of the client is lower than the bandwidth the file needs.

Another problem found, was when testing the playlist in a smartphone. Obviously, the connection of a smartphone is not as faster as the connection in a university even if is a connection for a household. So a possible solution found, was to make a copy of the video file but with a lower quality.

Here is when the multiple formats are at stake. In the theoretical part chapter, the “technology used” section, 3gp format has been explained. This format is for mobile devices and with a good conversion, it is possible to have a playlist available for the smartphones.
Currently, the software developed creates two kinds of playlists: the normal one, used specially in the University and with a good connections, and the other, created particularly for the smartphones.

The commands to achieve these two types of playlists, are:

\begin{verbatim}
  sudo ffmpeg -i /var/www/video/upload/$name.avi -acodec libfaac -aq 100 -vcodec libx264 -preset slow -crf 22 -threads 0 - /video/$name.mp4
\end{verbatim}

Figure 4.13: Command for convert the avi file into mp4

\begin{verbatim}
  sudo ffmpeg -i /var/www/video/upload/$name.avi -s qcf-r 12 -ac 1 -ar 8000 -b 300 -ab 12 - /video/$name2.3gp
\end{verbatim}

Figure 4.14: Command for convert the avi file into 3gp

Now, all the smartphones have the option to play these kinds of playlist. What was done with the name of the playlist created specially for the mobiles, was to put the same name but adding at the beginning an "m" for distinguish it from the normal one. So the URL for the mobile playlist is:

\begin{verbatim}
  Server@Server-$ vlc rtsp://164.8.22.227/mnappnameplaylist.sdp
\end{verbatim}

Figure 4.15: URL for the mobile playlists

The mobile playlists were tested with a Blackberry smartphone.

The conversion of audio files is done in another script. This file, like the other one, compares if the input files are in a non acceptable format for DSS, and if is necessary, converts the files into MP3. The part of prepare the playlist is the same as the video part.

Currently, the script is prepared for cheking if the file is in MP3 format, but in a future, it will be possible to have a script that converts from any format to the most currently used formats into MP3.

At the beginning, the conversion program only created playlists using the original file (MP3 format), but when the playlist was tested in a smartphone, only works without stopping the playlist by using WI-FI. Looking at the bandwidth that the smartphone needs, is a bandwidth higher than the downloading rate that can offered
in a 3G connection. The quality of the files converted, has dropped to a quarter of the original. With this reduction of the quality, a smartphone can reproduce these kinds of playlists.

The conversion program creates a copy of the files uploaded and changes the quality of the files. This is done, because lowering the quality, lets the client access with a smartphone to listen to the playlist created without the necessity of using WI-FI. Thus, it creates two playlist: one of them with the original quality and the other one, with the lower quality.

4.3 Interface for upload files

What has been explained until now, is about how to convert media files into an accepted format for Darwin Streaming Server. What lacks now, is how to upload files into the server.

The answer is to use another website. This web page has been created to bring to the user a simple way to upload files. One of the goals of this project is to give to the clients as much simplicity as possible. The user, doesn’t have to know how the project works, he only has to know how to upload files and how to reproduce them in a player.

As previously stated, the project has an easy way to upload files through the website. The web page is located on the same server. To access it, it is necessary to have a browser and write the URL below:

\[
\text{http://ip_server/index.html}
\]

Figure 4.16: URL for access to the upload web page

Let’s talk about the functionalities and the features of the website.

4.3.1 Features and functionality

The website not only offers the possibility to upload a file but also to upload some information about the file that is going to be uploaded. This information is stored in a database.
In this part of the project is when XBMC plays a role. XBMC has a default database that can be enlarged. The default database, is almost completed. The project stores all the information inserted in the web page, into the XBMC database. The main goal of the application is not only to upload a file, but also to grow this database.

The website is divided into three sections as can be seen in Figure 4.17: video, audio and files. These parts are programmed similarly. The things that change in each one, are the files accepted to upload, the size of the files and the access to the XBMC database. The view offered in Figure 4.18 is an example of how the video is part of the website. There are some fields that can be filled in or not, depending on the information on the video uploaded.

In the video site, the client can upload files with the extension .avi or .mp4. It is only possible to select one file for being uploaded. After selecting the file, the client can put some information about it in the fields that are for filling in. Then, there is only one missing step; the submit button. When it is pressed, the file starts to upload into the server and the same happens with the information filled. This process could take a few minutes, depending on the connection and the size of the file.

Once the file is uploaded, the conversion and the creation of the playlist that are explained in the previous section start. These actions, can take some minutes, before the file is prepared for being reproduced. The time of the conversion depends on the size of the file.

The audio part is quite the same as the video part except that it is possible to upload more than one file. The fields of information about the file/s are almost the same. It is interesting that in the audio, it is possible to upload multiple files because some clients may want to make a playlist with a set of files.
The structure of functionality is nearly the same as the video part. The steps to
follow are the same but it change the types of conversion and a little part of the
creation on the playlist.

Since now, have been talked about video and audio parts. The part that lacks, is
the file one. As can see in the Figure 4.17, the web page has a section dedicated to
upload files, like a FTP.

This section allows the user to upload files and to put some information about them.
4 A database called file_xbmc that stores the information filled in the fields of the
website is created expressly.

It is also possible to upload more than a file and the biggest permitted size is 300
Mb.

What this web page pretends, is to help the user as much as possible with the upload of
all the media files. In a future, the functionalities of the website, should be integrated
into the xbmc system.

4Due to the XBMC database only has the video and audio databases created after the installation
Chapter 5

Future functionalities

5.1 Introduction

The main goal of this project is to have a system that allows the clients to use media-files by DSS. The customers can access the website, upload files and play in a streaming way. But there are a set of functionalities that we lack and should be implemented in a future. The aim of this chapter is to explain these functionalities. The structure of the chapter is divided in three parts: the first one talks about the software implemented during the project, the second, is about the website created to upload files from a client to the server, and the third one, talks about the XBMC system.

All these suggestions that are described, have not been tested but their implementation has been studied. This chapter not only wants to suggest new functionalities, but also to improve the developed system. As told in the introduction, this project is a part of a whole project and in order to unify it into the main one, there are some functionalities that are indispensable.

5.2 Converting software

Currently, the software developed for converting the files already uploaded in the server, accepts the AVI and MP4 formats. This software converts the first type of format into MP4 and 3GP and then hints the file. In the case that the input file is MP4, the software converts it into 3GP and hints it if necessary (referring to MP4 files). Actually, this two formats are among the most common over the network but
are not the only ones. So, a functionality that could be interesting to develop, is to modify the code in order to increase the set of the input accepted formats. For the output formats it is not necessary to modify the code because there is no other output format that DSS accepts, except the MOV format, but his use is not as common as the two others.

5.3 Website Interface

The website, is a basic interface that currently allows the customer to upload three types of files: video, audio and files. Improvements that can be done are described below:

- This interface has a limitation, and it is that only allows the user to upload files without seeing which files are in the server. An improvement is to create another website that allows the client to see which files are in the server. It is a useful application because sometimes the user before uploading a file, wants to see if this file is already in the server or not. A viable proposition, is that the new website has a field for making a query into the server and show the requested files. This functionality not only will be useful for the client, but also for the server because it will prevent duplicated files. Furthermore it will save space into the server.

- The next improvement is an application that will help the client at the moment of uploading new files. Currently, the website does not have a GUI while a file is uploading that shows the progress of it. Actually, if a customer wants to upload a file with a large size, he must wait some minutes, and if the client does not know anything about networks, he can think that the server has been blocked or simply does not work. If the client has a GUI that shows the progress of an uploading file, it will be easier for him. This improvement should be done in all the pages that are called “upload.php” in the folder `/var/www/"type_of_media"`, where the types of media can be: video, audio or file.

5.4 XBMC

The last part of the improvements refers to the XBMC system. XBMC is the skeleton of the whole project. The goal of the whole project is to unify all the little projects, but before, all of them must be compatible with XBMC.
One idea is to internconnect the server where DSS is with XBMC. In order to do so we need to access the configuration of XBMC and select DSS in the "add some source" option. So one solution is to add the Server as a source, set the content type (movies, TV shows, music...) and each time the XBMC is started, the content will be updated automatically. It is important that both systems are in the same network. Currently, both systems are in the University, in the same LAN, although in the middle of both systems, there could be a firewall that doesn’t allow these types of connections.

Supposing that both systems use the same network, operation of the new project (the union between DSS and XBMC) would be: A user can connect to the web server interface (the server where is DSS) and upload some files. Then these files will be converted and would be ready for being played in a computer client or a terminal client. Also, this content would be updated in the XBMC system and the user working with it, would see the content automatically.

This option would be the easiest one. For having a complete XBMC system, it is necessary to modify the code in order to adapt the DSS into the XBMC system. In the interface, we should create a new module that allows a user to connect via HTTP into the XBMC. With this module, the web for uploading the content and all the modules that DSS contains for transmitting multimedia files, should be adapted.

The source code of the XBMC interface is located in the folder “720p”. All the .xml files are the config files. Once modified, the code must be compiled and built using this manual [14].

Another feature that XBMC can have, is XBMC web interface. It is an option that can be found in Network Settings, as shown in the Figure 5.1, and allows to control XBMC by a browser via HTTP.

Enabling this option, a user can control all these features:

An example of how the XBMC interface using the web interface option is seen, is shown in Figure 5.2.

This option would be a good idea as we will have the possibility to control the software without being in front of it.

However, this option has a restriction. With the XBMC interface, it is possible to start a new playlist, video or music but it can only be seen in the TV or computer screen where it is connected. The restriction is that it is not possible to show this content via the web browser.

---

1 In this case terminal client means a mobile device or PDA
2 Using the version 1.9.4
Current available features:

- Manage Movie, TV Show and Music databases
- Manage Genres, Covers and Fanarts
- Manage movie sets
- Manage Cast (actors)
- Manage Music
- Quick search
- HTML export for Movies List (all / watched / unwatched)
- Display video flags
- Manage watched / unwatched status
- Control XBMC scraper ("scan for new content" and update XBMC library)
- Browse XBMC video sources and search unscraped files
- Auto-fix path for moved files

Table 5.1: Current available features if the web interface option is enabled
Figure 5.2: Example of the XBMC interface
Chapter 6

Conclusions

One of the goals in the project was to create an easy access to the client in any of the programs installed. This means that the user must not know how the project works or is programmed. The customers only have to know how they can use it. Therefore, in this project programs, scripts and web pages, have been developed apart from the utility of Darwin Streaming Server, that have been adapted according to the necessities of the users.

Nowadays Darwin Streaming Server is a powerful software that can be implemented easily in a computer, independently of the platforms used. In our case, it is very useful because of the project not only having a free streaming server, but also an open source server which can develop new modules and therefore functionalities that adapt to the requirements of the customers. DSS not only has the main functionalities such as broadcasting stored media, but can also broadcast live content such as videoconferences or live speeches. These features are very useful both in the field of education and in business.

Increasingly, the use of mobile devices will grow due as most people opt to have a terminal that perform several functions, almost like a computer. For this reason, in this project the 3GPP project including the PSS and the MSS has been explained. It is necessary to expand the supports, in order to allow all terminals to be able to play streams created by DSS. An important feature that can be implemented in DSS, is the development of a module that adapts the workflow according to the features of the mobile and the scenario. Thus, it is an interesting feature that should be developed in a future because it is useful to have a transmission that adapts to the necessities of the terminal, reducing the quality of the stream but without stopping the connection.
The software choice, XBMC, is a good election because of the potential that this software has. The functionalities that the installation of the source code provide to the users, are numerous. However, fortunately, this software is not closed source. Due to it being an open source software, these functionalities can be improved or new ones created. A good characteristic is that all this open source software has forums that explain the new functionalities developed by different users. Most of them, can be downloaded and added into our XBMC system. For this reason it is a powerful software as it is not only the owner of the software who performs the updates, but also the voluntary developers.
Bibliography


[10] "3rd Generation Partnership Project". http://www.3gpp.org/About-3GPP


[14] XBMC developers "HOW-TO compile XBMC for Linux from source code". http://wiki.xbmc.org/?title=HOW-TO_compile_XBMC_for_Linux_from_source_code

