Universidade do Porto

FEUP Faculdade de Engenharia

Relatório Final

Preparação para a Dissertação

MIEEC 2010/2011

By

Nuno Mota

Table of Contents

1. Introduction **5**

2. State of the Art on Multimedia Streaming **6**

2.1. Introduction **6**

2.2. Video Technologies **6**

2.2.1. MPEG-2/H262 7

2.2.2. H.264/MPEG-4 AVC 9

2.2.3. Containers 10

2.2.4. Ogg 11

2.2.5. Transport Stream - MPEG-2 part 1 12

2.3. Network Protocols **12**

2.3.1. RTP/RTSP 12

2.3.2. RTCP 15

2.4. Web Services **16**

2.4.1. WSDL 16

2.4.2. SOAP 17

2.4.3. REST 19

2.5. Web Standards **20**

2.5.1. HTML5 20

2.5.2. CSS3 20

2.5.3. PHP 21

2.6. Software Streaming Solutions **21**

2.6.1. Darwin Streaming Server 21

2.6.2. VLC 21

2.6.3. LIVE555 Media Server 21

2.7. Programming Tools **22**

2.7.1. Qt 22

2.7.2. JavaFX 22

3. System architecture **24**

3.1. Server 24

3.2. WebAdmin 25

3.3. Client 26

4. Work Plan 27

5. References 28

List of Tables

Table 2.1 - Levels
Table 2.2 - Profiles
Table 2.3 - WSDL elements
Table 4.1 - Work Plan

List of Figures

Figure 2.1 - Quantization
Figure 2.2 - intra-frame
Figure 2.3 - B-frame
Figure 2.4 - Group Of Pictures
Figure 2.5 - Ogg container
Figure 2.6 - Mpeg-2 Transport Stream Hierarchy
Figure 2.7 - RTP packet
Figure 2.8 - RTSP Sequence
Figure 2.9 - RTST Sequence
Figure 2.10 - Options command
Figure 2.11 - Control commands
Figure 2.12 - RTP ports
Figure 2.13 - RTP protocol
Figure 2.14 - WSDL environment
Figure 3.1 - Server Diagram
Figure 3.2 - WebAdmin Diagram
Figure 3.3 - Client/Server Diagram

1. Introduction

Nowadays, millions of users have access to digital contents through many applications that didn’t exist a couple of years ago. Both on the internet as well as in digital television, the viewer transported himself from a passive usage, to a place where he can access any type of contents and information. The revolution of IPTV brought thousands of new possibilities, from choosing the camera view in a football match, to selecting the favorite shows, series, or a selection of movies without the need to wait for the scheduled time.

In the television market, these choices are dictated by the Service Providers and each service has an associated cost. Normally the user needs to rent a Set-top Box (STB) to be able to get the most out of these services, which a basic set-up doesn’t provide. However, in the internet these type of services have proliferated and in a few countries users have access to some free services. But most of these applications are paid, because digital contents also means copyright contents. In most of the cases the specific software for this type of applications is proprietary, which means you either buy a license or pay to create your own software.

The goal of this dissertation is to develop a tool that can provide videos and other contents in this kind environment, a Video-on-Demand (VoD) application. The users must be able to visualize the information of all the contents available and access them inside the same application. This information must be retrieved from the server, which collects information from the appropriate websites. The server should have the capacity to serve several users and at the same time be able to be configured through a web admin interface.

There is several open source software capable of streaming and receiving contents on the internet but they are just a mere startup point for this project, because they don’t fulfill the basic requisites of the application we are looking for. The solution is to use this existing tools and start building up our application.

2. State of the Art on Multimedia Streaming

2.1. Introduction

In this chapter I will present the state of the art of multimedia standards and streaming solutions. First I’m going to describe some of the most important video coding techniques available and the most relevant file formats for multimedia data.

Afterwords an overview of the main network streaming protocols will be given and some necessary tools to implement a web services communication. The software solutions will also be given, specially products where the General Public Licenses (GPL) applies or other free/open-source software that can provide the best results for our application.

2.2. Video Technologies

Video is a sequence of pictures, in which each picture is described by an array of pixels. The red, green and blue signals (RGB) can be expressed as luminance (Y) and chrominance (UV) components. The chrominance may be reduced according to the luminance without affecting the picture quality. The CCIR recommendation 601 defines how the YUV video signals can be described as pixels. Each of these pixels can have millions of colors associated to a number of bits. In each second, several of this pictures, called frames, can be reproduced and a minimum of 15 frames is necessary to obtain a “moving image”.

In the CCIR recommendation 601 a normal video would be 720pixels x 480 pixels x 30 frames and if each pixel had 16 bits of resolution, the video data rate would be around 165Mbit/s. This transmission rate is too high for a user-level application and also the CPU would take time to process such load. According to ANACOM, in Portugal more than 2 million users have cable internet, where speeds go up to 24Mbit/s.

To solve this problem several ways were created to compress video and audio, to reduce them to a size possible of being transmitted at lower speeds without loosing to much quality. The Moving Pictures Experts Group (MPEG), a committee created by the International Organization for Standardization (ISO), was established to create the standard of codification of digital content. The MPEG has 3 groups: MPEG-1,2 and 4. Each group is divided in several parts which include, systems component which specifies container formats, video coding specifications, etc. The most relevant to multimedia streaming are: MPEG-2 Systems and Video parts, MPEG-4 part 10 and 14. All the above standards requires a license granting rights to manufacture and sell products under this standards or use such products to provide video content for profit, otherwise if it is for non profit use only H264 is royalty free.

Besides MPEG there's also other type of video codecs and systems, open-source formats, that can be applied in this case.

2.2.1. MPEG-2/H262

This standard is used nowadays in all kinds of digital applications, for example Digital Video Broadcast(DVB). It involves four parts and it's primarily goal is coding CCIR 601 or higher resolution videos to achieve lower data rates, without compromising the quality of these videos. For now we will only discuss the part 2 of this standard that specifies the video coding.

It can achieve lower data rates from 4 Mbit/s up to 16Mbit/s, but for HDTV content and movie productions it goes up to 80Mbit/s. The principle is to remove redundant information prior to transmission. A main feature of this standard is the three types of compression it has: an intra-frame I-frame, a predictive frame P-frame and a bidirectional predicted frames B-frame. Two major techniques are employed: intra-frame Discrete Cosine Transform (DCT) coding and motion-compensated inter-frame prediction.

In intra-frame DCT a quantisation process is used to reduce the required number of bits to be transmitted in an image block. The quantized DCT block is then scanned for low-frequency coefficients and occurrences of zero-value coefficients. The list of values produced are entropy coded using a variable-lenght code (VLC).

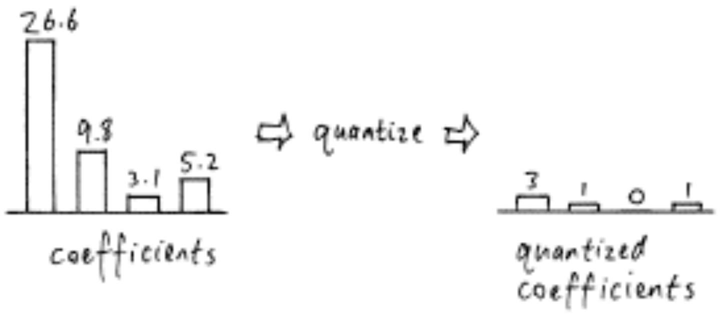


Figure 2.1 - Quantization

In other words, I-frame uses spatial reduction and takes advantage of the incapacity of the human eye, called the phsycovisual redundancy, to notice certain changes in a picture.

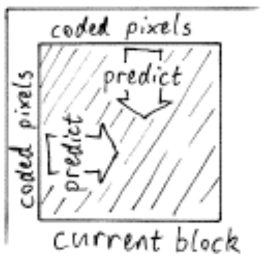


Figure 2.2 - intra-frame

P-frames can have a higher compression because they use motion-compensated inter-frame prediction, which means they're based on precedent frames, I-frames or P-frames. B-frames have the highest compression of the three frames because it uses the past and future frames as reference, but the B-frames itself cannot be used as a reference.

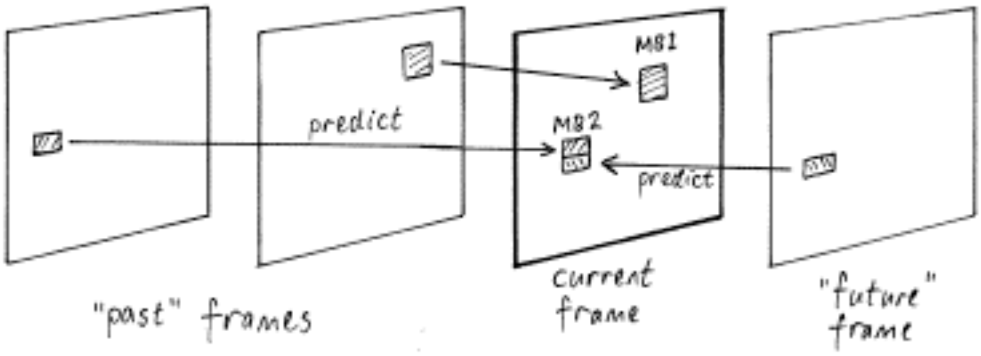


Figure 2.3 - B-frame

All these frames are ordered in a sequential way to create a Group Of Pictures (GOP). From figure 2.4 we can see all the layers that compose the video.

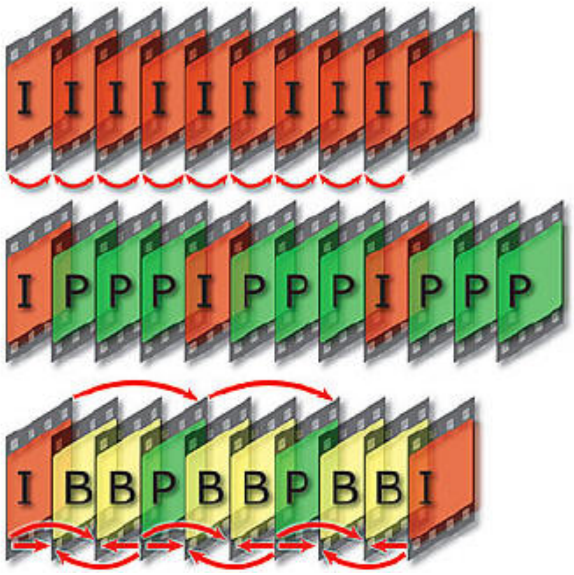


Figure 2.4 - Group Of Pictures

Given that most applications nowadays don't support the full implementation of the standard, some levels and profiles were created to satisfy these needs. In the profile are defined algorithmic tools for compression, and chromatic format. In the levels we can specify maximum resolution and maximum data rate transfer.

Levels	Max. width, pixels	Max. height, lines	Max. Frame Rate	Max bit rate, Mbit/s	Application
Low	352	288	30	4	Set-top boxes
Main	720	576	30	15	DVD, SD-DVB
High-1440	1440	1152	60	60	HDTV
High	1920	1152	60	80	Movie productions

Table 2.1 - Levels

Profiles	Picture Coding	Chroma Format	Scalable Modes	Application
Simple Profile	I, P	4:2:0	none	Video-conference
Main Profile	I, P, B	4:2:0	none	STB, DVD, HDTV
SNR Profile	I, P, B	4:2:0	signal to noise ratio scalable	TDT
Spatially Scalable Profile	I, P, B	4:2:0	spatial-scalable	HDTV
High Profile	I, P, B	4:2:0 e 4:2:2	SNR e Spatial	-

Table 2.2 - Profiles

2.2.2. H.264/MPEG-4 AVC

This standard was created with the goal of reducing substantially the data rate transmissions of the other standards, for example MPEG-2, without increasing too much its complexity and implementation costs. Studies show that if well implemented, it can reduce up to 50% of data rate when compared to other models. Another objective was to be able to use it in several types of applications with different networks, because an increasing number os services, for example HDTV, needed higher coding efficiency.

The standard specification is divided into two parts: the video coding layer (VCL) is responsible for coding the video, and the network abstraction layer (NAL), the part that formats the coded video in a way that can be used in several transport layers or storage media. This standard has similar specifications as other video codecs, because it uses inter-prediction with motion compensation, transform and encoding processes to achieve a H264 bitstream.

A macroblock is used to make a prediction of the previous coded data in two ways, from the same frame (intra prediction) or from already coded and transmitted frames (inter-prediction). These methods are more adjustable than other standards.

A reconstruction filter is applied to every macroblock in order to reduce blocking distortion. With this technique the images are improved and results in a small residual after prediction.

This standard also includes detailed information on how to represent video data and other information. The raw H264 stream consists of a series of pieces called the Network Application Layer Units (NAL unit). These can include two things: information to proper decode the stream called parameters, and the video frames itself called slices.

When it comes to profiles and levels, we now have 17 profiles with several improved features and 16 levels with maximum bit rates from 64 Kbit/s in the baseline profile to 960Mbit/s in the highest profile (High 4:4:4 Predictive Profile - Hi444PP). We can also have levels of resolution from 128x96@30.9 in the first level to 4096x2304@26.7 in the top level.

The blu-ray discs use this standard because the video has better quality at the same bit rate as others, providing more viewing hours. Most of the internet content providers also use this standard for video transmission. For example Youtube uses mostly H264, but with all the infrastructures costs, it now moving to a new standard called WebM that uses a video codec called VP8, which is open source. Nevertheless, H264 is still fully supported by Microsoft and Apple, and VP8 has still a long way to go to become fully developed. Some researches indicate that H264 has better quality with the same data rates and that the VP8 standard may have some patent issues because of its similarity to the H264 specifications. There's also an open source video codec called Theora, supported by Firefox and Opera but in compression-wise it's worst than VP8 and H264.

2.2.3. Containers

Most of the streaming software available supports all the standards described in this paper. Using containers has it advantages even if it increases the payload. With the appropriate containers, we can have audio, video and other information related (e.g subtitles) multiplexed into one single file adding the ability to seek the content more easily.

Like video codecs, video containers have much importance in a video. These containers describe how the video file is organized as a file in the computer and later network protocols are responsible for streaming this data over the internet. So the most important thing is to find a container suitable for video files which is going to be streamed form a server. The protocols that the files are supposed to be streamed are RTP with RTSP protocol for control purposes. This protocols are also detailed in this report in the Network Protocols section.

The containers that I found most important for this type of multimedia context are the following: OGG, Transport Stream (MPEG-2 part 1) , and Matroska. When we look at the specifications of the last one we can see that it's a good wrapper for HTTP streaming, however it's not meant to be used with RTP for one reason, RTP and Matroska both have timing and channel mechanism which would be unreasonable to use two times.

2.2.4. Ogg

Ogg is a free video container used for streaming or for data storage. It's supposed to be as simple as possible with only three major principles: framing, ordering and interleave. It has a simple seeking design when being played, which means an immediate stream capture only with 128kB of data. Choosing any timeline of the Ogg file should then be very quick. The Ogg container uses Ogg pages to build a stream with a unique serial number. Several elementary streams can be multiplexed in one single stream. This means that Ogg is a very powerful video container for media streaming where unreliable channels and information loss is very common.

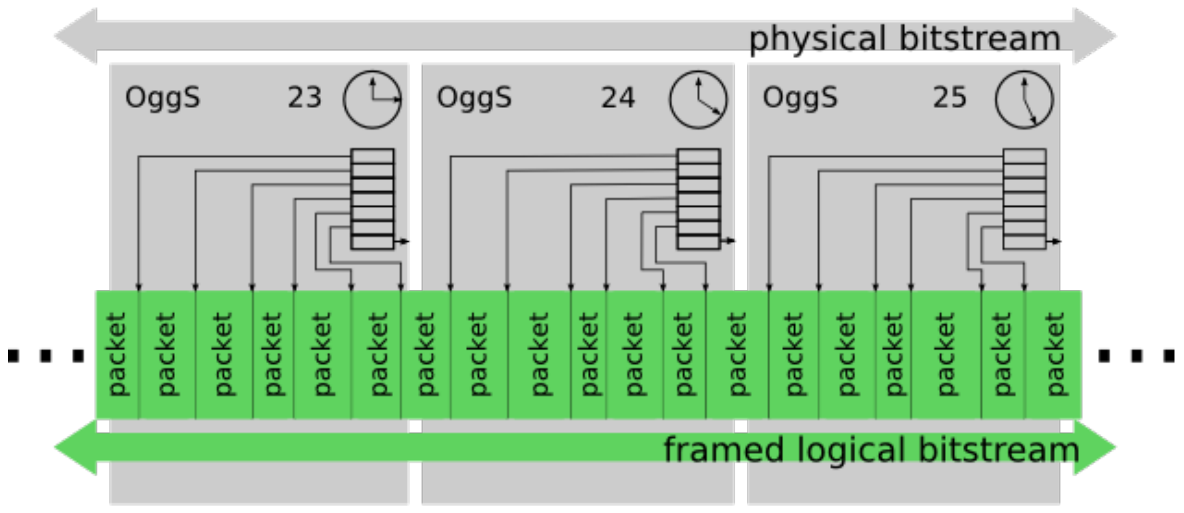


Figure 2.5 - Ogg container

2.2.5. Transport Stream - MPEG-2 part 1

The Transport Stream (TS) belongs to the MPEG-2 part 1: Systems. Its main characteristic is to allow multiplexing of several streams, like Ogg, to have a synchronized output. It's mostly used in DVB applications and offers an error correction system in channels where reliability is not an issue. The packets in the transport layer are 188 bytes in length, one synchronization byte (0x47), three one-bit flags a 13-bit packet identifier, followed by other options and payload data. Eventually, the communication medium may add error correction bytes to the packet, depending on the transmitting signal. The packet identifier is responsible for identifying the different programs/channels present in the transport stream. There's also a feature called Program Clock Reference, that enables a decoder to synchronize audio and video.

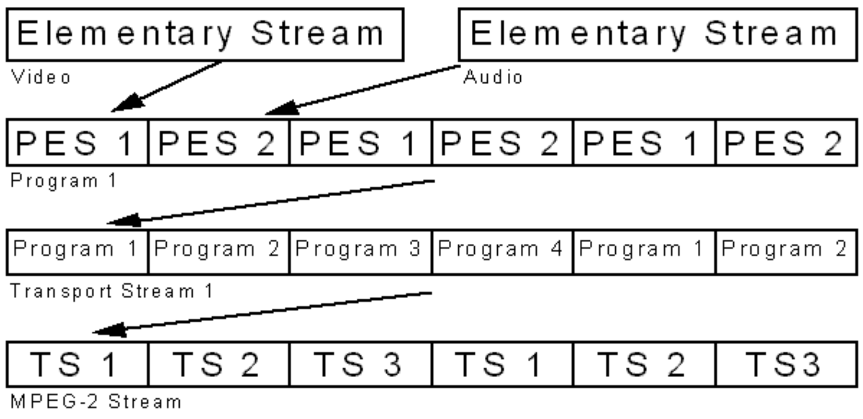


Figure 2.6 - Mpeg-2 Transport Stream Hierarchy

Broadcast industry nowadays uses this standard because multiplexing and demultiplexing of the streams can be done in hardware and because the encoder clock and decoder clock are synchronized.

2.3. Network Protocols

2.3.1. RTP/RTSP

The most important streaming protocol is the Real-time Transport Protocol (RTP). RTP is a standard that delivers real-time data streams, carrying audio and video over unicast or multicast network services. It's typically used on top of User Datagram Protocol (UDP), but it runs in other network or transport protocols. A RTP header is different from a UDP stream making it easier for a firewall to block and inspect the stream. RTP does not ensure that data is delivered sequentially, nor does it guarantee the delivery of the packets. The sequence numbers included in the RTP will tell the receiver how to properly reconstruct the sequence.

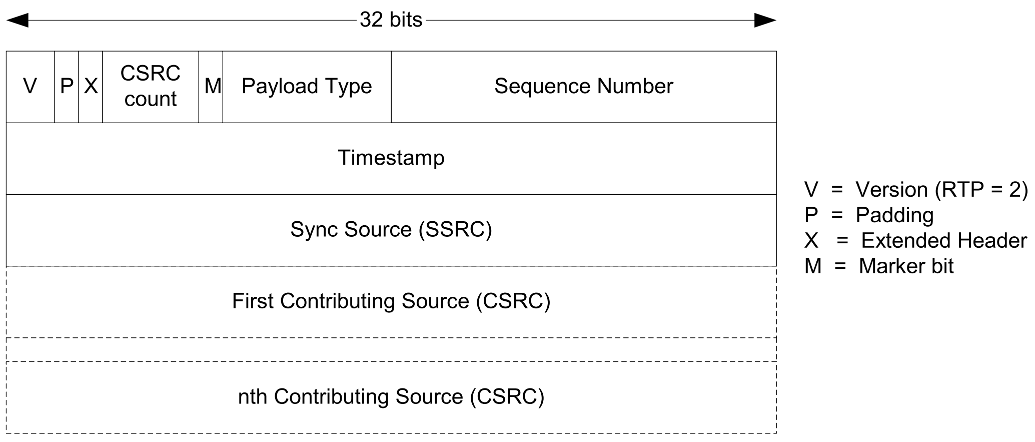


Figure 2.7 - RTP packet

This protocol supports different kinds of media types such as the ones described in this work: H.264 and MPEG-2. For each type of media, RTP has different ways of dealing with the payload.

Another important feature is Real Time Streaming Protocol (RTSP) which is an application level protocol. Real-time streaming Protocol is design to control the media streams, sending the directives to the streaming server. This protocol works like an HTTP connection, the only difference is that RTSP is a stateful protocol. In other words, a session identifier is created to keep track of sessions; so there’s no need for a permanent TCP connection.

A simple VLC server was setup and wireshark was used to log this protocols. In this case, the RTSP behavior observed in the wireshark logs is detailed in figure 2.8.

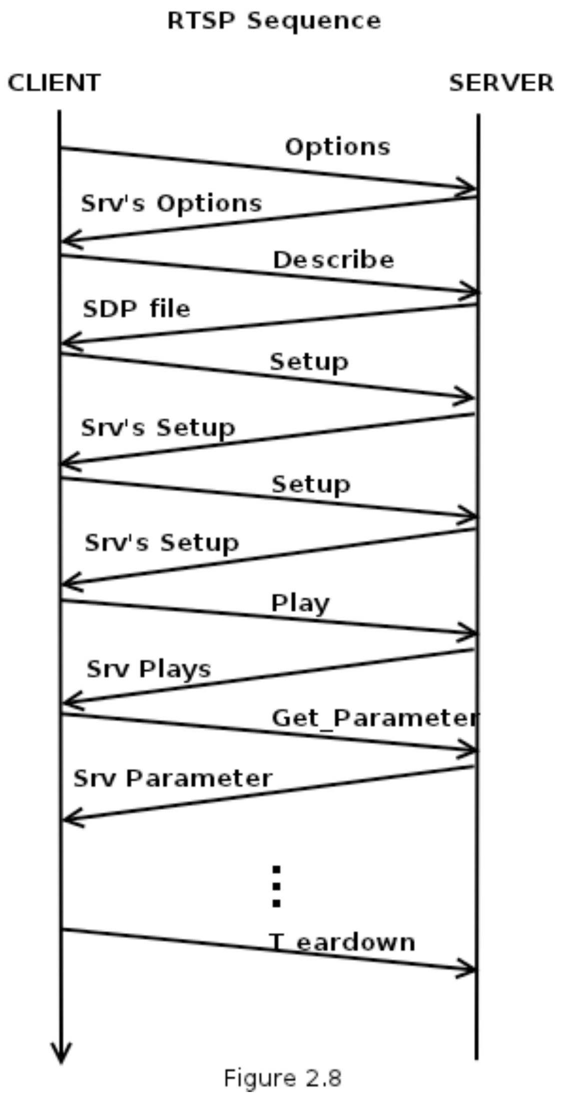


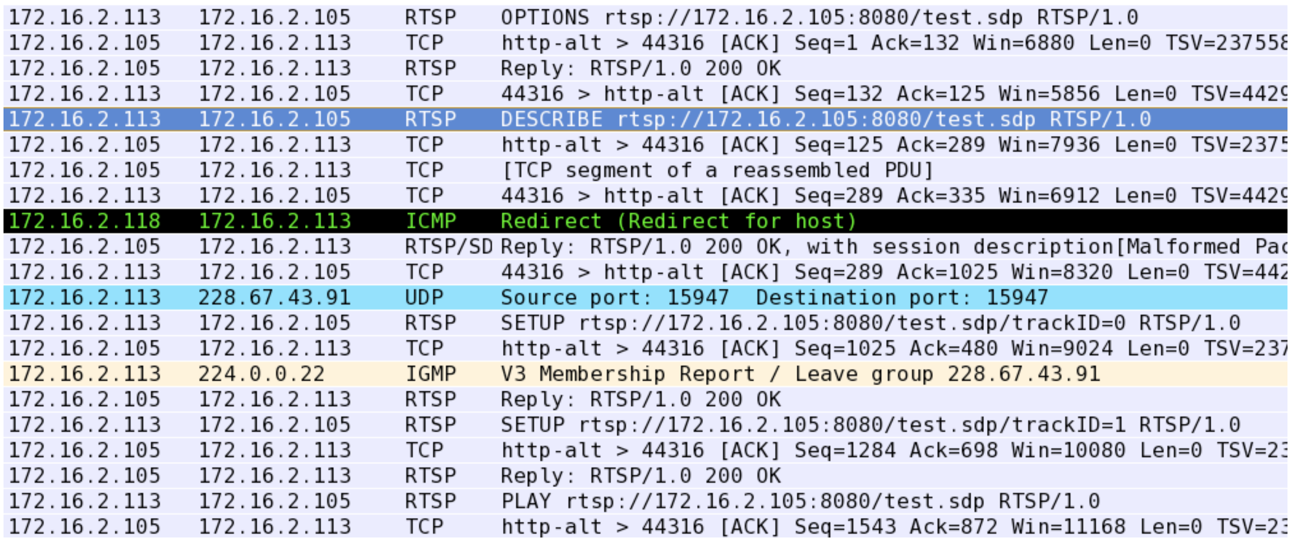
Figure 2.8

Initially the client asks the server for possible control commands with the command Option. Then he asks for a description of the URL in question and this is where the Session Description Protocol comes in. The SDP is a format used to describe the media communication sessions, with the intent of session announcement, session invitation and also parameter association. It doesn't deliver the media itself, it's only used to negotiate between end points all the media properties involved in the communication.

After receiving the description, the client needs to know how the media is going to be transported, so he uses the Setup command. Then he starts the streaming issuing the Play control, and begins to receive the data through UDP connections. The Get_Parameter is a control used to check the server's liveness. More commands could be performed, like Pause, Record and Set_Parameter.

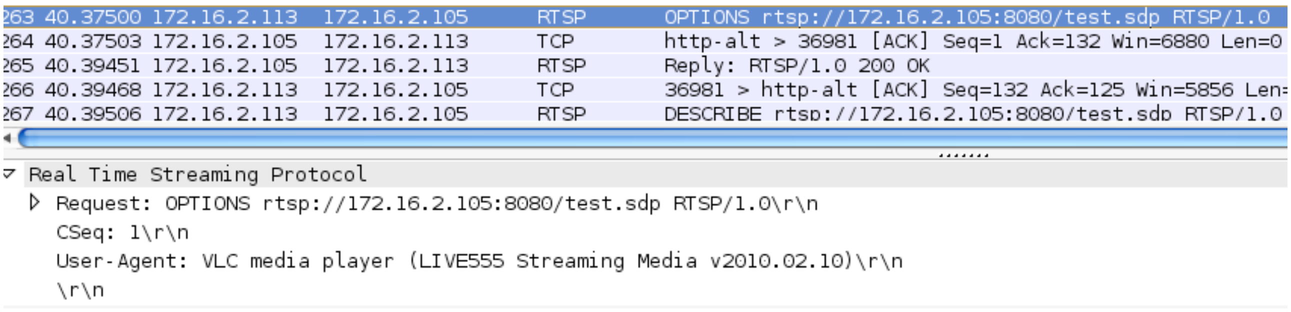
RTP protocol normally uses two UDP connection ports, each one for video and audio streams. The port numbers are even and given to the client in the Server's Setup.

In figure 2.9 we can see all the RTSP commands, exchanged between client and server. The TCP connections corresponds only to Server and Client controls, and no permanent connection is needed when exchanging data.



```
172.16.2.113   172.16.2.105   RTSP    OPTIONS rtsp://172.16.2.105:8080/test.sdp RTSP/1.0
172.16.2.105   172.16.2.113   TCP     http-alt > 44316 [ACK] Seq=1 Ack=132 Win=6880 Len=0 TSV=237558
172.16.2.105   172.16.2.113   RTSP    Reply: RTSP/1.0 200 OK
172.16.2.113   172.16.2.105   TCP     44316 > http-alt [ACK] Seq=132 Ack=125 Win=5856 Len=0 TSV=4429
172.16.2.113   172.16.2.105   RTSP    DESCRIBE rtsp://172.16.2.105:8080/test.sdp RTSP/1.0
172.16.2.105   172.16.2.113   TCP     http-alt > 44316 [ACK] Seq=125 Ack=289 Win=7936 Len=0 TSV=2375
172.16.2.105   172.16.2.113   TCP     [TCP segment of a reassembled PDU]
172.16.2.113   172.16.2.105   TCP     44316 > http-alt [ACK] Seq=289 Ack=335 Win=6912 Len=0 TSV=4429
172.16.2.118   172.16.2.113   ICMP    Redirect (Redirect for host)
172.16.2.105   172.16.2.113   RTSP/SD Reply: RTSP/1.0 200 OK, with session description[Malformed Pac
172.16.2.113   172.16.2.105   TCP     44316 > http-alt [ACK] Seq=289 Ack=1025 Win=8320 Len=0 TSV=442
172.16.2.113   228.67.43.91   UDP     Source port: 15947  Destination port: 15947
172.16.2.113   172.16.2.105   RTSP    SETUP rtsp://172.16.2.105:8080/test.sdp/trackID=0 RTSP/1.0
172.16.2.105   172.16.2.113   TCP     http-alt > 44316 [ACK] Seq=1025 Ack=480 Win=9024 Len=0 TSV=237
172.16.2.113   224.0.0.22     IGMP    V3 Membership Report / Leave group 228.67.43.91
172.16.2.105   172.16.2.113   RTSP    Reply: RTSP/1.0 200 OK
172.16.2.113   172.16.2.105   RTSP    SETUP rtsp://172.16.2.105:8080/test.sdp/trackID=1 RTSP/1.0
172.16.2.105   172.16.2.113   TCP     http-alt > 44316 [ACK] Seq=1284 Ack=698 Win=10080 Len=0 TSV=23
172.16.2.105   172.16.2.113   RTSP    Reply: RTSP/1.0 200 OK
172.16.2.113   172.16.2.105   RTSP    PLAY rtsp://172.16.2.105:8080/test.sdp RTSP/1.0
172.16.2.105   172.16.2.113   TCP     http-alt > 44316 [ACK] Seq=1543 Ack=872 Win=11168 Len=0 TSV=23
```

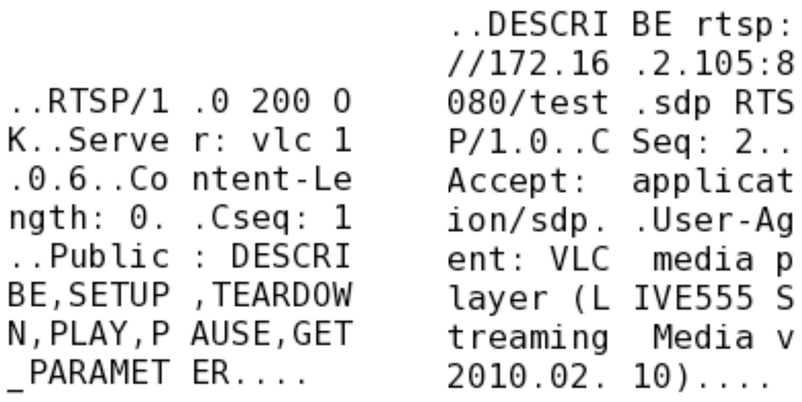
Figure 2.9 - RTSP Sequence



```
263 40.37500 172.16.2.113  172.16.2.105    RTSP     OPTIONS rtsp://172.16.2.105:8080/test.sdp RTSP/1.0
264 40.37503 172.16.2.105  172.16.2.113    TCP      http-alt > 36981 [ACK] Seq=1 Ack=132 Win=6880 Len=0
265 40.39451 172.16.2.105  172.16.2.113    RTSP     Reply: RTSP/1.0 200 OK
266 40.39468 172.16.2.113  172.16.2.105    TCP      36981 > http-alt [ACK] Seq=132 Ack=125 Win=5856 Len=
267 40.39506 172.16.2.113  172.16.2.105    RTSP     DESCRIBE rtsp://172.16.2.105:8080/test.sdp RTSP/1.0

▿ Real Time Streaming Protocol
  ▷ Request: OPTIONS rtsp://172.16.2.105:8080/test.sdp RTSP/1.0\r\n
    CSeq: 1\r\n
    User-Agent: VLC media player (LIVE555 Streaming Media v2010.02.10)\r\n
    \r\n
```

Figure 2.10 - Options command

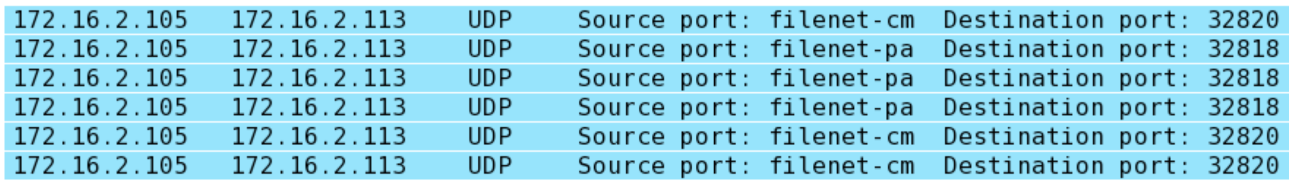


```
                       ..DESCRI BE rtsp:
                       //172.16 .2.105:8
..RTSP/1 .0 200 O      080/test .sdp RTS
K..Serve r: vlc 1      P/1.0..C Seq: 2..
.0.6..Co ntent-Le      Accept:  applicat
ngth: 0. .Cseq: 1      ion/sdp. .User-Ag
..Public : DESCRI      ent: VLC  media p
BE,SETUP ,TEARDOW      layer (L IVE555 S
N,PLAY,P AUSE,GET      treaming  Media v
_PARAMET ER....        2010.02. 10)....
```

Figure 2.11 - Control commands

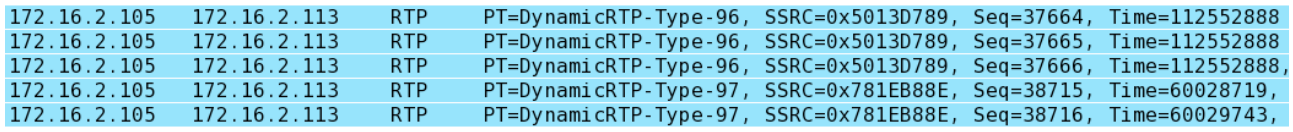
In figure 2.10 we can understand how the command information is sent. The rest of the information is visible through the bytes section in wireshark. An example of a RTSP reply and a Describe query is also visible.

However, RTSP protocol lacks some features. For example, after the reply options we can acknowledge that we only have two possible commands while viewing the stream: Play and Pause. Others like seeking, fast-forward and reverse play, also called Trick Play functionality, are not available. This means that our streaming software should feature this commands with the basic RTSP support.



```
172.16.2.105   172.16.2.113    UDP     Source port: filenet-cm  Destination port: 32820
172.16.2.105   172.16.2.113    UDP     Source port: filenet-pa  Destination port: 32818
172.16.2.105   172.16.2.113    UDP     Source port: filenet-pa  Destination port: 32818
172.16.2.105   172.16.2.113    UDP     Source port: filenet-pa  Destination port: 32818
172.16.2.105   172.16.2.113    UDP     Source port: filenet-cm  Destination port: 32820
172.16.2.105   172.16.2.113    UDP     Source port: filenet-cm  Destination port: 32820
```

Figure 2.12 - RTP ports



```
172.16.2.105   172.16.2.113    RTP     PT=DynamicRTP-Type-96, SSRC=0x5013D789, Seq=37664, Time=112552888
172.16.2.105   172.16.2.113    RTP     PT=DynamicRTP-Type-96, SSRC=0x5013D789, Seq=37665, Time=112552888
172.16.2.105   172.16.2.113    RTP     PT=DynamicRTP-Type-96, SSRC=0x5013D789, Seq=37666, Time=112552888,
172.16.2.105   172.16.2.113    RTP     PT=DynamicRTP-Type-97, SSRC=0x781EB88E, Seq=38715, Time=60028719,
172.16.2.105   172.16.2.113    RTP     PT=DynamicRTP-Type-97, SSRC=0x781EB88E, Seq=38716, Time=60029743,
```

Figure 2.13 - RTP protocol

As explained before, RTP protocol, sends its data through two UDP connections, one for video another for audio, as we can see in figure 2.12. In this example the two even ports used are: 32820 and 32818. Figure 2.13 refers to the same log, viewed in figure 2.12 after a protocol decoding, where we can see the sequence numbers.

2.3.2. RTCP

Real Time Transport Control Protocol (RTCP) is associated with RTP and its main goal is to provide a control channel for each media session. It provides information of reception statistics and current activities. With this information it is able to properly configure any problem with the connection due to its unreliability. RTCP is carried over the same protocol as RTP. This may be an important protocol if we are interested in an adaptive situation where, for example, different bit-rates may be achieved, or just for monitoring purposes.

It works on a report basis with two functions available: sender reports (SRs) and receiver reports (RRs). The sender reports, detail the number of packets exchanged which provides a way to calculate the proper mean data rate for the all session or for every transmission interval. The receiver reports includes statistics like: packet loss, highest sequence number received and a moving average of the inter-arrival jitter of the media packets, which gives an indirect view of the playout buffer used in the receiver.

2.4. Web Services

Web of Services is based on a message-type design found on the web in enterprise solutions and maintained by the World Wide Web Consortium (W3C). It's described as a machine-to-machine interaction. The basic platform used nowadays is HTTP + eXtensible Markup Language (XML) but others elements exist like Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL). In the Web 2.0 movement the big web services providers started to develop open Application Programming Interfaces using this technologies so developers could create new services.

2.4.1. WSDL

As web services and communications protocols are standardized, with so many new services created everyday it's important to describe this communications in a structured way. With aid of XML grammar, WSDL describes network services with a collection of network endpoints, called ports, capable of exchanging messages. Ports can be described as collections of operations supported by the service.

Element	Defines
<type>	The data types used by the webservice
<message>	The messages used by the webservice
<port-type>	The operations performed by the webservice
<binding>	The communication protocols used by the web service

Table 2.3 - WSDL elements

WSDL defines a binding mechanism to relate a specific protocol, data format or structure to an operation.

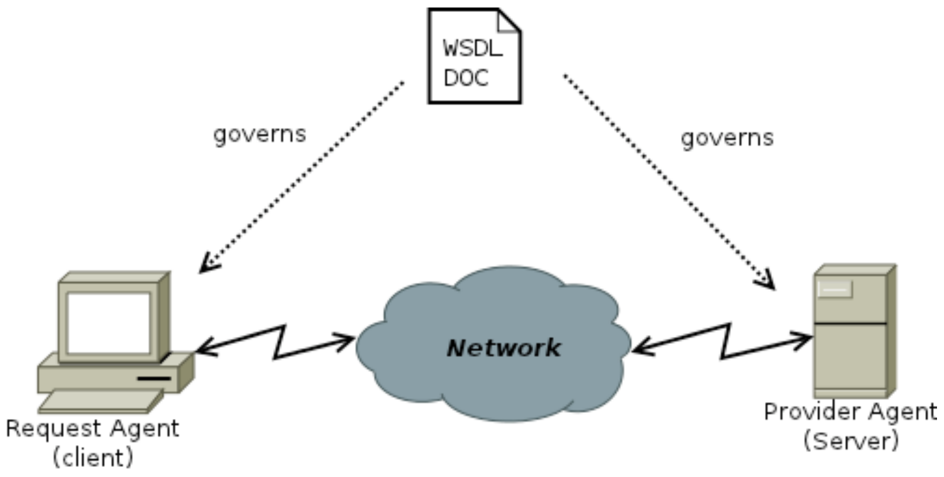


Figure 2.14 - WSDL enviroment

An example of a WSDL document follows:

```
<definitions>

<types>
  definition of types........
</types>

<message>
  definition of a message....
</message>

<portType>
  definition of a port.......
</portType>

<binding>
  definition of a binding....
</binding>

</definitions>
```

2.4.2. SOAP

Simple Object Access Protocol (SOAP) is a simple way to exchange structured and typed information between agents in a spread environment using XML grammar. It does not specify a programming model or implementation semantics but rather defines a way to encode data in packed modules. With this, SOAP can be used in a large variety of services.

The SOAP message consists of a mandatory Envelope with a SOAP Body and an optional SOAP Header. This envelope is the element that identifies the XML document as a SOAP message. The SOAP Body contains the call and response information. One element was described in the SOAP Body to handle error and status information, the Fault element. The SOAP Header provides a way to add extensions without the prior knowledge of both agents, examples include authentication, payment etc.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

The SOAP messages are exchanged through HTTP requests like HTTP POST or HTTP GET.

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

The server then processes the request and answers with an HTTP response.

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

2.4.3. REST

The Representational State Transfer (REST) was introduced by Roy Fielding. This architecture has nothing to do with the other webservices described above, but it shares some similarities like the use of XML as a document format and the use of HTTP forms. The motivation for REST was to capture the characteristics of the Web which has highly desirable architectural properties: scalability, performance, security, reliability, and extensibility.

There are four basic commands that REST uses: HTTP GET, POST, PUT and DELETE. This brings interesting properties, for example, HTTP GET has no side effects as it is only an information retrieval and a very simple one. However this architecture is not only to gather information but to process, update and delete resources. Any information that can be named can be a resource: a document, an image, an stock information. This resources are identified by a Uniform Resource Identifier (URI).

Example of an information retrieval:

```
http://stockquoteserver.example/query?symbol=MSFT
```

WSDL can also be used in REST services using an HTTP binding and all methods are supported:

```
<binding name="HttpBinding" interface="m:GetTemperature"
         type="http://www.w3.org/2005/08/wsdl/http">
  <operation ref="m:location" whttp:method="GET"
             whttp:location="{country}/{city}"/>
</binding>
```

This allows for an HTTP GET on http://weather.example/Sweden/V%C3%A4xj %C3%B6 with the respective response:

```
HTTP/1.1 200 Here's the temperature for you
Content-Type: application/xml
…

<weather xmlns="…">
  <temperature>23</temperature>
</weather>
```

2.5. Web Standards

2.5.1. HTML5

HyperText Markup Language (HTML) has been in use in the world wide web since 1990. Twenty one years have passed and a lot has changed. This standard was first introduced by Tim Berners-Lee and published in 1995 as HTML 2.0 by the Internet Engineer Task Force (IETF) in the RFC 1966 and has had several improvements over the years. The later HTML 4.0 has some features that now are obsolete and the need to improve some characteristics was increasing.

In 2009 the group that was developing the HTML joined with W3C to create the next generation of HTML the HTML5. This standard is not yet official because it’s considered a work in progress, however, most modern browsers have some HTML5 support. The goal of this new standard was to handle today’s internet use and so it needed to follow a few rules to prevent some mistakes of the past. For example, it was established that there was a need to reduce external plugins (like Flash), that the standard should be device independent to be able to adapt to any application and it should have better error handling.

New interesting features are now included in HTML5: the video and audio elements are incorporated for media playback, new input type attributes were created, new content specific elements were introduced like the article, footer, video, audio, progress and many others. New local data objects were created to handle large amounts of data because the previous feature, cookies, was not suitable. In the event section several new events were created to deal with window events, media events, keyboard and mouse events.

2.5.2. CSS3

Cascading Style Sheets (CSS) defines a way to display HTML elements. HTML was intended to contain the content of a document and never to contain tags for formatting. Adding color tags and other formatting tags brought a lot of difficulties to web developers. These styles were added by W3C to HTML 4.0 to overcome those problems and to save lot of work in design implementation. The external style sheets are stored in CSS files.

CSS3 is an improvement of past releases and is divided in several modules: Selectors, Box Model, Backgrounds and Borders, it’s able to deal with Text Effects, Animations and much, much more.

2.5.3. PHP

The Hypertext Preprocessor (PHP) is an open-source scripting language that can be embedded into HTML to provide a dynamic web page creation. Instead of writing lots of commands to output HTML we include instructions that do “something”. Using this language is very easy and brings a lot of new features to HTML design.

One advantage in using PHP is that many webservices can be easily developed because it supports several protocols. It can be used in all major operating systems, including Linux, and it has support for most of the web servers today.

2.6. Software Streaming Solutions

2.6.1. Darwin Streaming Server

Darwin Streaming Server (DSS) is the open source version of Apple’s Quicktime Streaming Server. It uses RTP and RTSP protocol to deliver media streams to clients across the Internet and with its webadmin interface it provides a highly configurable environment. Various platforms are supported and it is intended to stream Quicktime and MPEG-4 media. Features like Authorization, Spam Defense and RTSP redirection are included.

2.6.2. VLC

Almost everything that VLC plays it can be also streamed. VLC is the most famous open source media player and was created in 2001. It can play almost any media file available and stream most what it plays and can be used in every platform Windows, Mac, Linux, Unix, etc.

For encoding and decoding most of it video files it uses a library called libavcodec from the FFmpeg project, but it also includes its own muxer and demuxers. For its serving capabilities VLC uses the LiveMedia library from LIVE555 to support RTSP, RTP and SDP.

2.6.3. LIVE555 Media Server

LIVE555 Media Server is a complete RTSP server based on the LIVE555 Streaming Media library. This includes source-code set of C++ libraries for multimedia standards RTP/RTCP/RTSP suitable for embedded streaming applications. It can stream TS and H264 elementary files, among others. There’s also the possibility to stream to set-top boxes that require raw UDP streaming, rather than standard RTP streaming. It can also stream its RTP(and RTCP) packets over TCP for firewall purposes.

The server supports RTSP ‘trick play’ functionality for some media types. Seeking , Fast forward and Reverse play is possible for TS files for example. Some non-official developers have developed some extra functionalities like ‘trick play’ to other video formats, but still lack the official approval. The latest release included a tool to wrap H264 elementary streams to TS containers to take advantage of the trick play functionality. Other encoding and decoding tools are also available as test programs, which is very useful.

2.7. Programming Tools

2.7.1. Qt

Qt is a programming framework that brings several libraries to support multiple features and facilitate their integration in our applications. It has been in the market for 15 years and since then it has seen a major development. It’s indicated for advanced and highly innovative applications and devices.

The success behind this framework is that it brings all the tools needed to develop advanced GUI applications with embedded multimedia characteristics. The use of native APIs of each supported platform provides full advantage of the system resources with a native look and appeal. The Phonon Multimedia framework library makes it easy to include audio and video in Qt applications and besides that Qt also brings a native XML support library.

In terms of licensing Qt as three strands:

- Commercial license where we can create proprietary applications without the obligation to share the source code and modifications;
- LGPL license where proprietary applications are possible but under the LGPL license and all the source code must be provided;
- GPL license does not give the possibility of proprietary applications and all source code must be provided.

2.7.2. JavaFX

JavaFX is a way to create expressive, multi-rich content which brings capability, performance to our applications. It uses a set of essential technologies, tools and other resources required to develop and create powerful content, able to be used in multiple devices. It has also lots of flexibility because of its intuitive Java platform.

The Java language derives much from C and C++ but it has simpler object model and fewer low-level facilities. This language was created by Sun in 1995 and it’s known for its capability to run in any platform.

3. System architecture

The goal of this dissertation is to create a VoD environment, beginning from a multi-functional server, to an end-user appealing application. The system is to be composed of three major applications: the server interface, the webadmin interface, to control the server and the client application. All these applications must run in a Linux environment and use open-source software for development.

One important topic of this service is the Web Services Application Programming Interface (API), created to provide a set of rules to use the services and resources that this application will offer. This API will have two relevant groups, the administration and the client interface. A WSDL will be used to describe this set of rules of our web service.

3.1. Server

The server will have the following characteristics:

- the main interface which is responsible for communicating with both agents, the administration and the client and should also be able to control/monitor the streaming server for QoS , status and error messages;
- the streaming server, that should run as a stand-alone process and provide feedback to the server's main interface;
- the uploaded video files should be handled by the interface and the appropriate support created, from the uploaded file, using the appropriate tools;
- this multimedia data should be stored in the server's computer;
- the main interface will require a database to store all the clients information and all the video contents information like Title, Cast, Director and other retrieved information;

The best solution, so far for the streaming server, seems to be Live555 Streaming Server, because it's practical and accessible for further development. This server is built on a set of C++ libraries, which means that the Server's main interface should use the same programming language to facilitate interconnectivity and take advantage of some video codification tools. The use of a stand-alone server gives the possibility to build a scalable server where the main interface may control several streaming servers. This means that if the use of the server starts to increase, another computer and network connection may be setup to provide sufficient bandwidth for a reliable service. To provide the streaming server control, a TCP/IP connection must be used, to assure reliability and accessibility to multiple streaming servers.

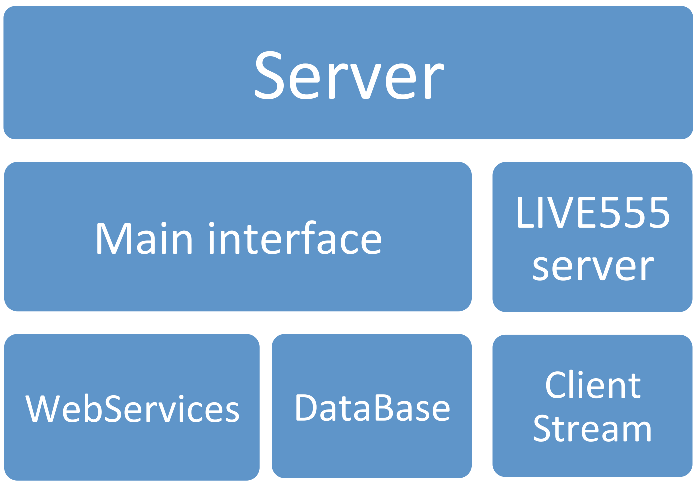


Figure 3.1 - Server Diagram

3.2. WebAdmin

To customize and configure the servers behavior and working requisites, a web interface will be created. This environment will consist of a webpage designed with HTML5, CSS3 and a PHP framework to implement all the needed tools. The interface will communicate with the server via a web services API and it's important to highlight that this type of dimensioning will give us a decentralized solution. With the admin interface, video files and contents which can be retrieved from the appropriate websites, will be uploaded to the servers local storage or database respectively.

The video's information should be retrieved from official websites like IMDB, this information must be serialized by the admin's interface and then uploaded to the servers DataBase.

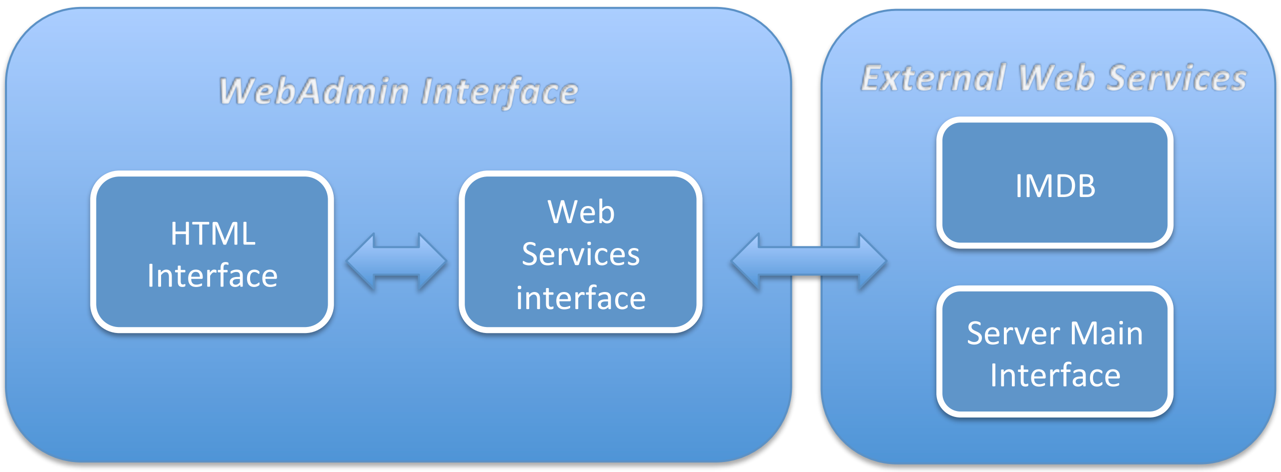


Figure 3.2 - WebAdmin Diagram

3.3. Client

The Client's only concern may be to properly play the video stream, but this application should use the state-of-art tools to create the best multimedia experience and usability. One of the programming tools ideal for this part of the project may be the JavaFX platform because of its capability to bring a feature-rich application.

Every communication will start in the client. After a successful login to the services, he will be able to search for the content available in the server. When it chooses a movie/ series, all the respective information will be shown, and if it's eligible he can play the content.

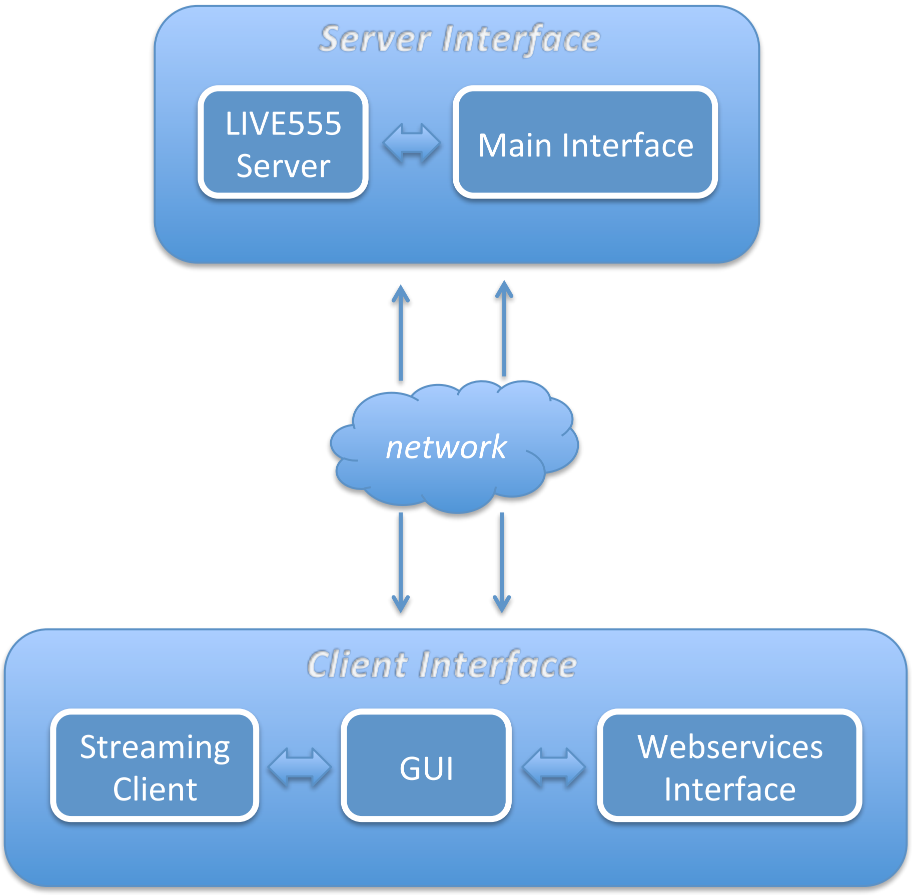


Figure 3.3 - Client/Server Diagram

4. Work Plan

The work will be divided in three major parts. Part 1 will integrate the literature revision among with the work planning and the web services methods definition. The second part will contain all the implementation work. Initially the streaming server solution and the interaction with main interface will be designed. Afterwords, the web services will be implemented, and at the same time it's important to build some of the the webadmin and the client's web services interface. The final work will contemplate some experimental tests to check that the application is fully functional and then the final report writing.

A weekly report will be made to keep track of the thesis work. This reports will be posted online, in a project website where all the relevant information will be present. A To-Do list will also be made with all the important tasks to be carried out each week.

Work		Deadline
Part 1	Literature Revision and Work Planning	20 Fev
	API methods definition	25 Fev
Part 2	Implementation of the server's main interface and streaming server interaction	18 Mar
	Implementation of the server's main interface web services	8 Apr
	Implementation of the webadmin interface	29 Apr
	Implementation of the client's interface	20 May
Part 3	Final test experiments for debugging purposes	23 May to 27 May
	Final Report Writing	30 May to 30 Jun

Table 4.1 - Work Plan

References

1. Free Foundation Software . [Updated 29 June 2007]. *GNU GENERAL PUBLIC LICENSE*. Available from http://www.gnu.org/licenses/gpl.html
2. P.N. Tudor. 1995. *MPEG-2 VIDEO COMPRESSION*. In: Electronics & Communication Engineering Journal, December 1995. Available from http://www.bbc.co.uk/rd/pubs/papers/paper_14/paper_14.shtml
3. ANACOM. 2010. *Serviço de Acesso à Internet - 3º trimestre de 2010*, November 2010. Available from http://www.anacom.pt/render.jsp?contentId=1059837
4. MPEG 2011. *Moving Picture Experts Group*. Available from http://mpeg.chiariglione.org/
5. Victor Lo. *A Beginners Guide for MPEG-2 Standard*. City University of Hong Kong. Available from http://www.fh-friedberg.de/fachbereiche/e2/telekom-labor/zinke/mk/mpeg2beg/beginnzi.htm
6. Iain Richardson, 2007-2008. *Overview of H.264 / AVC*. Vcodec whitepaper. Vcodex Limited. Available from http://www.vcodex.com/h264overview.html
7. S. Wenger,M.M. Hannuksela,T. Stockhammer, M. Westerlund and D. Finger. *RTP Payload Format for H.264 Video*. Internet RFC 3984 February 2005
8. Jason Garrett-Glaser. [Updated June 2010]. *Diary Of An x264 Developer*. Available from http://x264dev.multimedia.cx/archives/377
9. Xiph.Org, 2010. *Ogg Documentation*. Available from http://www.xiph.org/ogg/doc/oggstream.html
10. MultimediaWiki. *MPEG-2 Transport Stream.* MediaWiki. Available from http://wiki.multimedia.cx/index.php?title=MPEG-2_Transport_Stream
11. H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. Internet RFC 3550 July 2003
12. H. Schulzrinne, A. Rao and R. Lanphier. *Real Time Streaming Protocol (RTSP).* Internet RFC 2326 April 1998
13. J. Ott and C. Perkins. *Guidelines for Extending the RTP Control Protocol (RTCP).* Internet RFC 5968 September 2010
14. E. Christensen, F. Curbera, G. Meredith and S. Weerawarana. *Web Services Description Language (WSDL) 1.1*. [Updated 15 March 2001]. W3C 2011.
15. N. Mitra and Y. Lafon. *SOAP Version 1.2 Part 0: Primer (Second Edition)*. [Updated 27 April 2007]. W3C 2011

16. Fielding, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000. ch 5.
17. Hass, Hugo. *Reconciling Web Services and REST Services.* [Updated 2005]. W3C 2011
18. I. Hickson. *HTML5 A vocabulary and associated APIs for HTML and XHTML.* [Updated 19 February 2011]. W3C 2011
19. W3schools 2011. CSS3 Tutorial. Refsnes Data. Available from http://www.w3schools.com/css3/default.asp
20. M. Achour, F. Betz, A. Dovgal, N. Lopes, H. Magnusson, G. Richter, D. Seguy and J. Vrana. *PHP Manual.* [Updated 18 February 2011]. PHP Documentation Group
21. MAC OS FORGE. *Darwin Streaming Server* [Updated 16 Jun 2008]. Apple Inc. Available from http://dss.macosforge.org/
22. VideoLan project. *VLC*. VideoLAN organization. Available form http://www.videolan.org/
23. LIVE555. *LIVE555 Streaming Server*. Live Networks, Inc. Available from http://www.live555.com/mediaServer/
24. Qt Development Frameworks. *QT.* Nokia Corporation. Availabe from http://qt.nokia.com/products/
25. *JavaFX*. Oracle Corporation. Available from http://javafx.com/