Multimedia Terminal for Digital Television

Paulo Sousa Paiva

Dissertation submitted to obtain the Master Degree in Information Systems and Computer Engineering

Jury

Chairman: Prof. João Paulo Marques da Silva
Supervisor: Dr. Nuno Filipe Valentim Roma
Co-Supervisor: Dr. Pedro Filipe Zeferino Tomás
Members: Dr. Luís Manuel Antunes Veiga

May 2012
Acknowledgments

This thesis is dedicated to my parents, Herminio e Filomena Paiva, who taught me the value of education and will now be passed to my sister, Mariana Paiva. I am deeply indebted to them for their continued support and unwavering faith in me.

I would like to express my sincere thanks to Dr. Nuno Roma and to Dr. Pedro Tomás for giving me an opportunity to pursue a Master’s program and for their support, guidance and patience throughout this project. They seemed to know just when I needed encouragement and has taught me a great deal about all aspects of life.

I am eternally grateful for my girlfriend, Marta Santos for her constant love and strength throughout the year. Without her, and her ability to raise my spirits when I was most discouraged, I could never made it this far.
Abstract

Nowadays almost every home has some kind of television service offered by some provider. Frequently, providers use proprietary software to operate the audio/video/data content (e.g., recording and displaying a TV show). As a consequence, it is difficult for the end-user to add new features. In this thesis an open-platform multimedia terminal based on existing single purpose open-source solutions is presented. The objective of this platform is, along with the implementation of the common features already available, to provide a common platform that can be easily changed to integrate new features. In this sense a multimedia terminal architecture is proposed and implemented that natively supports: (i) multi-user remote management; (ii) broadcast of recorded contents to remote devices such as laptops and mobile phones; (iii) broadcast of real-time TV or video/surveillance cameras.

Keywords

Audio and Video Encoding, Multimedia Terminal, Remote Management, Open Source Software
Resumo

Atualmente quase todos nós temos em casa um serviço de televisão pago que é disponibilizado por inúmeras companhias de televisão privadas. Na maioria dos casos, estas companhias têm software especializado para a gestão dos serviços audio/video por eles disponibilizado (por exemplo, gravação e visualização de programas de televisão). Uma consequência directa da disponibilização deste software é o facto de a empresa que fornece o serviço também fornece o software para o operar, impossibilitando o utilizador final de alterar ou acrescentar novas funcionalidades ao software já existente. De forma a superar estas limitações, nesta tese é apresentada uma solução que agrega diversas ferramentas de software independentes e com licenças gratuitas, numa única plataforma aberta. O objectivo da tese prende-se no desenvolvimento de uma plataforma gratuita, que disponibiliza uma série de funcionalidades básicas oferecidas por este tipo de serviços. Adicionalmente deve permitir uma fácil alteração e adição de novas funcionalidades. Desta forma, a plataforma multimédia proposta e implementada fornece as seguintes funcionalidades: (i) gestão remota de conteúdos por vários utilizadores; (ii) transmissão de conteúdos gravados para diversos dispositivos (exemplo, computadores portáteis e telemóveis); (iii) transmissão em tempo real de programas de televisão e/ou câmaras de vigilância.

Palavras Chave

Codificação de Audio e Video, Terminal Multimedia, Gestão Remota, Software Aberto
# Contents

1 Introduction .......................................................... 1
   1.1 Objectives ...................................................... 3
   1.2 Main contributions ........................................... 5
   1.3 Dissertation outline .......................................... 5

2 Background and Related Work ................................... 7
   2.1 Audio/Video Codecs and Containers ....................... 8
      2.1.1 Audio Codecs ............................................ 9
      2.1.2 Video Codecs .......................................... 9
      2.1.3 Containers ............................................ 10
   2.2 Encoding, broadcasting and Web Development Software . 11
      2.2.1 Encoding Software .................................... 11
      2.2.2 Broadcasting Software ............................... 12
   2.3 Field Contributions .......................................... 15
   2.4 Existent Solutions for audio and video broadcast .... 15
      2.4.1 Commercial software frameworks .................... 16
      2.4.2 Free/open-source software frameworks .............. 17
   2.5 Summary ...................................................... 17

3 Multimedia Terminal Architecture ................................ 19
   3.1 Signal Acquisition And Control ............................ 21
   3.2 Encoding Engine ............................................. 21
      3.2.1 Audio Encoder & Video Encoder Modules ........... 21
      3.2.2 Profiler ............................................... 21
   3.3 Video Recording Engine ...................................... 22
   3.4 Video Streaming Engine ...................................... 23
   3.5 Scheduler ..................................................... 24
   3.6 Video Call Module ........................................... 24
   3.7 User interface ................................................ 25
   3.8 Database ..................................................... 25
## List of Figures

1.1 All-in-one multimedia solution. Multiple inputs processed by the server which supports multiple outputs. .......................................................... 3

3.1 Server and Client Architecture of the Multimedia Terminal .................................................. 20
3.2 Video Recording Engine - VRE .............................................................................. 23
3.3 Video Streaming Engine - VSE .............................................................................. 24
3.4 Video-Call Module - VCM .................................................................................. 25
3.5 Several user-interfaces for the most common operations .................................................. 26

4.1 Mapping between the designed architecture and software used ........................................ 30
4.2 Model-View-Controller interaction diagram ................................................................. 32
4.3 Multimedia Terminal MVC .................................................................................... 34
4.4 Authentication added to the project .............................................................................. 36
4.5 The administration controller actions, models and views ................................................. 37
4.6 AXN EPG for April 6, 2012 ................................................................................... 39
4.7 The recording controller actions, models and views ......................................................... 40
4.8 Time intersection graph ............................................................................................ 41
4.9 Recording validation pseudo-code .............................................................................. 42
4.10 Relation between Planet, Atmosphere and Flow ............................................................ 44
4.11 Flumotion simple streaming application architecture .................................................... 45
4.12 Manager basic XML configuration file ......................................................................... 45
4.13 Manager connections definition ................................................................................. 46
4.14 Manager components definition .................................................................................. 46
4.15 General Worker XML definition ................................................................................. 48
4.16 Some Flumotion possible setups ................................................................................. 49
4.17 Launching the Flumotion manager with the nohup command ...................................... 49
4.18 Stop Flumotion server script ...................................................................................... 49
4.19 Flumotion recorder component definition .................................................................... 51
4.20 Recording flow, algorithms and jobs ............................................................................ 52
List of Figures

5.2 Considered codecs assessment values .......................................................... 59
5.3 Comparison between the five considered samples encoded with H.264 codec. ... 60
5.4 CBR vs VBR assessment ............................................................................... 61
5.5 Encoding quality comparison between VP8 and H.264 high profile (source [69]). 64
5.6 Average execution time of the tested tasks .................................................... 72
5.7 Deviation time execution of testing tasks ....................................................... 72
5.8 Multimedia Terminal in Sony Xperia Pro ....................................................... 74

A.1 Execution times and errors for each user on each task (1st Time) .................. 86
A.2 Execution times and errors for each user on each task (2nd Time) ............... 87

B.1 General Question 1. Users age ................................................................... 90
B.2 General Question 2. Users sex .................................................................... 90
B.3 General Question 3. Users occupation ........................................................... 91
B.4 General Question 4. Users computer expertise ............................................. 91
B.5 Evaluation 1. No geral estou satisfeito com a facilidade de utilizacão do sistema. 92
B.6 Evaluation 2. Conseguí executar todas as tarefas eficazmente utilizando o sistema. 92
B.7 Evaluation 3. Conseguí executar todas as tarefas rapidamente ..................... 93
B.8 Evaluation 4. Senti-me confortável a utilizar o sistema ............................... 93
B.9 Evaluation 5. Sempre que encontrei erros consegui recuperar e corrigi-los facilmente. 94
B.10 Evaluation 6. A organizacão dos menus e itens é clara ............................... 94
B.11 Evaluation 7. A organizacão dos botões é clara ........................................... 95
B.12 Evaluation 8. Percebi imediatamente qual a função de cada botão ............. 95
B.13 Evaluation 9. Gostaria de utilizar este sistema em casa ............................. 96
B.14 Evaluation 10. Qual a nota final que daria ao sistema, em termos da sua funcionalidade e facilidade de Utilização? .............................................. 96

C.1 Movies - Encoding quality comparison between VP8 and H.264 high profile (source [69]) ................................................ 98
C.2 HDTV Movies - Encoding quality comparison between VP8 and H.264 high profile (source [69]) ........................................ 99
List of Tables

2.1 Comparison of the considered solutions ........................................... 18
4.1 Rails default project structure and definition .................................... 33
4.2 Create, Read, Update, Delete associated actions and concepts ............... 34
4.3 Flow components - function and parameters ..................................... 47
4.4 Channels list - code and name matching for TV Cabo provider ............... 50
5.1 Profiles conclusion ........................................................................... 60
5.2 Encoding properties and quality level mapped with the samples produced for the first evaluation attempt ......................................................... 62
5.3 User's evaluation of each sample ...................................................... 62
5.4 Encoding properties and quality levels mapped with the samples in the second questionnaire ............................................................... 62
5.5 User's evaluation of the three new samples ........................................ 63
5.6 Descriptions of the sequences used in the codec encoding comparison (source [69]) ................................................................. 65
5.7 CPU and memory usage samples ....................................................... 66
5.8 Medium quality network bandwidth samples ...................................... 66
5.9 Medium quality network throughput samples ..................................... 67
5.10 3 Clients simultaneously using the system at Medium quality ............... 67
5.11 Key features of the test users .......................................................... 69
5.12 Tested tasks .................................................................................... 70
5.13 Expected usability objectives .......................................................... 71
5.14 Average scores of the satisfaction questionnaire ................................ 73
List of Tables

TV  Television

NTSC  National Television System Committee

PAL  Phase Alternate Line

SECAM  Sequential Color with Memory

UHF  Ultra high frequency

VHF  Very high frequency

DVB  Digital Video Broadcast

HDD  Hard Disk Drive

IPTV  Internet Protocol television

MMT  Multimedia Terminal

SAAC  Signal Acquisition And Control

VSE  Video Streaming Engine

VRE  Video Recording Engine

VCM  Video Call Module

UI  User interface

OS  Operating System

RoR  Ruby on Rails

PSNR  Peak Signal-to-Noise Ratio

MVC  Model-View-Controller

DRY  Don’t Repeat Yourself

CRUD  Create, Read, Update, Delete

RDL  Relational Database Language

EPG  Electronic program guides

REXML  Ruby Electric XML

RTSP  Real Time Streaming Protocol

AAC  Advanced Audio Coding
List of Tables

IE  Internet Explorer

GPL  General Public Licence

CIF  Common Intermediate Format

QCIF  Quarter CIF
# Introduction

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Objectives</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Main contributions</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Dissertation outline</td>
<td>5</td>
</tr>
</tbody>
</table>
1. Introduction

Television (TV) is the major adopted communication means to disseminate all kinds of contents and information worldwide. Several TV providers exist that are specialized in a set of services and contents, from the basic TV streaming to the interactive TV play. It is the case of news channels, sports channels, art channels, government channels, among several others.

In the beginning of television, broadcast technology enforced users to use analog and black-and-white television. Low quality big televisions and big antennas to capture the signal were worldwide adopted. The signal was broadcast in NTSC, PAL or SECAM analog encoding and then modulated using VHF or UHF.

Today, analog transmission of TV signal is being replaced by audio/video digital broadcast technologies. The Digital Video Broadcast (DVB) system is subject to international standards and is maintained by the DVB Project, consisting of an international consortium with more than 270 broadcast members around the world. There are several DVB families, each one used in a specific zone. In Europe the prevalent standards are the following:

- Cable broadcast (DVB-C) to broadcast digital television over cable in MPEG-2 or MPEG-4 standards.
- Satellite broadcast (DVB-S) to broadcast through satellites that serves every continent around the world using the MPEG-2 standard.
- Terrestrial broadcast (DVB-T) to transmit compressed digital audio, video and other data in an MPEG transport stream.
- Handheld broadcast (DVB-H) adopted to provide digital television service to mobile handsets.

Along the las decade, a new technology has also emerge to broadcast digital television - the Internet Protocol television (IPTV). In an IPTV based system, all the audio and video data for representing audio/video in our display/speakers is passed trough Internet protocols. IPTV provides the traditional television view and also interactive channels, where the user can set all sort of configurations to increase the viewing and interaction experience.

As there are different standards, there are also many providers with all sorts of services. For this, each provider often has or uses proprietary software and hardware in order to deliver the services they offer. This limits the possibility for the end-user to implement new features or to modify current ones by changing the existing services.

Currently, the majority of users wants an all-in-one solution with the possibility of adding more features to it if/as necessary. For example, imagine a scenario where a user with one or more TV cards provided with a computer could: view TV on-demand, record TV shows (and as long as the user is provided with an Internet connection he can record any show without physical access to the computer), video-call (if the user has an Internet connection, web-camera and microphone).
This idea is represented in Fig. 1.1 where many different inputs are acceptable, from satellite broadcasts to a video-camera or surveillance-camera, and many types of devices can access this service with no extra costs for the user. Such a system would allow, for example, the user to see a TV show in his mobile device while in the train without extra costs beyond the Internet connection. To make this possible, the user would just have to connect to the Internet gateway installed at home, login and request a channel. The server would then select the input from its TV provider, change to the designated channel, compress the audio and video stream and send it to the user.

Finally, it is important to stress that this application should not be associated or even compared with other streaming applications spread around the Internet, where the main purpose is to stream illegal content (e.g., TVTuga, http://www.tvtuga.com) or to stream legal content in some specific countries (e.g., TVCatchup, http://www.tvcatchup.com where the users can view some selected channels and only works in the United Kingdom). This application is for strict personal usage and requires the user to have a subscription with a TV provider.

### 1.1 Objectives

Based on the all-in-one service illustrated in Fig. 1.1, the objective is to develop an open-platform entirely based on already available open-source tools and executed on off-the-shelf computers. This platform should also implement a set of basic services: TV play, TV record with some advanced options, such as recording a program by its name, and video-call.

To summarize, the proposed system should be characterized with:

- Input devices: satellite, video or radio antenna, cable TV, IPTV.
1. Introduction

- Outputs to: TV, computer, laptops, mobile devices, Hard Disk Drive (HDD) (recording), others (e.g. PlayStation Portable, Nintendo DS, . . . ) through a local connection, LAN and/or Internet.

- Supported services: on-demand TV, on-demand radio, voice and video calls, video conference, support for surveillance systems.

TV Streaming

One of the main utilities of digital video is to be reproduced. After correctly acquiring the TV signal, it is important to be able to play it in real-time or even to play it while recording. This reproduction facility of the audio/video signals is often denoted as streaming, where media is constantly received and displayed to the end-user through an active Internet connection, whether or not the user is next to the computer where the signal is acquired.

The universal nature of TV streaming services makes them available to users under normal circumstances, provided that they have access to the Internet. By using the facilities offered by any personal computer with an Internet connection, any TV with a data communication link, or even a mobile device (provided that the OS supports it), can allow the user to view any channel that the user owns at home. This is intended for strict personal use, meaning that only the legitimate user should be able to benefit from this, and it will not be permitted any kind of illegal broadcast service.

TV Recording

At least once, everyone has already missed a TV show that could not be recorded because there was nobody at home to do it and there were no means to remotely set the recording. This is a basic functionality that the proposed platform should provide, i.e., the ability to remotely manage the recording of any TV show or program that, by some reason, the user cannot see.

Video-call

Currently most TV providers also offer their customers a Internet connection. The simplest question is: if the customer has a television and an Internet connection, why cannot he do a video-call, by installing an additional web-camera and a microphone? This feature would not only be easy to implement as it could even be offered/sold as a new product package, together with a microphone and video camera.

There are numerous advantages in this service. For example, a mother with her laptop at work could easily check a sick child at home while he is in bed (considering that he has a TV, computer or mobile device next to him) without extra costs.
1.2 Main contributions

The contributions of this work are as follows:

- A free and fully operational framework with a high level of usability for regular users (e.g. users with no programming knowledge) and advanced users (e.g. programmers).

- Promote the integration of several existent open-source tools into the conceived framework with low effort.

- Full customizable all-in-one multimedia terminal, offered with some basic features.

- Overcoming of existent solutions in terms of supported features and of easiness to add additional ones.

The main contributions of this thesis were already published as a scientific paper entitled Open Source Platform for Remote Encoding and Distribution of Multimedia Contents in the proceedings of the Conference on Electronics, Telecommunications and Computers - CETC11 [90] in Lisbon, Portugal, November 2011. The published paper presented a multimedia platform for digital television based on existing single purpose open-source solutions, with the possibility of integrating new features or changing existing ones (e.g., recording format and coding standard).

1.3 Dissertation outline

This paper is organized as follows:

- **Chapter 2 - State of the art** - describes and analyzes the related work already carried out, focusing on the pros and constrains of the existent applications, commercial and free/open-source;

- **Chapter 3 - Multimedia Terminal Architecture** - presents the proposed framework architecture with detail, along with all the components that integrate the framework in question.

- **Chapter 4 - Multimedia Terminal Implementation** - describes all the used software, along with alternatives and the reasons that lead to the use of the chosen software; furthermore it details the implementation of the multimedia terminal and maps the conceived architecture blocks to the achieved solution;

- **Chapter 5 - Evaluation** - describes the methods used to evaluate the proposed solution; furthermore, it presents the results used to validate the platform functionality and usability in comparison to the proposed requirements;

- **Chapter 6 - Conclusions** - presents the limitations and proposes for future work along with all the conclusions reached during the course of this thesis;
1. Introduction

- **Bibliography** - All books, papers and other documents that helped in the development of this work;

- **Appendix A - Evaluation tables** - detailed information obtained from the usability tests with the users;

- **Appendix B - Users characterization and satisfaction results** - users characterization diagrams (age, sex, occupation and computer expertise) and results of the surveys where the users expressed their satisfaction;
Background and Related Work

Contents

2.1 Audio/Video Codecs and Containers ........................................ 8
2.2 Encoding, broadcasting and Web Development Software .............. 11
2.3 Field Contributions ............................................................... 15
2.4 Existent Solutions for audio and video broadcast ...................... 15
2.5 Summary .............................................................................. 17
2. Background and Related Work

Since the proliferation of computer technologies, the integration of audio and video transmission has been registered through several patents. In the early nineties, audio and video was seen as mean for teleconferencing [84]. Later, there was the definition of a device that would allow the communication between remote locations by using multiple media [96]. In the end of the nineties, other concerns, such as security, were gaining importance and were also applied to the distribution of multimedia content [3]. Currently, the distribution of multimedia content still plays an important role and there is still lots of space for innovation [1].

From the analysis of these conceptual solutions, it is sharply visible the aggregation of several different technologies, in order to obtain new solutions that increase the sharing and communication of audio and video content.

The state of the art is organized in four sections:

- **Audio/Video Codecs and Containers** - this section describes some of the considered audio and video codecs for real-time broadcast and the containers where they are inserted;

- **Encoding and Broadcasting Software** - here are defined several frameworks/softwares that are used for audio/video encoding and broadcasting;

- **Field Contributions** - some investigation has been done in this field, mainly in IPTV. In this section, this researched is presented while pointing out the differences to the proposed solution;

- **Existent Solutions for audio and video broadcast** - it will be presented a study of several commercial and open-source solutions, including a brief description of the solutions and a comparison between that solution and the proposed solution in this thesis.

2.1 Audio/Video Codecs and Containers

The first approach to this solution is to understand what are the audio & video available codecs and containers. Audio and video codecs are necessary in order to compress the raw data, while the containers include both or separated audio and video data. The term codec stands for a blending of the words “compressor-decompressor” and denotes a piece of software capable of encoding and/or decoding a digital data stream or signal. With such a codec, the computer system recognizes the adopted multimedia format and allows the playback of the video file (=decode) or to change to another video format (=(en)code).

The codecs are separated in two groups, the lossy codecs and the lossless codecs. The lossless codecs are typically used for archiving data in a compressed form while retaining all of the information present in the original stream, meaning that the storage size is not a concern. In the other hand, the lossy codecs reduce quality by some amount in order to achieve compression. Often, this type of compression is virtually indistinguishable from the original uncompressed sound or images, depending on the encoding parameters.

The containers may include both audio and video data, however the container format depends on the audio and video encoding, meaning that each container specifies the acceptable formats.
2.1 Audio/Video Codecs and Containers

2.1.1 Audio Codecs

The presented audio codecs are grouped in open-source and proprietary codecs. The developed solution will only take to account the open-source codecs, due to the established requisites. Nevertheless, some proprietary formats are also available and are described.

Open-source codecs

Vorbis – is a general purpose perceptual audio CODEC intended to allow maximum encoder flexibility, thus allowing it to scale competitively over an exceptionally wide range of bitrates. At the high quality/bitrate end of the scale (CD or DAT rate stereo, 16/24 bits) it is in the same league as MPEG-2 and MPC. Similarly, the 1.0 encoder can encode high-quality CD and DAT rate stereo at below 48kbps without resampling to a lower rate. Vorbis is also intended for lower and higher sample rates (from 8kHz telephony to 192kHz digital masters) and a range of channel representations (e.g., monaural, polyphonic, stereo, 5.1).

MPEG2 - Audio, AAC – is a standardized, lossy compression and encoding scheme for digital audio. Designed to be the successor of the MP3 format, AAC generally achieves better sound quality than MP3 at similar bit rates. AAC has been standardized by ISO and IEC, as part of the MPEG-2 and MPEG-4 specifications, ISO/IEC 13818-7:2006. AAC is adopted in digital radio standards like DAB+ and Digital Radio Mondiale, as well as mobile television standards (e.g., DVB-H).

Proprietary codecs

MPEG-1 Audio Layer III, MP3 – is a standard that covers audio, ISO/IEC-11172-3, and a patented digital audio encoding format using a form of lossy data compression. The lossy compression algorithm is designed to greatly reduce the amount of data required to represent the audio recording and still sound like a faithful reproduction of the original uncompressed audio for most listeners. The compression works by reducing accuracy of certain parts of sound that are considered to be beyond the auditory resolution ability of most people. This method is commonly referred to as perceptual coding, meaning that it uses psychoacoustic models to discard or reduce precision of components less audible to human hearing, and then records the remaining information in an efficient manner.

2.1.2 Video Codecs

The video codecs seek to represent a fundamentally analog data in a digital format. Because of the design of analog video signals, which represent luma and color information separately, a common first step in image compression in codec design is to represent and store the image in a YCbCr color space. The conversion to YCbCr provides two benefits:

1. It improves compressibility by providing decorrelation of the color signals; and
2. Separates the luma signal, which is perceptually much more important, from the chroma signal, which is less perceptually important and which can be represented at lower resolution to achieve more efficient data compression.
2. Background and Related Work

All the codecs presented below are used to compress the video data, meaning that they are all lossy codecs.

Open-source codecs

**MPEG-2 Visual**[10] – is a standard for “the generic coding of moving pictures and associated audio information”. It describes a combination of lossy video compression methods which permits the storage and transmission of movies using currently available storage media (e.g. DVD) and transmission bandwidth.

**MPEG-4 Part 2**[11] – is a video compression technology developed by MPEG. It belongs to the MPEG-4 ISO/IEC standards. It is based in the discrete cosine transform, similarly to previous standards such as MPEG-1 and MPEG-2. Several popular containers including DivX and Xvid support this standard. MPEG-4 Part 2 is a bit more robust than its predecessor, MPEG-2.

**MPEG-4 Part10/H.264/MPEG-4 AVC**[9] – is the ultimate video standard used in Blu-Ray DVD and has the peculiarity of requiring lower bit-rates in comparison with its predecessors. In some cases, one-third less bits are required to maintain the same quality.

**VP8**[81][82] – is an open video compression format created by On2 Technologies, bought by Google. VP8 is implemented by libvpx, which is the only software library capable of encoding VP8 video streams. VP8 is Google’s default video codec and the competitor of H.264.

**Theora**[58] – is a free lossy video compression format. It is developed by the Xiph.Org Foundation and distributed without licensing fees alongside their other free and open media projects, including the Vorbis audio format and the Ogg container. The libtheora is a reference implementation of the Theora video compression format being developed by the Xiph.Org Foundation. Theora is derived from the proprietary VP3 codec, released into the public domain by On2 Technologies. It is broadly comparable in design and bitrate efficiency to MPEG-4 Part 2.

2.1.3 Containers

The container file is used to identify and interleave different data types. Simpler container formats can contain different types of audio formats, while more advanced container formats can support multiple audio and video streams, subtitles, chapter-information, and meta-data (tags) — along with the synchronization information needed to play back the various streams together. In most cases, the file header, most of the metadata and the synchro chunks are specified by the container format.

**Matroska**[89] – is an open standard free container format, a file format that can hold an unlimited number of video, audio, picture or subtitle tracks in one file. Matroska is intended to serve as a universal format for storing common multimedia content. It is similar in concept to other containers like AVI, MP4 or ASF, but is entirely open in specification, with implementations consisting mostly of open source software. Matroska file types are .MKV for video (with subtitles and audio), .MK3D for stereoscopic video, .MKA for audio-only files and .MKS for subtitles only.
WebM – is an audio-video format designed to provide royalty-free, open video compression for use with HTML5 video. The project’s development is sponsored by Google Inc. A WebM file consists of VP8 video and Vorbis audio streams, in a container based on a profile of Matroska.

Audio Video Interleaved, AVI – is a multimedia container format introduced by Microsoft as part of its Video for Windows technology. AVI files can contain both audio and video data in a file container that allows synchronous audio-with-video playback.

QuickTime – is Apple’s own container format. QuickTime sometimes gets criticized because codec support (both audio and video) is limited to whatever Apple supports. Although it is true, QuickTime supports a large array of codecs for audio and video. Apple is a strong proponent of H.264, so QuickTime files can contain H.264-encoded video.

Advanced Systems Format – ASF is a Microsoft-based container format. There are several file extensions for ASF files, including .asf, .wma, and .wmv. Note that a file with a .wmv extension is probably compressed with Microsoft’s WMV (Windows Media Video) codec, but the file itself is an ASF container file.

MP4 – is a container format developed by the Motion Pictures Expert Group, and technically known as MPEG-4 Part 14. Video inside MP4 files are encoded with H.264, while audio is usually encoded with AAC, but other audio standards can also be used.

Flash – Adobe’s own container format is Flash, which supports a variety of codecs. Flash video is encoded with H.264 video and AAC audio codecs.

OGG – is a multimedia container format, and the native file and stream format for the Xiph.org multimedia codecs. As with all Xiph.org technology is it an open format free for anyone to use. Ogg is a stream oriented container, meaning it can be written and read in one pass, making it a natural fit for Internet streaming and use in processing pipelines. This stream orientation is the major design difference over other file-based container formats.

Waveform Audio File Format, WAV – is a Microsoft and IBM audio file format standard for storing an audio bitstream. It is the main format used on Windows systems for raw and typically uncompressed audio. The usual bitstream encoding is the linear pulse-code modulation (LPCM) format.

Windows Media Audio, WMA – is an audio data compression technology developed by Microsoft. WMA consists of four distinct codecs: lossy WMA, was conceived as a competitor to the popular MP3 and RealAudio codecs; WMA Pro, a newer and more advanced codec that supports multichannel and high resolution audio; WMA Lossless, compresses audio data without loss of audio fidelity; and WMA Voice, targeted at voice content and applies compression using a range of low bit rates.

2.2 Encoding, broadcasting and Web Development Software

2.2.1 Encoding Software

As described in the previous section, there are several audio/video formats available. Encoding software is used to convert audio and/or video from one format to another. Below are
2. Background and Related Work

presented the most used open-source tools to encode audio and video.

**FFmpeg**[37] – is a free software project that produces libraries and programs for handling multimedia data. The most notable parts of FFmpeg are:

- **libavcodec** is a library containing all the FFmpeg audio/video encoders and decoders.
- **libavformat** is a library containing demuxers and muxers for audio/video container formats.
- **libswscale** is a library containing video image scaling and colorspace/pixelformat conversion.
- **libavfilter** is the substitute for vhook which allows the video/audio to be modified or examined between the decoder and the encoder.
- **libswresample** is a library containing audio resampling routines.

**Mencoder**[44] – is a companion program to the MPlayer media player that can be used to encode or transform any audio or video stream that MPlayer can read. It is capable of encoding audio and video into several formats and includes several methods to enhance or modify data (e.g., cropping, scaling, rotating, changing the aspect ratio of the video’s pixels, colorspace conversion).

2.2.2 Broadcasting Software

The concept of streaming media is usually used to denote certain multimedia contents that may be constantly received by an end-user, while being delivered by a streaming provider, by using a given telecommunication network.

A streamed media can be distributed either by Live or On Demand. While live streaming sends the information straight to the computer or device without saving the file to a hard disk, on demand streaming is provided by firstly saving the file to a hard disk and then playing the obtained file from such storage location. Moreover, while on demand streams are often preserved on hard disks or servers for extended amounts of time, live streams are usually only available at a single time instant (e.g. during a football game).

2.2.2.A Streaming Methods

As such, when creating streaming multimedia, there are two things that need to be considered: the multimedia file format (presented in the previous section) and the streaming method.

As referred, there are two ways to view multimedia contents on the Internet:

- On Demand downloading;
- Live streaming.

**On Demand downloading**

On Demand downloading consists in the download of the entire file into the receiver’s computer for later viewing. This method has some advantages (such as quicker access to different parts of the file) but has the big disadvantage of having to wait for the whole file to be downloaded before
any of it can be viewed. If the file is quite small this may not be too much of an inconvenience, but for large files and long presentations it can be very off-putting.

There are some limitations to bear in mind regarding this type of streaming:

- It is a good option for websites with modest traffic, i.e. less than about a dozen people viewing at the same time. For heavier traffic a more serious streaming solution should be considered.

- Live video cannot be streamed, since this method only works with complete files stored on the server.

- The end user's connection speed cannot be automatically detected. If different versions for different speeds should be created, a separate file for each speed will be required.

- It is not as efficient as other methods and will incur a heavier server load.

**Live Streaming**

In contrast to On Demand downloading, Live streaming media works differently — the end user can start watching the file almost as soon as it begins downloading. In effect, the file is sent to the user in a (more or less) constant stream, and the user watches it as it arrives. The obvious advantage with this method is that no waiting is involved. Live streaming media has additional advantages such as being able to broadcast live events (sometimes referred to as a webcast or netcast). Nevertheless, true live multimedia streaming usually requires a specialized streaming server, to implement the proper delivery of data.

**Progressive Downloading**

There is also a hybrid method known as progressive download. In this method, the media content is downloaded but begins playing as soon as a portion of the file has been received. This simulates true live streaming, but does not have all the advantages.

**2.2.2.B Streaming Protocols**

Streaming audio and video, among other data (e.g., Electronic program guides (EPG)) over the Internet is associated to the IPTV [98]. IPTV is simply a way to deliver traditional broadcast channels to consumers over an IP network in place of terrestrial broadcast and satellite services. Even though IP is used, the public Internet actually does not play much of a role. In fact, IPTV services are almost exclusively delivered over private IP networks. At the viewer’s home, a set-top box is installed to take the incoming IPTV feed and convert it into standard video signals that can be fed to a consumer television.

Some of the existing protocols used to stream IPTV data are:

**RTSP - Real Time Streaming Protocol** [98] — developed by the IETF, is a protocol for use in streaming media systems which allows a client to remotely control a streaming media server, issuing VCR-like commands such as “play” and “pause”, and allowing time-based access to files on a server. RTSP servers use RTP in conjunction with the RTP Control Protocol (RTCP) as the transport protocol for the actual audio/video data and the Session Initiation Protocol, SIP, to set up, modify and terminate an RTP-based multimedia session.
2. Background and Related Work

**RTMP - Real Time Messaging Protocol**[64] – is a proprietary protocol developed by Adobe Systems (formerly developed by Macromedia) that is primarily used with Macromedia Flash Media Server to stream audio and video over the Internet to the Adobe Flash Player client.

### 2.2.2.C Open-source Streaming solutions

A streaming media server is a specialized application which runs on a given Internet server, in order to provide “true Live streaming”, in contrast to “On Demand downloading”, which only simulates live streaming. True streaming, supported on streaming servers, may offer several advantages, such as:

- The ability to handle much larger traffic loads;
- The ability to detect users’ connection speeds and supply appropriate files automatically;
- The ability to broadcast live events.

Several open source software frameworks are currently available to implement streaming server solutions. Some of them are:

**GStreamer Multimedia Framework, GST**[41] – is a pipeline-based multimedia framework written in the C programming language with the type system based on GObject. GST allows a programmer to create a variety of media-handling components, including simple audio playback, audio and video playback, recording, streaming and editing. The pipeline design serves as a base to create many types of multimedia applications such as video editors, streaming media broadcasters, and media players. Designed to be cross-platform, it is known to work on Linux (x86, PowerPC and ARM), Solaris (Intel and SPARC) and OpenSolaris, FreeBSD, OpenBSD, NetBSD, Mac OS X, Microsoft Windows and OS/400. GST has bindings for programming-languages like Python, Vala, C++, Perl, GNU Guile and Ruby. GST is licensed under the GNU Lesser General Public License.

**Flumotion Streaming Server**[24] – is based on the multimedia framework GStreamer and Twisted, written in Python. It was founded in 2006 by a group of open source developers and multimedia experts, Flumotion Services, S.A., and it is intended for broadcasters and companies to stream live and on demand content in all the leading formats from a single server or depending in the number of users it may scale to handle more viewers. This end-to-end and yet modular solution includes signal acquisition, encoding, multi-format transcoding and streaming of contents.

**FFserver**[7] – is an HTTP and RTSP multimedia streaming server for live broadcasts for both audio and video and a part of the FFmpeg. It supports several live feeds, streaming from files and time shifting on live feeds.

**Video LAN VLC**[52] – is a free and open source multimedia framework, developed by the VideoLAN project, which integrates a portable multimedia player, encoder, and streamer applications. It supports many audio and video codecs and file formats, as well as DVDs, VCDs, and various streaming protocols. It is able to stream over networks and to transcode multimedia files and save them into various formats.
2.3 Field Contributions

In the beginning of the nineties, there was an explosion in the creation and demand of several types of devices. It is the case of a Portable Multimedia Device described in [97]. In this work, the main idea was to create a device which would allow ubiquitous access to data and communications via a specialized, wireless multimedia terminal. The proposed solution is focused in providing remote access to data (audio and video) and communications, using day-to-day devices, such as common computer, laptops, tablets and smartphones.

As mentioned before, a new emergent area is the IPTV with several solutions being developed on a daily basis. IPTV is a convergence of core technologies in communications. The main difference to standard television broadcast is the possibility of bidirectional communication and multicast, offering the possibility of interactivity, with a large number of services that can be offered to the customer. The IPTV is an established solution for several commercial products. Thus, several work has been done in this field, namely the Personal TV framework presented in [65], where the main goal is the design of a Framework for Personal TV for personalized services over IP. The proposed solution differs from the Personal TV Framework [65] in several aspects. The proposed solution is:

- Implemented based on existent open-source solutions;
- Intended to be easily modifiable;
- Aggregates several multimedia functionalities, such as video-call, recording content;
- Able to serve the user with several different multimedia video formats (currently, the streamed video is done in WebM format, but it is possible to download the recorded content in different video formats, by requesting the platform to re-encode the content).

Another example, of an IPTV base system, is Play - “Terminal IPTV para Visualização de Sessões de Colaboração Multimédia” [100]. This platform was intended to give to the users the possibility, in their own home and without the installation of additional equipment, to participate in sessions of communication and collaboration with other users, connected though the TV or other terminals (e.g., computer, telephone, smartphone). The Play terminal is expected to allow the viewing of each collaboration session and additionally implement as many functionalities as possible like chat, video conferencing, slideshow, sharing and editing documents. This is also the purpose of this work, being the difference that Play is intended to be incorporated in a commercial solution, MEO, and the solution here in proposed is all about reusing and incorporating existing open-source solutions into a free extensible framework.

Several solutions have been researched through time, but all are intended to be somehow incorporated in commercial solutions given the nature of the functionalities involved in this kind of solutions. The next sections give an overview of several existent solutions.

2.4 Existent Solutions for audio and video broadcast

Several tools to implement the features previously presented exist independently, but with no connectivity between them. The main differences between the proposed platform and the tools
already developed, is that this framework integrates all the independent solutions into it and this solution is intended to be used remotely. Other differences are stated as follows:

- Some software is proprietary and, as so, has to be purchased and cannot be modified without incurring in a crime.

- Some software tools have a complex interface and are suitable only for users with some programming knowledge. In some cases, this is due to the fact that some software tools support many more features and configuration parameters than what is expected in an all-in-one multimedia solution.

- Some television applications cover only DVB and no analog support is provided.

- Most applications only work in specific world areas (e.g., USA).

- Some applications only support a limited set of devices.

In the following, a set of existing platforms is presented. It should be noted the existence of other small applications (e.g. other TV players, such as Xawtv). However, in comparison with the presented applications, they offer no extra feature.

2.4.1 Commercial software frameworks

**GoTV** [40]. GoTV is a proprietary and paid software tool that offers TV viewing to mobile-devices only. It has a wide platform support (Android, Samsung, Motorola, BlackBerry, iPhone) and only works in USA. It does not offer video-call service and no video recording feature is provided.

**Microsoft MediaRoom** [45]. This is the service currently offered by Microsoft to television and video providers. It is a proprietary and paid service, where the user cannot customize any feature; only the service provider can modify it. Many providers use this software, such as the Portuguese MEO and Vodafone and lots of others worldwide [53]. The software does not offer the video-call feature and it is only for IPTV. It also works through a large set of devices: personal computer, mobile devices, TV’s and with Microsoft XBox360.

**GoogleTV** [39]. This is the Google TV service for Android systems. It is an all-in-one solution, developed by Google, and works only for some selected Sony televisions and Sony Set-Top boxes. The concept of this service is basically a computer inside your television or inside your Set-Top Box. It allows developers to add new features through the Android Market.

**NDS MediaHighway** [47]. This is a platform adopted worldwide by many Set-Top boxes. For example, it is used by the Portuguese Zon provider [55], among others. It is a similar platform to Microsoft MediaRoom, with the exception that it supports DVB (terrestrial, satellite and hybrid), while MediaRoom does not.

All of the above described commercial solutions for TV have similar functionalities. However, some support a great number of devices (even some unusual devices, such as Microsoft XBox360), and some are specialized in one kind of device (e.g. GoTV, mobile devices). All share the same idea to charge for the service. None of the mentioned commercial solutions offer support for video-conference, either as a supplement or with the normal service.
2.4.2 Free/open-source software frameworks

Linux TV [43]. It is a repository for several tools that offers a vast set of support for several kinds of TV Cards and broadcast methods. By using the Video for Linux driver (V4L) [51], it is possible to view TV from all kinds of DVB sources, but none for analog TV broadcast sources. The problem of this solution is that, for a regular user with no programming knowledge, it is hard to setup any of the proposed services.

Video Disk Recorder, VDR [50]. It is an open solution, for DVB only, with several options such as regular playback, recording and video edition. It is a great application if the user has DVB and some programming knowledge.

Kastor! TV, K!TV [42]. It is an open solution for MS Windows to view and record TV content from a video card. Users can develop new plug-ins for the application without restrictions.

MythTV [46]. MythTV is a free open-source software for digital video recording (DVR). It has a vast support and development team, where any user can modify/customize it with no fee. It supports several kinds of DVB sources as well as analog cable.

Linux TV as explained represents a framework with a set of tools that allow the visualization of the content acquired by the local TV card. Thus, this solution only works locally and if the users uses it remotely it will be a one user solution. Regarding the VDR, as said it requires some programming knowledge and it is restricted to DVB. The proposed solutions aims for the support of several inputs, not being restrict to one technology.

The other two applications, K!TV and MythTV, fail to meet the following proposed requirements:

- Require the installation of the proper software;
- Intended for local usage (e.g., viewing the stream acquired from the TV card);
- Restricted to the defined video formats;
- They are not accessible through other devices (e.g., mobilephones);
- The user interaction is done through the software interface (they are not web-based solutions).

2.5 Summary

Since the beginning of audio and video transmission there is a desire to build solutions/devices with several multimedia functionalities. Nowadays this is possible and offered by several commercial solutions. Given the current devices development, now able to connect to the Internet almost anywhere, the offer of commercial TV solutions increased, based on IPTV but it is not visible other solutions based in open-source solutions.

Besides the set of applications presented, there are many other TV playback applications and recorders, each with some minor differences, but always offering the same features and oriented to be used locally. Most of the existing solutions run under Linux distributions. Some do not even
2. Background and Related Work

have a graphical interface: in order to run the application is needed to type the appropriate commands, in a terminal, and this can be extremely hard for a user with no programming knowledge whose intent is to only to view TV or to record TV. Although all these solutions work with DVB, few of them give support to analog broadcast TV. Table 2.1 summarizes all the presented solutions, according to their limitations and functionalities.

Table 2.1: Comparison of the considered solutions.

<table>
<thead>
<tr>
<th>Features</th>
<th>Commercial Solutions</th>
<th>Open Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TV View</td>
<td>GoTV</td>
<td>Microsoft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MediaRoom</td>
</tr>
<tr>
<td>TV Recording</td>
<td>X</td>
<td>V</td>
</tr>
<tr>
<td>Video Conference</td>
<td>X</td>
<td>V</td>
</tr>
<tr>
<td>Supported Devices</td>
<td>Microsoft MediaRoom</td>
<td>V</td>
</tr>
<tr>
<td>Supported Devices</td>
<td>Google MediaHighway</td>
<td>X</td>
</tr>
<tr>
<td>Supported Devices</td>
<td>NDS MediaHighway</td>
<td>V</td>
</tr>
<tr>
<td>Supported Devices</td>
<td>TVR</td>
<td>V</td>
</tr>
<tr>
<td>Supported Devices</td>
<td>mytvTVMediaHighbury</td>
<td>V</td>
</tr>
<tr>
<td>Supported Devices</td>
<td>Linux TV</td>
<td>V</td>
</tr>
<tr>
<td>Supported Devices</td>
<td>VDR</td>
<td>V</td>
</tr>
<tr>
<td>Supported Devices</td>
<td>KITV</td>
<td>V</td>
</tr>
<tr>
<td>Supported Devices</td>
<td>mytv</td>
<td>V</td>
</tr>
<tr>
<td>Supported Devices</td>
<td>Proposed MM-Terminal</td>
<td>V</td>
</tr>
<tr>
<td>Supported Input</td>
<td>MM-Terminal</td>
<td>V</td>
</tr>
<tr>
<td>Supported Input</td>
<td>TV-View</td>
<td>V</td>
</tr>
<tr>
<td>Supported Input</td>
<td>Computer</td>
<td>V</td>
</tr>
<tr>
<td>Supported Input</td>
<td>Mobile Device</td>
<td>V</td>
</tr>
<tr>
<td>Supported Input</td>
<td>Analogical</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>DVB-T</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>DVB-C</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>DVB-S</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>DVB-H</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>IPTV</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>Worldwide</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>Localized</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>Customizable</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>Mobile OS*</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>MS Windows CE</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>Android</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>Set-Top Boxes**</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>Linux</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>MS Windows</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>BSD</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>Mac-OS</td>
<td>X</td>
</tr>
<tr>
<td>Supported Input</td>
<td>Linux</td>
<td>X</td>
</tr>
</tbody>
</table>

Legend

v = Yes
x = No

* Android OS, iOS, Symbian OS, Motorola OS, Samsung bada.
** Set-Top Boxes can run MS Windows CE or some light Linux distribution, anyhow in the official page there is no mention to supported OS.
## Multimedia Terminal Architecture

### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Signal Acquisition And Control</td>
<td>21</td>
</tr>
<tr>
<td>3.2 Encoding Engine</td>
<td>21</td>
</tr>
<tr>
<td>3.3 Video Recording Engine</td>
<td>22</td>
</tr>
<tr>
<td>3.4 Video Streaming Engine</td>
<td>23</td>
</tr>
<tr>
<td>3.5 Scheduler</td>
<td>24</td>
</tr>
<tr>
<td>3.6 Video Call Module</td>
<td>24</td>
</tr>
<tr>
<td>3.7 User interface</td>
<td>25</td>
</tr>
<tr>
<td>3.8 Database</td>
<td>25</td>
</tr>
<tr>
<td>3.9 Summary</td>
<td>27</td>
</tr>
</tbody>
</table>
3. Multimedia Terminal Architecture

This section presents the proposed architecture. The design of the architecture is based on the analysis of the functionalities that this kind of system should provide, namely it should be easy to manipulate, remove or add new features and hardware components. As an example, it should support a common set of multimedia peripheral devices, such as video cameras, AV capture cards, DVB receiver cards, video encoding cards or microphones. Furthermore, it should support the possibility of adding new devices.

The conceived architecture adopts a client-server model. The server is responsible for signal acquisition and management, in order to provide the set of features already enumerated, as well as the reproduction and recording of audio/video and video-call. The client application is responsible for the data presentation and the interface between the user and the application.

Fig. 3.1 illustrates the application in the form of a structured set of layers. In fact, it is well known that it is extremely hard to create an application based on a monolithic architecture: maintenance is extremely hard and one small change (e.g., in order to add a new feature) implies going through all the code to make the changes. The principles of a layered architecture are: (1) each layer is independent, and (2) adjacent layers communicate through a specific interface. The obvious advantages are the reduction of conceptual and development complexity, easy maintenance and feature addition and/or modification.

![Layered Architecture Diagram](image)

Figure 3.1: Server and Client Architecture of the Multimedia Terminal.

As it can be seen in Fig. 3.1, the two bottom layers correspond to the Hardware (HW) and Operating System (OS) layers. The HW layer represents all physical computer parts. It is in this first layer that the TV card for video/audio acquisition is connected, as well as the web-cam and microphone (for video-call), and other peripherals. The management of all HW components is the responsibility of the OS layer.

The third layer (the Application Layer) represents the application. As it can be observed, there is a first module, the Signal Acquisition And Control (SAAC), that provides the proper signal to the modules above. After the acquisition of the signal by the SAAC module, the audio and video signals are passed to the Encoding Engine. There, they are encoded according to the predefined profile, which is set by the Profiler Module, accordingly to the user definitions. The profile may be saved in the database. Afterwards, the encoded data is fed to the components
3.1 Signal Acquisition And Control

above, i.e. the Video Streaming Engine (VSE), the Video Recording Engine (VRE) and the Video Call Module (VCM). This layer is connected to a database, in order to provide security, user and recording data control and management.

The proposed architecture was conceived in order to simplify the addition of new features. As an example, suppose that a new signal source is required, such as DVD playback. This would require the manipulation of the SAAC module, in order to set a new source to feed the VSE. Instead of acquiring the signal from some component or from a local file in HDD the module would have to access the file in the local DVD drive.

In the top level it is presented the user interface, which provides the features implemented by the layer below. This is where the regular user interacts with the application.

3.1 Signal Acquisition And Control

The SAAC Module is of great relevance in the proposed system, since it is responsible for the signal acquisition and control. In other words, the video/audio signal acquired from multiple HW sources (e.g., TV card, surveillance camera, webcam and microphone, DVD, ...) providing information in a different way. However, the top modules should not need to know how the information is provided/encoded. Thus, the SAAC Module is responsible to provide a standardized mean for the upper modules to read the acquired information.

3.2 Encoding Engine

The Encoding Engine is composed by the Audio and Video Encoders. Their configuration options are defined by the Profiler. After acquiring the signal from the SAAC Module, this signal needs to be encoded into the requested format for subsequent transmission.

3.2.1 Audio Encoder & Video Encoder Modules

The Audio & Video Encoder Modules are used to compress/decompress the multimedia signals being acquired and transmitted. The compression is required to minimize the amount of data to be transferred, so that the user can experience a smooth audio and video transmission.

The Audio & Video Encoder Modules should be implemented separately in order to easily allow the integration of future audio or video codecs into the system.

3.2.2 Profiler

When dealing with recording and previewing it is important to have in mind that different users have different needs, and each need corresponds to three contradictory forces: encoding time, quality and stream size (in bits). One could easily record each program in the raw format, outputted by the TV tuner card. This would mean that the recording time would be equal to the time required by the acquisition, the quality would be equal to the one provided by the tuner card and the size would obviously be huge, due to the two other constrains. For example, a 45 minutes recording would require about 40 Gbytes of disk space for a raw YUV 4:2:0 [93] format. Even though storage is considerably cheap nowadays, this solution is still very expensive. Furthermore, it makes no sense to save that much detail into the record file since the human eye has proven limitations [102] that prevent the humans to perceive certain levels of detail. As a consequence,
it is necessary to study what are the most suitable recording/previewing profiles, having in mind those three restrictions presented above.

On one hand, there are the users who are video collectors/preservers/editors. For this kind of users both image and sound quality are of extreme importance, so the user must be aware that for achieving high quality he either needs to sacrifice the encoding time in order to compress the video as much as possible (thus obtaining good quality-size ratio), or he needs a large storage space to store it in raw format. For a user with some concern about quality but with no other intention other than playing the video once, and occasionally saving it for the future, the constrains are slightly different. Although, he will probably require a reasonably good quality, he will not probably care about the efficiency of the encoding. On the other hand, the user may have some concerns about the encoding time, since he may want to record another video at the same time or immediately after. Another type of user is the one who only wants to see the video but without so much concerns about quality (e.g., because he will see it in a mobile device or low resolution tablet device). This type of user thus worries about the file size, and may have concerns about the download time, or limited download traffic.

By summarizing the described situations, the three defined recording profiles will now be presented:

- **High Quality (HQ)** - for users who have a good Internet connection, no storage constrains, and do not mind waiting some more time in order to have the best quality. This can provide support for some video edition and video preservation, but increases the time to encode and obviously the final file size. The frame resolution corresponds to 4CIF, i.e., 704x576 pixels. This quality is also recommended for users with large displays. This profile can even be extended in order to support High Definition (HD), where the frame size would be changed to 720p (1280x720 pixels) or 1080i (1920x1080) pixels;

- **Medium Quality (MQ)** - intended for users with a good/average Internet connection, a limited storage and a desire for a medium video/audio quality. This is the common option for a standard user, good ratio between quality-size and an average encoding time. The frame size corresponds to CIF, i.e., 352x288 pixels of resolution;

- **Low Quality (LQ)** - targeted for users that have a lower bandwidth Internet connection, a limited download traffic and do not care so much for the video quality. They just want to be able to see the recording and then delete it. The frame size corresponds to QCIF, i.e., 176x144 pixels of resolution. This profile is also recommended for users with small displays (e.g., a mobile device);

### 3.3 Video Recording Engine

[VRE] is the unit responsible for recording audio/video data coming from the installed TV card. There are several recording options but the recording procedure is always the same. First, it is necessary to specify the input channel to record as well as the beginning and ending time. Afterwards, accordingly to the Scheduler status, the system needs to decide, if it is an acceptable recording or not (verify if there is some time conflict, i.e., simultaneous records in different channels with only one audio/video acquisition device). Finally, it tunes the required channel and starts the recording with the desired quality level.

The [VRE] component interacts with several other models as illustrated in Fig 3.2. One of such modules is the database. If the user wants to select the program that will be recorded by specifying its name, the first step is to request the database recording time and the user permissions to
3.4 Video Streaming Engine

The VSE component is responsible for streaming the captured audio/video data provided by the SAAC Module or for streaming any video recorded by the user that is presented in the server’s storage unit. It may also stream the web-camera data, when the video-call scenario is considered.

Considering the first scenario, where the user just wants to view a channel, the VSE has to communicate with several components before streaming the required data. Such procedure involves:

1. The system must validate the user’s login and user’s permission to view the selected channel;
2. The VSE communicates with the Scheduler, in order to determine if the channel can be played at that instant (the VRE may be recording and cannot display other channel);
3. The VSE reads the requests profile from the Profiler component;
4. The VSE communicates with the SAAC unit, acquires the signal and applies the selected profile to encode and stream the selected channel.

Viewing a recorded program is basically the same procedure. The only exception is that the signal read by the VSE is the recorded file and not the SAAC controller. Fig. 3.3(a) illustrates all the components involved in the data streaming, while Fig. 3.3(b) exemplifies the described procedure for both input options.
3. Multimedia Terminal Architecture

3.5 Scheduler

The Scheduler component manages the operations of the VSE and VRE and is responsible for scheduling the recording of any specific audio/video source. For example, consider the case where the system would have to acquire multiple video signals at the same time with only one TV card. This behavior is not allowed because it will create a system malfunction. This situation can occur if a user sets multiple recordings, at the same time, or because a second user tries to access the system while it is already in use. In order to prevent these undesired situations, a set of policies have to be defined:

**Intersection**: Recording the same show in the same channel. Different users should be able to record different parts from the same TV show. For example: User 1, wants to record only the first half of the show, User 2 wants to record the both parts, and User 3 only wants the second half. The Scheduler Module will record the entire show encode it and in the end split the show according to each user needs.

**Channel switch**: Recording in progress or different TV channel request. With one TV card only one operation can be executed at the same time. This means that if some User 1 is already using the Multimedia Terminal (MMT) only he can change channel. Other possible situation is: the MMT is recording, only the user that request the recording can stop it and in the meanwhile changing channel is lock. This situation is different if the MMT possesses two or more TV capture cards. In that case other policies need to be defined.

3.6 Video Call Module

Video call applications are currently used by many people around the world. Families that are separated by thousands of miles can chat without extra costs.

The advantages of offering a Video-Call service through this multimedia terminal is: (1) the user already has an Internet connection that can be used for this purpose; (2) most laptops sold...
today already have an incorporated microphone and web-camera, this guarantees the sound and video acquisition; (3) the user obviously has a display unit. With all this facilities already available, it seems natural to add this service to the list of features offered by the conceived multimedia terminal.

To start using this service, the user first needs to authenticate himself in the system with his username and password. This is necessary to guarantee privacy and to provide each user with its own contact list. After correct authentication, the user selects an existent contact (or introduces one new) to start the video-call. At the other end, the user will receive an alert that another user is calling and has the option to accept or decline the incoming call.

The information flow is presented in Fig 3.4 with the involved components of each layer.

3.7 User interface

The User interface (UI) implements the means for the user interaction. It is composed by multiple web-pages with a simple and intuitive design, accessible through an Internet browser. Alternatively, it can also be provided through a simple ssh connection to the server. It is important to refer that the UI should be independent from the host OS. This allows the user to use whatever OS desired. This way, multi-platform support is provided (in order to make the application accessible to smart-phones and other).

Advanced users can also perform some tasks through an SSH connection to the server, as long as their OS supports this functionality. Through SSH, they can manage the recording of any program in the same way as they would do in the web-interface. In Fig. 3.5 some of the most important interface windows are represented as a sketch.

3.8 Database

The use of a database is necessary to keep track of several data. As already said, this application can be used by several different users. Furthermore, in the video-call service it is expected that different users may have different friends and want privacy about their contacts. The same
3. Multimedia Terminal Architecture

User common Interfaces

(a) Multimedia Terminal HomePage authentication.

(b) Multimedia Terminal HomePage. In the right side there is a quick access panel for channels. In the left side are the possible features, e.g., Menu.

(c) TV Interface.

(d) Recording Interface.

(e) Video-Call Interface.

(f) Example of one of the Multimedia Terminal Properties Interface.

Figure 3.5: Several user-interfaces for the most common operations.
can be said for the user’s information. As such, it can be distinguished different usages for the database, namely:

- Track scheduled programs to record, for the scheduler component;
- Record each user information, such as: name and password, friends contacts for video-call;
- Track, for each channel, their shows and starting times, in order to provide an easier interface to the user, by recording a show and channel by its name;
- Recorded programs and channels over time, for any kind of content analysis or to offer some kind of feature (e.g. most viewed channel, top recorded shows, …);
- Define shared properties for recorded data (e.g. if an older user wants to record some show non suitable for younger users, he may define the users he wants to share this show);
- Provide features like parental-control for time of usage and permitted channels.

In summary, the database may be accessed by most components in the Application Layer, since it collects important information that is required to ensure a proper management of the terminal.

### 3.9 Summary

The proposed architecture is based on existent single purpose open-source software tools and was defined in order to make it easy to manipulate, remove or add new features and hardware components. The core functionalities are:

- **Video Streaming**, allowing real-time reproduction of audio/video acquired from different sources (e.g., TV cards, video cameras, surveillance cameras). The media is constantly received and displayed to the end-user through an active Internet connection.
- **Video Recording**, providing the ability to remotely manage the recording of any source (e.g. a TV show or program) in a storage medium;
- **Video-call**, considering that most TV providers also offer their customers an Internet connection, it can be used together with a web-camera and a microphone, to implement a video-call service.

The conceived architecture adopts a client-server model. The server is responsible for signal acquisition and management of the available multimedia sources (e.g., cable TV, terrestrial TV, web-camera, etc.), as well as the reproduction and recording of the audio/video signals. The client application is responsible for the data presentation and the user interface.

Fig. 3.1 illustrates the architecture in the form of a structured set of layers. This structure has the advantage of reducing the conceptual and development complexity, allows easy maintenance, and permits feature addition and/or modification.

Common to both sides, server and client, is the presentation layer. The user interface is defined in this layer and is accessible both locally and remotely. Through the user interface it should be possible to login as a normal user or as an administrator. The common user uses the interface to view and/or schedule recordings of TV shows or previously recorded content and to do a video-call. The administrator interface allows administration tasks, such as retrieving passwords, disable or enable user accounts or even channels.

The server is composed of six main modules:
3. Multimedia Terminal Architecture

- **Signal Acquisition And Control (SAAC)**, responsible for the signal acquisition and channel change;

- **Encoding Engine**, which is responsible for channel change and for encoding audio and video data with the selected profile, i.e. different encoding parameters;

- **Video Streaming Engine (VSE)**, which streams the encoded video through the Internet connection;

- **Scheduler**, responsible for managing multimedia recordings;

- **Video Recording Engine (VRE)**, which records the video into the local hard drive, for posterior visualization, download or re-encoding;

- **Video Call Module (VCM)**, which streams the audio/video acquired from the web-cam and microphone.

In the client side there are two main modules:

- **Browser and required plug-ins**, in order to correctly display the streamed and recorded video;

- **Video Call Module (VCM)**, to acquire the local video+audio and stream it to the corresponding recipient.
4
Multimedia Terminal Implementation

Contents

4.1 Introduction ........................................... 30
4.2 User Interface ......................................... 31
4.3 Streaming ............................................. 41
4.4 Recording ............................................. 51
4.5 Video-Call ............................................. 53
4.6 Summary ............................................... 53
4. Multimedia Terminal Implementation

The Implementation chapter describes how the previously conceived architecture was developed in order to originate this new multimedia terminal framework. The chapter starts with a brief introduction stating the principal characteristics of the used software and hardware, then each module that composes this solution is explained in detail.

4.1 Introduction

The developed prototype is based on existent open-source applications released under the General Public Licence (GPL) [57]. Since the license allows for code changes, the communities involved in these projects, are always improving them.

The usage of open-source software under the GPL represents one of the requisites of this work. This has to do with the fact that having a community contributing with support for the used software ensures future support for upcoming systems and hardware.

The described architecture is implemented by several different software solutions, see Figure 4.1.

Figure 4.1: Mapping between the designed architecture and software used.

To implement the UI it was used the Ruby on Rails (RoR) framework and the utilized database was SQLite3 [20]. Both solutions work perfectly together due to RoR SQLite support.

The signal acquisition, encoding engine, streaming and recording engines as well as the video-call module are all implemented through the Flumotion Streaming Server, while the signal control
(i.e., channel switching) is implemented by V4L2 framework \[51\]. To manage the recordings schedule it is used the Unix Cron \[31\] scheduler.

The following sections describe in detail the implementation of each module and the motives that lead to the utilization of the described software. This chapter is organized as follows:

- Explanation of how the UI is organized and implemented;
- Detailed implementation of the streaming server with all the tasks associated, audio/video acquisition and management, streaming, recording and recording management (schedule);
- Video-call module implementation.

### 4.2 User Interface

One of the main concerns, while developing this solution, was the development of a solution that would cover most of the devices and existent systems. The UI should be accessible through a client browser, regardless of the OS used, plus a plug-in to allow viewing of the streaming content.

The UI was implemented using the RoR Framework \[49\] \[75\]. RoR is an open-source web application development framework, that allows agile development methodologies. The programming language is Ruby and it is highly supported and useful for daily-tasks.

There are several others web application frameworks that would also serve for this purpose, frameworks based on Java (e.g., Java Stripes \[63\]), nevertheless RoR presented some solid reasons that stood out along with the desire to learning a new language. The reasons that lead to the use of RoR were:

- Ruby programming language, is a object-oriented language, easy readable and with an unsurprising syntax and behaviour;
- The Don’t Repeat Yourself (DRY) principle, leads to concise and consistent code that is easy to maintain;
- Convention over configuration principle, using and understanding the defaults speeds development, less code to maintain, and it follows the best programming practices;
- High support for integrating with other programming languages, e.g., Ajax, PHP, JavaScript;
- Model-View-Controller (MVC) architecture pattern to organize application programming;
- Tools that make common development tasks easier “out of the box”, e.g., scaffolding that can automatically construct some of the models and views needed for a website;
- Includes WEBrick, which is a simple Ruby web server and it is utilized to launch the developed application;
- With Rake, stands for Ruby Make, it is possible to specify task that can be called either inside the application or from a console which is very useful for management purposes;
- It has several plug-ins designated as gems that can be freely used and modified;
- ActiveRecord management, which is extremely useful for database driven applications, in concrete the management of the multimedia content.
4. Multimedia Terminal Implementation

4.2.1 The Ruby on Rails Framework

RoR adopts the MVC pattern that modulates the development of a web application. A model represents the information (data) of the application and the rules to manipulate that data. In the case of Rails, models are primarily used for managing the rules of interaction with a corresponding database table. In most cases, one table in the database will correspond to one model in the application. The views represent the user interface of your application. In Rails, views are often HTML files with embedded Ruby code that perform tasks related solely to the presentation of the data. Views handle the job of providing data to the web browser or other tool that are used to make requests from the application. Controllers are responsible for processing the incoming requests from the web browser, interrogating the models for data, and passing that data on to the views for presentation. In this way controllers are the bridge between the models and the views.

![Model-View-Controller interaction diagram.](image)

The procedure triggered by an incoming request from the browser is as follows (see Figure 4.2):

- The incoming request is received by the controller which decides either to send the requested view or to invoke the model for further process;
- If the request is a simple redirect request with no data involved, then the view is returned to the browser;
- If there is data processing involved in the request, the controller gets the data from the model, invokes the view that processes the data for presentation and then returns it to the browser.

When a new project is generated in RoR it builds the entire project structure and it is important to understand that structure in order to correctly follow Rails conventions and best practices. Table 4.1 summarizes the project structure along with a brief explanation of each file/folder.

4.2.2 The Models, Controllers and Views

According to the MVC pattern, some models along with several controllers and views had to be created in order to assemble a solution that would aggregate all the system requirements: real-time streaming of a channel, the possibility to change the channel and the broadcast quality; management of recordings, recorded videos, user information, channels; and video-call functionality. Therefore, to allow the management of recordings, videos and channels these three objects generate three models:


4.2 User Interface

Table 4.1: Rails default project structure and definition.

<table>
<thead>
<tr>
<th>File/Folder</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gemfile</td>
<td>This file allows the specification of gem dependencies for the application.</td>
</tr>
<tr>
<td>README</td>
<td>This file should include the instruction manual for the developed application.</td>
</tr>
<tr>
<td>Rakefile</td>
<td>This file contains batch jobs that can be ran from the terminal.</td>
</tr>
<tr>
<td>app/</td>
<td>Contains the controllers, models, and views of the application.</td>
</tr>
<tr>
<td>config/</td>
<td>Configuration of the application's runtime rules, routes, database, ...</td>
</tr>
<tr>
<td>config.ru</td>
<td>Rack configuration for Rack based servers used to start the application.</td>
</tr>
<tr>
<td>db/</td>
<td>Shows the database schema and the database migrations.</td>
</tr>
<tr>
<td>doc/</td>
<td>In-depth documentation of the application.</td>
</tr>
<tr>
<td>lib/</td>
<td>Extended modules for the application.</td>
</tr>
<tr>
<td>log/</td>
<td>Application log files.</td>
</tr>
<tr>
<td>public/</td>
<td>The only folder seen to the world as-is. Here are the public images, javascript, stylesheets (CSS), and other static files.</td>
</tr>
<tr>
<td>script/</td>
<td>Contains the Rails scripts to starts the application.</td>
</tr>
<tr>
<td>test/</td>
<td>Unit and other tests.</td>
</tr>
<tr>
<td>tmp/</td>
<td>Temporary files.</td>
</tr>
<tr>
<td>vendor/</td>
<td>Intended for third-party code, e.g., Ruby Gems, the Rails source code and plugins containing additional functionalities.</td>
</tr>
</tbody>
</table>

- Channel model - holds the information related to channel management: channel name, code, logo image, visible and timestamps with the creation and modified date;

- Recording model - for the management of scheduled recordings. It contains the information regarding the user that scheduled that recording, the start and stop date and time, the channel and quality to record and finally the recording name;

- Video model - holds the recorded videos information: the video owner, video name, creation and modification date;

Also for users management purposes there was the need to define:

- User model - holds the normal user information;

- Admin model - for the management of users and channels;

The relation between the described models is: the user, admin and channel models are independent, there is no relation between them. For the recording and video models, each user can have several recordings and videos, while a recording and a video belongs to a user. In Relational Database Language (RDL) [66] this is translated to: the user has many recordings and videos while a record and a video belongs to one user, specifically it is a one to many association.

Regarding the controllers, for each controller there is a folder named after it where each file corresponds to an action defined in that controller. By default each controller should have an index action corresponding to the index.html.erb file, this is not mandatory but it is a Rails convention.

Most of the programming is done in the controllers. The information management task is done through a Create, Read, Update, Delete (CRUD) approach is adopted, which follows Rails conventions. Table 4.2 resumes the mapping from the CRUD to the actions that must be implemented. Each CRUD operation is implemented as a two action process:

- Create first action is new which is responsible for displaying the new record form to the user, while the other action is create which processes the new record and if there are no errors it is saved.
4. Multimedia Terminal Implementation

Table 4.2: Create, Read, Update, Delete associated actions and concepts.

<table>
<thead>
<tr>
<th>CRUD</th>
<th>Action</th>
<th>Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE</td>
<td>new</td>
<td>Display new record form.</td>
</tr>
<tr>
<td></td>
<td>create</td>
<td>Processes the new record form.</td>
</tr>
<tr>
<td>READ</td>
<td>list</td>
<td>List records.</td>
</tr>
<tr>
<td></td>
<td>show</td>
<td>Display a single record.</td>
</tr>
<tr>
<td>UPDATE</td>
<td>edit</td>
<td>Display edit record form.</td>
</tr>
<tr>
<td></td>
<td>update</td>
<td>Processes edit record form.</td>
</tr>
<tr>
<td>DELETE</td>
<td>delete</td>
<td>Display delete record form.</td>
</tr>
<tr>
<td></td>
<td>destroy</td>
<td>Processes delete record form.</td>
</tr>
</tbody>
</table>

- The Read operation first action is list which lists all the records in the database and show action shows the information for a single record.

- Update first action, edit, displays the record while the action update processes the edited record and saves it.

- Delete could be done in a single action, but to offer the user to give some thought about his action, this action is implemented in a two step process also. So the delete action shows the selected record to delete and the destroy removes record permanently.

The next figure, Figure 4.3, presents the project structure and the following sections describes them in detail.

![Figure 4.3: Multimedia Terminal MVC.](image)

4.2.2. A Users and Admin authentication

Ruby has several gems to implement recurrent tasks in a simple and fast manner. It is the case of the authentication task. To implement the authentication feature it was used the Devise gem. Devise is a flexible authentication solution for Rails based on Warden, it implements the full MVC for authentication and it’s modular concept allows the usage of only the needed modules. The decision to use Devise over other authentication gems was due to the simplicity of configuration, management and for the features provided. Although some of the modules are not used in the current implementation, Device as the following modules:
4.2 User Interface

- **Database Authenticatable**: encrypts and stores a password in the database to validate the authenticity of a user while signing in.

- **Token Authenticatable**: signs in a user based on an authentication token. The token can be given both through query string or HTTP basic authentication.

- **Confirmable**: sends emails with confirmation instructions and verifies whether an account is already confirmed during sign in.

- **Recoverable**: resets the user password and sends reset instructions.

- **Registerable**: handles signing up users through a registration process, also allowing them to edit and destroy their account.

- **Rememberable**: manages generating and clearing a token for remembering the user from a saved cookie.

- **Trackable**: tracks sign in count, timestamps and IP address.

- **Timeoutable**: expires sessions that have no activity in a specified period of time.

- **Validatable**: provides validations of email and password. It is an optional feature and it may be customized.

- **Lockable**: locks an account after a specified number of failed sign-in attempts.

- **Encryptable**: adds support of other authentication mechanisms besides the built-in Bcrypt.

The dependency of Devise is registered in the Gemfile in order to be usable in the project. To set-up the authentication and create the user and administrator role, the following commands were used in the command line at the project directory:

1. `$bundle install` - checks the Gemfile for dependencies, downloads them and installs;
2. `$rails generate devise_install` - installs devise into the project;
3. `$rails generate devise User` - creates the regular user role;
4. `$rails generate devise Admin` - creates the administrator role;
5. `$rake db:migrate` - for each role it creates a file in `db/migrate` folder containing the fields for each role. The `db:migrate` creates the database with the tables, representing the model, and the fields, representing the attributes of the model.
6. `$rails generate devise:views` - generates all the devise views, `/app/views/devise`, allowing customization.

The result of adding the authentication process is illustrated in Figure 4.4. This process created the user and admin models, all the views associated to the login, user management, logout, registration, are available for customization at the views.

The current implementation of devise authentication is done through HTTP. This authentication method should be enhanced through the utilization of a secure communication, SSL. This known issue is described in the Future Work chapter.
4. Multimedia Terminal Implementation

4.2.2.B Home controller and associated views

The home controller is responsible for deciding to which controller the logged user should be redirected to. If the user logs as a normal user, he is redirected to the mosaic controller, else the user is an administrator and the home controller redirects him to the administrator controller.

The home view is the first view invoked when a new user accesses the terminal. This configuration is enforced by the command `root :to => 'home#index'`, being the root and all other paths defined at `/config/routes.rb`, see Table 4.1.

4.2.2.C Administration controller and associated views

All controllers with data manipulation are implemented following the CRUD convention, and the administration controller is no exception as it manages the users and channels information.

There are five views associated to the CRUD operations:

- `new_channel.html.erb` - blank form to create a new channel;
- `list_channels.html.erb` - list all the channels in the system;
- `show_channel.html.erb` - displays the channel information;
- `edit_channel.html.erb` - shows a form with the channel information allowing the user to modify it;
- `delete_channel.html.erb` - shows the channel information and allows the user to delete that channel.

For each of these views there is an associated action in the controller. The new channel view presents the blank form to create the channel, while the action `create` creates a new channel object to be populated. When the user clicks on the create button the action `create channel` at the controller validates the inserted data and if it is all correct the channel is saved, else the new channel view is presented with the corresponding error message.

The `_form.html.erb` view is a partial page which only contains the format to display the channel data. Partial pages are useful to restrain a section of code to one place, reducing code repetition and lowering management complexity.

The user management is done through the `list_users.html.erb` view that lists all the users and shows the option to activate or block a user, `activate_user` and `block_user` actions. Both
4.2 User Interface

actions after updating the user information invoke the list_users action in order to present all the users with the proper updated information.

All of the above views are accessible through the index view. This view only contains the management options that the administrator can access.

All the models, controllers and views, with the associated actions, involved are presented in Figure 4.5.

Figure 4.5: The administration controller actions, models and views.

4.2.2. D Mosaic controller and associated views

The mosaic controller is the regular user’s home page and it is named mosaic because in the first page channels are presented as a mosaic. This controller unique action is index, which creates a local variable with all the visible channels and this variable is used in the index.html.erb page to present the channels image in a mosaic design.

An additional feature is to keep track of the last viewed channel by the user. This feature is easily implemented through the following steps:

1. Add to the users data scheme a variable to keep track of the channel, last_channel;
2. Every time the channel changes the variable is updated.

This way the mosaic page displays the last viewed channel by the user.

4.2.2. E View controller and associated views

The view controller is responsible for several operation, namely:

- The presentation of the transmitted stream;
- Presenting the EPG [74] for a selected channel;
- Changing channel validation.

The EPG is an extra feature, extremely useful whether for recording purpose or to view/consult when a specific programme is transmitted.
4. Multimedia Terminal Implementation

The view controller `index` action redirects the user request to the `streaming` action associated to the `streaming.html.erb` view. In the streaming action, besides presenting the stream, two different tasks are performed. The first task is to get all the visible channels in order to present them to the user allowing him to change channel. The second task is to present the name of the current and next programme of the transmitted channel. To get the [EPG](#) for each channel it is used XMLTV open-source tool [34] [88].

**EPG**

XML TV file format was originally created by Ed Avis and it is currently maintained by the XMLTVProject [35]. The XMLTV consists in the acquisition of channels programming guide, in XML format, from a web server, having several servers available throughout the world. Initially the used XMLTV server in Portugal was `www.tvcabo.pt`, but this server stopped working and the information was obtained from the `http://services.sapo.pt/EPG` server. So XMLTV generates several XML documents, one for each channel, containing the list of programmes, the starting and ending time and in some cases the programme description.

Each day the channel's EPG is downloaded from the server. This task is performed by a batch script, `getEPG.sh`, located at `/lib/epg` under the multimedia terminal project. The script behaviour is: eliminate all EPGs older then 2 days (currently there is no further use for these information); contact the server and download the EPG for the next 2 days. The elimination of older EPGs is necessary to remove unnecessary files from the computer since that the files occupy a significant disk space (about 1MB each day).

Rails has a native tool to process XML, Ruby Electric XML (REXML) [33]. The user streaming page displays the actual programme being watched and the next one (in the same channel). This feature is implemented in the streaming action and the steps to acquire the information are:

1. Find the file that corresponds to the channel currently viewed;
2. Match the programmes time to find the actual one;
3. Get the next programme in the EPG list.

The implementation has an important detail. If the viewed programme is the last of the day, the actual EPG list does not contains the next programme. The solution is to get the tomorrow's EPG and present the first programme in the list.

Another use for the EPG is to show to the user the entire list of programmes. The multimedia terminal allows the user to view the yesterday, today and tomorrow's EPG. This is a simple task: after choosing the channel, `select_channel.html` view, the `epg` action grabs the corresponding file, according to the channel and the day, and displays it to the user, Figure 4.6.

In this menu the user can schedule the recording of a programme by clicking in the record button near the desired show. The `record` action gathers all the information to schedule the recording: start and stop time, channel's name and id, programme name. Before adding the recording to the database, it has to be validated and only then the recording is saved (recording validation is described in the Scheduler Section).

**Change Channel**

Another important action in this controller is `setchannel` action. This action is responsible for invoking the script that changes the channel viewed by every user (explained in detail in the Streaming section). In order to change the channel the next conditions need to be met:

- No recording is in progress (the system gives priority to recordings);
- Only the oldest logged user has permission to change the channel (first come first get strategy);
4.2 User Interface

Additionally, for logical purposes, the requested channel can not be the same that the actual transmitted channel.

To assure the first requirement, every time a recording is in progress, the process ID and name is stored at `/lib/streamer_recorder/PIDS.log` file. This way, the first step is to check if there is a process named `recorderworker` in the `PIDS.log` file. The second step is to verify if the user that requested the change is the oldest in the system. Each time a user logs into the system successfully, the user email is inserted into a global control array and removed when he logs out. The insertion and removal of the users is done in the `session controller` which is an extension of the previous mentioned Devise authentication module.

Verified the above conditions, i.e., no recording ongoing, the user is the oldest and the channel required is different from the actual, the script to change the channel is executed and the page `streaming.html.erb` is reloaded. If some of the conditions fail a message is displayed to the user stating that the operation is not allowed and the reason for it.

To change the quality there are two links that invoke the `set_size` action with different parameters. Each user as a session variable, resolution, indicating the quality of the stream he desires to view. Modifying this value changes the viewed stream quality by selecting the corresponding link in the view `streaming.html.erb`. The streaming and all its details is explained in the Streaming Section.

### 4.2.2.F Recording Controller and associated Views

The recording controller is responsible for the management of recordings and recorded videos (the CRUD convention was once again adopted in this controller, thus the same actions have been implement). For recording management there are the actions: new and create, list, edit and update, and delete and destroy all followed by the suffix recording. Figure 4.7 presents the models, views and actions used by the recording controller.

Each time a new recording is inserted it as to be validated, through the Recording Scheduler, and only if there is no time/channel conflict the recording is saved. The saving process also includes adding to the system scheduler, Unix Cron, the recording entry. This is done by means of the Unix `at` command [23], where it is given the script to run and the date/time (year, month, day, hour, minute) it should run, syntax `at -f recorder.sh -t time`.

There are three other actions applied to videos that were not mentioned, namely:

- **View_video action** - plays the video selected by the user;
4. Multimedia Terminal Implementation

- Download_video action - allows the user to download the requested video, and this is accomplished using Rails send_video method [30];

- Transcode_video and do_transcode, first action invokes the transcoding script with the user id and the filename as arguments. The transcoding processes is further detailed in the Recording Section.

4.2.2.G Recording Scheduler

The recording scheduler as previously mention is invoked every time a recording is request and when some parameter is modified.

In order to centralize and to facilitate the algorithm management, the scheduler algorithm lies at \lib\recording\methods.rb and it is implemented using ruby. There are several steps in the validation of the recording, namely:

1. Is the recording in the future?

2. Is the recording ending time after it starts?

3. Find if there are time conflicts (Figure 4.8). If there are no intersections the recording is scheduled, else there are two options, the recording is in the same channel or the recording is in a different channel. If the recording intersects another previously saved recording and it is the same channel there is no conflict, but if it is in different channels the scheduler does not allow that setup.

The resulting pseudo-code algorithm is presented in Figure 4.9

If the new recording passes the tests, it is returned the true value and the recording is saved, else the message corresponding to the problem is shown.
4.3 Streaming

Figure 4.8: Time intersection graph.

4.2.2.H Video-call Controller and associated Views

The video-call controller actions are: index - invokes the index.html.erb view which allows the user to insert the local and remote streaming data; and present_call action - invokes the view named after it with the inserted links, allowing the user to view side by side the local and remote streams. This solution is further detailed in the Video-Call Section.

4.2.2.I Properties Controller and associated Views

The properties controller is where the user configuration lies. The index.html.erb page contains the links for the actions the user can execute: change the user default streaming quality, change_def_res action, and restart the streaming server in case it stops streaming.

This last action, reload, should be used if the stream stops or if after some time there is no video/audio, which may occasionally occur after requesting a channel change (the absence of audio/video relates to the fact that sometimes, when the channel changes, the streaming buffer takes some time to acquire the new audio/video data). The reload action invokes two bash scripts: stopStreamer and startStreamer, which, as the name indicates, stops and starts the streaming server (see next section).

4.3 Streaming

The streaming implementation was the hardest to do due to the requirements previously established. The streaming had to be supported by several browsers and this was a huge problem. In the beginning it was defined that the video stream should be encoded in H.264 [9] format using the GStreamer Framework tool [41]. A streaming solution was developed using GStreamer Real Time Streaming Protocol (RTSP) [29] Server [25], but viewing the stream was only possible using
4. Multimedia Terminal Implementation

def is_valid_recording(recording):
    @new = recording
    # recording the pass?
    if (Time.now > Recording.start_at):
        Display.Message "Wait... You can't record things from the pass..."
    end

    # stop time before start time
    if (Recording.stop_at < Recording.start_at):
        Display.Message "Wait... You can't stop recording before starting..."
    end

    # recording is set to the future - now check for time conflict
    from = Recording.start_at
    to = Recording.stop_at
    # go trough all recordings
    For each Recording - rec
        # check the rest if it is a just once record in another day
        if (rec.periodicity == "Just Once" and Recording.start_at.day != rec.start_at.day)
            next
        end

        @start = rec.start_at
        @stop = rec.stop_at
        # outside, check the rest (Figure 4.8)
        if @to < @start or @from > @stop
            next
        end

        # intersection! (Figure 4.8)
        if (@from < @start and @to < @stop) or
           (@from > @start and @to < @stop) or
           (@from < @start and @to > @stop) or
           (@from > @start and @to > @stop)
            if (channel is the same)
                next
            else
                Display.Message "Time conflict! There is another recording at that time!"
            end
        end
    end

    return true
end

Figure 4.9: Recording validation pseudo-code.

tools like VLC Player [52]. VLC Player had a visualization plug-in for Mozzila Firefox [27] that
did not work properly and it was a limitation to the developed solution, it would work only in some
browsers. The browsers that supported H.264 video with Advanced Audio Coding (AAC) audio
format in a MP4 container were [92]:

- Safari [16] to Macs and Windows PCs (3.0 and later) support anything that QuickTime [4]
supports. QuickTime does ship with support for H.264 video (main profile) and AAC audio
in an MP4 container.

- Mobile phones, e.g., Apple’s iPhone [15] and Google Android phones [12], support H.264
video (baseline profile) and AAC audio (“low complexity” profile) in an MP4 container;

- Google Chrome [13] dropped H.264 + AAC in a MP4 container support since version 5, due
to H.264 licensing requirements [56]
4.3 Streaming

After some investigation about the supported formats by most browsers [92], it was concluded that the most feasible video and audio format would be: video encoded in VP8 [81], audio Vorbis [87], both mixed in a WebM [32] container. At the time GStreamer did not support support VP8 video streaming.

Due to this constraints, using GStreamer Framework was no longer a valid option. To overcome this major problem another open-source tool was researched, Flumotion open-source Multimedia Streaming Server [24]. Flumotion was founded in 2006 by a group of open source developers and multimedia experts, and it is intended for broadcasters and companies to stream live and on demand content in all the leading formats from a single server. This end-to-end and yet modular solution includes signal acquisition, encoding, multi-format transcoding and streaming of contents. This way, with a single softwate solution, it was possible to implement most of the modules defined previously in the architecture.

Due to Flumotion multiple format support, it overcomes the limitations encountered when using GStreamer. To maximize the number of supported browsers, the audio and video are streamed using the WebM [32] container format. The reason to use the WebM format has to do with the fact that HTML5 [91] supports it natively. WebM format is supported by the following browsers:

- Internet Explorer (IE) 9 will play WebM video if it is installed a third-party codec, e.g., WebM/VP8 DirectShow Filters [18] and OGG codecs [19], which is not installed by default on any version of Windows;
- Mozilla Firefox (3.5 and later) supports Theora [58] video and Vorbis [87] audio in an Ogg container [21]. Firefox 4 also supports WebM;
- Opera (10.5 and later) supports Theora video and Vorbis audio in an Ogg container. Opera 10.60 also supports WebM;
- Google Chrome latest versions offer full support for WebM;
- Google Android [12] support the WebM format from version 2.3 and later.

WebM defines the file container structure where the video stream is compressed with the VP8 [81] video codec, the audio stream is compressed with the Vorbis [87] audio codec and mixed together into a Matroska [89] like container named WebM. Some benefits of using WebM format are: openness, innovation and optimized for the web. Addressing WebM openness and innovation, its core technologies, such as HTML, HTTP, and TCP/IP, are open for anyone to implement and improve. Being the video the central web experience, a high-quality and open video format choice is mandatory. As for optimization, WebM runs in low computational footprint in order to enable playback on any device, (i.e., low-power netbooks, handhelds, tablets), it is based in a simple container and offers a high quality and real-time video delivery.

4.3.1 The Flumotion Server

Flumotion is written in Python using GStreamer Framework and Twisted [70], an event-driven networking engine also written in Python. A single Flumotion system is called a Planet. It contains several components working together, some of these called Feed components. The feeders are responsible for receiving data, encoding, and ultimately streaming the manipulated data. A group of Feed components is designated as a Flow. Each Flow component outputs data that is taken as an input by the next component in the Flow, transforming the data step by step. Other components may perform extra tasks, such as restricting access to certain users, or allowing users to pay for
access to certain content. These other components are known as Bouncer components. The aggregation of all these components results in the Atmosphere. The relation of this components is presented by Fig. 4.10.

There are three different types of Feed components belonging to the Flow:

- **Producer** - A producer only produces stream data, usually in a raw format, though sometimes it is already encoded. The stream data can be produced from an actual hardware device (webcam, FireWire camera, sound card, ...), by reading it from a file, by generating it in software (e.g., test signals), or by importing external streams from Flumotion servers or other servers. A feed can be simple or aggregated. An aggregated feed might produce both audio and video. As an example, an audio producer component provides raw sound data from a microphone or other simple audio input. Likewise, a video producer provides raw video data from a camera.

- **Converter** - A converter converts stream data. It can: encode or decode a feed; combine feeds or feed components to make a new feed; change the feed by changing the content, overlaying images over video streams, compressing the sound, ... For example, an audio encoder component can take raw sound data from an audio producer component and encode it. The video encoder component encodes data from a video producer component. A combiner can take more than one feed, for instance, the single-switch-combiner component can take a master feed and a backup feed. If the master feed stops supplying data then it will output the backup feed instead. This could show a standard “Transmission Interrupted” page. Muxers are a special type of combiner component, combining audio and video to provide one stream of audiovisual data, with the sound synchronized correctly to the video.

- **Consumer** - A consumer only consumes stream data. It might stream a feed to the network, making it available to the outside world, or it could capture a feed to disk. For example, the http-streamer component can take encoded data and serve it via HTTP for viewers on the Internet. Other consumers, such as the shout2-consumer component can even make Flumotion streams available to other streaming platforms, such as IceCast.

There are other components that are part of the Atmosphere. They provide additional functionality to flows and are not directly involved in creation or processing of the data stream. It is the example of the Bouncer component that implements an authentication mechanism. It receives
authentication requests from a component or manager and verifies that the requested action is allowed (communication between components in different machines).

The Flumotion system consists of a few server processes (daemons) working together. The Worker creates the Components processes while the Manager is responsible for invoking the Worker processes. Fig. 4.11 illustrates a simple streaming scenario involving a Manager and several Workers with several processes. After the manager process starts, an internal Bouncer component is used to authenticate workers and components; it waits for incoming connections from workers to command them to start their components. These new components will also log in to the manager for proper control and monitoring.

![Flumotion simple streaming application architecture.](image)

**4.3.2 Flumotion Manager**

Flumotion is an administration user interface but also supports input from XML files for the Manager and Workers configuration. The Manager XML file contains the planet definition, which in turn contains nodes for the Planet’s manager, atmosphere and flow, which themselves contain component nodes. The typical structure of a XML manager file is presented by Fig 4.12, where the three distinct sections: manager, atmosphere and flow are part of the planet.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<planet name="planet">
  <manager name="manager">
    <!-- manager configuration -->
  </manager>

  <atmosphere>
    <!-- atmosphere components definition -->
  </atmosphere>

  <flow name="default">
    <!-- flow component definition -->
  </flow>
</planet>
```

![Manager basic XML configuration file.](image)
4. Multimedia Terminal Implementation

In the manager node, it can be specified the manager’s host address, the port number and the transport protocol that should be used. Nevertheless, the defaults should be used, if no specification is set. The default SSL transport protocol should be used to ensure secure connections, unless Flumotion is running on an embedded device with very restrict resources or in a private network. The defined manager configuration is shown in Figure 4.13.

```xml
<manager name="tv-manager">
  <host>146.193.56.43</host>
  <port>8642</port>
  <transport>tcp</transport>

  <component name="manager-bouncer" type="htpasswdcrypt-bouncer">
    <property name="data">paiva:DF4wh3SMa4q/2</property>
  </component>

</manager>
```

Figure 4.13: Manager connections definition.

After defining the manager configurations, it comes the definition of the atmosphere and the flow. In the manager’s atmosphere it is defined the porter and the htpasswdcrypt-bouncer. The porter is the component that listens to a network port on behalf of other components, e.g. the http-stream, while the htpasswdcrypt-bouncer is used to ensure that only authorized users have access to the streamed content. This components are defined as shown in Figure 4.14.

```xml
<atmosphere>

  <component name="porter-http" type="porter" label="porter-http" worker="generalworker">
    <property name="username">dijlQWcYVCAv</property>
    <property name="socket-path">flu-4cif.socket</property>
    <property name="password">bpHOubrGkkrC</property>
    <property name="port">8001</property>
  </component>

  <component name="streamer-bouncer" type="htpasswdcrypt-bouncer" label="htpasswdcrypt-bouncer" worker="generalworker">
    <property name="data"><someone:5jKUrPBXbzos></property>
  </component>

</atmosphere>
```

Figure 4.14: Manager components definition.

The manager’s flow defines all the components related to the audio and video acquisition, encoding, muxing and streaming. The used components, parameters and corresponding functionality are given in Table 4.3.

4.3.3 Flumotion Worker

As previously explained, the worker is responsible for the creation of the processes that execute/materialize the components defined in the manager. The workers XML configuration file contains the information required by the worker in order to know which manager it should login to and what information it should provide to authenticate it self. The parameters of a typical worker are defined in three nodes:

- manager node - were lies the the manager’s hostname, port and transport protocol;
Table 4.3: Flow components - function and parameters.

<table>
<thead>
<tr>
<th>Component</th>
<th>Function</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>soundcard-producer</td>
<td>Captures a raw audio feed from a sound-card.</td>
<td>channels (default 2), depth (default 16), device, sample rate (in Hz. default: 44100), source-element.</td>
</tr>
<tr>
<td>webcam-producer</td>
<td>Produces a raw video feed from a Web-cam or TVCard.</td>
<td>format, framerate, height, width, mime, element-factory, device*.</td>
</tr>
<tr>
<td>overlay-video</td>
<td>Overlays text and images onto a raw video feed.</td>
<td>eater (produced component name), showtext, fluendo-logo.</td>
</tr>
<tr>
<td>pipeline-converter</td>
<td>A generic GStreamer pipeline converter.</td>
<td>eater and a partial GStreamer pipeline (e.g. videoscale ! video/x-raw-yuv,width=176,height=144).</td>
</tr>
<tr>
<td>vorbis-encoder</td>
<td>An audio encoder that encodes to Vorbis.</td>
<td>eater, bitrate (in bps), channels and quality if no bitrate is set.</td>
</tr>
<tr>
<td>vp8-encoder</td>
<td>Encodes a raw video feed using vp8 codec.</td>
<td>eater feed, bitrate, keyframe-maxdistance, quality, speed(defaults to 2) and threads (defaults to 4).</td>
</tr>
<tr>
<td>WebM-muxer</td>
<td>Muxes encoded feeds into an WebM feed.</td>
<td>eater: video and audio encoded feeds.</td>
</tr>
<tr>
<td>http-streamer</td>
<td>A consumer that streams over HTTP.</td>
<td>eater muxed audio and video feed, porter username and password, mount point, burst on connect, port to stream, bandwidth and clients limit.</td>
</tr>
</tbody>
</table>

- **authentication node** - contains the username and password required by the manager to authenticate the worker. Although the password is written as plaintext in the worker’s configuration file, using the SSL transport protocol ensures that the password it is not passed over the network as clear text;

- **feederport node** - it specifies an additional range of ports that the worker may use for unencrypted TCP connections, after a challenge/response authentication. For instance a component in the worker may need to communicate with components in other workers, to receive feed data from other components.

There were defined three distinct workers. This distinction was due to the fact that there were some tasks that should be grouped and other that should be associated to a unique worker, it is the case of changing channel where the worker associated to the video acquisition should stop to allowed a correct video change. The three defined workers were:

- **video worker**, responsible for the video acquisition;
- **audio worker**, responsible for the audio acquisition;
- **general worker**, responsible for the remaining tasks: scaling, encoding, muxing and streaming the acquired audio and video.

In order to clarify the worker XML structure, it is presented the definition of the generalworker.xml in Figure 4.15 (the manager that it should login to, authentication information it should provide and the feederports available for external communication).
Figure 4.15: General Worker XML definition.

4.3.4 Flumotion streaming and management

Defined the Flumotion Manager along with its Workers it is necessary to define the possible set-ups for streaming. Figure 4.16 shows three different setups for Flumotion that can run separately or all together. The possibilities are:

- Stream only in a high size. Corresponds to the left flow in Figure 4.16 where the video is acquired in the desired size and encoded with no extra processing (e.g. resize), muxed with the acquired audio after encoded and HTTP streamed;

- Stream in a medium size, corresponding to the middle flow visible in Figure 4.16. If the video is acquired in the high size it as to be resized before encoding, afterwards it is the same operations as described above;

- Stream in a small size, represented by the operations in the right side of Figure 4.16.

- It is also possible to stream in all the defined formats at the same time; however this increases computation and required bandwidth.

It is also visible an operation named Record in Fig. 4.16. This operation is described in the Recording Section.

In order to enable and control all the processes underlying the streaming, it was necessary to develop a solution that would allow the startup and termination of the streaming server as well as the changing channel functionality. The automation of these three task: startup, stop and change channel, was implement using bash script jobs.

To start the streaming server the defined manager and workers XML structures have to be invoked. The manager as well as the workers are invoked by running the command flumotion-manager manager.xml or flumotion-worker worker.xml from the command line. To run this tasks from within the script and to make them unresponsive to logout and other interruptions the nohup command is used [28].

A problem that was occurring when the startup script was invoked from the user interface, was that the web-server would freeze and become unresponsive to any command. This problem was
due to the fact that when the `nohup` command is used to start a job in the background, it is to avoid the termination of a job. During this time the process refuses to lose any data from/to the background job, meaning that the background process is outputting information of it’s execution and awaiting for possible input. To solve this problem all three I/O methods: normal execution outputted information, error outputted information and possible inputs had to be redirected to the `/dev/null` to be ignored and to allow the expected behaviour. Figure 4.17 presented the code for launching the manager process (the workers follow the same structure).

![Figure 4.17: Launching the Flumotion manager with the nohup command.](image)

To stop the streaming server, the designed script, `stopStreamer.sh`, reads the file containing all the launched streaming processes in order to stop them. This is done by executing the script in Figure 4.18.

![Figure 4.18: Stop Flumotion server script.](image)
4. Multimedia Terminal Implementation

Table 4.4: Channels list - code and name matching for TV Cabo provider.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>E5</td>
<td>TVI</td>
</tr>
<tr>
<td>E6</td>
<td>SIC</td>
</tr>
<tr>
<td>SE19</td>
<td>NATIONAL GEOGRAPHIC</td>
</tr>
<tr>
<td>E10</td>
<td>RTP2</td>
</tr>
<tr>
<td>SE5</td>
<td>SIC NOTICIAS</td>
</tr>
<tr>
<td>SE6</td>
<td>TVI24</td>
</tr>
<tr>
<td>SE8</td>
<td>RTP MEMORIA</td>
</tr>
<tr>
<td>SE15</td>
<td>BBC ENTERTAINMENT</td>
</tr>
<tr>
<td>SE17</td>
<td>CANAL PANDA</td>
</tr>
<tr>
<td>SE20</td>
<td>VH1</td>
</tr>
<tr>
<td>S21</td>
<td>FOX</td>
</tr>
<tr>
<td>S22</td>
<td>TV GLOBO PORTUGAL</td>
</tr>
<tr>
<td>S24</td>
<td>CNN</td>
</tr>
<tr>
<td>S25</td>
<td>SIC RADICAL</td>
</tr>
<tr>
<td>S26</td>
<td>FOX LIFE</td>
</tr>
<tr>
<td>S27</td>
<td>HOLLYWOOD</td>
</tr>
<tr>
<td>S28</td>
<td>AXN</td>
</tr>
<tr>
<td>S35</td>
<td>TRAVEL CHANNEL</td>
</tr>
<tr>
<td>S38</td>
<td>BIOGRAPHY CHANNEL</td>
</tr>
<tr>
<td>22</td>
<td>Euronews</td>
</tr>
<tr>
<td>27</td>
<td>Odisssea</td>
</tr>
<tr>
<td>30</td>
<td>Mezzo</td>
</tr>
<tr>
<td>40</td>
<td>RTP AFRICA</td>
</tr>
<tr>
<td>43</td>
<td>SIC MULHER</td>
</tr>
<tr>
<td>45</td>
<td>MTV PORTUGAL</td>
</tr>
<tr>
<td>47</td>
<td>DISCOVERY CHANNEL</td>
</tr>
<tr>
<td>50</td>
<td>CANAL HISTORIA</td>
</tr>
</tbody>
</table>

Switching channels

The most delicate task was the process to change the channel. There are several steps that need to be followed for correctly changing channel, namely:

- Find in the PIDS.log file the PID of the videoworker and terminate it (this initial step is mandatory in order to allow other applications to access the TV card, namely the v4lctl command);
- Invoke the command that switches to the specified channel. This is done by using the command v4lctl [51] used to control the TV card;
- Launch a new videoworker process to correctly acquire the new TV channel.

The channel code argument is passed to the changeChannel.sh script by the [UI]. The channel list was created using another open-source tool: XawTV [54]. XawTV was used to acquire the list of codes for the available channels offered by the TV-Cabo provider, see Table 4.4. To create this list it was used the XawTV auto-scan tool, scantv, with the identification of the TV-Card (-C /dev/vbi0) and the file to store the results, -o output_file.conf. Running this command generates a list of channels presented in Table 4.4 that is used in the entire application. The result of the scantv* tool was the list of available codes, which is later translated into the channel name.
4.4 Recording

The recording feature should not interfere in the normal streaming of the channel. Nonetheless, to correctly perform this task it may be necessary to stop streaming, due to channel changing or quality setup, in order to correctly record the contents. This feature is also implemented using the Flumotion Streaming Server. One of the other options available, beyond streaming, is to record the content into a file.

Flumotion Preparation Process

To allow the recording of a streamed content it is necessary to add a new task to the Manager XML file, as explained in the Streaming section, and create a new Worker to execute the recording task defined in the manager. To materialize this feature a component named disk-consumer, responsible for saving the streamed content to disk, should be added to the manager configuration (see Figure 4.19).

```xml
<component name="disk-audio-video" type="disk-consumer" worker="recordworker">
    <eater name="default">
        <feed alias="default">muxer-audio-video:default</feed>
    </eater>
    <property name="directory">/tmp</property>
    <property name="filename">tvcapture</property>
    <property name="start-recording">True</property>
</component>
```

Figure 4.19: Flumotion recorder component definition.

As for the worker it should follow a similar structure to the ones presented in the Streaming Section.

Recording Logic

After defining the recording functionality in the Flumotion Streaming Server, it is necessary an automated control system for executing a recording when scheduled. The solution to this problem was to use the Unix at command, as described in the UI Section, with some extra logic in a Unix job. When the Unix system scheduler finds that it is necessary to execute a scheduled recording, it follows the procedure represented in Figure 4.20 and detailed below.

The job invoked by Unix Cron [31], recorder.sh, is responsible for executing a Ruby job, :start_rec. This Ruby job is invoked through rake command; it goes through the scheduling database records and searches for the recording that should start.

1. If no scheduling is found then nothing is done (e.g. the recording time was altered or removed).
2. Else, it invokes in background the process responsible for starting the recording - invoke_recorder.sh. This job is invoked with the following parameters: recordingID, to remove the scheduled recording from the database after it starts; the user ID, in order to know to which user this recording belongs to; the amount of time to record; the channel to record and the quality; and finally the recording name for the resulting recorded content.

After running the :start_rec action and finding that there is a recording that needs to start, the recorder.worker.sh job procedure is as follows:
4. Multimedia Terminal Implementation

Figure 4.20: Recording flow, algorithms and jobs.

1. Check if the file `.progress` has some content. If the file is empty there are no current recordings in progress, else there is a recording in progress and there is no need to set up the channel and to start the recorder;

2. When there is no recordings in progress, the job changes the channel to the one scheduled to record, by invoking the `changeChannel.sh` job. Afterwards the Flumotion recording worker job is invoked accordingly to the defined quality to record and the job waits until the recording time ends;

3. When the recording job "wakes up" (`recorder.worker`), there are two different flows. After checking that there is no other recording in progress, the Flumotion recorder worker is stopped, using the FFmpeg tool the recorded content is inserted into a new container, moved into the `/public/videos` folder and added to the database. The need of moving the audio and video into a new container has to do with the Flumotion recording method. When it starts to record, the initial time is different from zero and the resultant file cannot be played from a selected point (index loss). If there are other recordings in progress in the same channel, the procedure is similar. The streaming server continues the previous recording and then, using FFmpeg with the start and stop times, the output file is sliced, moved into the `/public/videos` folder and added to the database.

**Video Transcoding**

There is also the possibility for the users to download their recorded content and to transcode that content into other formats (the recorded format is the same as the streamed format, in order to reduce computational processing, but it is possible to re-encode the streamed data into another format if desired). In the transcoding sections the user can change the native format, VP8 video and VORBIS audio in a WebM container, into other formats, like H.264 video and AAC audio in a Matroska container and to any other format by adding it to the system.

The transcode action is performed by the `transcode.sh` job. Encoding options may be added by using the last argument passed to the job. Actually the existent transcode is from WebM to...
4.5 Video-Call

The video call functionality was conceived in order to allow users to interact simultaneously through video and audio in real time. This kind of functionality normally assumes that the video-call is established through an incoming call, originated from some remote user. The local user naturally has to decide whether to accept or reject the call.

To implement this feature in a non traditional approach the Flumotion Streaming Server was used. The principle of using Flumotion is that, in order for the users communicate between themselves, each user needs Flumotion Streaming Server installed and configured to stream the content captured by the local webcam and microphone. After configuring the stream, the users exchange between them the link where the stream is being transmitted and insert it into the fields in the video-call page. After inserting the transmitted links, the web server creates a page where the two streams are presented simultaneously, representing a traditional video-call with the exception of the initial connection establishment.

To configure the Flumotion to stream the content from the webcam and the microphone the users need to do the following actions:

- In a command line or terminal, invoke the Flumotion through the command: `$flumotion-admin`;
- A configuration window will appear and it should be selected the “Start a new manager and connect to it” option;
- After creating a new manager and connecting to it, the user should select the “Create a live stream.” option;
- The user then selects the video and audio input sources, webcam and microphone respectively, defines the video and audio capture settings, encoding format and then the server starts broadcasting the content to any other participant.

This implementation allows multiple user communication. Each user starts his content streaming and exchanges the broadcast location. Then the recipient users insert the given location into the video-call feature which will display them.

The current implementation of this feature still requires some work in order to make it easier to use and to require less work from the user end. The implementation of a video-call feature is a complex task, given its enormous scope and it requires an extensive knowledge of several video-call technologies. In the Future Work section (Conclusions chapter) it is presented some possible approaches to overcome and improve the current solution.

4.6 Summary

In this section it was described how the framework prototype was implemented and how each independent solution was integrated with each other.

The implementation of the UI and some routines was done using RoR. The solution development followed all the recommendations and best practices in order to make a robust, easy to modify and above all easy to integrate new and different features.
4. Multimedia Terminal Implementation

The most challenging components were the ones related to: streaming acquisition, encoding, broadcasting and recording. From the beginning there was the issue with the selection of a free, working, supportive open-source application. In a first stage, a lot of effort was done to get GStreamer Server [25] to work. Afterwards, when finally the streamer was properly working, there was the problem with the representation of the stream that could not be exceeded (browsers did not support video streaming in the H.264 format).

To overcome this situation, an analysis of which were the audio/video formats most supported by the browsers was conducted. This analysis lead to the vorbis audio [87] and VP8 [81] video streaming format, WebM [32], and hence to the use of the Flumotion Streaming Server [24], that given its capabilities was the suitable open-source software to use.

All the obstacles were exceeded using all available sources:

- The Ubuntu Unix system offered really good solutions regarding the components interaction. As each solution was developed as a "stand-alone", there was the need to develop the means to glue altogether and that was done using bash scripts;
- The RoR framework was also a good choice thanks to ruby programming language and to the rake tool.

All the established features were implemented and work smoothly, the interface is easy to understand and use thanks to the usage of the developed conceptual design.

The next chapter presents the results of applying several tests, namely: functional, usability, compatibility and performance tests.
Evaluation

Contents

5.1 Transcoding: codec assessment .................................. 56
5.2 VP8 and H.264 comparison ...................................... 63
5.3 Testing Framework ................................................. 65
5.4 Conclusions .......................................................... 73
5. Evaluation

These chapter presents the tests that where conducted to evaluate the developed solution and the respective evaluation of the results. It begins by describing the evaluation of several video codecs and profiles in order to determine which configurations offered the best quality vs encoding time for the transcoding feature. After the definition of the encoding profiles, it is describe the testing framework and follows by presenting the obtained results along with the corresponding conclusions.

The conducted tests were intended to prove the functionality of the solution implemented as well as the system performance. The presented tests are divided into four types: performance testing, functional testing, usability testing and compatibility testing. These tests give an insight of the potential of the developed architecture. No security tests were conducted due to the personal usage nature of the solution, and because no sensitive data is currently held.

5.1 Transcoding: codec assessment

Some work was done in order to evaluate several codecs and presets. This was conducted in order to determine the optimal conditions of video encoding. Initially these results were supposed to be used for video streaming; however, when it was conclude that the GStreamer solution was not feasible (view section 4.3 Streaming), the results were used to re-encode the acquired video (transcoding).

Encoding assessment tools

MEncoder

Pros

- For the same encoding parameters it often offers a better performance, which herein is measured based on the Peak Signal-to-Noise Ratio (PSNR). This method is an objective video quality measure that consists on the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. In other words, it is a measure of the quality of the reconstruction of lossy compression codecs.

Cons

- It has a slow development cycle and support from the community. In fact, many questions that were addressed to the community, received the answer that this application was not actively maintained and would probably be discontinued. In some cases, the users were even encouraged to the use FFmpeg.

- Another problem is the provided documentation. The technical report is often very poor in the explanation and lacks a lot in important details.

- For the same operations performed in FFmpeg, MEncoder took twice the time.
• The command line of this application is also very complicated and confusing, when compared to FFmpeg. While in FFmpeg we can clearly distinguish the options we are addressing, in MEncoder the options come all together making it hard to understand them.

**FFmpeg**

**Pros**

• This application is constantly being improved and new functionalities are being added; it has several integrated solutions such as: (1) ffmpeg, a command line tool to convert the encoding format of multimedia files; (2) ffserver, a multimedia streaming server for live broadcasts; (3) ffplay, a simple media player, based on Simple Direct media Layer (SDL [85]) and the FFmpeg libraries.

• FFmpeg supports a great variety of libraries that intend to provide maximum support to the users. It supports: (1) libavutil, a library containing functions for simplifying programming, including random number generators, data structures, mathematics routines and much more; (2) libavcodec, a library containing decoders and encoders for audio/video codecs; (3) libavformat, a library containing demuxers and muxers for multimedia container formats; (4) libavdevice, a library containing input and output devices for grabbing from and rendering to many common multimedia input/output software frameworks, including V4L [51], Video for Windows (VfW) , and ALSA [36]; (5) libswscale, a library performing highly optimized image scaling and color space/pixel format conversion operations.

• The manual is very complete and explanatory, although the documentation for some codecs is not included in the official web page (e.g. H.264 [9]). In order to map the existing parameters, other documentation was used [38].

**Setup**

The experiments were conducted using several types of samples, each one with unique characteristics. It was used a 60 second sample of: cartoons, movie, music, news and soccer. The system were these experiments were conducted had the following characteristics: Intel Core2Quad Q9300 CPU @ 2.5 GHz, 4GB of memory running Ubuntu 10.10 as OS. The encoding time was measured using the user time value outputted by the UNIX time function. As for quality, the values were obtained by consulting the outputted encoding logs from the encoding tool.

After defining the encoding tools and the possible usable standards, several experiments were conducted in order to determine which encoding tool was better, having in account the referred pros and cons described. Thus the conducted experiments regarded the encoding of a file, with several different common options, and analyzing the quality (PSNR) and the encoding time. The results are presented in Fig. 5.1. As it can be observed, there is a difference in quality of 1dB between FFmpeg and Mencoder.
5. Evaluation

This difference is mainly due to the buffer control. It should be noticed that video encoding standards only define how to decode a video stream. For the encoding program, this means that only the output is standardized. How the program encodes a video sequence into the output stream is of no importance as long as the output fulfills the standard. This leaves many variables including quantization step, number of intra and prediction frames and motion estimation procedures. Despite this negative result, FFmpeg was chosen because of the pros previously mentioned.

Having defined the encoding tool, the next step was to conduct a series of experiments, in order to determine the most suitable encoding parameters that would match the defined profiles (HQ, MD and LQ). It was defined a set of possible bit-rates to assess the profiles:

- For the HQ, the bit-rate would go from 500kbps up to 1500kb/s, with steps of 50kbps;
- The MQ would go from 100kbps to 1000kbps, also with steps of 50kbps;
- For LQ, it would go from 25kb/s up to 500kb/s, with steps of 25kbps.

Assessment

The assessment of the candidate codecs, was done by considering the quality of the encoded stream (measured by the PSNR value) and the time necessary to encode the file. The obtained values of this assessment can be observed in Fig. 5.2. As can be observed, the quality gain of H.264 was always superior to the other considered codecs. The same can be said for the resulting file size. By analyzing Fig. 5.3 it can be concluded that the worst encoding values obtained with the H.264 codec belongs to the news sample. This is mainly due to the extreme conditions of this type of content: during the news there are several sudden changes of scenario which are hard to encode.

After deciding the most suitable codec (H.264) a new analysis of the FFmpeg documentation was made to determine the parameters to configure the H.264 codec. In the documentation is was
5.1 Transcoding: codec assessment

described the existence of some specific profiles, called presets. To assess the several codecs, the same encoding process was once again used for each described preset. The presets are the following: ultrafast, superfast, veryfast, faster, fast, medium, slow, slower, veryslow and placebo (the slowest preset with higher quality).

After carefully analyzing the obtained data, it was decided that: LQ would use the veryfast preset, MQ the medium preset and the HQ would encode with the slower preset. These decision was made based on the ratio PSNR-Time and on the fact that the encoding time should not overcome the recording time.

Another decision that had to be made regards to the use of variable bit-rate (VBR) instead of constant bit-rate (CBR). The difference between VBR and CBR is: the file encoding is made in one single pass when using CBR; VBR encodes in two ore more passes to assure the best quality.

Figure 5.2: Considered codecs assessment values.
5. Evaluation

and restrain the total file size. The advantage of CBR is to often be faster than VBR. To assess which method was better, the previously described procedure was adopted. The news sample was again encoded with some presets. The values using CBR (1 pass encode), were already obtained in the assessment of the preset to adopt; for the VBR (2 pass encode) it was used the fast, medium, slow and slower presets. The results are summarized by Fig. 5.4. Observing Fig. 5.4 it is clear that for HQ and MQ the CBR method is better. For the LQ setup, the chosen preset (CBR) has a higher encoding time. Nevertheless, from an application uniformization point of view all the content will be encoded using CBR.

Conclusion. After this experimental procedure, the profiles to be adopted were mostly defined. There were some doubts about the exact bit-rate to adopt, because the PSNR was changing slowly. The conclusions drawn by this study are presented in Table 5.1. It can be observed that the exact value for the bit-rate has not yet been decided, because there is a very small difference between the quality levels provided for this range. In order to decide which bit-rate to use, it was decided to conduct a questionnaire to a set of target users to help the selection process of the best quality parameters. That study and its end-results are described in the next section. As for the 2-pass encoding, it may be considered in the future.

Table 5.1: Profiles conclusion.

<table>
<thead>
<tr>
<th>FFmpeg-Preset</th>
<th>Bit-Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>HQ</td>
<td>slower 950-1100kb/s</td>
</tr>
<tr>
<td>MQ</td>
<td>medium 200-250kb/s</td>
</tr>
<tr>
<td>LQ</td>
<td>veryfast 100-125kb/s</td>
</tr>
</tbody>
</table>

Profile Definition

As mentioned in the previous subsection, after considering several different configurations (different bit-rates and encoding options) three concrete setups with an acceptable bit-rate range were selected. In order to choose the exact bit-rate that would fit the users needs, it was prepared
In a first approach, a 30 seconds clip was selected from a movie trailer. This clip was characterized by rapid movements and some dark scenes. That was necessary because these kinds of videos are the worst to encode, due to the extreme conditions they present. Videos with moving scenes are harder to encode with lower bit-rates; they have many artifacts and the encoder needs to represent them in the best possible way with the provided options. The generated samples are mapped with the encoding parameters defined in Table 5.2.

In the questionnaire, the users were asked to view each sample (without knowing the target bit-rate) and classify it in a scale from 1 to 5 (very bad to very good). As it can be seen, in the HQ samples the corresponding quality differs by only 0.1dB, while for MQ and LQ they differ almost 1dB. Surprisingly, the quality difference was almost unnoticed by the majority of the users, as
Table 5.2: Encoding properties and quality level mapped with the samples produced for the first evaluation attempt.

<table>
<thead>
<tr>
<th>Quality</th>
<th>Bit-rate (kb/s)</th>
<th>Sample</th>
<th>Encoder Preset</th>
<th>PSNR (db)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HQ</td>
<td>950</td>
<td>D</td>
<td>veryfast</td>
<td>36,1225</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>A</td>
<td>medium</td>
<td>36,2235</td>
</tr>
<tr>
<td></td>
<td>1050</td>
<td>C</td>
<td>medium</td>
<td>36,3195</td>
</tr>
<tr>
<td></td>
<td>1100</td>
<td>B</td>
<td>medium</td>
<td>36,4115</td>
</tr>
<tr>
<td>MQ</td>
<td>200</td>
<td>E</td>
<td>medium</td>
<td>36,5135</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>F</td>
<td>medium</td>
<td>36,6195</td>
</tr>
<tr>
<td>LQ</td>
<td>100</td>
<td>G</td>
<td>slower</td>
<td>37,807</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>H</td>
<td>slower</td>
<td>38,795</td>
</tr>
</tbody>
</table>

Table 5.3: User’s evaluation of each sample.

<table>
<thead>
<tr>
<th>Sample</th>
<th>Sample B</th>
<th>Sample C</th>
<th>Sample D</th>
<th>Sample E</th>
<th>Sample F</th>
<th>Sample G</th>
<th>Sample H</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>


Looking at each sample average, it is noticeable that these results were inconclusive to determine the exact bit-rate to adopt for each profile. Thus the MQ and LQ, as there were only 2 options, it was decided to use the bit-rate corresponding to the best quality presented in the questionnaire. The MQ would use 250kbps and the LQ 125kbps. Concerning the HQ, a new questionnaire was prepared with a wider difference between the selected bit-rates and with a different sample. This time, it was used an advertisement video as test sample, since it had lots of color texture and movement. The new correlation between the received information is given by Table 5.4

Table 5.4: Encoding properties and quality levels mapped with the samples in the second questionnaire.

<table>
<thead>
<tr>
<th>Quality</th>
<th>Bit-rate (Kbps)</th>
<th>Sample</th>
<th>Encoding Preset</th>
<th>PSNR (db)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HQ</td>
<td>850</td>
<td>B</td>
<td>slower</td>
<td>37.038</td>
</tr>
<tr>
<td>HQ</td>
<td>1050</td>
<td>C</td>
<td>slower</td>
<td>37.523</td>
</tr>
<tr>
<td>HQ</td>
<td>1200</td>
<td>A</td>
<td>slower</td>
<td>37.822</td>
</tr>
</tbody>
</table>

The evaluation of these three samples was scaled from 1 to 3 (bad to very good). This time, the observed difference between the samples was clear to the users. The results are presented in Table 5.5
5.2 VP8 and H.264 comparison

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>Sample A</th>
<th>Sample B</th>
<th>Sample C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 3</td>
<td>3 3 3</td>
<td>2 2 2</td>
<td>2 2 2</td>
</tr>
<tr>
<td>1 – Bad</td>
<td>2 2 2</td>
<td>2 2 2</td>
<td>2 2 2</td>
</tr>
<tr>
<td>3 – Very Good</td>
<td>2 2 3</td>
<td>3 3 3</td>
<td>3 3 3</td>
</tr>
<tr>
<td>Average</td>
<td>2.21</td>
<td>2.14</td>
<td>2.50</td>
</tr>
</tbody>
</table>

**Conclusion.** With these last values, the profiles defined are as follows:

1. HQ - 1050kbps bit-rate using the slower preset.
2. MD - 250kbps using the medium preset.
3. LQ - 125kbps using the veryfast preset.

### 5.2 VP8 and H.264 comparison

Since the announcement of the VP8 video codec there has been several works in order to assess this codec and compare it with the H.264 competitor. In the assessment made by the VP8 developers [81] [82], it is visible that the VP8 video compression codec achieves very competitive quality results against the H.264/AVC encoder. Figure 5.5 shows the results corresponding to encoding quality comparison between VP8 and H.264 (high profile), by using a number of standard test video test clips. For VP8, the used encoding software was libvpx from the WebM Project; H.264 uses a high profile, encoded using the latest x264 encoder software (considered the best encoder implementation for H.264/AVC by the authors).

The authors also inform that VP8 encoder implementation from the WebM Project is not yet making full use of all the VP8 features described in the paper [81]. Some used techniques in other modern video encoders, such as advanced rate control strategies, rate distortion optimization methods, motion estimation methods, are directly applicable to VP8. Thus, there is great potential for innovations in future versions of VP8 encoder and decoder.

Another similar assessment was conducted by the Graphics and Media Lab - CMC department, Lomonosov Moscow State University [69]. This group has performed a deep analysis and comparison between H.264, VP8 and other codecs. The figures in the Appendix B presents the comparison between H.264, VP8 and XviD codecs quality, while Table 5.6 shows the details of the video sequences used in their comparison. The assessment is based in the SSIM Index [104] measure. SSIM is based on measuring three components (luminance similarity, contrast similarity and structural similarity) and combining them into a result value. The obtained results show that
VP8 video codec still as not achieved the encoding quality of its competitor, H.264. The obtained values are always inferior to the H.264 high-quality preset. VP8 developers argue that (source [69]):

- Most input sequences had been previously compressed using other codecs, and these sequences have an inherent bias against VP8 in recompression tests;

- H.264 and MPEG-like encoders have slight advantages in reproducing some of their own typical artifacts; this helps their objective measurement numbers, but not necessarily improve visual quality. This is reflected by the relatively better results for VP8 when the sample is an uncompressed input sequence;

- VP8 delivered better results against other encoders, although VP8 has not been specifically optimized for SSIM, as some other codecs have.

Nevertheless, VP8 codec is presented as a good solution and a direct competitor against H.264. As a consequence, lots of software and devices developed by Google offer native support for WebM (e.g., Google Chrome and all Android 2.3 or later devices).
5.3 Testing Framework

5.3.1 Testing Environment

The server side test where conducted using a personal computer with the following characteristics: Dual Core AMD Opteron 170 processor @ 2.0GHz, with 2 GiB of memory, running Ubuntu 10.04 OS. The two cores were used during server testing.

The client computer had the following characteristics: Intel Core2Quad Q9300 CPU @ 2.50GHz, 4GiB of memory, running Ubuntu 10.10 and Windows 7 with a 8 MBit/s ADSL Internet connection. Additional load tests were made using a personal computer equipped with Intel Core i3 processor @ 2.4GHz, 4GiB of memory and Windows 7 OS.

5.3.2 Performance Tests

The performance tests aim to evaluate the system performance in terms of processing, memory and network bandwidth. The tests where performed at the client side and at the server side. For the collection of the testing data, the top application \cite{17} was used to monitor the CPU and memory utilization. The top application provides a dynamic real-time view of a running system; it displays system summary information as well as a list of tasks currently managed by the Linux kernel. To monitor the network bandwidth it was used the iftop application \cite{14}. The iftop monitors all network activity, and displays a table of current bandwidth.

CPU and Memory usage

It is important to know the amount of resources consumed by the Flumotion Server when there is no user connected, 1 user and several users simultaneously connected. The top output was captured 6 times at a 5 second rate and it was measured the percentage of CPU and memory used by Flumotion (see Table \ref{tab:cpu_memory_usage}).

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Number of frames</th>
<th>Frame rate</th>
<th>Resolution</th>
<th>Bitrates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ice Age</td>
<td>2014</td>
<td>24</td>
<td>720x480</td>
<td></td>
</tr>
<tr>
<td>Indiana Jones</td>
<td>5000</td>
<td>30</td>
<td>704x288</td>
<td>Movies (bitrates of 500-2000 kbps)</td>
</tr>
<tr>
<td>State Enemy</td>
<td>6500</td>
<td>24</td>
<td>720x304</td>
<td></td>
</tr>
<tr>
<td>Up</td>
<td>1920</td>
<td>24</td>
<td>720x480</td>
<td></td>
</tr>
<tr>
<td>Amazon</td>
<td>1200</td>
<td>24</td>
<td>1280x720</td>
<td>High-definition television (bitrates of 0.7-10 mbps)</td>
</tr>
<tr>
<td>Iron Man</td>
<td>600</td>
<td>24</td>
<td>1920x1080</td>
<td></td>
</tr>
<tr>
<td>Mobile Calendar</td>
<td>504</td>
<td>50</td>
<td>1280x720</td>
<td></td>
</tr>
<tr>
<td>Troy</td>
<td>300</td>
<td>24</td>
<td>1920x1072</td>
<td></td>
</tr>
</tbody>
</table>

CPU and memory usage conclusions: as expected, there is a slightly increase of resource consumption when additional users are connected to the server. Even if no user is connected, Flumotion has to acquire audio and video, encode the data, mix both feeds and stream them. This is a demanding process that requires some computational power and memory. Nevertheless, the
Table 5.7: CPU and memory usage samples.

<table>
<thead>
<tr>
<th>Users</th>
<th>0 Users</th>
<th>1 User</th>
<th>3 Users</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Memory (%)</td>
<td>CPU (%)</td>
<td>Memory (%)</td>
</tr>
<tr>
<td>1</td>
<td>17.6</td>
<td>54.0</td>
<td>20.9</td>
</tr>
<tr>
<td>2</td>
<td>18.9</td>
<td>53.6</td>
<td>17.8</td>
</tr>
<tr>
<td>3</td>
<td>17.6</td>
<td>52.5</td>
<td>17.7</td>
</tr>
<tr>
<td>4</td>
<td>18.3</td>
<td>57.8</td>
<td>18.9</td>
</tr>
<tr>
<td>5</td>
<td>18.9</td>
<td>54.6</td>
<td>19.7</td>
</tr>
<tr>
<td>6</td>
<td>19.7</td>
<td>53.5</td>
<td>17.7</td>
</tr>
<tr>
<td>Average</td>
<td>18.5</td>
<td>54.3</td>
<td>18.8</td>
</tr>
</tbody>
</table>

The difference in used resources when no user is connected and when 3 users are connected is small (only, approximately, 5% increase of CPU and 0.4% of memory usage).

Network bandwidth

Currently there are two streaming qualities defined:

- the medium quality video resolution is 352x288 pixels, Common Intermediate Format (CIF), and the bitrate is 400Kb/s;
- the low quality video resolution is 176x144 pixels, Quarter CIF (QCIF), and the bitrate is 200Kb/s.

The evaluation of the network bandwidth required to stream the defined qualities was done using the iftop application. The command that allowed to monitor the network traffic on the client was:

```
iftop -i eth1 -f 'host 146.193.56.43'.
```

Since there are two different streaming qualities (low and medium quality) are being offered, the consumed network bandwidth by the client is different. The values were obtained at intervals of 10 seconds during 1 minute, corresponding to 6 samples per minute.

Table 5.8 presents the samples obtained while 1 user is viewing television in medium quality (data obtained at the client end).

Table 5.8: Medium quality network bandwidth samples.

<table>
<thead>
<tr>
<th>Sample</th>
<th>Outgoing (Kb)</th>
<th>Incoming (Kb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>17.1</td>
<td>367</td>
</tr>
<tr>
<td>1</td>
<td>21.9</td>
<td>507</td>
</tr>
<tr>
<td>2</td>
<td>23.2</td>
<td>508</td>
</tr>
<tr>
<td>3</td>
<td>22.5</td>
<td>505</td>
</tr>
<tr>
<td>4</td>
<td>21.5</td>
<td>484</td>
</tr>
<tr>
<td>5</td>
<td>21.7</td>
<td>481</td>
</tr>
<tr>
<td>6</td>
<td>20.9</td>
<td>485</td>
</tr>
<tr>
<td>Average</td>
<td>21.26</td>
<td>476.71</td>
</tr>
</tbody>
</table>

Table 5.9 presents the samples obtained while 1 user is viewing television in low quality (data obtained at the client end).
### 5.3 Testing Framework

#### Table 5.9: Medium quality network throughput samples.

<table>
<thead>
<tr>
<th>Sample</th>
<th>Outgoing (Kb/s)</th>
<th>Incoming (Kb/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>14.2</td>
<td>168</td>
</tr>
<tr>
<td>1</td>
<td>19.1</td>
<td>283</td>
</tr>
<tr>
<td>2</td>
<td>19.9</td>
<td>287</td>
</tr>
<tr>
<td>3</td>
<td>20.9</td>
<td>295</td>
</tr>
<tr>
<td>4</td>
<td>20.7</td>
<td>298</td>
</tr>
<tr>
<td>5</td>
<td>20.3</td>
<td>288</td>
</tr>
<tr>
<td>6</td>
<td>20.7</td>
<td>282</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>19.4</strong></td>
<td><strong>271.57</strong></td>
</tr>
</tbody>
</table>

To evaluate the network bandwidth it is also important to assess the consumption when several users are connected. A test similar to the previous was therefore conducted to assess the server network usage when 3 users were simultaneously viewing medium quality television, Table 5.10 presents the results.

#### Table 5.10: 3 Clients simultaneously using the system at Medium quality.

<table>
<thead>
<tr>
<th>Sample</th>
<th>Outgoing (Mb/s)</th>
<th>Incoming (Kb/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.98</td>
<td>28.5</td>
</tr>
<tr>
<td>1</td>
<td>1.39</td>
<td>37.5</td>
</tr>
<tr>
<td>2</td>
<td>1.36</td>
<td>38.3</td>
</tr>
<tr>
<td>3</td>
<td>1.23</td>
<td>35.2</td>
</tr>
<tr>
<td>4</td>
<td>1.41</td>
<td>33.5</td>
</tr>
<tr>
<td>5</td>
<td>1.20</td>
<td>37.0</td>
</tr>
<tr>
<td>6</td>
<td>1.33</td>
<td>36.6</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>1.271</strong></td>
<td><strong>30.44</strong></td>
</tr>
</tbody>
</table>

**Network usage conclusions:** the observed differences in the required network bandwidth when using different streaming qualities are clear as expected. The medium quality uses about 476.71Kb/s, while the low quality uses 271.57Kb/s (although Flumotion is configured to stream MQ at 400Kb/s and LQ at 200Kb/s, Flumotion needs some more bandwidth to ensure the desired video quality). As expected, the variation between both formats is approximately 200Kb/s.

When the 3 users were simultaneously connect the increase of bandwidth was as expected. While 1 user needs about 470Kb/s to correctly play the stream, 3 users were using 1.271Mb/s; in the latter each client was getting around 423Kb/s. These results prove that the quality should not be significantly affected when more than one user is using the system; the transmission rate was almost the same and visually there were no visible differences when 1 user or 3 users were simultaneously using the system.

#### 5.3.3 Functional Tests

To assure the proper functioning of the implemented functionalities, several functional tests were conducted. These tests had the main objective of ensuring that the behavior is the expected, i.e., the available features are correctly performed without performance constrains. These functional tests focused on:
5. Evaluation

- login system;
- real-time audio & video streaming;
- changing the channel and quality profiles;
- first come first served priority system (for channel changing);
- scheduling of the recordings either according to the EPG or with manual insertion of day, time and length;
- guaranteeing that channel change was not allowed during recording operations;
- possibility to view, download or re-encode the previous recordings;
- video-call operation.

All these functions were tested while developing the solution and then re-test when the users were performing the usability tests. During all the testing no unusual behavior or problem was detected. It is therefore concluded that the functionalities are in compliance with the architecture specification.

5.3.4 Usability Tests

This section describes how the usability tests were designed, conducted and it also presents the most relevant findings.

Methodology

In order to obtain real and supportive information from the tests it is essential to choose the appropriate number and characteristics of each user, the necessary material and the procedure to be performed.

Users Characterization

The developed solution was tested by 30 users: one family with six members, three families with 4 member and 12 singles. From this group, 6 users were less than 18 years, 7 were between 18 and 25, 9 between 25 and 35, 4 between 35 and 50 and 4 users were older than 50 years. This range of ages cover all age groups to which the solution herein presented is intended. The test users had different occupations, which lead to different levels of expertise with computers and Internet. Table 5.11 summarizes the users description and maps each user age, occupation and computer expertise. Appendix A presents the detail of the users information.
Table 5.11: Key features of the test users.

<table>
<thead>
<tr>
<th>User</th>
<th>Sex</th>
<th>Age</th>
<th>Occupation</th>
<th>Computer Expertise</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Male</td>
<td>48</td>
<td>Operator/Artisan</td>
<td>Medium</td>
</tr>
<tr>
<td>2</td>
<td>Female</td>
<td>47</td>
<td>Non-Qualified Worker</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>Female</td>
<td>23</td>
<td>Student</td>
<td>High</td>
</tr>
<tr>
<td>4</td>
<td>Female</td>
<td>17</td>
<td>Student</td>
<td>High</td>
</tr>
<tr>
<td>5</td>
<td>Male</td>
<td>15</td>
<td>Student</td>
<td>High</td>
</tr>
<tr>
<td>6</td>
<td>Male</td>
<td>15</td>
<td>Student</td>
<td>High</td>
</tr>
<tr>
<td>7</td>
<td>Male</td>
<td>51</td>
<td>Operator/Artisan</td>
<td>Low</td>
</tr>
<tr>
<td>8</td>
<td>Female</td>
<td>54</td>
<td>Superior Qualification</td>
<td>Low</td>
</tr>
<tr>
<td>9</td>
<td>Female</td>
<td>17</td>
<td>Student</td>
<td>Medium</td>
</tr>
<tr>
<td>10</td>
<td>Male</td>
<td>24</td>
<td>Superior Qualification</td>
<td>High</td>
</tr>
<tr>
<td>11</td>
<td>Male</td>
<td>37</td>
<td>Technician/Professional</td>
<td>Low</td>
</tr>
<tr>
<td>12</td>
<td>Female</td>
<td>40</td>
<td>Non-Qualified Worker</td>
<td>Low</td>
</tr>
<tr>
<td>13</td>
<td>Male</td>
<td>13</td>
<td>Student</td>
<td>Low</td>
</tr>
<tr>
<td>14</td>
<td>Female</td>
<td>14</td>
<td>Student</td>
<td>Low</td>
</tr>
<tr>
<td>15</td>
<td>Male</td>
<td>55</td>
<td>Superior Qualification</td>
<td>High</td>
</tr>
<tr>
<td>16</td>
<td>Female</td>
<td>57</td>
<td>Technician/Professional</td>
<td>Medium</td>
</tr>
<tr>
<td>17</td>
<td>Female</td>
<td>26</td>
<td>Technician/Professional</td>
<td>High</td>
</tr>
<tr>
<td>18</td>
<td>Male</td>
<td>28</td>
<td>Operator/Artisan</td>
<td>Medium</td>
</tr>
<tr>
<td>19</td>
<td>Male</td>
<td>23</td>
<td>Student</td>
<td>High</td>
</tr>
<tr>
<td>20</td>
<td>Female</td>
<td>24</td>
<td>Student</td>
<td>High</td>
</tr>
<tr>
<td>21</td>
<td>Female</td>
<td>22</td>
<td>Student</td>
<td>High</td>
</tr>
<tr>
<td>22</td>
<td>Male</td>
<td>22</td>
<td>Non-Qualified Worker</td>
<td>High</td>
</tr>
<tr>
<td>23</td>
<td>Male</td>
<td>30</td>
<td>Technician/Professional</td>
<td>Medium</td>
</tr>
<tr>
<td>24</td>
<td>Male</td>
<td>30</td>
<td>Superior Qualification</td>
<td>High</td>
</tr>
<tr>
<td>25</td>
<td>Male</td>
<td>26</td>
<td>Superior Qualification</td>
<td>High</td>
</tr>
<tr>
<td>26</td>
<td>Female</td>
<td>27</td>
<td>Superior Qualification</td>
<td>High</td>
</tr>
<tr>
<td>27</td>
<td>Male</td>
<td>22</td>
<td>Technician/Professional</td>
<td>High</td>
</tr>
<tr>
<td>28</td>
<td>Female</td>
<td>24</td>
<td>Operator/Artisan</td>
<td>Medium</td>
</tr>
<tr>
<td>29</td>
<td>Male</td>
<td>26</td>
<td>Operator/Artisan</td>
<td>Low</td>
</tr>
<tr>
<td>30</td>
<td>Female</td>
<td>30</td>
<td>Operator/Artisan</td>
<td>Low</td>
</tr>
</tbody>
</table>

Definition of the environment and material for the survey

After defining the test users, it was necessary to define the used material with which the tests were conducted. One of the concepts that surprised all the users submitted to the test, was that their own personal computer was able to perform the test and there was no need to install extra software. Thus, the equipment used to conduct the tests was a laptop with Windows 7 installed and the browsers Firefox and Chrome to satisfy the users.

The tests were conducted in several different environments. Some users were surveyed in their house, others in the university (applied to some students) and in some cases in the working environment. These surveys were conducted in such different environments, in order to cover all the different types of usage that this kind of solution aims.

Procedure

The users and the equipment (laptop or desktop depending on the place) were brought together for testing. To each subject it was given a brief introduction about the purpose and context
5. Evaluation

of the project and an explanation of the test session. It was then given a script with the tasks to perform. Each task was timed and the mistakes made by the user were carefully noted. After these tasks were performed, the tasks were repeated with a different sequence and the results were re-registered. This method aimed to assess the users learning curve and the interface memorization, by comparing the times and errors of the two times that the tasks were performed. Finally, it was presented a questionnaire where they tried to quantitatively measure the user satisfaction towards the project.

The Tasks

The main tasks to be performed by the users attempted to cover all the functionalities in order to validate the developed application. As such, 17 tasks were defined for testing. These tasks are numerated and described briefly in Table 5.12.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log into the system as regular user with the username <a href="mailto:user@test.com">user@test.com</a> and the password user123</td>
<td>General</td>
</tr>
<tr>
<td>2</td>
<td>View the last viewed channel.</td>
<td>View</td>
</tr>
<tr>
<td>3</td>
<td>Change the video quality to the Low Quality (LQ).</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Change the channel to AXN.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Confirm that the name of the current show is correctly displayed.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Access the electronic programming guide (EPG) and view the today’s schedule for SIC Radical channel.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Access the MTV EPG for tomorrow and schedule the recording of the third show.</td>
<td>Recording</td>
</tr>
<tr>
<td>8</td>
<td>Access the manual scheduler and schedule a recording with the following configuration: Time: from 12:00 to 13:00 hours; Channel: Panda; Recording name: Test de Gravação; Quality: Medium Quality.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Go to the Recording Section and confirm that the two defined recordings are correct.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>View the recorded video named “new.webm”</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Transcode the “new.webm” video into H.264 video format.</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Download the “new.webm” video.</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Delete the transcoded video from the server.</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Go to the initial page.</td>
<td>General</td>
</tr>
<tr>
<td>15</td>
<td>Go to the Users Properties.</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Go to the Video-Call menu and insert the following links into the fields: Local: “<a href="http://localhost:8010/local%E2%80%9D">http://localhost:8010/local”</a> Remote: “<a href="http://localhost:8011/remote%E2%80%9D">http://localhost:8011/remote”</a></td>
<td>Video-Call</td>
</tr>
<tr>
<td>17</td>
<td>Log out from the application</td>
<td>General</td>
</tr>
</tbody>
</table>

Usability measurement matrix.

The expected usability objectives are given by Table 5.13. Each task is classified according to:

- Difficulty - level bounces between: easy, medium and hard;
- Utility - values: low, medium or high;
5.3 Testing Framework

- Apprenticeship - how easy is to learn;
- Memorization - how easy is to memorize;
- Efficiency - how much time should it take (seconds);
- Accuracy - how many errors are expected.

Table 5.13: Expected usability objectives.

<table>
<thead>
<tr>
<th>Task</th>
<th>Difficulty</th>
<th>Utility</th>
<th>Apprenticeship</th>
<th>Memorization</th>
<th>Efficiency</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Easy</td>
<td>High</td>
<td>Easy</td>
<td>Easy</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Easy</td>
<td>Low</td>
<td>Easy</td>
<td>Easy</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Easy</td>
<td>Medium</td>
<td>Easy</td>
<td>Easy</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Easy</td>
<td>High</td>
<td>Easy</td>
<td>Easy</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Easy</td>
<td>Low</td>
<td>Easy</td>
<td>Easy</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Easy</td>
<td>High</td>
<td>Easy</td>
<td>Easy</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>Medium</td>
<td>High</td>
<td>Easy</td>
<td>Easy</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
<td>120</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>Medium</td>
<td>Medium</td>
<td>Easy</td>
<td>Easy</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>Medium</td>
<td>Medium</td>
<td>Easy</td>
<td>Easy</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>Hard</td>
<td>High</td>
<td>Medium</td>
<td>Easy</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>Medium</td>
<td>High</td>
<td>Easy</td>
<td>Easy</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>Medium</td>
<td>Medium</td>
<td>Easy</td>
<td>Easy</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>Easy</td>
<td>Low</td>
<td>Easy</td>
<td>Easy</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>Easy</td>
<td>Low</td>
<td>Easy</td>
<td>Easy</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Hard</td>
<td>High</td>
<td>Hard</td>
<td>Hard</td>
<td>120</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>Easy</td>
<td>Low</td>
<td>Easy</td>
<td>Easy</td>
<td>15</td>
<td>0</td>
</tr>
</tbody>
</table>

Results

Figure 5.6 shows the results of the testing. It presents the mean time of execution of each tested task, the first and second time and the acceptable expected results according to the usability objectives previously defined. The vertical axis represents time (in seconds) and on the horizontal axis the number of the tasks.

As expected, in the first time the tasks were executed, the measured time in most cases was slightly superior to the established. In the second try it is clearly visible the time reduction. The conclusions drawn from this study are:

- The UI is easy to memorize and easy to use.

The 8th and 16th tasks were the hardest to execute. The scheduling of a manual recording requires several inputs and took some time until the users understood all the options. Regarding to the 16th task, the video-call, is implemented in an unconventional approach, this presents additional difficulties to the users. In the end all users acknowledge the usefulness of the feature and suggested further development to improve the feature.

In Figure 5.7 it is presented the standard deviation of the execution time of the defined tasks. It is also noticeable the reduction to about half in most tasks from the first to the second time. This shows that the system interface is intuitive and easy to remember.
By the end of the testing sessions, it was delivered to each user a survey to determine their level of satisfaction. These surveys are intended to assess how users feel about the system. The satisfaction is probably the most important and influential element regarding the approval, or not, of the system.

Thus, it was presented to the users who tested the solution a set of statements that would have to be answered quantitatively 1-6, with 1 being “I strongly disagree” and 6 “I totally agree.” The list of questions and statements were:

Table 5.14 presents the average values of the answers given by users for each question. Appendix B details the responses to each question. It should be noted that the average of the given answers is above 5 values, which expresses a great satisfaction by the users during the system test.
Table 5.14: Average scores of the satisfaction questionnaire.

<table>
<thead>
<tr>
<th>Number</th>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In general I am satisfied with the usability of the system.</td>
<td>5.2</td>
</tr>
<tr>
<td>2</td>
<td>I executed the tasks accurately?</td>
<td>5.9</td>
</tr>
<tr>
<td>3</td>
<td>I executed the tasks efficiently?</td>
<td>5.6</td>
</tr>
<tr>
<td>4</td>
<td>I felt comfortable while using the system?</td>
<td>5.5</td>
</tr>
<tr>
<td>5</td>
<td>Each time I made a mistake it was easy to get back on tracks?</td>
<td>5.5</td>
</tr>
<tr>
<td>6</td>
<td>The organization/disposition of the menus is clear?</td>
<td>5.4</td>
</tr>
<tr>
<td>7</td>
<td>The organization/disposition of the buttons/links are easy to understand?</td>
<td>5.4</td>
</tr>
<tr>
<td>8</td>
<td>I understood the usage of every button/link?</td>
<td>5.7</td>
</tr>
<tr>
<td>9</td>
<td>I would like to use the developed system at home?</td>
<td>5.6</td>
</tr>
<tr>
<td>10</td>
<td>Overall, how do I classify the system according to the implemented functionalities and usage?</td>
<td>5.3</td>
</tr>
</tbody>
</table>

5.3.5 Compatibility Tests

Since there are two applications running simultaneously (the server and the client) both have to be evaluated separately.

The server application was developed and designed to run under a Unix based OS. Currently the OS is Linux, distribution Ubuntu 10.04 LTS Desktop Edition, yet other Unix OS that supports the software described in the implementation section should also support the server application.

A huge concern while developing the entire solution, was the support of a large set of Web-Browsers. The developed solution was tested under the latest versions of:

- Firefox version;
- Google Chrome version;
- Chromium;
- Konqueror;
- Epiphany;
- Opera version.

All these Web-Browsers support the developed software with no need for extra add-ons and independently of the used OS. Regarding to MS Internet Explorer and Apple Safari, although the latest versions also support the implemented software, they require the installation of a WebM plug-in in order to display the streamed content. Concerning to other type of devices (e.g., mobile phones or tablets) any device with Android OS 2.3 or later offer full support, see Figure 5.8.

5.4 Conclusions

After thoroughly testing the developed system and after taking into account the satisfaction surveys carried out by the users, it can be concluded that all the established objectives have been achieved.

The set of tests that were conducted show that all tested features meet the usability objectives. Analyzing the execution times for the mean and standard deviation of the tasks (first and second attempt), it can be concluded that the framework interface is easy to learn and easy to memorize.
Regarding the system functionalities the objectives were achieved, some exceeded the expectations, while others still need more work and improvements.

The conducted performance test showed that the computational requirements are high but perfectly feasible with off-the-shelf computers and an usual Internet connection. As expected, the computational requirements do not grow significantly as the number of users grow. Regarding the network bandwidth, the transfer debt is perfectly acceptable with current Internet services.

The codecs evaluation brought some useful guidelines to video re-encoding, although the initial purpose was the video streamed quality. Nevertheless, the results helped in the implementation of other functionalities and to understand how VP8 video codec performed in comparison with the other available formats (e.g., H.264, MPEG4 and MPEG2).
6
Conclusions
6. Conclusions

It was proposed in this dissertation the study of the concepts and technologies used in IPTV, i.e., protocols, audio/video encoding, existent solutions, among others, in order to deepen the knowledge in this area, that is rapidly expanding and evolving, and to develop a solution that would allow users to remotely access their home television service and overcome all existent commercial solutions. Thus this solution offers the following core services:

- **Video Streaming**, allowing real-time reproduction of audio/video acquired from different sources (e.g., TV cards, video cameras, surveillance cameras). The media is constantly received and displayed to the end-user through an active Internet connection.

- **Video Recording**, providing the ability to remotely manage the recording of any source (e.g. a TV show or program) in a storage medium;

- **Video-call**, considering that most TV providers also offer their customers an Internet connection, it can be used together with a web-camera and a microphone, to implement a video-call service.

Based on these requirements, it was developed a framework for a "Multimedia Terminal", using existent open-source software tools. The design of this architecture was based on a client-server model architecture and composed by several layers.

The definition of this architecture has the following advantages: (1) each layer is independent, and (2) adjacent layers communicate through a specific interface. This allows the reduction of conceptual and development complexity and eases maintenance and feature addition and/or modification.

The implementation of the conceived architecture was solely implemented by open-source software and using some Unix native system tools (e.g., cron scheduler [31]).

The developed solution implements the proposed core services: real-time video streaming; video recording and management, and video-call service (even if it is an unconventional approach). The developed framework works under several browsers and devices, as it was one of the main requirements of this work.

The evaluation of the proposed solution consisted in several tests that ensured its functionality and usability. The evaluations produced excellent results, overcoming all the objectives set and usability metrics. The users experience was extremely satisfying as proven by the inquiries carried out at the end of the testing sessions.

In conclusion, it can be said that all the objectives proposed for this work have been met and most of them overcome. The proposed system can compete with existent commercial solutions and because of the usage of open-source software the actual services can be improved by the communities and new features may be incorporated.
6.1 Future work

While the objectives of the thesis was achieved, some features can still be improved. Below it is presented a list of activities to be developed in order to reinforce and improve the concepts and features of the actual framework.

Video-Call

Some future work should be considered regarding the Video-Call functionality. Currently, the users have to setup the audio&video streaming using the Flumotion tool, and after creating the streaming they have to share through other means (e.g., e-mail or instant message) the URL address. This feature may be overcome by incorporating a chat service, allowing the users to chat between them and provide the URL for the video-call. Another solution is to implement a video-call based on video-call protocols. Some of the protocols that may be considered are:

Session Initiation Protocol, SIP [78] [103] – is an IETF-defined signaling protocol widely used for controlling communication sessions such as voice and video calls over Internet Protocol. The protocol can be used for creating, modifying and terminating two-party (unicast) or multiparty (multicast) sessions. Sessions may consist of one or several media streams.

H.323 [80] [83] – is a recommendation from the ITU Telecommunication Standardization Sector (ITU-T) that defines the protocols to provide audio-visual communication sessions on any packet network. The H.323 standard addresses call signaling and control, multimedia transport and control, and bandwidth control for point-to-point and multi-point conferences.

Some of the possible frameworks that may be used and which implement the described protocols are:

openH323 [61] – the project had as goal the development of a full featured, open source implementation of the H.323 Voice over IP protocol. The code was written in C++ and supports a broad subset of the H.323 protocol.

Open Phone Abstraction Library, OPAL [48] – is a continuation of the open source openh323 project to support a wide range of commonly used protocols used to send voice, video and fax data over IP networks rather than being tied to the H.323 protocol. OPAL supports H.323 and SIP protocol, it is written in C++ and utilises the PTLib portable library that allows OPAL to run on a variety of platforms including Unix/Linux/BSD, MacOSX, Windows, Windows mobile and embedded systems.

H.323 Plus [60] – is a framework that evolves from OpenH323 and aims to implement the H.323 protocol exactly as described in the standard. This framework provides a set of base classes (API) that helps the application developer of video conferencing build their projects.
6. Conclusions

Described some of the existent protocols and frameworks, it is necessary to conduct a deeper analysis to better understand which protocol and framework is more suitable for this feature.

SSL security in the framework

The current implementation of the authentication in the developed solution is done through HTTP. The vulnerabilities of this approach are that the username and passwords are passed in plain text; it allows packet sniffers to capture the credentials; and each time the the user requests something from the terminal the session cookie is also passed in plain text.

To overcome this issue the latest version of RoR 3.1, natively offers SSL support, meaning that porting the solution from the current version, 3.0.3, into the latest will solve this issue (additionally some modifications should be done to Devise to ensure SSL usage [59]).

Usability in small screens

Currently the developed framework layout is set for larger screens. Although, being accessible from any device it can be difficult to view the entire solution on smaller screens, e.g., mobilephones or small tablets. It should be created a light version of the interface, offering all the functionalities, but rearranged and optimized for small screens.
Bibliography


Bibliography


Bibliography


[https://github.com/hassox/warden](https://github.com/hassox/warden)


International Workshop on Acoustics and Video Coding and Communication.


Bibliography


Appendix A - Evaluation tables
<table>
<thead>
<tr>
<th>User</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>22</td>
<td>12</td>
<td>57</td>
<td>55</td>
<td>114</td>
<td>1</td>
<td>56</td>
<td>53</td>
<td>55</td>
<td>27</td>
<td>27</td>
<td>17</td>
<td>18</td>
<td>105</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>1</td>
<td>18</td>
<td>0</td>
<td>20</td>
<td>1</td>
<td>70</td>
<td>1</td>
<td>68</td>
<td>2</td>
<td>140</td>
<td>3</td>
<td>69</td>
<td>1</td>
<td>66</td>
<td>1</td>
<td>34</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>62</td>
<td>0</td>
<td>60</td>
<td>1</td>
<td>120</td>
<td>0</td>
<td>61</td>
<td>0</td>
<td>58</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>54</td>
<td>0</td>
<td>52</td>
<td>0</td>
<td>104</td>
<td>0</td>
<td>53</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>52</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>54</td>
<td>0</td>
<td>52</td>
<td>0</td>
<td>104</td>
<td>1</td>
<td>53</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>52</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>66</td>
<td>1</td>
<td>64</td>
<td>0</td>
<td>128</td>
<td>0</td>
<td>65</td>
<td>0</td>
<td>62</td>
</tr>
<tr>
<td>7</td>
<td>22</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>25</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>60</td>
<td>1</td>
<td>58</td>
<td>0</td>
<td>120</td>
<td>2</td>
<td>59</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>58</td>
<td>0</td>
<td>56</td>
<td>1</td>
<td>116</td>
<td>1</td>
<td>57</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>53</td>
<td>1</td>
<td>51</td>
<td>0</td>
<td>106</td>
<td>0</td>
<td>52</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>53</td>
<td>0</td>
<td>51</td>
<td>0</td>
<td>106</td>
<td>0</td>
<td>52</td>
</tr>
<tr>
<td>11</td>
<td>22</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>40</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td>93</td>
<td>2</td>
<td>91</td>
<td>1</td>
<td>186</td>
<td>2</td>
<td>91</td>
</tr>
<tr>
<td>12</td>
<td>27</td>
<td>1</td>
<td>26</td>
<td>0</td>
<td>25</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>113</td>
<td>2</td>
<td>111</td>
<td>2</td>
<td>226</td>
<td>3</td>
<td>112</td>
</tr>
<tr>
<td>13</td>
<td>17</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>73</td>
<td>1</td>
<td>71</td>
<td>0</td>
<td>146</td>
<td>1</td>
<td>72</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>65</td>
<td>0</td>
<td>63</td>
<td>0</td>
<td>130</td>
<td>2</td>
<td>64</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>53</td>
<td>1</td>
<td>51</td>
<td>0</td>
<td>106</td>
<td>0</td>
<td>52</td>
</tr>
<tr>
<td>16</td>
<td>15</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>65</td>
<td>0</td>
<td>63</td>
<td>0</td>
<td>130</td>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td>17</td>
<td>14</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>61</td>
<td>0</td>
<td>59</td>
<td>0</td>
<td>122</td>
<td>3</td>
<td>60</td>
</tr>
<tr>
<td>18</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>18</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>49</td>
<td>0</td>
<td>47</td>
<td>0</td>
<td>130</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>19</td>
<td>14</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>65</td>
<td>1</td>
<td>63</td>
<td>0</td>
<td>126</td>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>61</td>
<td>0</td>
<td>59</td>
<td>0</td>
<td>118</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>21</td>
<td>11</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>53</td>
<td>0</td>
<td>60</td>
<td>0</td>
<td>120</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>22</td>
<td>13</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>61</td>
<td>1</td>
<td>59</td>
<td>0</td>
<td>118</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>23</td>
<td>16</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>60</td>
<td>0</td>
<td>58</td>
<td>0</td>
<td>120</td>
<td>2</td>
<td>59</td>
</tr>
<tr>
<td>24</td>
<td>15</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>65</td>
<td>0</td>
<td>63</td>
<td>0</td>
<td>126</td>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td>25</td>
<td>15</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>65</td>
<td>0</td>
<td>63</td>
<td>0</td>
<td>126</td>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>26</td>
<td>14</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>61</td>
<td>0</td>
<td>59</td>
<td>0</td>
<td>118</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>27</td>
<td>12</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>53</td>
<td>0</td>
<td>51</td>
<td>0</td>
<td>102</td>
<td>1</td>
<td>52</td>
</tr>
<tr>
<td>28</td>
<td>13</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>57</td>
<td>1</td>
<td>55</td>
<td>0</td>
<td>114</td>
<td>2</td>
<td>56</td>
</tr>
<tr>
<td>29</td>
<td>17</td>
<td>0</td>
<td>14</td>
<td>1</td>
<td>16</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>62</td>
<td>0</td>
<td>60</td>
<td>1</td>
<td>124</td>
<td>2</td>
<td>61</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
<td>1</td>
<td>18</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>70</td>
<td>2</td>
<td>68</td>
<td>2</td>
<td>143</td>
<td>3</td>
<td>69</td>
</tr>
</tbody>
</table>

**Figure A.1:** Execution times and errors for each user on each task (1st Time).
Figure A.2: Execution times and errors for each user on each task (2nd Time).
A. Appendix A - Evaluation tables
Appendix B - Users characterization and satisfaction results
B. Appendix B - Users characterization and satisfaction results

1. Age

![Age Distribution](image)

Figure B.1: General Question 1. Users age.

2. Sex

![Sex Distribution](image)

Figure B.2: General Question 2. Users sex.
3. Occupation

Figure B.3: General Question 3. Users occupation.

4. Computer Expertise

Figure B.4: General Question 4. Users computer expertise.
B. Appendix B - Users characterization and satisfaction results

Figure B.5: Evaluation 1. No geral estou satisfeito com a facilidade de utilização do sistema.

Figure B.6: Evaluation 2. Consegui executar todas as tarefas eficazmente utilizando o sistema.
Figure B.7: Evaluation 3. Consegui executar todas as tarefas rapidamente.

Figure B.8: Evaluation 4. Senti-me confortável a utilizar o sistema.
B. Appendix B - Users characterization and satisfaction results

Figure B.9: Evaluation 5. Sempre que encontrei erros consegui recuperar e corrigi-los facilmente.

Figure B.10: Evaluation 6. A organização dos menus e itens é clara.
Figure B.11: Evaluation 7. A organização dos botões é clara.

Figure B.12: Evaluation 8. Percebi imediatamente qual a função de cada botão.
Figure B.13: Evaluation 9. Gostaria de utilizar este sistema em casa.

Figure B.14: Evaluation 10. Qual a nota final que daria ao sistema, em termos da sua funcionalidade e facilidade de Utilização?
Appendix D - Codecs Assessment: VP8 vs H.264
Figure C.1: Movies - Encoding quality comparison between VP8 and H.264 high profile (source [69]).
Figure C.2: HDTV Movies - Encoding quality comparison between VP8 and H.264 high profile (source [69]).