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

Abstract: This paper presents a multimedia platform for digital television based on existing single purpose open-source solutions. The advantage of the proposed platform regarding existing ones is the possibility to integrate new features or change existing ones (e.g., recording format and coding standard). In this sense, the proposed terminal 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.

1 INTRODUCTION

Nowadays, the commercial trend for multimedia terminals are all-in-one solutions that are able to display real-time TV shows, record TV programs and manage other multimedia contents. However, the latest advances in multimedia services will soon integrate other features, such as video-call or home surveillance, which will require more customizable platforms, where the user has the possibility to add new features.

Some platforms for the distribution of multimedia contents already exist. Example of such platforms include: commercial solutions, such as Microsoft Mediaroom [17], do not allow any change due to proprietary rights. Example of open-source solutions include Linux TV [15], which may have a large support for devices and formats, but require programming knowledge by the end-user, making it hard to customize or change.

In contrast, the platform that is proposed allows different inputs, from satellite broadcasts to video-camera or surveillance-cameras. The produced content is handled by an encoding and streaming server, allowing different types of devices to access the service with no extra costs for the user other than the Internet connection. Moreover, its strict personal usage is assured by convenient security (using SSL/TLS protocols) and user authentication mechanisms.

To summarize, the proposed system is characterized by offering support for:

- input devices: satellite, video or radio antenna, cable TV, IPTV;
- output devices: TV, computer, laptops, mobile devices, Hard Disk Drive (HDD) (for recording purpose), others (e.g. tablets ) through a local connection, LAN and/or Internet.

Furthermore, it supports the following core services:

- **TV Streaming** – after acquiring the TV signal, it may be displayed in real-time or even displayed while recording. This reproduction facility of the audio/video 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.

- **TV Recording** – this basic functionality provides the ability to remotely manage the recording of any TV show or program that, by some reason, cannot be seen at that time.

- **Video-call** – since the audio/video acquisition is implemented for the TV Streaming functionality, setting it up for a web-cam and microphone represents a small change in the system. This way, the video-call can also be offered as a service.
2 RELATED WORK

Several tools that implement the previously presented features already exist independently, but with no connectivity between them. The main difference between the developed platform and the tools already developed is that this framework integrates several independent solutions into it. Other differences are:

- Some proprietary solutions cannot be modified without violating intellectual property.
- Many software tools have a complex interface and are suitable only for experienced users or users with some programming knowledge. In some cases, this is due to the fact that these tools support many more features and configuration parameters than what is expected in an all-in-one multimedia solution.
- Some TV applications cover only DVB, while 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.

A new emergent area is IPTV (Internet Protocol Television), with several solutions being developed in a daily basis, but all related to IP solutions. It is the case of a Personal TV framework presented in [1], where the main goal is the design of a Framework for Personal TV for personalized services over IP.

The solution presented in this paper differs from the Personal TV [1] in several aspects. The solution here in presented is: implemented based on existent open-source solutions; intended to be easily modifiable; aggregate several multimedia functionalities, such as video-call, recording content; and to serve the user with several different multimedia video formats (the streamed video is done in WebM format, but it is possible to download the recorded content in different video formats, after requesting the platform to re-encode the content).

In the following, a set of existing platforms is presented. It should be noted the existence of other small applications just for video display (e.g., TV players such as Xawtv [6]) However, in comparison with the proposed application, there is no solution that covers all the features offered by the proposed solution.

Commercial Solutions Several commercial solutions exist, but none is legally modifiable. GoTV [9] is a proprietary and paid software tool, which offers TV viewing specifically to mobile-devices (e.g., Android, iPhone, ...) and only works in the USA. Microsoft MediaRoom [16] is a proprietary and paid IPTV service available to television providers. Many providers use this software (including Portuguese MEO and Vodafone) and it is also accessible through a large set of devices (personal computer, mobile devices, TV’s, Microsoft XBox360). GoogleTV [7] is an IPTV service for Android based systems. It is an all-in-one solution, allows developers to add new features, through the Android Market, and currently works only for some selected Sony televisions and Sony Set-Top boxes. NDS MediaHighway [20] is a platform adopted worldwide by many Set-Top boxes (e.g., Portuguese Zon provider). The difference between NDS and MediaRoom is that NDS supports DVB (terrestrial, satellite and hybrid), while MediaRoom does not.

All of the above commercial solutions have similar functionalities and charge their usage. However, some support a great number of devices (MS MediaRoom), and some are specialized in one kind of device, e.g. GoTV - mobile devices. None of the mentioned commercial solutions offer support for video-conference, either as an add-on or within the normal service.

Free/open-source software Linux TV [15] is a repository for several tools that offers a vast set of support for several kinds of TV Cards and broadcasts methods. By using the Video for Linux driver (V4L) [2], it is possible to watch 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 solutions. Video Disk Recorder (VDR) [14] is another open-solution for DVB, with the common options (regular playback, recording and video edition). However, it requires some programming knowledge. Kastor! TV (K!TV) [22] is an open solution for MS Windows to view and record TV content from a video card. MythTV [19] is a free open-source software for digital video recording (DVR). It has a vast support and development team, and any user can modify/customize it with no fee.

In general, the existent open-source software offer similar functionalities in comparison with the proposed solution. The major restrictions of using these solutions are: the user needs some programming knowledge (Linux TV); the acquired content can only be viewed in the machine where the signal is acquired (VDR); and other solutions (MythTV) offer the possibility for remote usage, but the remote user needs specific software installed in order to properly view the remote content.
3 ARCHITECTURE

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. 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.

3.1 Server Side

As shown in Fig. 1, the server is composed of six main modules:

- **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 Streamer Engine (VSE)**, which streams the encoded video through the Internet connection;
- **Scheduler**, responsible for managing multimedia recordings;
- **Video Recorder Engine (VRE)**, which records the video into the local hard drive, for posterior visualization, download or re-encoding;
- **Video-Call Module (VMC)**, which streams the audio/video acquired from the web-cam and microphone.

Using a bottom-up approach, the server side modules, presented in Fig. 1 will now be described in the next subsection.

3.1.1 Signal Acquisition And Control

The SAAC Module is responsible for the signal acquisition and control. Video/audio signal can be acquired from multiple Hardware (HW) sources (e.g., TV card, surveillance camera, web-cam and microphone, DVD). It can also be acquired in different formats. Thus, the top modules should not be concerned about how the information is provided/encoded. Thus, the SAAC Module is responsible for providing a standardized mean for the upper modules to read the acquired information.

3.1.2 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.

The Audio & Video Encoder Modules are used to compress/decompress the multimedia signals being acquired. 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. Both audio and video encoder modules should be implemented separately, in order to easily allow the integration of future audio or video codecs into the system.

The Profiler is the module that specifies the parameters for audio and video encoding. This module is represented as independent unit, since it could be integrated into the database.

3.1.3 Video Streamer Engine

The VSE component is responsible for streaming the captured audio/video data provided by the SAAC
module and for streaming any content previously recorded. It may also stream the web-camera data, when the video-call scenario is considered.

3.1.4 Video Recorder Engine

The VRE is the unit responsible for recording audio/video data coming from the available source. 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, according to the Scheduler status, the system needs to decide if it is an acceptable recording or not. Finally, it tunes the required channel and starts the recording with the defined quality level.

3.1.5 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. Consider that the system would have to acquire multiple video signals at the same time, with only one TV card (multiple recordings at the same time but with different channels). These behavior should not be allowed, because it will lead to unexpected/undesired results, while using only one TV Card. However, this might occur if the system had several input devices. In order to prevent this undesired situations, a set of policies have to be defined in the Scheduler. Those polices may be: the first recording dictates the remaining, it is not possible to record multiple sources (from the same TV Card) at the same time. These policies may be defined in a bash file created for the effect.

3.1.6 Video-Call Module

The process that implements this feature is very similar to the process that implements the audio and video streaming feature. The difference is the HW where the signals are acquired. Hence, VCM is composed of: a module for video and audio acquisition, like SAAC; a module for audio and video encoding according to the defined profile (Encoding Engine); and a module to transmit the encoded stream (VSE).

3.2 Client Side

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**, to acquire the local video+audio and stream it to the corresponding recipient.

3.2.1 Browser and required plug-ins

The base software that implements this solution is a Web Browser and, if necessary, a plug-in for audio/video display. One of the main concerns is to support as many Web Browsers as possible, provided that there are plug-ins available for a proper functioning.
3.2.2 Video-Call Module

In order for the client to support the VCM, it is necessary the installation of some software (equivalent to the server-side) for acquisition, encoding and transmission of the local web-cam and microphone sources. The operation mode is the same as described in the server side VCM.

4 IMPLEMENTATION

The whole system was implemented over Linux Ubuntu operating system, using a set of open-source packages for the multimedia services. The installed open-source software packages were: GStreamer [10] core and its base, good, ugly and bad plug-ins to add support to Flumotion [5]; libvpx, to add support to WebM’s VP8 [26, 12] video format; Flumotion [5]; Ruby on Rails (RoR) Framework [3]; XMLTV [24, 18]; the latest version of a web browser; and, for data management, it was used a SQLite data-base, which implements a self-contained, server-less, zero-configuration, transactional SQL database engine.

4.1 User Interface and Authentication

The User Interface (UI) module was developed using the RoR Framework [3], which is an open-source web application development framework, that allows agile development methodologies. The UI is accessible through a browser with support to HTML5 (e.g., latest versions of Firefox and Chrome), in order to allow the displaying of the streaming content.

The user authentication was implemented using Devise [21], a flexible, easy to configure and manage authentication solution that is based on Warden [11].

4.2 Audio&Video acquisition, encoding and broadcasting

The audio and video acquisition (SAAC) is implemented by Flumotion, which acquires the signals from the available HW. It also makes use of several Bash Scripts that tune the HW (TV card) to the selected channel.

The Encoding Engine module is implemented by Flumotion. Flumotion is based in the concept of having one (or more) manager(s), where the tasks are defined, and one (or more) worker(s), associated to the defined tasks. This way, the manager is responsible for managing the workers (i.e., start and stop). Both manager and workers are defined in a XML document, by following a structure defined by Flumotion software. To implement the Encoding Engine, the following tasks were defined in Flumotion:

- **Producer**, responsible for producing stream data (usually in a raw format) and to feed it to other components.
- **Consumer**, which consumes the 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.
- **Converter**, converts stream data. It can: encode or decode a feed; combine feeds to make a new feed (e.g., mux audio and video feeds); change the feed by changing the content, (e.g., take a master feed and a backup feed and if the master feed stops supplying data then it will output the backup feed), etc.

Thus, the producer component acquires the data from the HW, passes the data to a consumer component for encoding purposes (the video data may go to a converter component if it is necessary to scale down the frame) and then passes the stream to the consumer component responsible for the muxing. The audio stream is encoded with Vorbis codec and the video is encoded with VP8 code. It is at this stage that the profiles are set. Acquired video in a large format (4CIF), can be scaled down to CIF and QCIF, encoded to VP8 with different parameters according to the desired quality. After having the two streams in the desired format (Vorbis for audio and VP8 for video) they are muxed into the WebM container and streamed through Real Time Protocol (RTP) to the web.

Mapping the described implementation into the architecture in Fig. it results that Flumotion covers the SAAC module, the Encoding Engine and the VSE module. The management of the Flumotion manager and workers is done by Bash Scripts.

There is still one manager where all the described tasks are defined (signal acquisition, encoding and broadcasting), associated to different workers. The need for several workers is because it simplifies the management process in the scripts. When a request to change channel is made, the following procedure is done:

1. the system verifies if the user that requested the change has permission for this action. The permission is currently based on who was first using the system, but it can be added special permissions to users;
2. assuming that the user can change the channel, the worker which was acquiring the video stream from the HW is terminated;
3. the script to change the channel is invoked with the channel code. The codes are defined in the database and were acquired using the XawTV software;

4. the video acquisition worker is launched again;

5. the web page is reloaded and the new channel content is displayed.

To control the workers and manager, a file is created when they are launched to keep track of their PID. This is useful for restarting and changing the channel.

4.3 Recording Management

The VRE is also implemented by Flumotion software. The streamed video and audio can be recorded to disk by a consumer component. This task is defined in the manager XML file and associated to a worker.

To evaluate if a recording is valid, it has to pass the scheduler tests. The scheduler has a set of rules to evaluate a request. Those rules are:

a) there cannot be two simultaneous recordings at the same time in different channels (unless there are two or more TV cards available);

b) it is possible to record subsets of a previous defined recording;

c) while a recording is ongoing, it is impossible to change the channel;

d) the recording always has the highest priority, meaning that if there is a user watching a different channel prior to the recording, when the recording starts the channel will be changed to the one defined in the recording; this ensures a first come, first serve priority system.

The implementation of these rules is done by ruby scripts, which simplify this task. When a recording is classified as valid, the request is inserted into the database and a job is added to the Linux Cron table through the at command. When a scheduled job starts, the following procedure is done:

1) verify if some recording is ongoing (check the PID file for the recording worker);

2) if not:
   • change to the channel defined in the recording (if necessary);
   • launch the recorded worker;
   • wait until the recording time ends.

If a recording is ongoing, then:

3) when the recording time ends, there are two different scenarios:

   • there is no more recordings in progress: the recorder worker is terminated, the resulting file is renamed, copied to the videos folder, and added to the database. At this time the user will be able to view the recording;

   • there are other recordings in progress: a subset of the file content is copied to the video folder using the FFmpeg tool [4] (the start time and end time are passed as parameters to get the subset) and the video is added to the database.

The recorded content, at the server, may then be downloaded. The original format of the recorded file is VP8+Vorbis in a WebM container (it is the streamed format and this way no more extra complexity is added to the recording process). However, a re-encoding option is available. The user may transcode the resultant file into H.264 video format and ACC audio format, or even add other formats. This functionality, media manipulation and coding blocks, is implemented using the FFmpeg tools.

4.4 Video-Call

Considering its personal use nature, with a strict peer-to-peer topology, the video-call module was implemented by reusing the Flumotion streaming server to broadcast the audio/video acquired from the local web-camera and microphone. This feature allows two or more users to share the streamed content between them. For this, each user needs the Flumotion software installed. After running the Flumotion Server and setting up the stream of the local web-cam and microphone, the application gives an URL which should be exchanged between the users and inserted into the fields. Local and Remote, in the Multimedia Terminal web-page. After the insertion of the two URL’s, the terminal presents in the same window the local and remote users, like a traditional video-call. This is a rudimentary solution which requires some future work. Future extensions are the integration of this feature with existent messaging programs as well as the encoding using a video-call protocol, such as Session Initiation Protocol (SIP) [25] and H.323 [13].

4.5 Programming Guide

A feature that was also implemented to add extra functionalities to the developed software was the addiction of a Electronic Programming Guide (EPG) [8]. The EPG was implemented using the XMLTV software [25], which connects to a Internet host and
retrieves several XML files (XMLTV [18]), one for each channel available in the country, Portugal (several countries are available). This information is used for displaying to the user the current and next show at the viewed channel and it also allows the user to set recordings while viewing the show list for a selected channel.

5 EVALUATION

This section presents the evaluation of the developed solution according to three different perspectives: (i) supported devices and Operating Systems (OS); (ii) resources usage; (iii) user usability and modifiability.

5.1 Supported Devices and OS

The server, where the software resides, needs to be a Unix based OS. Currently the OS where the server is running is Ubuntu 10.04 LTS Desktop Edition, but any other Unix OS that supports the software described in the implementation section should also support the developed software.

For the user interaction, the solution was tested under Firefox, Google Chrome, Chromium, Konqueror, Epiphany and Opera (latest versions). All of these Web-Browsers support the developed software with no need for extra add-ons. Regarding MS Internet Explorer and Apple Safari, the latest versions also support the software. However, they require the installation of a WebM plug-in in order to display the streamed content. Regarding the user interaction, any device with Android OS 2.3 or later, should offer full support. The user interface is represented in Fig. 2.

5.2 Resources Usage

The CPU computational load is mainly due to the audio&video acquisition, scaling, encoding and broadcasting. The server where this evaluation was conducted, was a Dual Core AMD Opteron Processor 170, where the two cores were used during the process, and with 2 GiB of memory. The CPU usage was measured in three different stages: no clients, one client and ten clients (being this solution for personal usage, ten people represent the size of a ten member family, which is uncommon) for the medium quality (video bit-rate 400 kbit/s and audio bit-rate 64kbit/s). The results are the following:

- with no clients, the total CPU usage was around 33%;
- with one client, the total CPU usage was 33.5%;
- with ten clients, the total CPU usage was 38.5%.

5.3 Usability and Modifiability

After inquiring several users about the usability of the developed solution (this analysis was conducted with several families, one family with six members, three with 4 and 12 singles), the average of the obtained results was that the user interface was easy to use (90% of a universe of 30 people), the recording functionality was also easy and intuitive to use (87%) and the video-call was clear about how to use after a brief explanation (70%). However, the current implementation was not the most suitable for the current user’s (video-call). The improvements for the video-call are described in the Conclusions section.

Regarding the user modifiability, the conclusions were made after explaining to the users how the system was designed and how RoR works and the reaction to the developed solution was:

- in order to easily perform modifications, it is required a user modification manual, with the implementation details;
- the users agreed that the usage of the RoR good practices were very useful (e.g., using obvious names for the functions, using clear and intuitive names for the views and controllers, the organization of the project, ...) for further development;
- users with small knowledge about programming language considered that the used programming language was quite easy to understand and with the developed features it would be easy to add other features by simply editing the existent code (RoR, bash scripts and the XML description of the Flumotion server).

Figure 2: User streaming interface.
As an example, to add video a surveillance feature the modifications would be: replicate the existent manager XML code, edit the video acquisition source, and workers name and used ports; reuse the video streaming web-page, edit the streaming URL, and add a link to the new video surveillance feature page in the global menu.

6 CONCLUSIONS

The proposed application comes with some base features, such as: TV streaming, TV recording and Video-Call. Nevertheless, its modular structure was designed to easily support other features (e.g. video surveillance services). The architecture is based in a client-server model. The server application lies in a Linux platform, where the signal is acquired, being possible to selected several sources (e.g., TV card, web-cam, microphone, surveillance cameras, etc.) and it can be accessed remotely, using the Internet. Due to the incorporated access control and authentication mechanisms this application can be used in a multi-user environment, with an easy and intuitive user interface, allowing an experienced user to modify any aspect he needs to edit, either by programming or by editing the configurations or the provided features.

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, instant message,...) the URL address. This feature may be overcome by incorporating a chat service, allowing the users to chat between them and provid the mean to share the URL for the video-call. Another solution is to implement a video-call based on protocols such as SIP signaling protocol, widely used for controlling communication sessions, such as voice and video calls over Internet Protocol (IP); or H.323 standard, which addresses call signaling and control, multimedia transport and control, and bandwidth control for point-to-point and multi-point conferences.

REFERENCES