Multimedia Source Management for Remote and Local Edition Environments

Alejandro González, Ana Cerezo, David Jiménez, and José Manuel Menéndez

Grupo de Aplicación de Telecomunicaciones Visuales, Universidad Politécnica de Madrid, Avda. Complutense. 30, 28040 Madrid, Spain
{agn,ace,djb,jmm}@gatv.ssr.upm.es

Abstract. This paper aims to detail an innovative multimedia edition system. A special functionality provides this solution with different ways of managing audiovisual sources. It has been specially designed for media centralized environments dealing with large files and groups of users that access contents simultaneously for editing and composing. Mass media headquarters or user communities can take advantage of the two working modes, which allow online and offline workflows. The application provides a user edition interface, audiovisual processing and encoding based on GPL tools, communication via SOAP between client and server, independent and portable edition capabilities and easy adaptable source handling system for different technologies.

Keywords: Source Management, Multimedia Edition, Centralized Data, Multiple Users, SOAP, XML.

1 Introduction

The full networking development of work environments leads to the necessity of content digitalization and ubiquitous access to it. The combination of both aspects reveals new requirements. Large capacity for data storage is immediately recognized as one of the most urgent necessities, so new resources management models for optimization are needed. This requirement is a key factor in environments that need to manage large amount of data, like mass media, publishing and advertising companies.

Daily, these companies are constantly generating multimedia data and its associated metadata with pending edition works. Even if that information is not used in shortly, it could be necessary to stored it for a possible future use.

Furthermore, the storage of these contents is often centralized in data servers and it has to be easily accessible and editable from each user terminal. This exchange of files involves a great computational load that could saturate both the edition terminals (which capacities are meaningful lower than servers) and the servers (if there are many file requests simultaneously), so this implies a new problem.

Consequently, it is necessary to change the way the audiovisual content is edited. This work expounds a software solution based on a client-server model for multimedia edition avoiding long download times and freeing resources for a more efficient system.

2 Functionalities

The main object of this work is to define a source management model for multimedia content within a new application for multimedia edition based on tools under a GPL license. The features covered by this kind of software are related to multimedia basic editing and encoding. The system functionalities can be classified in two categories: source management and multimedia edition.

2.1 Source Management

The application is based on a client-server model. It allows multimedia edition on files not necessarily stored in the client computer, but also in a server. Two source management work modes are allowed, according to the user necessities:

Offline Mode. It is the standard working mode. The importing module downloads low quality sources from the server catalog when they are requested. The multimedia files are stored in the client computer and no connection is required between client and server for the editing process. When the edition is over, its parameters are sent to the server in an XML[1] file. The edition is done using low resolution files within the client and the server repeats it with the high quality media once the process is over.

Online Mode. This mode is aimed to permanently connected environments, such as local networks. The importing module does not download any source file. Instead of this, it saves a reference to each imported multimedia event. This mode has a stronger bandwidth requirement, as the server core has to generate previews that are sent back to the client via streaming during the whole edition process.

2.2 Sources Origin and Prosumer Capabilities

The system supports several kinds of sources attending to their origin.

FTP Catalog. An online multimedia files set which is actually the main source provider for the editor. The catalog can be either private or public.

HTTP. Multimedia files available through the web can be imported to the projects directly, and referenced to the server as well in order to include external content.

Personal Content. It is also possible for the user to upload personal content to the FTP server. The user becomes then a “prosumer”, content consumer and producer at the same time. The access to the uploaded content is also configurable, so the system orientation allows many different points of view, including editor communities for content sharing.

2.3 Multimedia Edition

As the main target of the application is to solve multimedia source management, its audiovisual edition capabilities are supported by some tools under a GPL license. Two external applications are used in order to satisfy these functionalities: AviSynth and MEncoder. AviSynth [2] is a *frameserver* capable of processing local multimedia files in real time. The user develops custom scripts using basic functions in order to edit the source content. In the other hand, MEncoder [3] provides the final solution with its encoding capabilities. It manages several codecs and containers, and accepts AviSynth scripts as input stream. MEncoder's software package also includes MPlayer, a multimedia player that is used in this application for the preview function.

A custom Java Swing user interface has been developed to allow easy multimedia edition. It generates project structures that will be translated either into an XML file or into an AviSynth script. Thus, the XML file can be sent to the server for high quality processing and the script can be executed for local previewing of the project. The complete list of its functionalities is explained in section 3.3.

3 Architecture

In order to support the functionalities previously explained, different structures have been defined and created. In this chapter, these structures and the communication between them are explained.

A general scheme of the work is shown in Fig. 1. As the diagram shows, the process implies several connections between client and server. The server, or an external media server, contains two versions of the sources: the master in original quality and another in low quality. The importing module either downloads or references (depending on the work mode) the desired multimedia events from the light sources catalog. This can be done using several communication protocols, such as http or ftp.

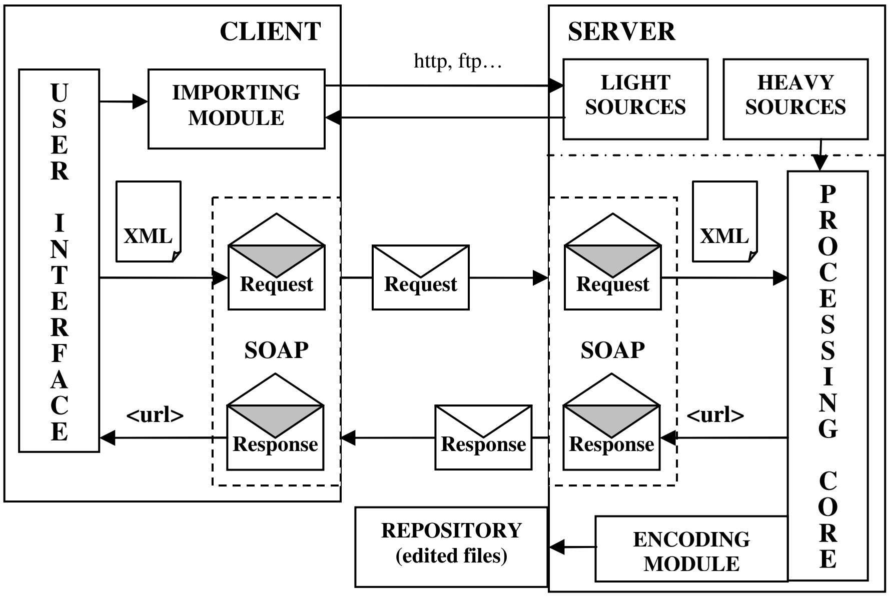

Fig. 1. General architecture scheme

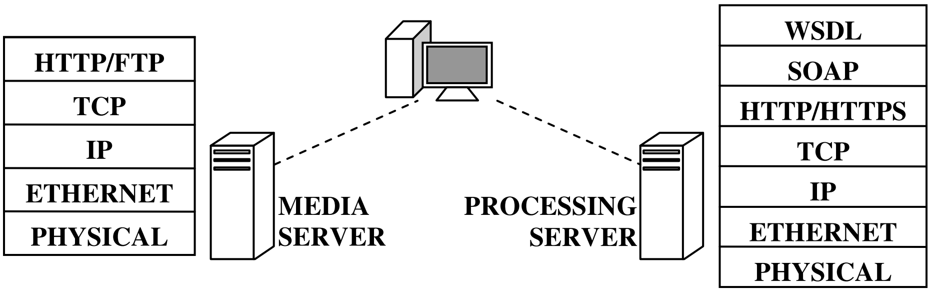

Fig. 2. Protocol stacks for media download and edition processing connections

An edition project can be built graphically by the user using the interface. Once the project is completed, the resulting XML edition file is enveloped following the SOAP standard [4] and sent as a Request message to the server. The server extracts the XML and processes its instructions using the original quality videos. An internal Java application automatically builds an AviSynth script following the directions established by the client in the XML file. Afterwards, this script's output stream is encoded by MEncoder with the extracted parameters. The resulting stream is encoded and stored in a repository that could be a public site, a private shared folder or any other accessible target. A SOAP Response message containing the URL to the generated file is sent back to the client.

SOAP is used due to its portability. In conjunction with the project structure definition, it allows alternative client solutions to work properly even if they are implemented in different programming languages. In addition, SOAP is based on XML, as the edition instructions file, so the application is easily adapted to this standard. The system protocol stacks are shown in Fig. 2.

3.1 The Processing Core

The multimedia processing core is in charge of extracting the XML parameters and building a valid AviSynth script that could be played or encoded. The whole work has been implemented using Java. The schema in Fig. 3 shows how it works.

The core's input is an XML file with the editing and encoding parameters. These parameters are extracted using JDOM [5], a Java library for handling XML structures.

The project builder uses the extracted parameters to build a Java structure of classes that defines each aspect of the edition. The project structure stands by itself and it is independent from AviSynth or any other external application used in this work. This means that a technological change on the multimedia processing within the application does not imply any change in the elements on the figure's left column.

The project structure is processed in order to write an AviSynth script that is ready to be played or encoded. Depending on how the processing core is accessed to, MPlayer or MEncoder is launched using the parameters specified in the XML file.

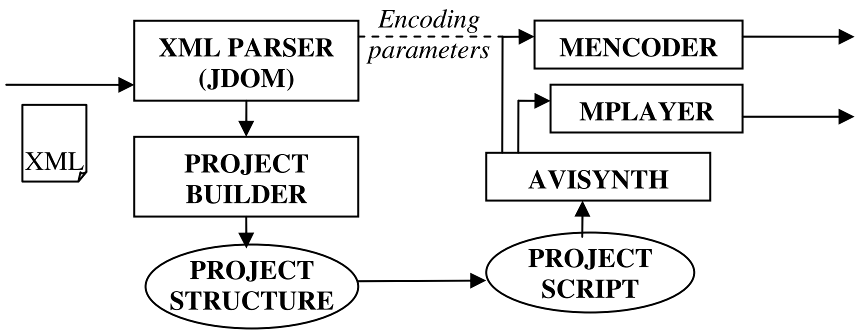

Fig. 3. The multimedia processing

3.2 The Project Structure

In this section, the structure used to define an edition project is described. It is based in parental relations easily exportable to an XML document. These relations are shown in Fig. 4 and they are shared by the implemented XML and Java structure.

The main node is *Project*, which contains the essential video and audio parameters that will be inherited by every node. These parameters are: project name, frames per second, video resolution, audio sampling rate and color system. Each new event added to the project will be treated in order to match these parameters. This treatment is included in the AviSynth script generation process and is supported by customized scripts. The most notable custom script is the *Autofit* function that, if used, matches different video sizes taking up the most of the available screen blank space without affecting on quality or aspect relation. A *Project* is made up of audio and video

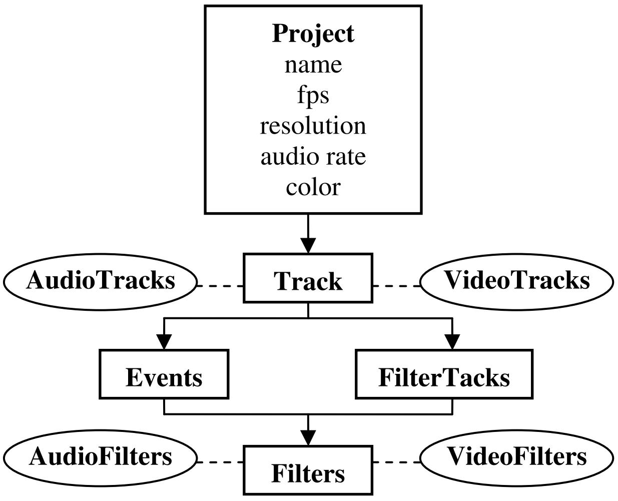

Fig. 4. The project structure

Tracks, which are independently configurable. Each *Track* is filled with *Events*, i.e., audio and video fragments distributed through time. These fragments can be modified trimming its content, changing its position in time or applying independent filters. These *Filters* were previously explained in this paper and can be applied directly to events, to certain tracks, to the final output, or by using *FilterTracks*, which allow the user to modify filter parameters through time.

As we introduced at the beginning of this section, the project structure is always exportable to an XML document or to an AviSynth script. The XML feature makes possible to implement an easy and fast *save* system, to recover a previously started project; and an *undo* system, in order to fix some undesired changes during the edition process. In the other hand, the AviSynth feature makes possible the Preview functionality, which is based in playing the resulting script.

3.3 The User Interface

The user interface's design and functionalities are based on professional edition tools, and provides the user with friendly commands that support a main part of the editing needs.

It may be important to emphasize the necessity for a friendly user interface. The final user is not going to be a computers' expert, nor an engineer or a scientist. Actually, the typical user will be an editor with experience using this kind of software and, for that reason, it is important to preserve the way of working developed by the industry standard edition tools, and avoid compromising it because of the different source management technology used by this system.

The interface includes the following features:

File Importing Module. An embedded window allows the user to explore the accessible sources and select the desired multimedia event to be included in the project. Accessible sources include web content, private and public FTP content and personal sources. Another tab in the module shows the already added sources with a snapshot preview.

Multi-track System. Multimedia events (video and audio clips) are represented in the interface by bars. They can be freely moved through a customizable set of video and audio tracks in order to build the final stream. The bars' width is proportional to the events duration, and there is a time ruler that helps the user distributing and ordering events during the editing process. Each track has its own parameters; so many different configurations can coexist in the same project. For example, each audio track has independent equalization, stereo balance and gain controls.

Filter Manager. Filters and effects are common tools in multimedia editing. They change the video and audio properties of the output stream. For example, a filter could modify video brightness or normalize an audio event. The application offers two ways of applying filters. Firstly, a single event or track can be assigned a filter in order to change its properties without affecting the rest of events. Secondly, filter tracks can be created to apply filters or arrays of filters to a certain range of a track or a project.

Some of the filters that are supported by the developed application are: *sharp* (blur adjustment), *bricon* (brightness and contrast adjustment), *levels* (black and white levels, and gamma correction), *crop*, *space denoiser*, *temporal denoiser*...

Preview Module. Video editing strongly requires previewing the result frequently. For this purpose, the application includes a module for playing the edited stream. It is based on the GPL tool MPlayer, which accepts AviSynth scripts as input. It is important to notice that big projects and complex editing features could require some parts to be rendered, this means previously encoded. This limitation is shared with professional editors, because it mainly depends on the machine where the application is running.

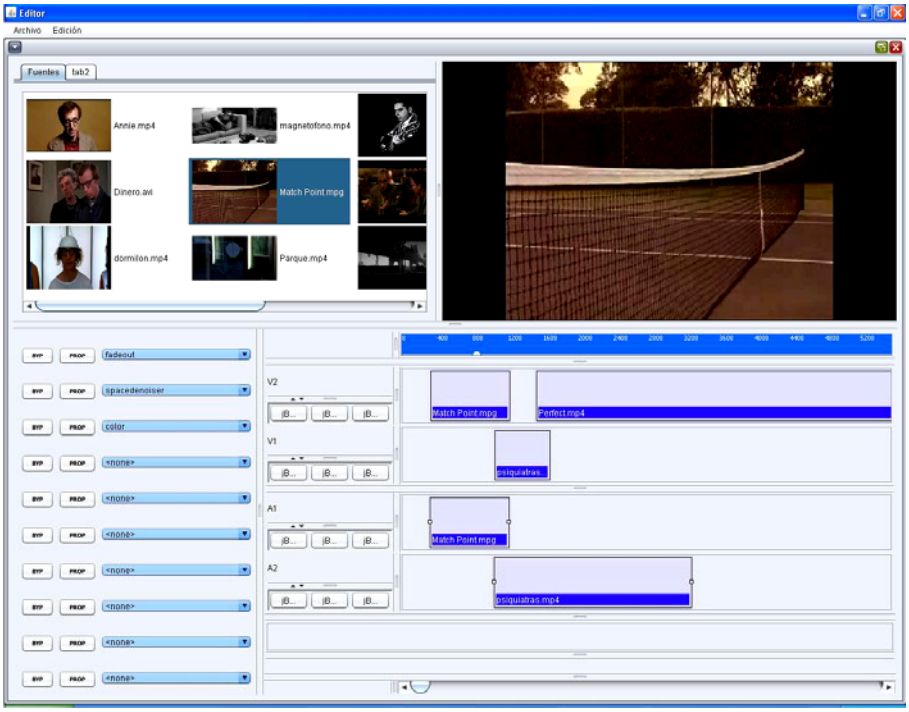

Fig. 5. User Interface screenshot

4 Test and Evaluation

This chapter shows how this system can tangibly bring some improvements into a multimedia edition environment. Several variables influence on the time spent on an entire edition process. Depending on the situation, different values can be found for sources bit rate, low quality sources bit rate, available bandwidth, sources duration, percentage of the sources' duration used in the final output stream or time spent by the user working in the edition. These values will determine the savings implied by the use of the solution.

An example should help on quantifying these savings. The high quality bit rate in this example will be 4 Mbps, which is less than the average bit rate for a DVD, but it is enough for a standard SD contribution; so it is not a very high quality value. The low quality bit rate will be 512 Kbps, for both downloading and streaming, which is enough for editing in the most of the situations. The percentage of the sources' length actually used for the project will be 40%.

It is necessary to study the work modes separately.

Offline Mode. Obviously, the main time saving is due to the difference between bit rates on high quality and low quality sources. In our example, a 4 Mbps high quality bit rate and a 512 Kbps low quality bit rate imply an 87.5% download time saving. As the resulting video has not to be uploaded again, a new time and bandwidth saving appears. In the example, the final file's length is the 40% of the sum of every source length. If we had to upload this file in high quality, time saving using the application's offline mode becomes then a 91.08%. Fig. 6 shows the difference implied by the use of the application's offline mode, using a 3 Mbps symmetric connection.

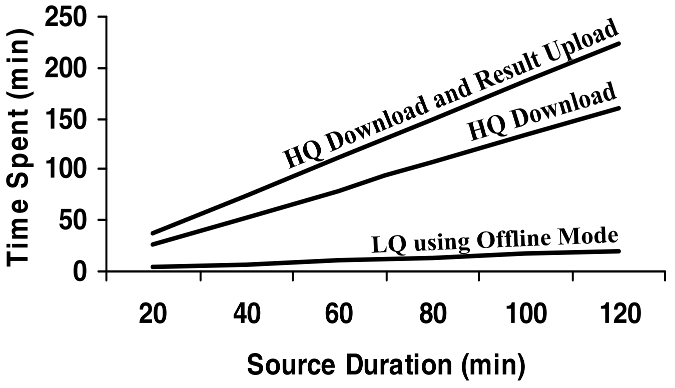

Fig. 6. Comparison between using the system with no improvement and using the offline mode

Online Mode. In this mode, the most notable advantage is that no previous downloads are needed. This means that the only spent time in an edition process is that invested by the user. The disadvantage is the necessity of a large bandwidth in order to give access simultaneously to a certain number of users. Anyway, this mode can be more efficient than the original way of working. Downloading the sources and uploading the result implies the same load than previewing the entire project more than ten times.

5 Conclusion

The explained application provides multiple user environments with a necessary source management system, flexible, and configurable. It defines as well a portable edition structure and coordinates different multimedia processing tools to work together, without making the final solution conditional to the selected tools.

The existence of two work modes reveals several uses for this application. For example, the online mode is useful for press offices, edition classrooms or multimedia file directories while the offline mode is useful for foreign correspondents, online production and edition educational applications or common edition workspaces, like replays on TV live shows or sports.

The improvements over a traditional edition system are notable. Less data load implies less spent time, and this implies the work to be done earlier and easier. This source management system may be a very useful tool for networking on content production and sharing not only for a professional use, but also for personal issues. If applied to more powerful multimedia edition software, it may result in a standalone application for online content edition and production.

References

1. Extensible Markup Language (XML) 1.0 (Fifth Edition), `http://www.w3.org/TR/REC-xml/`
2. AviSynth, `http://avisynth.org/mediawiki/Main_Page`
3. MEncoder/MPlayer, `http://www.mplayerhq.hu/design7/news.html`
4. SOAP Specifications, `http://www.w3.org/TR/soap/`
5. JDOM Project, `http://www.jdom.org/`