An Optimized Framework for Interactive TV Services Broadcasting

José Antonio Hurtado-Dueñas
jahurtado@citic.es

Bernardo Ruiz-Villalobos
bruiz@citic.es

Rocío Padilla-Conejo
rpadilla@citic.es

J. R. Salinas
jsalinas@citic.es

Centro Andaluz de Innovación y Tecnologías de la Información y las Comunicaciones (CITIC).
6 Marie Curie, Parque Tecnológico de Andalucía, 29590 Campanillas (Málaga), Spain. +34 952028610

ABSTRACT

Great research efforts have been focused on Digital Terrestrial Television (DTT) technology in the last years. The digital modulation and codification features of DTT have made it possible to offer new interactive services and applications. The ability of researchers and producers to develop and optimize these new applications in a fast and reliable manner to cover the new users' needs will determine the future of DTT. Several standards and middleware platforms have been developed to build interactive applications. An example of such a platform is the Multimedia Home Platform (MHP), which is established in most European countries. In this paper, a new MHP based optimized framework to develop and broadcast interactive television applications is presented. The system has been successfully tested and compared in terms of time and size efficiency against another platform belonging to a Spanish state-owned television channel.

Categories and Subject Descriptors

D.3.3 [**Programming Languages**]: Language Constructs and Features – *Frameworks, modules, data types and structures.*

General Terms

Performance, Design, Reliability, Human Factors.

Keywords

Digital Terrestrial Television, interactive applications framework, Multimedia Home Platform.

1. INTRODUCTION

Digital Terrestrial Television (DTT) is the new technology that will replace the traditional analog television. It is based on the use of digital encoding and modulation to manage audio and video contents, achieving a more efficient bandwidth use.

Besides the audio and video content, another kind of digital information can be sent to provide new services and applications that might run on the users' receivers or set-top-boxes. The Multimedia Home Platform (MHP) standard [4] [9] has been chosen to develop these applications in most countries in Europe.

The new DTT services and applications provide very interesting interactive features. The user does not have a passive role anymore as he can freely change the contents and information he is receiving from the television. Basically, there are two different types of interactivity: (a) local interactivity, which is related to the users that may interact locally with data stored in the set-top-box, and (b) real interactivity, that allows the user to provide response data and obtain data by using a return channel.

DTT Technology also helps to reduce the problem of the digital gap, which refers to the difference in opportunities for development between those who have access to new communication technologies and those who do not, thanks to its universality, set-top-boxes low cost and ease of learning.

Due to the DTT and MHP market growth, an efficient and robust application development framework becomes a key production factor to become competitive and allow fast implementation of new interactive services and functionalities.

2. STATE OF THE ART

As an initiative of the Digital Video Broadcasting Project (DVB), the Multimedia Home Platform (MHP) defines a generic interface between interactive digital applications and the terminals on which those applications execute. This interface de-couples different providers' applications from the specific hardware and software details of different MHP terminal implementations. The terminal can be a set-top-box, a television set or a desktop computer. Therefore, MHP envisions a truly horizontal market in contents, applications and services over multiple delivery systems including cable, satellite, terrestrial broadcasting and even third generation mobile phones.
Other standards for interactive TV around the world are the OpenCable Applications Platform (OCAP) [13] that is the software environment standard developed by the North American cable TV industry to enable interactive applications to run on cable and the Multimedia-Hypermedia Experts Group (MHEG) [10] that consists in an open standard interactive TV designed to provide interactive services in UK. Different transmission systems may adopt MHP middleware through the Globally Executable MHP (GEM) standard [3], which is a core of MHP APIs, where the DVB-transmission specific elements were removed.

DVB defines a suite of APIs that includes most of the Java TV™ API [17], HAVi (user interface) [5], DAVIC APIs [2] and DVB APIs. The applications downloaded to the set-top-box are Java™ applications called Xlets, built on a suite of APIs, tailored specifically for the interactive TV environment.

In [18], a browser approach for efficient and rapid deployment of MHP applications based on HTML/XML resources is presented. The first MHP browser based on DVB-HTML was Pontegra, developed by Nionex [12] in 2002. Commercial tmplayer from

Tmira [19] and open source Yambo developed by Cineca [1] are also examples of recent MHP browsers that rely on XML descriptions to build simple applications. In this paper, we present a new optimized framework to develop and broadcast interactive TV applications based on a similar approach and focused on achieving a proper performance in terms of time and size.

3. PROPOSED FRAMEWORK

3.1 Architecture Overview

The proposed framework is shown in Figure 1. This framework introduces an additional software layer on top of the MHP layer in order to run applications described in XML documents.

This additional layer, known as a "XML Browser", is a standard Xlet, which can be present with others Xlets without any problem. This browser makes it possible to run a complete interactive application, written entirely in XML format with the possibility of adding new feature extensions through plugins written in Java.

The XML Browser provides a higher level of abstraction for application development making it easy to write several tasks, which are more tedious to write in Java. The largest part of the code needed to perform typical operations with the MHP API is placed in the browser code, which is shared with all applications, thereby saving space in the object carousel [6] and reducing the startup time. The different components in the browser architecture are described in the following sections.

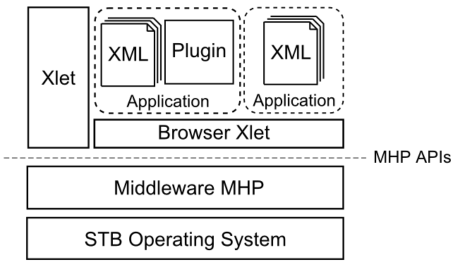


Figure 1. Software Architecture Diagram

3.2 XML Application Description

XML format was chosen to describe applications because it is extensible and there exist xml parsing libraries for Java, such as nanoxml [11], which has a small size and a good performance. In this framework, all applications are described in one or more XML documents which are processed at runtime by the browser.

Using XML for describing applications minimizes the amount of required space in the object carousel. The browser size is around 300Kb, which is rather high for a typical interactive application. However, XML applications are quite small because most of the needed functionality in those applications is implemented as high level functions in the browser code, which is shared by all XML applications. Sharing the browser code between applications also improves the loading time, because most of the needed Java classes are already loaded in the memory of the set-top-box.

XML files can be loaded from the object carousel or from the return channel via http/https protocols. They even can be generated dynamically by an application server. This feature brings the possibility of running server side applications with dynamic contents making it possible to use the TV as a true interface to the World Wide Web.

3.3 User Interface Component Set

The system has a set of graphical components that implement the user graphic interface. The components consist basically of enhanced text using styles, text input for the entry of text on the part of the user, containers, images, menus, buttons, animations, tables, comboboxes, checkbox, and textlists. These widgets are not included in the typical widgets of HAVi.

The XML description of the user graphic interface includes all the information associated with the graphical components set of the proposed framework, defining properties of each component, user events, and actions associated with those. An example of XML description is showed in List 1.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<application>
  <screen id="pulsa_rojo" background="" focus="pulsa_rojo">
    <video x="0" y="0" width="720" height="576" />
    <animation id="botonrojo" x="46" y="500" width="240"
      height="49" time="500">
      <image file="citic/images/pulsa_rojo.png" />
    </animation>
    <keyevent key="VK_COLORED_KEY_0"
      command="exit()" />
  </screen>
<application>
```

List 1. Example of Browser XML code

3.4 User Interface Builder

The user interface builder is the module that takes the XML description files as the input and processes all XML elements in order to build the screens on TV. As an initial step, the XML file is parsed and loaded into the memory of the device. Then, the browser takes the initial screen element and it starts the building process of that screen in order to show it on TV.

The time needed to build a screen depends on the number of components defined in it. If an application has a large amount of screens, processing all of them at the beginning could take a lot of time. For this reason, every screen is built at the moment of being showed, hence speeding up the loading time of the application.

3.5 Command Interpreter

In the proposed architecture, a user event may associate an action to a button on the remote control. The actions are defined by commands with a particular syntax that are processed by an interpreter integrated inside the framework. The commands can be included inside the XML file descriptor to perform a specific action of the component of the application, such as a script.

The command interpreter is also capable of handling variables, used as arguments of the commands and useful for its capacity to work on different screens, menus or components. It may also process scripts, estimate each command and execute all the actions associated to them.

3.6 Event Manager

Every component of the screen and even the screen itself may contain user event handling elements in order to perform a custom action when the user presses a key of the remote control.

The event manager is in charge of processing the user event handlers defined in the XML application description. The event handling process depends on the selected component at the moment of pressing the key. This event handling mechanism covers the requirements for most applications in DTT. In other cases, the event handling can be processed with a browser plugin written entirely in Java code using AWT event listeners.

3.7 Plugin Extensions

The XML browser solves the aspects associated with the MHP graphical interface, although it is not able to accomplish more complex tasks, such as the execution of algorithms or the use of a smartcard. Thus, it is necessary to implement some mechanisms to create hybrid applications that use the browser application to define the graphical interface and its behavior and, on the other hand, other specific elements written on Java from scratch. The proposed system uses plugins to solve the specific components implementation and allows invoking their own methods.

3.8 Security

An internet access profile was introduced in MHP 1.1 that enables the access of the applications to the internet from the set-top-box. This fact brings new opportunities to provide web services through TV. Then it is possible to load XML applications over the return channel via HTTP protocol from an application server.

Some services may require secure access to the application server. In that case, it is possible to make a secure connection using SSL sockets because MHP 1.1 supports them.

DDT receivers often have a smartcard slot, which can be accessed using standard SATSA APDU API [7] in order to store user information as passwords, user settings, encryption keys, etc. XML Applications can be broadcasted in an encrypted form in order to grant access only for users with the smartcard which contains the proper keys for decryption.

4. SYSTEM EVALUATION

In this section, the proposed system performance is compared with another interactive applications platform developed for a Spanish state-owned television channel. In the following sections, we will refer to this platform as "reference system".

4.1 Software and hardware resources

Several software and hardware resources were used to perform this evaluation. StreamGURU MPEG Analyzer [15] was used to extract files contained in carousels. TSReader [20] was used to obtain elemental streams bitrates from transport streams. iMux multiplexer from MIT-xperts [8] was used for frame generation, while interactive applications were developed using MHP 1.1.2 middleware from Osmosys [14]. Finally, Strong 5510MHP devices were used as set-top-boxes [16].

4.2 Evaluation initial issues

Certain issues related with the following concepts were taken into account before both systems were compared. Global bitrate affects directly to the applications loading time. Thus, it was established to 790.94 Kbps for both set of applications, in order to evaluate both systems on equal terms.

The reference system contains applications that are designed on a traditional Xlet manner, which implies that every application is written from scratch, whereas the applications running in the proposed system are services that run over the browser.

The interval of time in which an application is completely emitted on the broadcast channel is called cycle time. It depends on bitrate and applications size. The proposed system uses a single carousel where the bitrate is distributed in a fair way amongst every single application. Therefore, the same cycle time is achieved for all the applications in the carousel. On the other hand, the reference system uses several carousels with different and specific bitrates, so each application has a specific cycle time.

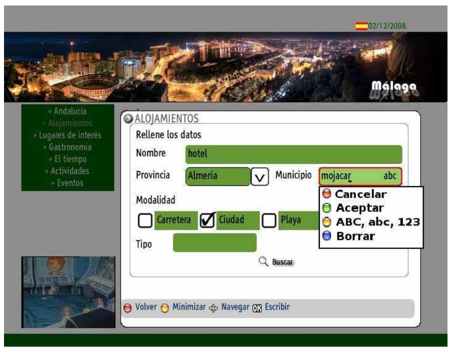
The applications running in the proposed system were tested, using the three caching approaches described in MHP standard, but no significant differences were found in the tests

Loading time is defined as the interval of time from the moment an application is launched until it can be executed. Increasing the number of applications on the transport stream implies a bigger amount of data and a loading time increase, as bitrate is fixed. On the proposed system, 11 applications are delivered. The browser application is excluded from this number, as it cannot provide any service on its own. The reference system delivers 6 applications, including a launcher application.

4.3 Applications analysis

The applications to be evaluated were previously analyzed in terms of size, bitrate and cycle time. To analyze these parameters on the proposed system, iMux multiplexer was used to generate a single carousel with a bitrate of 790.94 Kbps. Software tools described on section 4.1 were used to compute the values shown in Table 1. A screenshot from an application developed and running over the proposed system is shown in figure 2.

For the reference system, parameter values were measured on a frame of 3 minutes of duration, which was captured from the broadcast channel on 2008, October 24, at 9:34. It is important to remember that, in this case, each application is carried on a different carousel so each application has a different cycle time.



Nombre: hotel
Provincia: Almería
Municipio: mojacar abc
Modalidad: [ ] Carretera [✓ Ciudad] [ ] Playa
Tipo:

Figure 2. Application running on the proposed system

4.4 Systems comparison

To achieve an adequate comparison between both platforms, time measures were accomplished in two different terms. First, we measured the maximum interval of time in which an application is started from the launcher application, when the decoder has not been able to cache any files from the carousel. Second, we measured the minimum interval of time in which an application is started once the launcher application is loaded. In this case, the decoder has been able to use its caching mechanisms to optimize the loading time. The measures obtained are shown in Table 2.

Table 1. Applications parameters

	App	Size (Kb)	Bitrate (Kb)	Cycle time (s)
Proposed system	Browser	339	218.02	12.74
	Sports	87	55.99	12.74
	Weather	166.9	107.35	12.74
	Traffic	78	50.19	12.74
	EPG	71	45.64	12.74
	Chemists	56.1	36.11	12.74
	Games	77.8	50.04	12.74
	Puzzle	112.4	72.31	12.74
	Sudoku	73.3	47.15	12.74
	Launcher	10.6	6.79	12.74
	News	93.6	60.19	12.74
	Services	64	41.16	12.74
		Total	**Total**	**Mean**
		1229.7	790.94	12.74
Reference system	Teletext	323.6	148.48	17.85
	Weather	140	98.5	11.64
	EPG	343.2	98.5	28.55
	Launcher	213.7	198.48	8.82
	Traffic	416.3	98.49	34.62
	EmpleaT	473	148.49	26.1
		Total	**Total**	**Mean**
		1909.8	790.94	21.26

Table 2. Maximum and minimum start time for comparable applications in both platforms

	App	Proposed system (s)	Reference system (s)	Difference (s)
Maximum application start time	Launcher	30	22	8
	EPG	33	97	-64
	Weather	41	56	-15
	Traffic	35	176	-141
Minimum application start time	EPG	2	3	-1
	Weather	10	29	-19
	Traffic	2	89	-87

As shown on table 2, every application on the proposed system presents a lower loading time than the same application running on the reference system, with the exception of the launcher application. The reason is that the loading time for the launcher application includes the loading time for the browser application, however such a delay will not occur anymore when loading the other applications. Furthermore, the bitrate assigned to the launcher application on the reference system is much larger than the bitrate assigned to the others. Thus, the loading time for the launcher application is improved while the loading time for the other applications get worse.

If we consider the loading time of the launcher application together with another application, the loading time for any of the applications on the proposed system is much lower than any of the applications in the reference platform. Once the launcher application is loaded, the proposed platform applications have a small and acceptable loading time.The proposed system permits the delivery of a greater number of applications on the carousel while avoiding a global bitrate increase and maintaining a proper loading time. Furthermore, although the proposed system delivers 11 applications instead of 6 applications for the reference system, the total size for the applications is reduced in 680 Kb.

5. CONCLUSIONS

In this paper, a comparative based on time performance between the proposed system and a reference system has been presented. The proposed system delivers better performance indicators, achieving better time responses while minimizing the applications size. Therefore, more interactive applications can be offered while using the same bitrate and bandwidth.

Apart of performance results, the benefits of this framework for new kinds of interactive TV services has been exposed, making it possible to use the TV as an interface to the World Wide Web.

This framework also presents benefits in development costs of interactive TV services because the complexity is reduced due to software abstraction based on XML description.

6. REFERENCES

[1] Cineca, http://www.cineca.tv

[2] DAVIC 1.4 Specification Part 9. 1998.

[3] Document A108r1 - GEM 1.2.1 (Including IPTV). 2008.

[4] DVB-MHP Official website, http://www.mhp.org

[5] HAVi SPECIFICATION Version 1.1. May 2001.

[6] ISO Standard 13818-6, Generic Coding of Moving Pictures and Associated Audio Information: Extension for Digital Storage Media Command and Control. 1996.

[7] MIDP: SATSA-APDU API Developer's Guide. 2007.

[8] MIT-xperts GmbH, http://www.mit-xperts.com

[9] Morris S., Smith-Chaigneau A. 2005. Interactive TV Standards. Focal Press, Elsevier.

[10] Multimedia and Hypermedia information coding Expert Group, http://www.mheg.org

[11] NanoXML parser, http://sourceforge.net/projects/nanoxml

[12] Nionex, http://www.nionex.de

[13] Open Cable Applications Platform, http://www.opencable.com

[14] Osmosys SA, http://www.osmosys.tv

[15] Streamguru MPEG Analyzer, http://www.streamguru.de

[16] Strong Digital TV, http://www.strongsat.com

[17] Sun Mycrosystems, http://www.sun.com

[18] MHP Knowledge Project. 2006. Guidelines on migration.

[19] Tmira Solutions, http://www.tmira.com

[20] TSReader TS Analyzer, http://www.coolstf.com/tsreader