DELIVERING MULTI-DEVICE SYNCHRONISED INTERACTIVE SERVICES IN A BROADCASTING ENVIRONMENT THROUGH UIML DEPLOYMENT

P. Leroux, V. Verstraete, F. De Turck, P. Demeester
Ghent University -IBBT
Department of Information Technology
Gaston Crommenlaan 8 bus 201
9050 Gent, Belgium
philip.leroux@intec.ugent.be

K. Thys, K. Luyten
Hasselt University - IBBT
Expertise Centre for Digital Media
Wetenschapspark 2
3590 Diepenbeek, Belgium
kris.luyten@uhasselt.be

Abstract - As the variety in network service platforms and end user devices grows rapidly, content providers must constantly adapt their production system to support these new technologies. A factor that highly complicates this process is the need for interactivity. In the Multimedia Content Distribution Platform (MCDP) Project, an architecture for deploying highly interactive and synchronised television programs over a diverse collection of broadcast networks and end user devices was developed. It consists of a middleware platform with pluggable support for new broadcast networks and end-user devices, whilst preserving synchronised interactivity. In order to support a maximum of functionality and use cases; downloadable applications are used to provide the interactive services. As the user interface of such applications may vary depending on the capabilities of the different target devices, the MCDP middleware uses UIML for the description of generic user interfaces: The MCDP middleware is also used as the main platform for the delivery of synchronized, interactive services in a IP Datacasting system combining DVB-H and UMTS.

I. INTRODUCTION

Two trends are clearly visible for the current consumption of digital information. On the one hand, the number and diversity of digital services is rapidly increasing as interactive services are added to the digital multimedia. On the other hand, more and more telecommunication and broadcast networks are being built up. These networks may have very different characteristics (low vs. high bandwidth, unidirectional vs. bidirectional, wireless vs. wired, reliable vs. best-effort, etc.) This huge variety in available networks and related business models has also led to significant differences in end-user devices. These range from fixed devices such as a high-end PC or set-top box to wireless, portable or handheld devices. All these devices have very different characteristics in terms of screen size, network interface, user interaction model, etc.

This diversity exists nowadays and is expected to keep growing in the forthcoming years. This has a great influence on content providers. It will become more and more difficult to continuously adapt their production system to the increasingly complex world of heterogeneous digital networks and the associated service platforms.

The MCDP project [1] has been defined to study the possible architectures for a multimodal multimedia content distribution system that can cope with this complexity today and in the near future. One of the main questions is how to preserve interactivity and the associated synchronisation, defined once at the content provider and delivered to a multitude of end-user devices over a variety of broadcast networks.

Strongly related to our research are the standardisation efforts in different domains. These standards are often related to each other. In the MCDP platform, we make an attempt to reuse the common strengths of certain standards. Two important related standards are e.g. MHP and OCAP [2]. A lot of the concepts behind these standards are incorporated in MCDP. When it comes to generic user interfaces, two approaches are common: model-based user interface development (MBUID [3]) or High Level User Interface Description Languages (HLUID) such as UIML [4]. In the MCDP middleware solution, we focused on the integration of a HLUID in order to describe a device independent user interface.

Currently, in a new project, the MCDP middleware is used as the main platform for the integration of synchronized, interactive services into an IP Datacasting system that combines a broadcast network (DVB-H) with a bidirectional unicast network (UMTS). Strong focus is also on the integration of new (mobile) application platforms into the MCDP middleware architecture. A good example of such an application platform is the embedded Rich Client Platform (eRCP) [5].

The rest of the paper is organized as follows. In section 2, we give an overview of the system architecture of MCDP. Sections 3 and 4 present the use case that was chosen to illustrate the operation of the MCDP platform, and the evaluation architecture on which the interactive quiz was deployed. In section 5, the results are presented. In section 6, we will introduce how we are currently integrating the MCDP middleware into a IP Datacast set-up. Finally, section 7 states our conclusions and future work.

II. SYSTEM ARCHITECTURE

A. MCDP Server Architecture

The functional design of the MCDP server is presented in figure 1. Content is ingested by the Content Provider in a well-specified format. This content consists of four basic parts:

- Multimedia - The video & audio content of the television or radio program.
- App - An interactive application, possibly synchronised with the television program.
- UIML - The user interface of the interactive application, where every page is described with UIML.
- Metadata - The glue to bind these three elements together.

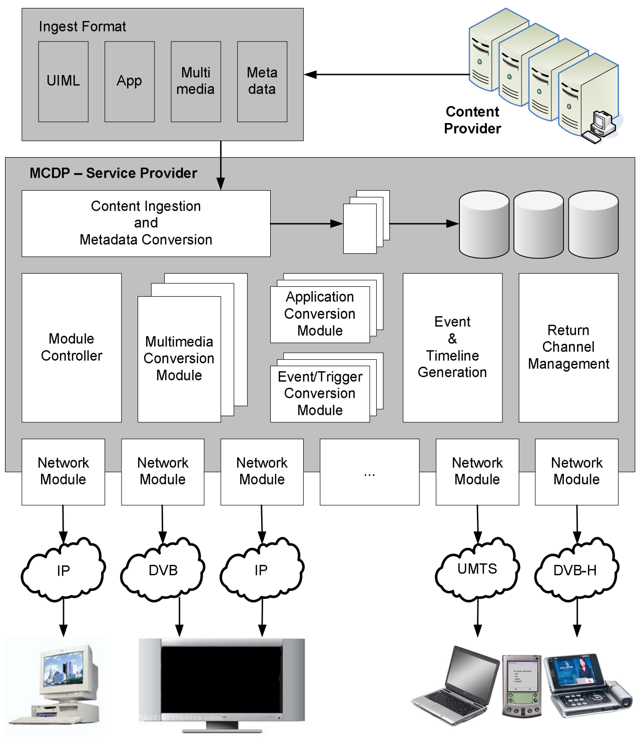

Figure 1: The MCDP server architecture.

It is important to notice that the interactive services are handled by application logic that has to be sent to, and installed on the end user's device. The main advantages of using downloadable applications are the support of a maximum of functionality and the unlimited and middleware version independent number of use cases that may be implemented. In this version of the MCDP middleware, Java application logic is used.

The content that is ingested by the content provider, is saved in a database, and retrieved when the Electronic Programming Guide (EPG), generated by the content provider or the service provider, indicates that the specified television program is to be broadcasted. When this happens, several other functional blocks come into play.

The *Multimedia*, *Application* and *Event/Trigger Conversion* Modules convert the multimedia, applications, events and triggers to a suitable format for the targeted devices, thus providing the user-device independence of both the multimedia content and the interactivity related data. An *Event* is the description of a specific action that can be executed by a downloaded application. A *Trigger* will inform the application logic on the client device when a specific event needs to be executed. There are two types of triggers:

- *Do-It-Now (DIN) Event* Triggers, where the associated event occurs immediately.
- *Scheduled Event (SE)* Triggers, these triggers contain a clock value that indicates when the event has to occur. They are sent in advance, and possibly more than once.

The Event & Timeline Generation Module handle the synchronization and on-time delivery of events to the clients. The Event/Trigger mechanism and the timeline generation are discussed in more detail in section 2.D. *Return Channel Management* collects and processes all the incoming return data and finally the *Network* Module will send the multimedia, the applications and the generated descriptors over the network to the clients. These Network Modules may support both unicast and broadcast scenarios.

B. MCDP Client Architecture

On the end-user device, a middleware framework should be installed that can handle all the extra functionality that is provided through the MCDP server middleware. The implementation of the client middleware may vary based on the terminal's capabilities, however the general architecture and functionality should be the same on all terminals.

As already mentioned, Java application logic is used to provide the interactive services. This requires the dynamic installing and management of Java software components on a device, which is exactly where OSGI[6] comes into play. Our client middleware layer is thus developed on top of the OSGI framework, while using the provided functionality of the OSGI framework in order to install and manage extra Java application logic on the end user's device. The client's middleware architecture is shown on Figure 2.

The initialization, starting, stopping and destroying of the Applications is handled by the *Application Manager*. The *Event Manager* provides the dynamic definition of events, and the delegation to the correct *Applications*. The *Return Channel Manager* provides an interface for the Applications to the (IP) *Unicast Return Channel* module (if available). The *Display* module renders the UIML pages on the screen and the *User Input* module handles the user input. The *Timeline* module is used in order to synchronize the Scheduled Events with the broadcasted media content. Finally, a *Context Manager* module is implemented to store the current context (which channel is currently watched, which applications are running, etc.).We will now discuss the user-device interaction and the client-server synchronised event triggering mechanism in more detail.

C. User-Device Interaction

The User Interface Markup Language (UIML [7]) is used to support user-device interaction on a variety of devices and platforms with a minimum of effort from the content or application provider. UIML is a high-level XML-based user interface description language that facilitates user interface creation and portability.

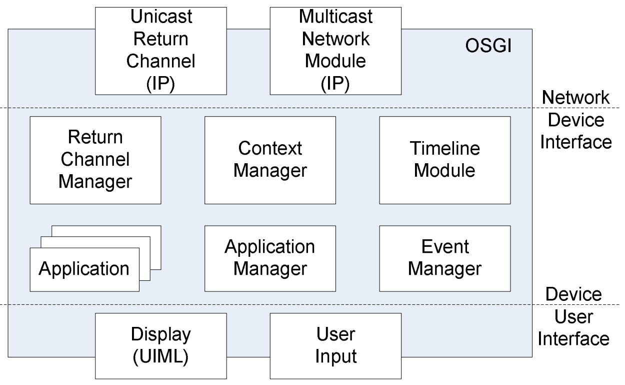

Figure 2: The MCDP client architecture.

A UIML document consists of four distinctive parts that allow to specify different aspects of the user interface independently:

1. *structure*: defines the different parts of the user interface and their hierarchical structure
2. *style*: describes properties of the parts defined in the structure. These properties can be both device specific properties (use device specific capabilities) or generic properties (reusable for a range of devices).
3. *content*: separates the content of the interface from the other parts, e.g. to create language independent user interfaces
4. *behavior*: defines a rule-based system with actions that are triggered when a certain condition is met.

In order to generate the actual user interface, the UIML document is linked to a vocabulary. This vocabulary defines how generic user interface parts (button, label) are mapped to widgets from a specific widget set (JButton, JLabel). This mapping is done by a UIML renderer, described in section 4.

The main benefit of this approach is its flexibility: since the user interface is not hard-coded on the device and the application logic is invoked remotely, new consumer devices that become available are easily integrated (e.g. new types of mobile phones or set-top boxes). The content provider can also update the user interface without requiring the user to install software updates. This flexibility is exactly what is required by the MCDP platform.

D. Synchronised Client-Server Interaction

In order to support full-fledged interactivity, the client framework must be able to dynamically load and unload applications, and to dynamically specify event types to which these applications can react. This allows for the most basic form of interactivity: DIN Event triggering.

An occurence of an event has to be triggered. In the DIN case, such a trigger is generated once on the server, sent over the network and processed by the client, resulting in the event occurence. This is a best-effort system, but provides no guarantee on the timely occurence of the event.

DIN triggering is sufficient for a wide range of services, and is the degree of interactivity that quite a few interactive television providers support. If, however, we want timely and possibly frame-accurate occurence of events to be guaranteed, we must go one step further, and relate the triggers to a timeline that is linked with the multimedia timeline. This can either be the multimedia timeline itself (e.g. the System Time Clock or STC in MPEG-2) or a derived timeline.

A derived timeline that is used in several widespread specifications (e.g. MHP and OCAP) is the Normal Playing Time (NPT). This notion is defined in the DSM-CC extensions to MPEG-2 [8]. The NPT is based on the STC but adds an extra level of abstraction. While the STC always increments by 1, the NPT can be used to reflect pausing, fast forwarding, rewinding and discontinuities in the multimedia stream.

At the client side, the NPT timeline is built and maintained based on NPT descriptors. In a non-broadcasting environment, it is sufficient to provide a descriptor for every change (e.g. resuming a paused NPT). In a broadcasting environment however, it is not known when the user will tune in on the channel, thus intermediate descriptors have to be generated by the server. If not, an NPT will only be started at the next change. The finer the granularity of the intermediate descriptors, the sooner the user will be able to enjoy the associated interactive services.

In order to add synchronised interactivity to the MCDP platform, we have chosen to use the NPT, and to relate the SE Triggers to this timeline. The Event & Timeline Generation Module generates NPT descriptors, both predefined and intermediate, and Scheduled Event Triggers. These descriptors and triggers are then sent to the different clients over the connected networks. It is important to note that the choice to use the NPT standard is not limiting. When support is needed for another synchronisation method, the generated NPT descriptors can dynamically be translated.

III. Use Case

The final goal of the MCDP project is an interactive, synchronised user experience over several devices and networks. As a proof-of-concept for the MCDP platform we selected a quiz application. In this quiz format, users are able to participate in an existing television show (“Test the nation”) using digital television or a mobile device which allows the participants to test their I.Q. based on questions asked by a quiz master.

In MCDP, this quiz application is reused and extended as synchronisation of the questions with the televisions show is added to the picture. When the quiz master asks a specific question, a dynamic overlay interface (rendered at the user’s device) is presented to all the viewers who want to participate. Depending on the target device, this interface may contain the possible answers, a repetition of the question, extra related media,etc. The viewer can then answer the questions by pressing on the correct colour-button of their remote control or by selecting the correct answer on the screen. Synchronised with the quiz master giving away the correct answer, all the participants receive an evaluation of their answer at the same time. Note that only the correct answer has to be send to the end user’s terminal, as the evaluation of the user’s answer is evaluated by the interactive application logic that has been downloaded to the end devices. The same application logic also keep track of the user’s score and finally the return channel is used in order to give the user feedback on how well he performed in relation with the other participants.

During this use case, targeting two mutual very different end user devices, all the necessary synchronisation parameters had only to be defined once as the MCDP middleware handled all the device dependencies.. Furthermore, the effort of generating different user interfaces for all the supported end-user devices is reduced by using a User Interface Description Language (UIDL). These two extensions make the synchronised define-once-play-everywhere scenario of MCDP possible.

IV. EVALUATION SETUP

In order to evaluate the MCDP platform, we plugged in two very different types of clients (see figure 3):

- a set-top box (STB), connected through an IP multicast network over Ethernet, and
- a mobile phone (GSM) with constrained resources, more specifically a Nokia 6680 mobile phone. Interaction with the MCDP server is established via an IP unicast connection over UMTS.

On neither of these platforms, middleware software was available that supports synchronised interactivity nor UIML. Therefore, on the set-top box we developed an OSGi based framework in Java that supports the NPT. We also implemented a similar (but currently non-OSGi) framework on the mobile phone using the J2ME technology, more specifically the MIDP 2.0 Profile [9]. Concurrently, a device independent UIML renderer was developed [10].

Combined with the device-specific widget sets, this allows us to render UIML pages on both the set-top box and the MIDP mobile phone. For both systems, we also integrated an emulator in the evaluation architecture. This allowed us to monitor device-specific influences. Finally, a multimedia server is used to stream the A/V content to the set-top box and its emulator. The synchronisation between the MCDP server and the multimedia server is done by means of the Network Time Protocol (NTP) [11] and a common EPG.

Figure 4 shows how the quiz applications finally looked on both the end-user devices. As you can see, the rendered content is specifically adapted to the hardware parameters of the end device (of which the screen size is definitely one of the most important parameters).

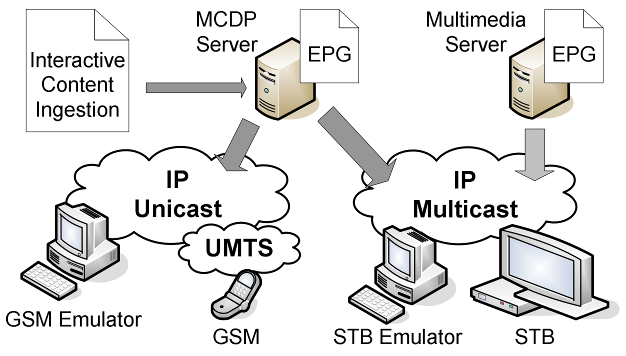

Figure 3: MCDP evaluation setup.

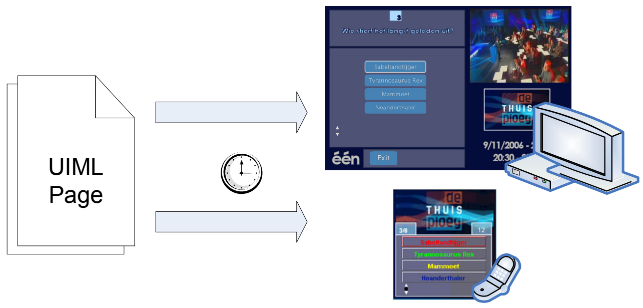

Figure 4: The MCDP use case.

V. EVALUATION RESULTS

When deploying the interactive quiz on the evaluation architecture, we measured two delays that have a big influence on the synchronisation: DIN delay & Rendering time. The results of these measurements are presented in figure 5.

SE triggers are being sent in advance to the client devices, resolved and added to the NPT timeline. When their trigger point is reached, they can be delegated almost instantly to the application. This is not the case for DIN triggers. When generated at the server, they have to be sent over the network, analysed and resolved. We can make an abstraction of the network delay as the multimedia will (in most cases) suffer the same network delay, but not of the processing time. As our graph shows, this has a rather limited influence that is device-dependent.

The rendering time is the amount of time it takes for the UIML renderer to build and display a page on screen. We have calculated this time span for all the quiz and result UIML pages, and set out the mean value for the four types of clients. As can be seen, rendering may take up to 1-2s, especially on devices with limited resources. Another factor that comes into play is the complexity of the UIML pages. The pages on the set-top box and emulator contained a large background image (see figure 4), which is why they took longer to render compared to the mobile phone and its emulator. 1-2 seconds compares to 25-50 frames (on a 50Hz PAL system), so extensive measures (caching, rendering in advance, etc.) will have to be taken in order to support (nearly) frame-accurate interactivity.

VI. INTEGRTEGRATING THE MCDP MIDDLEWARE IN A IP DATACAST ARCHITECTURE

The Digital Video Broadcast-Handheld (DVB-H) standard [12] provides an efficient way of carrying multimedia services over digital terrestrial broadcasting networks to handheld terminals. IP Datacast (IPDC) over DVB-H integrates DVB-H in a hybrid network structure consisting of both a mobile communications network such as GPRS or UMTS and an additional DVB-H downstream.

As the MCDP middleware was designed with network and device independence in mind, it forms an ideal basis for the integration of synchronised, interactive services in a IPDC set-up. Only some IP Datacast specific issues have to be taken care of. The most important issue to be resolved, is related to the synchronisation mechanism.

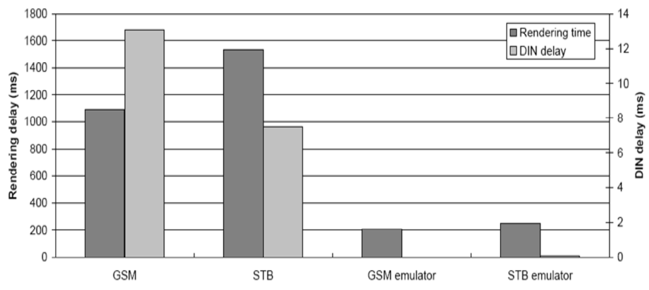

Figure 5: Rendering and DIN delays.

As a standard IPDC protocol stack prescribes the embedding of audio and video into Real-Time Transport Protocol (RTP) [13] packets, a dynamic translation should be made of the NPT descriptors to the RTP timestamp values. This mapping [14] can be done by analyzing the output (i.e. the RTP streams and the related RTCP Sender Reports) of the encoders that are located in a typical IP Datacast headend. Thus, the most important component that should be added to the existing MCDP middleware framework is a component that is able to analyze the output of the DVB-H headend's encoders in order to remap the MCDP synchronization mechanism to the RTP timestamp values of the multimedia content that is broadcasted over DVB-H.

Besides the translation module, 2 new Network modules should be developed. The first Network module is a typical multicast module for popular interactive services that are broadcasted via the DVB-H network. The second Network module is a unicast module that is able to send (less popular) interactive services to the same (IP Datacast) clients, but this time over a UMTS connection. The client middleware that has to be installed on a DVB-H terminal doesn't differ from the client middleware architecture that was introduced in section 2.B.

We can conclude that the effort that is needed to add synchronized interactivity to a new network or handheld device has been greatly reduced thanks to the MCDP middleware platform.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a middleware framework system (as part of the MCDP platform) to facilitate distribution of rich interactive multimedia content to a variety of end-user devices using heterogeneous network infrastructures, whilst preserving synchronisation between the application and the associated multimedia. The implementation of a proof-of-concept quiz application illustrates the feasibility of this platform and allows to evaluate the performace of the platform. In the near future, research will be done on how to reduce this rendering time (caching, pre-rendering, etc.) and adapt it to the requirements of different applications.

Currently we have integrated the MCDP middleware architecture into an IP Datacasting set-up. Only few extensions to the existing MCDP middleware are made in order to support this new set-up. As DVB-H is especially targeting mobile devices, we also focus more on the client middleware integration on mobile devices. As MIDP 2.0 currently does not have the right capabilities in order to add new application logic to an already installed framework, we focus now on new technologies such as the embedded Rich Client Platform (eRCP).

eRCP has a Core Runtime that provides OSGi support, a generic workbench which manages the launching and display of eRCP applications and also provides eSWT, a subset of the desktop Standard Widget Toolkit (SWT) API. The MCDP middleware may be applied to such a new platform when a mapping can be made of UIML descriptions to eSWT graphical components and when an eRCP bundle is used for the rendering of these eSWT related UIML pages. The downloadable application logic that is responsible for the interactive services are thus small eRCP bundles that can easily be installed and managed on the embedded target devices.

VIII. ACKNOWLEDGEMENTS

The authors would like to thank Siemens for the definition of the MCDP and MADUF [15] projects, which allowed us to do this research, and for their hardware support. Part of the research at EDM and IBCN is funded by EFRO (European Fund for Regional Development), the Flemish Government, the Flemish Interdisciplinary institute for Broadband technology (IBBT) and the industrial partners in MCDP (VRT, Siemens, Belgacom) and MADUF (VRT, Siemens, Belgacom, Telenet, Option).

REFERENCES

[1] MCDP, Multimedia Content Distribution Platform, "http://projects.ibbt.be/mcdp/", 2007.

[2] J. Piesing. The DVB Multimedia Home platform (MHP) and Related Specifications. *Proceedings of the IEEE*, 94(1), 2007.

[3] F.Paterno, C.Santoro. A Unified Method for Designing Interactive Systems Adaptable to Mobile and Stationary Platforms. *Interacting with Computers*, 15(3), 2003.

[4] M. Abrams and J. Helms. User Interface Markup Language (UIML) Specification version 3.1. Technical report, Oasis UIML TC, 2004.

[5] eRCP, embedded Rich Client Platform, "http://www.eclipse.org/ercp/", 2007.

[6] OSGi, "http://osgi.org/", 2007.

[7] M. Abrams and J. Helms. User Interface Markup Language (UIML) Specification version 3.1. Technical report, Oasis UIML TC, 2004.

[8] ISO/IEC 13818-6 Internat. Standard. Generic Coding of Moving Pictures and Associated Audio: Digital Storage Media Command and Control. Technical report, 1996.

[9] MIDP 2.0, JSR-118 Mobile Information Device Profile 2.0, "http://jcp.org/en/jsr/detail?id=272", 2007.

[10] K. Luyten, K. Thys, J. Vermeulen and K. Coninx. A Generic Approach for Multi-Device User Interface Rendering with UIML. *Computer-Aided Design of UserInterfaces*, 2006.

[11] Network time protocol (version 3) specification, implementation and analysis," 1992, RFC 1305. ETSI EN 302 304 V1.1.1,

[12] Digital Video Broadcasting (DVB), Transmission System for Handheld Terminals (DVB-H) ,2004-11.

[13] IETF RFC 3550: RTP, A Transport Protocol for Real-Time Applications, July 2003.

[14] P. Leroux, V. Verstraete, F. De Turck and P. Demeester,"Synchronized Interactive Services for Mobile Devices over IPDC/DVB-H and UMTS", Proc. *IEEE Broadband Convergence Networks, Munchen,*2007

[15] MADUF, Maximizing DVB Usage in Flanders, "http://projects.ibbt.be/maduf/", 2007.