A Model for the Delivery of Interactive Applications over Broadcast Channels

Jean-Claude Moissinac Cyril Concolato
Institut Télécom ; Télécom ParisTech ; CNRS LTCI
39 rue Dareau
75014 Paris - France
{jean-claude.moissinac, cyril.concolato}@telecom-paristech.fr

ABSTRACT

A good way to deliver video and multimedia services to mobile terminals is to broadcast them. But, methods to broadcast interactive applications are very crude. In current broadcasting systems, the available bandwidth for interactive applications is and will remain heavily constrained by the associated video and audio streams. To overcome these constraints, traditional techniques rely on carrousels to deliver the application in fragments, with some implied latency. The application starts when the base fragments have been received and the whole application is available when all the fragments have been delivered. It is crucial for broadcasters to control how an application is fragmented. Such fragmentation is in general made by hand, specifically for each application and differently for each broadcast technology. In this paper, we analyzed some typical interactive applications and derive a model, close to the application level, that can drive the broadcasting of the application independently from the type of application and from the broadcast technology. We also present a piece of software developed based on this model that validates the concepts on some applications.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia – *Architectures*; I.7.2 [**Document and Text Processing**]: Document Preparation – *Hypertext/hypermedia, Languages and systems, Multi/mixed media, Scripting languages, Standards*.

General Terms

Algorithms, Design.

Keywords

Broadcast, Carousel, Interactive Applications, Interactive TV, Mutimedia systems.

1. INTRODUCTION

Delivery of interactive applications together with audio-video streams is a key feature of many multimedia systems. In broadcast environments, the delivery of interactive application is heavily constrained. The application must be delivered so that new comers can tune in at any time and view the application as quickly as possible, or with a reasonable latency. A typical solution is to use the concept of carousel, as defined in DVB [3], in which the application is delivered periodically. The carousel period should therefore be as short as possible. However, interactive applications are delivered together with audio/video streams and these streams require most of the bandwidth. Given an available bit rate, it is often impossible to deliver the application both as a whole and at the given carrousel period. Hence, application developers or broadcasters have to design the application so that it can be delivered within the constraints.

Interactive applications can be designed in two ways. They can be described using programming languages such as Java, as in DVB-MHP applications. They can also rely on declarative languages such as HTML, CSS or SVG, together with some ECMAScript code, as in the upcoming HBBTV standard[1]. For both cases, we can consider that an application, also called interactive document, is a structured set of elements. These elements may be files (e.g. Java files, XML or HTML files) or parts of files (e.g. XML elements or sub-trees). Additionally, a key feature of an interactive application is that its content is usually dynamic and changes over time. Changes can be insertion, deletion, or replacement of application elements. Therefore, such applications or documents can usually be fragmented (into separate files or into separate updates to files or trees). The fragments can be delivered and presented progressively to the user, with a different carrousel period, depending on the nature and importance of each fragment. For example, in an application displaying an electronic TV program guide, the application should enable the user to browse all programs for all channels at all time, but it is acceptable, if at first, only the running programs are displayed and if the additional programs arrive a bit later.

In general, the fragmentation process is constrained by the programming or declarative language used to describe the application. It is also affected by the underlying protocol used to deliver the application. This makes it difficult and costly to manage the delivery of the application independently from the language and network used. In this paper, we present our work in the delivery of interactive applications in a broadcast or multicast context. We propose to model the properties of interactive applications relevant for their broadcasting. This model enables the separation of the application design from the preparation of the delivery, still allowing the efficient delivery of the application

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MoViD'10, October 25, 2010, Firenze, Italy.
Copyright 2010 ACM 978-1-4503-0165-7/10/10...$10.00.

[1] http://www.hbbtv.org

given bandwidth constraints, and independently from the representation of the application (Java, HTML, MPEG-4 BIFS) and from the delivery mechanism (DSM-CC, FLUTE, T-DMB).

The rest of this paper is organized as follows: Section 2 surveys related works; Section 3 presents the model and associated broadcasting algorithm; Section 4 discusses the implementation and results of our proposal. Finally, Section 5 concludes this paper and presents future work.

2. RELATED WORKS

In related works, we can first describe the many standards that exist to broadcast interactive applications. Among most of them, the MPEG-2 Transport Stream standard is the core. There are two mechanisms in MPEG-2 transport streams that allow the broadcasting of interactive applications. The first is called Sections. Sections are transport units that provide some important low level features for broadcast delivery. Sections are coded with a checksum which allows detecting errors not corrected or detected at the physical level. This feature is important because interactive application data does not contain much redundancy compared to audio or video signals and is therefore very sensitive to errors. Then, Sections also enable versioning of the broadcasted data. This is in particular important for receivers already tuned-in to detect changes in the version of the data being broadcasted or to ignore data already received. Sections are used in many systems, either as a basis for low level signaling (e.g. PMT) or for delivery of raw data (e.g. EIT), of files (as in most systems in DVB systems) or of application streams (as in the T-DMB standard).

The upper level construct of MPEG-2 TS for broadcasting applications is DSM-CC. Based on DSM-CC, two carousel mechanisms are derived: the low level Data Carousel and the high level Object Carousel. The Data Carousel provides means to deliver data in so-called modules with a versioning mechanism. The Object Carousel, which is built on top of the Data Carousel, enables the transmission of a virtual file system, using modules, with the possibility to add, update, and delete files. These mechanisms are used in many (if not all) television systems such as MHP, ACAP, OCAP, DASE as described in [4]. Interested readers can manipulate these technologies and create MPEG-2 Transport streams with sections and carousels using the Open Source software framework described in [1].

For IP-based broadcast or multicast systems, used in mobile or mixed mobile satellite systems such as DVB-H or DVB-SH, the file delivery protocol FLUTE is the well-known technology. It has similar characteristics to the MPEG-2 based systems in the sense that they provide error detection mechanisms or versioning.

Finally, we can cite the MPEG-4 Systems standard which defines a carousel mechanism with additional features compared to MPEG-2 or FLUTE, enabling the detection of crucial/non-crucial errors reducing the number of times receivers enter an invalid state and need to recover the full carrousel.

Among the academic research work, several papers focus on server issues related to the scheduling of data, the multiplexing of streams to minimize the latency or energy [5] or related to improving the existing carousel protocols [8]. Other types of work focus on the receiver, in order to improve the response time and reduce the latency for the viewer. For example, in [2], the authors discuss the notion of Hypermedia Temporal Graph to enable receivers to derive structures that will help retrieve the media required by the interactive application at the right time; while in [6], the authors discuss caching strategies.

To the best of our knowledge, up to now, there has not been any work on the formalization of the properties of interactive applications that are required for their delivery in broadcast.

3. MODEL AND ARCHITECTURE

The goal of this work is to enable the separation between the design of the application, its delivery preparation and its actual delivery, as depicted in Figure 1. In a broadcast production chain, once the application has been created with an Authoring Tool (AT), depending on the application, there are several characteristics that should be provided to the Broadcasting Tool (BT) to enable adequate delivery of the application. We consider that the Broadcast Preparation Tool (BPT) is there to mark the elements of the interactive application or document according to the properties defined the model below. The BT uses the markers to deliver the elements on channels with appropriate signaling according to the protocol and provides feedback to the BPT.

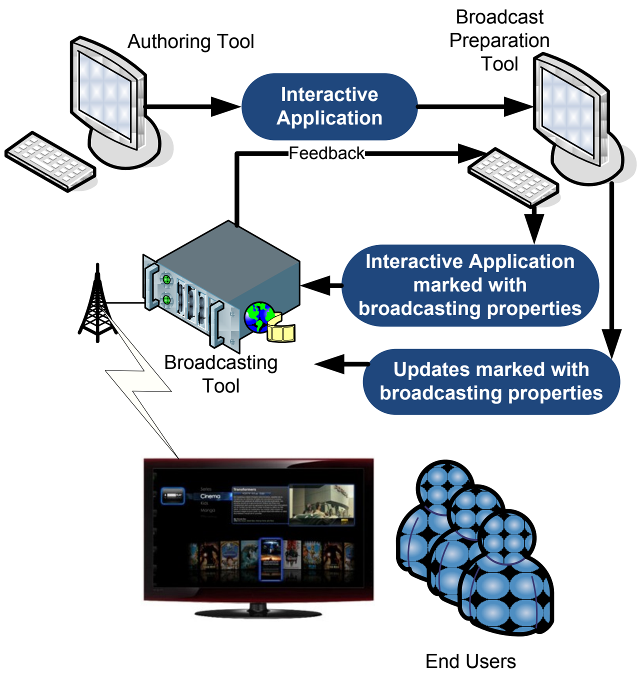
Authoring Tool
Broadcast Preparation Tool
Interactive Application
Feedback
Interactive Application marked with broadcasting properties
Broadcasting Tool
Updates marked with broadcasting properties
End Users

Figure 1 Interactive application production chain

3.1 Broadcast properties

We define here the abstract characteristics of an interactive application that are needed to best fragment and deliver it. As indicated in the introduction, we consider an interactive application or document as a set of elements to be delivered.

The first characteristic of these elements is ***synchronization***. When preparing the delivery of the application, one has to indicate for each element if it must be synchronized with the associated audio/video streams or not.

Then, for application elements that do not need to be synchronized, an important characteristic is the ***periodicity*** of the data to be sent, i.e. if it needs to be sent periodically and in this case what is the carousel period. The carousel period is the upper bound for the acceptable latency in the display of an element. This upper bound can also be constrained by its lifetime.

Interactive application elements may also be delivered differently based on their ***necessity***. Among the elements that compose an application, some elements form the basis of this application and are required. Other elements that provide additional information can be considered as enhancement elements and are not strictly required to access the application. This notion of displaying enhancement elements of an interactive application after the base element can be put in parallel with the scalable coding of video streams or progressive loading techniques for web pages.

3.2 Broadcasting tool architecture

According to the previous section, there are 3 binary markers that correspond to the questions: should an element be delivered periodically or not, should it be delivered synchronously with the video and is it a mandatory element for the application to work. These 3 markers actually define 8 types of channels that the BT will process differently. However, if we assume that a periodic element cannot be delivered synchronously with the video, we end up with the following 6 types of channels:

C1 Channels from which mandatory elements must be delivered periodically (without synchronization with the video). The base document or class representing the application is typically received on this type of channel.

C2 Channels from which elements are optional and should be delivered periodically. Typically, enhancement elements to the base elements are received on this type of channel.

C3 Channels from which elements are mandatory for the application and must be delivered synchronously. Typically, updates to the base document that must happen at precise instants in the video are received on this type of channel.

C4 Channels from which non-mandatory elements are received that must be delivered synchronously. Typically, temporary modifications to the base application are received on this type of channel.

C5 Channels from which mandatory but non periodic and non synchronized elements are received.

C6 Channels from which optional elements are received. Typically, low priority enhancement updates are received on this type of channels and are delivered only if the bandwidth allows.

However, we can group these types of channels and consider that the BT conceptually uses two types of input channels:

- Channels on which the data is sent periodically and for which versioning is important. In an application, there must be at least one channel of this type (C1 and C2).
- Channels on which the data is made of updates, which are not sent periodically, but are placed in a queue and sent depending on their mandatory and synchronized properties (C3 to C6). These updates typically modify elements that are sent periodically.

So the BT maintains a link between the channels of type C3-C6 and of type C1/C2. When an update is received, it is dealt with in the following manner:

- U1 It is only applied to the carouseled elements and will be sent as part of the next carousel delivery,
- U2 It is applied to the carouseled element and delivered both in the carousel and in a non-carouseled channels (with a proper signaling) for non- and already-connected clients,
- U3 It is only delivered and not applied to the carouseled elements.

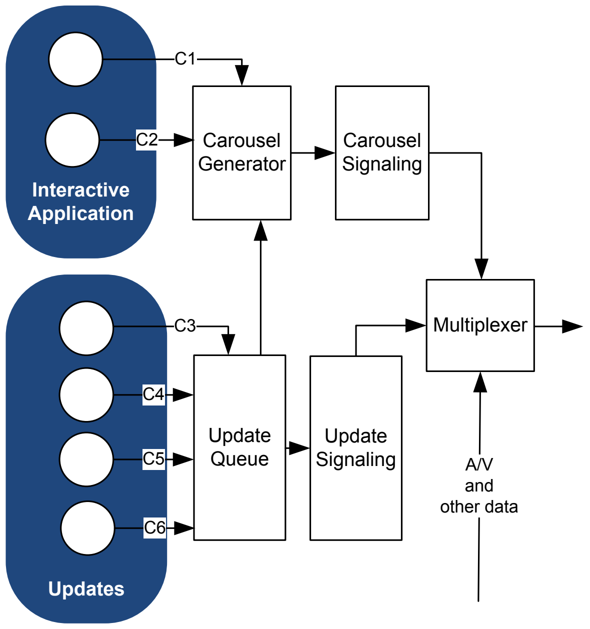

Figure 2 Architecture of the broadcasting tool

The BT uses queues for processing synchronized updates: one for mandatory (C3) and one for non-mandatory (C4), in which the updates are ordered according to their timestamps. Additionally, each update has an expiration date, after which its content is not needed anymore and can be removed from the queue.

3.3 Broadcasting algorithm

We describe here a possible algorithm for the BT based on the architecture described before. We assume that the available bandwidth for the output channel is constant or at least has a known minimum.

Each channel of type C1 and C2 has an associated carousel period P_I. We consider that the BT will accumulate data from the input channels during a period of time P_B and will then produce the output. This creates some latency in the process but is necessary for the ordering of the input data. The algorithm is as follows.

First, the updates that remain in the queues of channels of type C3 are added to the output and removed from the queue, An exception is raised if some updates are late or expired.

Then, elements from channels of type C1 are processed. If the data was not fully sent during the previous period P_B, the left over is processed again. If no data from this channel has been added to the output since a period P_I, a new carousel delivery is started and the corresponding elements are processed. The processing consists in sending the element in the output channel added until either it is full or until all elements are sent.

Then, updates from channels of type C4 are processed, expired updates are removed from the queue, and updates whose timestamp fit in the current period are added to the output channel.

Finally, updates from channels of type C5 and C6, and carousels from channels of type C2 are processed.

This algorithm defines a possible priority between input channels, but there may be other possible configurations for the priority. This is a possible extension of our model. Additionally, the priority could also be determined dynamically based on the feedback provided by the BT to the BPT.

4. IMPLEMENTATION AND RESULTS

4.1 Implementation

Based on the principles described above, we have implemented a BT within the GPAC open source framework [7]. It is capable of managing interactive applications developed using the MPEG-4 BIFS standard and of broadcasting them according to the T-DMB standard. It first receives a configuration which signals the number of input channels to be set up. For each input channel, a carousel period can be provided indicating channels of type C1 or C2. The tool currently assumes that only the first periodic channel is of type C1 and the others are of type C2. Other channels, for which a period is not provided, are assumed to be of type C3 and C5. The distinction between the two types is made by the use of timestamps on the data given on the channel. This tool currently also assumes that the data provided will fit in the available output bitrate.

It therefore does not handle the priority or the (non-)mandatory property of channels. It rather focuses on applying the data received on C3 and C5 channels in the C1 and C2 carousel channels. For that purpose, each received data on a channel C3 and C5 is marked with an update type (U1 to U3).

4.2 Results

Using this implementation, we have applied the above algorithm on the following example of a cycling race. The program is composed of live audio/video of the race and of an interactive application. This latter provides access to two pages of information: one is the overall standing and the second one is a personalized page showing the standings of the viewer's favorites cyclists. This application can be designed as follows:

- C1: the elements that form the structure of the application, the basis of the two pages, the name of all cyclists and all the elements needed for interactions (e.g. buttons). These elements are periodic and mandatory. Their lifetime is the duration of the application. The carousel period of these elements define the access time to the application.
- C1: the elements that provide the ranking for all cyclists. They are mandatory for the application to be meaningful. These elements should be sent periodically and since the lifetime of these elements is short, the carousel period should be short.
- C5: instant messages like messages about the fall of a competitor…
- C2: elements providing enhancements for the application such as the flags of the country of each cyclist. These elements are not mandatory for the application. They should be sent periodically, with a long period, only if the bandwidth allows.

As a sample, let us look at the result for two scenarios with the same data: one with a simple carousel; and another one, where a carousel is set up for several data types. Here we consider three types: a base scene (type C1), a periodic update of the base scene (type C1) and data for amelioration (type C2). For simplicity, we suppose here that the cost of the protocol doesn't affect significantly the relative cost of the transport of each type of data. The table below shows the data for this scenario.

Available bandwidth (kb/s)	10
Base scene (kB)	50
Race data (kB)	0,5
Improvement data (kB)	80

If used with a simple carousel with the same duration for all the data, the result is illustrated by the table below.

Total bandwidth (kb/s)	17,4
Base scene (kb/s)	6,67
Race data (kb/s)	0,07
Improvement data (kb/s)	10,67

If we adjust the carouselling by data type, the table below shows the result.

Total bandwidth (kb/s)	9,6
Main carousel cycle (s)	60
Base scene (kb/s)	6,67
Carousel for race data (s)	5
Race data (kb/s)	0,8
Carousel for improvement data (s)	300
Improvement data (kb/s)	2,13

In the first case, we need nearly twice the available bandwidth to transport the scene; the periodic update of the main information is not very short (60s). In the second case, a user needs to wait at most 60s to get the scene; each user has an updated information each 5s; an available bandwidth of 0,4 kb/s remains available for instant messaging.

A demonstration of this example can be viewed on this video[2].

5. CONCLUSION

In this paper, we have presented our proposal to separate the design of an interactive application from its adaptation to the delivery on broadcast channels. Our proposal is based on the use

[2] http://vimeo.com/9323680

of an intermediary tool, on a model describing the properties of the application and on the use of such description by the broadcasting tool. Our experiment, with a static definition of the priority between the channels, shows us the pertinence of the model. It validates the fact that building of the dynamic interactive application is simplified by this new tool between the source of the data to broadcast and the broadcasting tool. The data source only has to be aware of the separation of channels of data and a choice of the period of each periodic channel to fit the available bandwidth. In future work, we want to address more complex scenarios where a dynamic control of the channels based on feedback from the broadcast tool is possible.

6. REFERENCES

[1] Berger, A., Pallara, L., and Schatz, R. 2008. An open source software framework for DVB-* transmission. In *Proc. of the 16th ACM international Conference on Multimedia* (Vancouver, British Columbia, Canada, October 26 - 31, 2008). MM '08. ACM, New York, NY, 1093-1096.

[2] Costa, R. *M., Moreno, M. F., and Gomes Soares, L. F. 2008. Interme*dia synchronization management in DTV systems. In *Proc. of the 8th ACM Symposium on Document Engineering* (Sao Paulo, Brazil, September 16 - 19, 2008). DocEng '08. ACM, New York, NY, 289-297.

[3] Digital Video Broadcasting (DVB); DVB specification for data broadcasting. ETSI EN 301 192 V1.4.1 (2004-11).

[4] Saleemi, M. M., Björkqvist, J., and Lilius, J. 2008. System architecture and interactivity model for mobile TV applications. In *Proc. of the 3rd international Conference on Digital interactive Media in Entertainment and Arts* (Athens, Greece, September 10 - 12, 2008). DIMEA '08, vol. 349. ACM, New York, NY, 407-414.

[5] Hsu, C. and Hefeeda, M. 2009. On statistical multiplexing of variable-bit-rate video streams in mobile systems. In *Proc. of the 17th ACM international Conference on Multimedia* (Beijing, China, October 19 - 24, 2009). MM '09. ACM, New York, NY, 411-420.

[6] Park, D., Ku, T., and Moon, K. 2005. Data Broadcasting Software Architecture supporting Real-Time Caching and Monitoring in Interactive TV. In *Proc. of the 4th Annual ACIS international Conference on Computer and information Science* (July 14 - 16, 2005). IEEE Computer Society, Washington, DC, 593-597.

[7] Le Feuvre, J., Concolato, C., and Moissinac, J. 2007. GPAC: open source multimedia framework. In Proceedings of the 15th international Conference on Multimedia (Augsburg, Germany, September 25 - 29, 2007). MULTIMEDIA '07. ACM, New York, NY, 1009-1012. DOI= http://doi.acm.org/10.1145/1291233.1291452

[8] Lin F.; Sun J. 2007. An Interactive Service Platform Solution Based On Enhanced Data Carousel Scheme. In IEEE T. Consum. Electr. 53, 2 (May 2007), 675-682.