Synchronization Management in DTV Applications

Romualdo Monteiro de Resende Costa
Departamento de Informática – PUC-Rio
Rua Marquês de São Vicente, 255
Rio de Janeiro – 22453-900 – Brazil
romualdo@telemidia.puc-rio.br

ABSTRACT

Synchronization support has been one of the most important QoS issues in DTV environments. In order to preserve the spatio and temporal synchronization among media objects that compound a DTV application, this PhD thesis proposes several abstract data structures to guide the synchronization control during application's life cycle. The proposed structures can be used to control the applications' playout stage at content producer and application execution at receivers in many different platforms, including those with multiple devices. The specification of applications where media objects, including those with declarative and imperative code, can be distributed to run in several different devices, recursively, is another important focus of this thesis.

Categories and Subject Descriptors

I.7.2 [**Document Preparation**]: Languages and systems, markup languages, multimedia, hypermedia, standards. D.3.2 [**Language Classifications**]: Specialized application languages

General Terms

Design, Standardization, Languages.

Keywords

Hypermedia, temporal graph, NCL, digital TV, Ginga.

1. INTRODUCTION

Temporal and spatial synchronization management in DTV applications deals not only with predictable events[1] (like the end of a media segment presentation with known duration and known beginning time), but also with unpredictable events (like viewer interactions). Content and content-presentation adaptations, which are usually performed during runtime, are other sources of unpredictability that should be supported.

In DTV applications where unpredictability is common, synchronization management is usually based on the event-driven paradigm (also called constraint/causality paradigm). Different from timeline, the event-driven paradigm bases its synchronization support on the relative spatiotemporal positioning of events, independent from when (the absolute moment in time) the synchronization happens and even if it happens.

Synchronization specifications should use high level constructs to support the authoring process. These constructs must favor relationships among media objects that exist in the author's mind model (application logical semantic). With this focus, the use of time-based declarative languages is favored when spatiotemporal event-driven synchronization needs to be specified.

Declarative languages emphasize the declarative description of an application rather than its decomposition into an algorithmic implementation, as it is done when using imperative languages. Such declarative descriptions generally are a high-level specification, and thus they are easier to be designed than imperative ones, which usually require a programming expert. Examples of declarative languages focusing on media object synchronization are NCL (Nested Context Language) [1], the standard language of the Brazilian Terrestrial DTV System – SBTVD-T [1] and ITU-T Consented Recommendation for IPTV services [11], and SMIL (Synchronized Multimedia Integration Language) [14], a W3C Recommendation.

Usually, DTV applications are transmitted to client receivers where application language engines must try to guarantee the author's (the programmer's) descriptions. Unlike authoring constructs, presentation data structures should be closer to the execution engine and should offer low-level primitives in order to make easy the presentation scheduling. Furthermore, some support should be offered to DTV application control. As an example, it is desirable that viewers could explicitly pause a DTV application and then resume it at some later time. It is also desirable that an application could be started at any moment in time. Moreover, it would be desirable to support a viewer that changes the TV channel, starts another application in the new channel, but then regrets and returns to the previous channel, resuming the application and inheriting all actions previously triggered.

In order to assure a synchronized presentation, application data transmissions, from servers to receivers, should also be management, maintaining the minimal needed QoS.

Although authoring, presentation, and transmission goals are the same (to guarantee the synchronized presentation of media objects), these different phases of an application life cycle normally require different data structures, computed from the application specification, to support client and server sides of a DTV system [8].

In agreement with the previous paragraph, this PhD thesis proposes the use of different data structures. However, they are derived from a unique parent data structure called HTG (Hypermedia Temporal Graph) the main focus of this PhD thesis. In this paper, the use of these data structures in multiple device platforms is emphasized, since it is the current work in progress. The proposed approaches were partially put into practice in the current open source reference implementation of the Ginga (the Brazilian Terrestrial DTV) middleware [1].

[1] In this paper, an *event* denotes any occurrence in time with finite or infinitesimal duration

The paper is organized as follows. Section 2 presents some related work. Section 3 discusses the proposed structures and introduces the use of multiple devices. The application flow control and the thesis's current contribution are then discussed. Section 4 is reserved for final remarks and future directions.

2. RELATED WORK

XHTML-based languages, such as those used in BML [2], ACAP-X [3] and DVB-HTML [10] allow a declarative description of relationships involving unpredictable events; in fact, only viewer interactions. Synchronization in its broad sense can only be achieved using imperative coding, commonly written in ECMAScript [9]. In applications specified using the aforementioned languages, DOM events [13] are responsible for triggering unpredictable events. As far as we know, no one of these middleware implementations have a data structure similar to HTG to guide its synchronization tasks. Therefore, no one offers the aforementioned facilities. Indeed, it would be very difficult, if not impossible, to infer a data structure like HTG when imperative languages (in the case ECMAScript) are responsible for adaptations and spatiotemporal synchronization.

Several players are available for traditional Web-based multimedia applications [4][5]. Their data structures are fair enough for playing the majority of applications. However, DTV broadcasting has some specific features that must be taken into account. The fine-grained presentation control requested by TV channel tunings is one of them. The support for live content generation and synchronization is another. Other requirements include the fact that applications may run in receivers with limited resources. The thesis's structures for synchronization management were proposed in agreement with all these requirements [8].

The use of exhibition devices other than the traditional television commanded by a remote control, can improve viewer experience in different ways [6][7], some of them implemented by Ambulant player and Annotator [5][6]. Among other features, this player allows the exhibition of optional content in available devices, extending the application control; allows including viewer's annotations during a presentation; and allows transferring a presentation to mobile devices, without presentation discontinuity, even when the viewers are moving out (session mobility).

Non-monolithic rendering [6], with multiple devices being simultaneously used to render the application content and with multimodal interaction devices available, is one of the current research issues being addressed in this thesis. The solution proposed uses NCL as a glue language. As a glue language, NCL does not restrict or prescribe any media-object content type. In this sense, we can have the usual media objects (text, image, video, audio, etc), imperative objects (objects with imperative code content) and declarative objects (objects with declarative code content), as NCL media objects. As an example, the Ginga middleware reference implementation provides support to SMIL, X-HTML, embedded NCL, Lua and Java objects. As a consequence, it is possible to create NCL applications that can use other applications (as NCL media objects) specified using the aforementioned declarative and imperative languages as well as to specify relationships among these applications [12].

The set of NCL objects can be distributed to be processes in several different devices, under a master control. Nested NCL objects received by devices can also be again distributed, in a recurrent process, as will be discussed in the next section.

3. THESIS WORK OVERVIEW

3.1 Synchronization Management

This research has focused on the development of several abstract data structures to support synchronization management in DTV systems. The main structure is a directed time graph model called Hypermedia Temporal Graph (HTG), which represents relationships among events in an application. When used to represent NCL applications, three types of events are recognized: presentation event, corresponding to playing a content anchor (whole media-object content, or part of this content); selection event, corresponding to a viewer interaction (selection of a content anchor); and attribution event, corresponding to setting a value to a media-object's property (variable). Each event defines a state machine, as show in Figure 1, which should be maintained by the receiver user agent. An event can be in the sleeping, occurring or paused state, and change its state upon receiving actions: start, stop, pause, resume and abort.

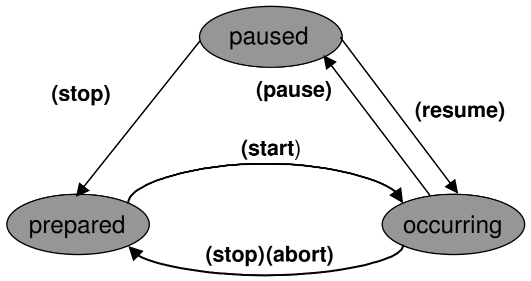

Figure 1. Event State Machine.

HTG is composed by vertices, which represent actions (for state changes) performed on event state machines, and directed edges that represent relationships among actions. An edge is labeled by a condition that must be satisfied in order to trigger the action specified in the edge's output vertice.

HTG defines simple and compound conditions. A simple condition is defined by a temporal interval that must be waited before firing the edge traverse (and so the action defined in the edge's output vertice), or by a variable that must be evaluated in relation to a desirable value, or still by external actions, such as viewer interactions. Compound conditions are defined through logical operators (or, and, not) binding two or more conditions.

After defining an application starting point, HTG can be used to derive other structures related with the synchronization management. Table 1 summarizes these structures (plans) in both client and server sides. How these plans are built and their functional details can be found in reference [8].

Table 1. Plans to support DTV synchronization management

Server Side	Client Side
Pushed-Data (Carousel) Plan	Presentation Plan
	Player-load Plan
	Pre-fetching Plan
	QoS Plan

When applications contain only predictable events, HTG edges are labeled only by temporal intervals. From the document starting time, the graph traverse identifies every action that must be applied to media players. These actions can have their moment in time computed taking into account the time intervals required to satisfy conditions, from the HTG entry point to the corresponding action vertices. This set of actions and corresponding moments in time compose the presentation plan.

The same procedure aforementioned can be used to compute actions and their corresponding moments in time for all predictable events, from an application starting point to an unpredictable event; and from each unpredictable event to the next unpredictable event in the graph traverse. In this last case, the computed moments in time will be relative to the moment in time that the starting unpredictable event of the traverse path happens. During an application presentation, as soon as an unpredictable event time is known, the presentation plan is updated changing all moments in time relative to this event to be now relative to document starting time.

Presentation plan preserves the past and predict possible future time moments of events of an application. Using this presentation-plan feature, applications can be paused and later resumed, or still started at different runtime positions, preserving all interactions applied before the interruption.

Since the considered devices usually have resource limitations, their media players should only be instantiated when necessary. (Ginga reference implementation [1][12] is component based and allows dynamic linking). They normally cannot stay instantiated after being used, waiting for a next possible utilization. On the other hand, the time need to instantiate a media player can introduce a delay long enough to cause loss of synchronizations. A player-load plan can be used to support player instantiation management and updating, avoiding undesirable delays during the application execution.

Player-load plan is computed from the presentation plan, disregarding events other than presentation and the transitions other than the start and resume. The plan construction must take into account the delay for each specific player and platform.

3.2 Multiple Device Support

Two types of device classes were defined during the research in Ginga-NCL middleware, the presentation engine responsible for playing NCL applications: those with devices able to run object players (known as active classes), and passive classes, whose devices are only requested to exhibit content processed in other devices (registered in an active class).

Many device classes can be created. Each one must specify parameter values for the exhibition output. These parameters include the screen size, the screen graphic size and the audio type supported (mono, stereo etc). Active classes must also specify the media players they supported. Devices associated with a specific class must support all its defined parameter values. A device can be registered in more than one class, of both types.

NCL objects (including media, imperative and declarative objects) are distributed among the available devices as defined by the application authors. NCL Layout module defines the *regionBase* element whose *device* attribute allows specifying to which class an object must be delivered. By Ginga-NCL default, when only one device is addressed by an application, the *device* attribute do not need to be explicitly declared. Also by default, the class index values “1” and “2” are reserved to passive and active classes, respectively.

A device in an active class can receive an imperative or declarative object to be presented. If the object code language allows, the object’s content processing can also be distributed, and so on. For example, if a device in an active class receives an NCL object with NCL code (that is, an NCL object nested in the NCL application), the processing of this object must be identical to any NCL application processing. Therefore, objects embedded in this NCL object can also be distributed in a recurrent process. In order to prevent loops, a device can not receive contents originated from its distribution process. Furthermore, devices in the same class cannot receive content from more then one device (parent device) simultaneously. These rules defines a tree structure processing distribution and will avoid several undesirable behaviors, like content presentation superposition (zIndex and graphic plan management), confusing navigation procedure (specially remote control key navigation), etc.

In Ginga-NCL the tree structure also defines to which device an interaction input must be addressed. When an NCL application is started, all input coming from any registered device must be handled by the device that runs the NCL application (the root device). That is, all devices act as input device of the root device. When a device registered in an active class receives a content to be presented and also the focus control, it gains the control of its input devices and all input devices of its descendent classes. When the object presentation finishes or when the focus control is explicitly given back, the input devices it controls are passed to the control of the parent device in the tree structure, and so forth. Objects presented in passive classes are always under control of the parent device, in all aspects.

Figure 2 illustrates a user case. In the scenario we have a TV set receiver; two headphones registered in a passive class (1); two PDAs registered in an active class (2); and a notebook, registered in an active class (3). The notebook defines another device domain, with a sound player registered in the passive class (4).

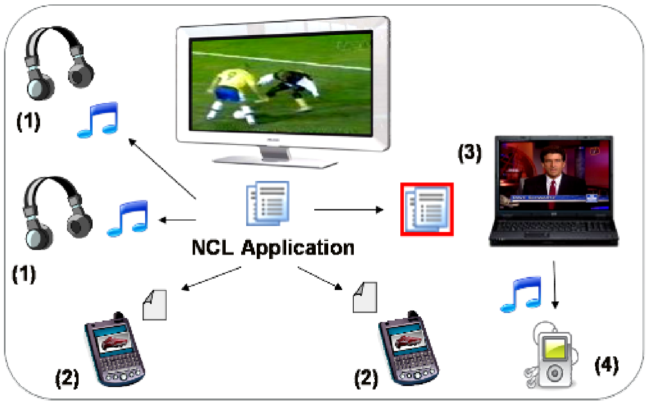

Figure 2. Example of DTV Multiple Devices Scenario.

The NCL application starts with the soccer video being presented in the TV set. During the match, audio tracks coming from the soccer field open microphone are captured and they compose the content of an audio object. This object is sent to exhibition in the passive device class (1). As a consequence, both headphones decode the audio stream sent and controlled by the TV set, simultaneously.

During the game an interactive advertisement appears. In order to avoid annoying viewers with additional information, the advertisement is an HTML document to be played in the device class (2). As soon as this HTML object is started in class (2), each PDA instantiates an HTML user agent for its own instance of the object. The focus is also placed in these HTML object instances, so that a device in class (2) can navigate through the HTML document without interfering with what is being exhibited in the TV set or in the other PDA. For example, additional information about the product being presented can be obtained and a purchase can be done.

Instead of an HTML document, the aforementioned advertisement could be composed of a set of synchronized media objects. In this case, each PDA instantiates a specific player for each object instance received. The presentation plan, in the root device, must be used to guide the objects distribution to each PDA, where the player-load plan must be calculated to be used to support player instantiation, avoiding undesirable delays.

Also during the match, important news comes in the form of an NCL application. This application is nested in the original NCL application as one of its object. When the time to exhibit this application arrives, the NCL object is sent to the (active) device class (3). The news is then presented in the notebook under its control. During the news presentation, the NCL player in the notebook can send objects to other device classes in its own domain. For example, an audio can be played in device class (4).

In www.telemidia.puc-rio.br/~romualdo, the NCL specification of this application and the specification of the NCL object (news example) can be found.

4. FINAL REMARKS AND FUTURE DIRECTIONS

This paper presents some results and contributions of this PhD thesis and also some work in progress. Work done includes an enhanced set of data structures to support synchronization management in a DTV system. The proposed data structures preserve all temporal relationships among events, including unpredictable relationships, such as viewer interactions and those requiring on-the-fly content adaptations. Moreover, they allow starting or resuming an application at any point in time.

Current synchronization management work involves enhancing the algorithms used to calculate and maintain the synchronization plans. Elastic time computation algorithms, for stretching and shrinking media object presentations, to compensate unpredictable delays are in our plans.

One of the main motivations of this work is to provide support distributed NCL applications, in which NCL objects are played in different exhibition devices. These objects can be the usual media objects or objects whose content are imperative and declarative code spans.

As regarding multiple exhibition devices, many issues still remains to be solved. Although the main entities (classes) and relationships among them, including the presentation and focus distribution, have been defined, several implementation issues must be addressed. The next main step is to define the communication protocol among devices in the same domain, and also among devices in different domains. Implementation of other services like session mobility and viewer's annotations is being also analyzed. As proofs of concept, implementations of our proposal for Windows Mobile, iPhone OS (Operation System) and Symbian OS are under development.

5. ACKNOWLEDGMENTS

I would like to thank my advisor Professor Luiz Fernando Gomes Soares for its supervision. This work is being supported by Telemidia Labs and PUC-Rio.

6. REFERENCES

[1] ABNT 2007. Digital Terrestrial Television Standard 06: Data Codification and Transmission Specifications for Digital Broadcasting, Part 2 – GINGA-NCL: XML Application Language for Application Coding (Brazil, November, 2007).

[2] ARIB Standard B-24 Data Coding and Transmission Specifications for Digital Broadcasting, version 4.0, 2004.

[3] ATSC Advanced Television Systems Committee. 2000. ATSC Data Broadcasting Standard - A/90, 2000.

[4] Buchanan M.C., Zellweger P.T.Specifying Temporal Behavior in Hypermedia Documents. Proceedings of European Conference on Hypertext (Milan, Italy, December 1992). ECHT'92.

[5] Bulterman D.C.A., Jansen J., Kleanthous K., Blom K., Benden D. AMBULANT: A Fast, Multi-Platform Open Source SMIL Player. In Proceedings of ACM International Conference on Multimedia (New York, USA, 2004).

[6] Cesar, P.; Bulterman, D.C.A.; Obrenovic, Z.; Ducret, J.; Cruz-Lara, S. 2007. Non-Intrusive User Interfaces for Interactive Digital Television Experiences. In Proceedings of European Interactive TV Conference (Amsterdam, Netherlands, May). EuroiTV 2007.

[7] Cesar, P.; Bulterman, D.C.A.; Jansen, J.; 2008. Usages of the Secondary Screen in an Interactive Television Environment: Control, Enrich, Share and Transfer Television Content. In Proceedings of European Interactive TV Conference (Salzburg, Austria, July). EuroiTV 2008.

[8] Costa, R.M.R.; Moreno, M.F.; Soares, L.F.G. 2008. Intermedia Synchronization Management in DTV Systems. In Proceedings of ACM Symposium on Document Enginnering (São Paulo, Brazil, September). DocEng 2008.

[9] ECMA International - 1999. ECMA – 262 – ECMAScript Language Specification. 3rd Edition.

[10] ETSI European Telecommunication Standards Institute. 2006. ETSI TS 102 812 V1.2.2 Digital Video Broadcasting "Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.1.1".

[11] ITU-T Consented Recommendation H.761. Nested Context Language (NCL) and Ginga-NCL for IPTV Services. 2009.

[12] Moreno, M.F. Ginga-NCL: Relating Imperative, Declarative and Media Objects. Doctoral Consortium of European Interactive TV Conference (Belgium, June). EuroiTV 2009.

[13] W3C World-Wide Web Consortium. 2004. Document Object Model – DOM Level 3 Specification. W3C Recommendation

[14] W3C World-Wide Web Consortium. 2008. Synchronized Multimedia Integration Language – SMIL 3.0 Specification, W3C Recommendation.