# Implementing spatio-temporal relations for hypermedia presentations using an HTML-like language

C. Bouras[1,2], V. Kapoulas[1,2],D. Moiras[3], V. Ouzounis[4], P. Spirakis[1,2], A. Tatakis[2]

[1] Computer Technology Institute, Patras, Greece

[2] Computer Engineering Dept., University of Patras, Greece

[3] University College of London, London, UK

[4] GMD Fokus, Berlin, Germany

E-mail: bouras@cti.gr

## Abstract

Hypermedia is currently the trend in delivering information composed of various forms of media. Up to now most systems lack the ability to present the user with hypermedia information containing various media that have specific spatio-temporal relations. In this paper we propose and describe a model that can describe such kinds of hypermedia presentations. The core of the model is an HTML-like language that supports tags for the description of timing relations between the information objects that compose a hypermedia presentation.

## Introduction

In our days multimedia tend to be a basic part of all software applications. The reasons for this trend in software, stem from the need of being more natural and friendly to the end user. However the demands of the user are getting higher and higher, thus making software developers produce high quality applications.

Developing a hypermedia application imposes several problems concerning the presentation of the hypermedia information. The spatio-temporal relationships among the different media that comprise the presentation is the most important aspect of a high quality presentation [3],[4],[5]. Synchronisation of all these media requires a model for defining the exact place where these media will be presented and the exact time that their presentation will start.

In this paper we propose a new model for structuring hypermedia documents. The core of this model is a hypermedia mark-up language which supports tags for the synchronised presentation of the various multimedia objects that comprise the hypermedia document. The need for a model meeting these requirements has been mentioned early in this decade. As a result of this identification several standards, aiming in satisfying these goals, have been proposed. (e.g. MHEG [11], Hytime [12]).

In the direction of incorporating hypermedia and especially continuous media, new transport protocols have been developed. The most important of these protocols are the Real Time Protocol (RTP) [6], the Real Time Streaming Protocol (RTSP) [7] and the Resource Reservation Protocol [8]. The basic idea behind these protocols is the usage of a streaming client-server system. The difference between client side streaming and the Web server, is that the last cannot control the rate at which a file is sent. The integration of these new protocols is hoped that will give solutions for achieving real-time delivery of multimedia.

In the following sections we will present the language we propose and some of the implementation issues it entails.

## Our proposal

In order to interchange multimedia objects in a distributed environment and provide a hypermedia document with the ability to regain the original spatio-temporal presentation scenario of its components, there is the need for the existence of a flexible model that should address basic features, which among others include the following:

- Association between the media contents (e.g. text's content) and their presentational characteristics, like text attributes (fonts, size, bold, italics, paragraphs, separators, etc.), colours (background, foreground), etc.
- Placement in space and time (spatio-temporal presentation) of the several media data that compose a hypermedia document.
- Synchronisation in real time of media that should be presented/played together, following the presentation scenario.
- Ability to link the hypermedia objects or the individual components of hypermedia documents, in order to guide the presentation through the various objects.

We started designing a model that would be easier to implement and use than MHEG and which provides modules for a synchronised presentation of hypermedia objects and would cooperate with protocols for real time delivery of hypermedia objects. The first step in this attempt was to design a mark-up language that would include support for timing features and would be capable of handling intermedia synchronisation (i.e. synchronisation between different media).

This language is the core of the model and offers a simple yet flexible way to represent a pre-orchestrated hypermedia presentation. This hypermedia mark-up language is influenced by HTML and follows its simplicity in constructing documents. A hypermedia document, as described by the model, is a text file which contains several *tags* and *keywords* to indicate the specific form of the document, such as media type (text, image, audio, etc.), media placement and annotation, paragraph structuring, text form (fonts, size, alignment, headings, etc.) and other presentation options.

Using this language it is possible to describe the exact place where all the multimedia objects that comprise the presentation will appear, and the exact time that this will happen.

In traditional hypermedia systems, navigation is enabled via either a "sequential" way or an "explorational" way. In sequential navigation, users have to follow the links defined by the creator of the hypermedia document in order to maintain a logical sequence of the selected information topic. On the other side, "explorational" navigation can be used to override the logical sequence.

In our language we include both of these methods and we expanded the notion of sequential navigation by incorporating timing characteristics in the linking method. This means that a specific link may automatically be followed after the expiration of a time period that is properly defined by the presentation scenario.

## Description of the language

In the following paragraphs we give a presentation of the formal grammar that describes the language and a brief discussion of the language itself.

### Grammar of the language

We present the main parts of the language using the Backus-Naur Form (BNF), where terminal symbols appear in capital letters and boldface, while non-terminals appear between < >.

```
<Hdocument> :: = TITLE STRING END_TITLE <HSentence>
<Hsentence> :: = /* empty */
             | <Headings> <Main> <Separator> <Hsentence>
<Next> ::= /* empty */
```

```
            |<HyperLink>
<Headings> ::= /* empty */
            | <Heading1>
            | <Heading2>
            | <Heading3>
<Heading1> ::= H1 STRING END_H1
<Heading2> ::= H2 STRING END_H2
<Heading3> ::= H3 STRING END_H3
<Main> ::= <Par> <Body>
<Separator> ::= /* empty */
            | SEPARATOR
<Par> ::= /* empty */
            | PARAGRAPH
<Body> ::= /* empty */
 | <Document> <Body>
 | <Image> <Body>
 | <Audio> <Body>
 | <Video> <Body>
 | <Audio_Video> <Body>
 | <HyperLink> <Body>
<Document> ::= TEXT <Text> END_TEXT
<Text> ::= /* empty */
            |STRING <Text>
<Image> ::= IMG <ImgOptions> <Source> <Id> <Note> END_IMG
<Audio> ::= AU <AuOptions> <Source> <Id> <Note> END_AU
<Video> ::= VI <ViOptions> <Source> <Id> <Note> END_VI
<Audio_Video>  ::=  AU_VI  <Au_ViOptions>  <Au_ViSource>  <Au_Vi_Id>  <Note>
END_AU_VI
<HyperLink> ::=HLINK <to_HyperText> <TimeOption><Note> END_HLINK
            |HLINK <to_OtherHost> <TimeOption><Note> END_HLINK
<ImgOptions> ::= <TimeOption>
                  |<TimeOption> <OtherImgOptions>
<AuOptions> ::= <TimeOption>
                  |<TimeOption> <OtherAuOptions>
<ViOptions> ::= <TimeOption>
                  |< TimeOption> <OtherViOptions>
<Au_ViOptions> ::= <SyncOption>
                  | <SyncOption> <OtherAu_ViOptions>
<TimeOption> ::= /*empty*/
                  |STARTIME STRING
                  |STARTIME STRING DURATION STRING
<SyncOption>::= STARTIME STRING STARTIME STRING
<OtherImgOptions>::= HEIGHT STRING WIDTH STRING
```

```
<OtherAuOptions> ::= /* empty for the time being...*/
<OtherViOptions> ::= /* empty for the time being...*/
<OtherAu_ViOptions>::= /* empty for the time being...*/
<Source> ::= SOURCE <Filename>
<Au_ViSource> ::= SOURCE <Filename> SOURCE <Filename>
<Id> ::= ID STRING
<Au_Vi_Id> ::= ID STRING ID STRING
<to_HyperText> ::= <Filename>
<to_OtherHost> ::= STRING < HyperLink >
<Note> ::= NOTE STRING
<Filename> ::= STRING
```

Tags may include specific keywords, like TEXT, IMG, etc. to indicate the formation of a presentation component. For example, whatever is included between IMG and END_IMG is the description of an image component of the document.

To include another media type component in the presentation scenario, one can just append it in the document's mark-up, using one of the IMG (image), AU (audio), VI (video), AU_VI (audio & video). The timing characteristics of these media can be described using the STARTIME and DURATION tags while for the AU_VI media type it is also possible to define the synchronisation options with the use of the <SyncOption> rule.

The SOURCE tag defines the files that contain the media data. ID refers to the identification key of the specific component media. It is used to distinguish the various media streams that arrive from the server and need to be processed. Naturally, each component of a hypermedia object has a unique identification number. NOTE refers to an annotation text.

Hyperlinks may also acquire time characteristics. This means that a specific link will be automatically followed after the expiration of a time period that is properly defined by the presentation scenario. This feature, can preserve the sequential nature or "writer's way" of presentation, in the absence of user involvement. This sequence may change if the user chooses another hyperlink at will. The timing characteristics of a link can be defined using the <TimeOption> rule that can be used with the HLINK tag.

We also introduce the notion of media "relative playout start time" and "media playout duration". These relative start time instants represent the playout deadline for a specific stream that should be satisfied in order to achieve a coherent synchronised presentation. We can include in the presentation scenario these set of time characteristics for every media component. By extracting this information in the receiving edge, we can construct a playout schedule of every media stream that follows the initial scenario, as designed by the author.

**Discussion of use**

In order to better explain what the language is supposed to do we will use the following scenario as an example. A hypermedia document contains formatted text that is always shown throughout the presentation. At the start of the presentation (relative time) the image $I_1$ is shown having presentation duration $d_{i1}$. After the appearance of $I_1$, in the time instant $t_{i2}$ after the presentation start, image $I_2$ is shown and kept on the display for duration $d_{i2}$. At time $t_{a1}$ an audio segment $A_1$ should be heard synchronised with a video V of duration $d_V$. The two media should start and stop playing at the same time. In the time $t_{a2}$ an audio segment $A_2$ plays out with duration $d_{a2}$. Figure 1 illustrates the graphical presentation of the scenario and its correspondence in term of playout timelines.

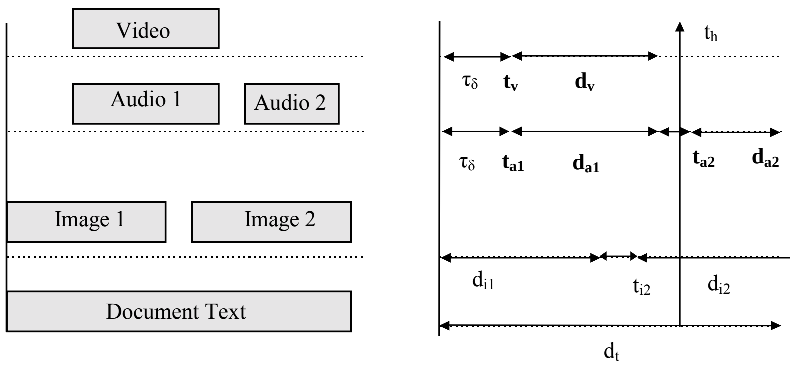
Video
Audio 1
Audio 2
Image 1
Image 2
Document Text
$t_h$
$\tau_\delta$ $t_v$ $d_v$
$\tau_\delta$ $t_{a1}$ $d_{a1}$ $t_{a2}$ $d_{a2}$
$d_{i1}$ $t_{i2}$ $d_{i2}$
$d_t$

**Figure 1:** Graphical and timing illustration of a simple hypermedia scenario

Trying to transform the graphical notation of the scenario in terms of the described mark-up language, we result to the following mark-up description:

**TEXT** text_body **END_TEXT**
**IMG** STARTIME=0 DURATION=$d_{I1}$ SOURCE=$I_1$ ID=... **END_IMG**
**IMG** STARTIME=$t_{I2}$ DURATION=$d_{I2}$ SOURCE=$I_2$ ID=... **END_IMG**
**AU** STARTIME=$t_{A1}$ DURATION=$d_{A1}$ SOURCE=$A_1$ ID=... **END_AU**
**AU** STARTIME=$t_{A2}$ DURATION=$d_{A2}$ SOURCE=$A_2$ ID=... **END_AU**
**VI** STARTIME=$t_V$ DURATION=$d_V$ SOURCE=V ID=... **END_VI**

A client receiving this description, in order to extract all the necessary timing information for the determination of the playout schedule, pre-processes the received presentation scenario. During this pre-processing, every media stream $S_i$ is recognised by its corresponding language rule and a structure $E_i$ is informed. This structure contains stream's $S_i$ timing parameters like start time $t_i$ and duration $d_i$, corresponding data position in the temporary storage mechanisms (media buffers), and other useful information. After acquiring this information, the playout scheduler process can arrange the presentation of each media stream according to its playout deadlines (that are expressed from the time instant $t_i$). For each media stream $S_i$, a concurrent presentation process is created, to play $S_i$ when its deadline is arrived at.

## Implementation issues

In order check the validity of the above described architecture we designed and implemented a prototype browser for this language. We also developed a teletraining system which is based on this architecture and uses the Real Time Protocol (RTP) for the delivery of continuous media.

The lessons are hypermedia documents and are constructed using the hypermedia mark-up language described previously. To integrate the transfer of data and the application protocol, we use the *Real-time Transfer Protocol (RTP)* as a transport mechanism. RTP is an Internet standard, and is used as an intermediate protocol that provides end-to-end transport functions suitable for applications transmitting real-time data such as audio and video. RTP is augmented by a control protocol (RTCP) to allow monitoring of the data delivery.

We use these mechanisms to gain intramedia and intermedia synchronisation using the timing information they provide through packet timestamping, as well as calculating statistical measurements about network's parameters like the transmission delay, jitter and packet loss. We use RTCP to send this feedback information to media servers and to realise the service's application protocol.

### Buffering Scheme

In the receiving edge of the service, the existence of a buffering mechanism is necessary. Incoming media data are temporally stored in multi-thread buffers. Every buffer thread has a different size according to the stored data characteristics. Media data are temporarily stored in the buffer to pass through a decoding procedure before being played. The main usage of a buffering scheme is to accommodate, using a statistical resource reservation policy or by just upgrading or degrading queued media quality, measured or negotiated connection's delay invariant behaviour. Such variant delay may lead to synchronisation or media presentation anomalies (jerky video, gaps in audio) due to buffer underflow.

## Conclusions and Future Work

We proposed a model for hypermedia presentations whose core is an HTML-like language with extended timing features, capable to manage the synchronisation between the objects that comprise it. In the future we are going to extend the capabilities of the language with more presentation features concerning both static and continuous media. Additionally, the structuring of presentations will be given in a more formal way, especially concerning the distribution of multimedia objects and we will try to provide more interaction to the model. Complementary to the augmentation of the language we are going to develop an authoring tool to facilitate the creation of hypermedia presentations. Finally, we would like to gather statistical data about the performance of the RTP layer in order to be able to evaluate its suitability to the task.

## References

[1] C.Bouras, et. al. "An Interactive Cooperative Teleworking Environment: Telemathea". In proceedings of EF-96, Boston, Mass., USA, June 17-22,1996.

[2] C.Bouras, et. al. "An on-Demand Multimedia\Hypermedia Service over Broadband Networks". In proceedings of 5th IEEE International Conference on High Performance Distributed Computing, Syracuse, N.Y., USA, 6-9 August,1996.

[3] R.Steinmetz "Synchronisation properties in a multimedia system", IEEE Selected Areas In Communications, vol8 ,no3, Apr.1990, pp.401-412.

[4] Chun-Chuan Yang and Jay-Hsiung Huang, "A Multimedia Synchronisation Model and its implementation in Transport protocols". IEEE J. Select. Areas on Communications, vol14 ,no1, pp 212-225, Jan 1996.

[5] Herng-Yow Chen & Ja-Ling Wu, "Multisync : A synchronisation model for multimedia systems," IEEE Journal on Selected Areas in Communications, vol 14, No.1 , January 1996.

[6] H.Schulzrinne and et. al. RTP: A Transport Protocol for Real-Time Applications, March 1995. Internet Draft Internet Engineering Task Force.

[7] Real Time Streaming Protocol (RTSP). Internet Draft. Available at: http://www.prognet.com/prognet/rt/protocol.txt

[8] Reservation Protocol (RSVP). Available at: http://www.isi.edu/div7/rsvp/rsvp-home.html

[9] T.Berners-Lee Hypertext Transfer Protocol (HTTP). Working Draft of the Internet Engineering Task Force, 1993.

[10] T.Berners-Lee and Connolly. Hypertext Markup Language (HTML). Working Draft of the Internet Engineering Task Force, 1993.

[11] Thomas Beyer-Boudnik. Wolfgang Effelsbberg. MHEG explained, IEEE Multimedia, Spring 1995, pp. 26-38.

[12] Newcombet al., 1991. The HyTime Hypermedia/Time-based Document Structuring Language, Communications of the ACM, November 1991.