On-demand hypermedia/multimedia service using pre-orchestrated scenarios over the Internet

Christos Bouras* — **Vaggelis Kapoulas*** — **Dimitris Miras**** — **Vaggelis Ouzounis***** — **Paul Spirakis*** — **Antonis Tatakis******

** Computer Engineering and Informatics Department, University of Patras, 26500 Rion (Patras), Greece, and Computer Technology Institute, Riga Feraiou 61, 26221 Patras, Greece*

{bouras,kapoulas,spirakis}@cti.gr

*** University College of London, UK*

**** GMD-FOKUS, Germany*

***** Radiant Technologies SA, Greece*

ABSTRACT: *In this paper we present a unified approach for delivering hypermedia/multimedia objects over broadband networks. The described service consists of several multimedia servers and a set of functions that intend to present to the end user interactive information in real-time. The hypermedia documents are structured using a hypermedia markup language that keeps information of the spatiotemporal relationships among document's media components. In order to deal with the variant network behavior, buffering manipulation mechanisms and grading of the transmitted media quality techniques are proposed to smooth presentation and synchronization anomalies.*

RÉSUMÉ: *Dans ce document nous présentons une approche unifiée pour la fourniture d'objets hypermédias/multimédias sur des réseaux à larges bandes. Le service décrit est constitué de plusieurs serveurs multimédias ainsi que d'un ensemble de fonctions qui visent à présenter à l'utilisateur final une information interactive en temps réel. Les documents hypermédias sont structurés en utilisant un language "mark-up" qui garde l'information sur les liens spaciotemporels entre les différents composants des médias du document. Afin de faire façe à la réaction variable du réseau, des mécanismes de protection contre des erreurs de manipulation ainsi que des améliorations des qualités tecniques des médias transmis sont proposés. Ils visent à atténuer les anomalies de présentation et de synchronisation.*

KEY WORDS: *media synchronization, media on-demand.*

MOTS-CLÉS: *synchronisation de médias, médias en demandant.*

1. Introduction

Recent years have introduced an essential development in several areas of computing technology; high speed networks (broadband ISDN, FDDI, ATM) are dominating, workstations are becoming more and more powerful, new media coding, compression and storage techniques are emerging. At the same time, multimedia technologies are becoming the edge, and new operating system aspects and protocols to deal with multimedia information are being proposed.

All these developments have significantly contributed to the emergence of innovative applications that make the transmission of multimedia data over a network feasible. Such applications have diverse requirements concerning the network, the workstation characteristics, the playout mechanisms, multimedia generation and presentation methods, and may be applicable to distance education, telemedicine, multimedia news services, electronic magazines, remote access to virtual galleries, etc. These applications should be considered as services and may involve such issues as subscription, authentication, pricing and privacy.

A multimedia service may invoke several functional components throughout its operation; database systems, retrieval, transmission and buffering mechanisms, media coding techniques and presentation devices. Each of them requires specific manipulation and may impose presentational malfunction, thus, user dissatisfaction. When dealing with distributed environments, the accomplishment of synchronization among the different media that compose a multimedia object is of major importance. In the following, we identify some key issues concerning an on-demand delivery of multimedia objects.

1. Multimedia objects or documents should be considered as a composition of different types of media (text, images, graphics, audio, video) that are appropriately placed in space and time to form a presentation scenario. In addition, in order to relate various multimedia objects and construct an information web, the notion of *hypermedia* is introduced. To manage effectively such presentations, several models for the integrated modeling of hypermedia documents have been proposed and adopted. *HTML* is widely accepted, yet lacks real-time dimensions in its presentation mechanisms. *HyTime* and *MHEG* are becoming standard. The *Dexter Reference Model* [HAL 94] provides a basis for interchange and interoperability standards. The *Amsterdam Hypermedia Model* [HAR 94] extends the Dexter Model, featuring notions of time, high level presentation and link context. Furthermore, several methods have been proposed for the modeling of storage, retrieval and synchronization of distributed media streams [LI 92, LIT 90, LIT 93, STE 93]. In [STE 90] general issues on synchronization are discussed.

2. Different media streams that participate in a multimedia presentation have diverse transmission requirements concerning the bandwidth, the tolerance to the network's delays, the error rates. Upon connection to the service, a negotiation process should be invoked to reserve the required network parameters. This negotiation process should take into account the user's desirable *Quality of Service (QoS)* parameters

and the network's condition.

3. The existence at the receiving end of a buffering mechanism is essential in order to provide temporary storage for the incoming data before their presentation, as well as to smooth presentation anomalies due to the network's load conditions and probabilistic behavior. Several techniques to handle such buffering schemes in conjunction with the synchronization problem among various media have been suggested, involving buffer sizes and mechanisms that should be activated on the buffer's overflow or underflow conditions, e.g. [LIT 92].

The area of multimedia synchronization continues to receive considerable attention, both in the definition of models and architectures (see e.g. [LAM 96, NAN 97, YAN 99]), and in the implementation of systems and tools (see e.g. [GEO 97, BAI 97]). Also, the issue of providing media synchronization while allowing interactive adjustment of the multimedia presentations has been studied [HUA 98].

In the rest of the paper, we discuss our approach to the aforementioned issues, and we present a design schema for applications that acquire real-time characteristics and require media synchronization support.

2. Service overview

A multimedia/hypermedia service should be considered as a set of functions that intend to present to the end user interactive information that involves a variety of media. A minimal set of such functions that addresses the user involvement should be:

– the subscription to the service as well as authentication and pricing primitives,

– the determination of the user's desired presentation parameters (e.g. the audio or video quality that the user desires),

– the retrieval of a document or a topic of interest,

– the navigation through different hypermedia documents,

– the adjustment of the synchronization and presentation options.

The service consists of a set of multimedia servers distributed over a broadband network. Hypermedia documents are stored in these servers. Each multimedia server may consist of various media servers (image servers, audio servers, etc.). The internal structural presentation of a hypermedia object is stored in a multimedia server, while the inline data that compose the document may reside on their own media servers attached to the multimedia server.

Multimedia documents can be linked via hyperlinks to provide a logical interconnection among related documents. Linked documents may be stored in the same or in different servers.

Requested documents, as well as their media data are transmitted to the connected user's workstation, where specific processes are invoked to handle, buffer and present/play the incoming streams.

The presented design approach addresses the above characteristics, specifying in a transparent way how hypermedia documents are organized and stored, how interactions and data delivery are managed, and how the presentation of the hypermedia documents is performed.

3. Structuring hypermedia objects

In order to effectively store, retrieve, represent and interchange hypermedia objects between distributed places, and also regain their original spatial and temporal presentation scenario at the receiving edge, a reference model should exist to meet the above needs. Such a model should feature:

– media content encapsulation in the integrated context (e.g. a text),

– association between media content and its presentational attributes (e.g. the text's fonts, the presentation's background or foreground colors, the image's dimensions),

– spatio-temporal internal representation of a document's entities, that also implies synchronization among related in time media,

– linking notions among the various hypermedia documents or among the related inline media; the clue feature of the hypermedia abstraction.

Our model is divided into four logical abstractions, namely *content*, *layout*, *synchronization* and *interconnection*.

The content refers to the inline media entities and specifies their characteristics, such as, where they are stored, the encoding standard that is used, etc. The layout consists of a set of rules that internally specify how the different media will be presented on the user's desktop. The synchronization specifies the temporal (time) relationships among the various media. The interconnection provides a method for connecting hypermedia documents with other ones. This is achieved using *hyperlinks*. Hyperlinks may be divided in two categories, namely *sequential* and *explorational*. Sequential links indicate the document that should be followed by the user in order to preserve the logical sequence (or the author's sequence) of the selected information topic. Explorational links can be used to override the logical sequence and to provide access to related information.

We have designed a hypermedia markup language that is influenced by HTML and offers a set of primitives for the presentation and synchronization of the inline media and the interconnection among documents. A document is a composition of different media that are appropriately placed in time and space to form a playout scenario. Every single media (text, image, graphics, audio, video) that takes place in the formation of the playout scenario, is described by the markup language. Each media has its static characteristics, such as the playout or the content structure, and its time characteristics, such as the relative time this particular media is played out according to the presentation scenario, the playout duration, etc. The representation of a document

by the markup language is actually a text file. Several *tags* and *keywords* are used to denote a specific meaning or operation. The conceptual ideas for representing the layout structure are borrowed by HTML. The tags and the keywords are used to indicate a specific form of a document, such as, indication of media type (text, image, etc.), media placement and annotation, paragraph structuring, text alignment, headings and other presentation primitives.

Despite HTML's wide acceptance, its big disadvantage is the lack of synchronization primitives for media with time dependent characteristics (e.g. audio, video). We try to overcome this problem by adding to the markup language time features. Issues on the proposed markup language are discussed in [BOU 97, BOU 98a].

The key point is the introduction of the "media relative start time". The different media have a specific presentation start time, relative to the presentation start time of the whole document, and a duration. Thus, by allowing each media to have its own start time and embedding this feature in the markup language, we gain a simple way of representing media in time. A simple synchronization can be achieved using this "start time" feature.

Using this "startime" indication of each media that participates in a scenario, and taking into account its presentation duration, a playout scheduler at the client side, invokes a concurrent playout process to present the media data on the proper device before their deadline, as it is dictated by the "startime" attribute.

It should be noted that the W3C has proposed the SMIL language as a standard for describing interactive synchronized multimedia distributed on the Web [W3C98]. SMIL is based on XML and provides users with a lot of functionality. On the other hand our approach aims at simplicity. Also, SMIL requires a SMIL player, and does not integrate well with HTML (i.e. cannot be used to synchronize the presentation of media within an HTML page)[1]. Our proposal extends HTML, and with an equivalent extension of the browsers capabilities can handle synchronization of media within an HTML page.

3.1. *Description of the language*

We briefly describe the basic principles and construction elements of the language, in order to show how it achieves the internal coding of the hypermedia document structure. We do not intend to give the full description of the language's usage, but to address some basic concepts to base on for further discussion. The grammar of the language is shown in Figure 1.

The basic elements of the model are keywords, such as `TITLE`, `H1`, `H2`, `H3`, `PAR`, `SEP`, etc. and tags such as <, >, /. In Table 1, the most useful keywords of the language are presented.

1. This disefficiency has been recognized by W3C, and in the next version of SMIL, it will be broken into modules that can be used individually in XML document sets

```
<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> <Note> END_HLINK
                         | HLINK <to_OtherHost> <Note> END_HLINK
<ImgOptions>         ::= <TimeOption>
                         | <TimeOption> <OtherImgOptions>
<AuOptions>          ::= <TimeOption>
                         | <TimeOption> <OtherAuOptions>
<ViOptions>          ::= <TimeOption>
                         | <TimeOption> <OtherViOptions>
<Au_ViOptions>       ::= <SyncOption>
                         | <SyncOption> <OtherAu_ViOptions>
<TimeOption>         ::= STARTIME 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
```

Figure 1. *Grammar of the language in BNF notation*

Keyword	Description
TITLE	Document title indicator
H1, H2, H3, ...	Heading indicators
PAR, SEP	Paragraph and separator indicators
TEST, IMG, AU, VI	Media type indicators
SOURCE, ID	Media source and id indicators
STARTIME, DURATION	Media time characteristics indicators
I, B, U	Boldface, italics, underline characters
NOTE	Annotation indicator

Table 1. *Description of basic keywords*

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 </IMG> is the description of an image component of the document.

Using structural indicators, we can determine the layout format of a media, for example the following rules determine a text with a title, a heading of type 1, and two text segments separated by a new paragraph between them.

```
<TITLE> This is a title </TITLE>
<H1> This is a heading 1 </H1>
<TEXT> This is a text segment </TEXT>
<PAR>
<TEXT> This is another text segment. <B> This is
boldface. </B> <I> And this is in italics. </I> </TEXT>
```

Text included between <B> and </B> is in boldface, and that between <I> and </I> is in italics.

To include another media type component in the presentation scenario, one can just append it in the documents rules, determining all the required characteristics of this media, e.g. to describe an image component, one may use:

<IMG> SOURCE=*retrieval_options* WHERE=*coordinates* ID=*component_id* NOTE=*annotation*
</IMG>

The SOURCE keyword introduces several retrieval options concerning the associated media stream. Such options are information about the storage of data and other options that are based on the database model used by the service to store and retrieve multimedia data. Keyword WHERE introduces placing attributes in media's representation, such as image's coordination on the display device. ID refers to the identification key of the specific component media. Its importance is great, since it is used to distinguish the various media streams that arrive from remote servers and need to be processed. Naturally, each component of a hypermedia object has a unique identification number. NOTE may refer to an annotation text.

As shown above, this markup language provides an easy way to build the layout structure of a document, as well as the document's form, the media objects' participation in the scenario, and the spatial relationship determination among the various components. The model preserves its simplicity and flexibility, and it is easily extendible to more complex presentation requirements, and work is in progress towards this.

Linking capabilities between different documents is hypermedia's most exceptional feature. Interconnectivity advances navigation opportunities to a wide web of connected hyperdocuments. Hyperlinks that realize this feature, can be divided into two categories, namely sequential and explorational. Sequential links indicate the document that should be followed by the user to preserve the logical sequence of the selected information topic. Exploration links can be used to override the logical sequence and to provide access to related information.

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. All the above are realized in the described model using the keyword HLINK, e.g. as follows:

<HLINK> AT *time_instance linked_document* NOTE=*annotation* </HLINK>

The keyword AT (optional) is used to activate the hyperlink to the next, based on the scenario, hypermedia document if the defined time has elapsed. This is how the sequencing of presentation units can be preserved.

Despite HTML's wide acceptance, its big disadvantage is the lack of time dimensions in the presentation of its inline described media, and consequently the lack of synchronization among related media (e.g. audio and video).

We attempt to overcome this problem by adding basic timing features to the specification of markup language. This is achieved by introducing 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 synchronized presentation. Bearing this in mind, for every media stream S_i that participates in the multimedia t_i refers to S_i's relative playout start time and d_i its playout duration. 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 is coherent to the initial scenario, as designed by the author.

The markup language expression that indicates the above time values is, e.g. for a video media description:

<VI> SOURCE=*retrieval_options* ID=*component_id* STARTIME=*startime*
DURATION=*playout_duration* NOTE=*annotation* </VI>

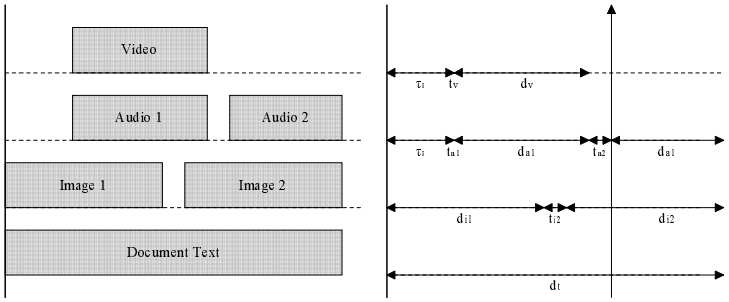

Figure 2. *Graphical and timing illustration of a simple multimedia scenario*

Where the keywords STARTIME and DURATION are introduced to indicate a video's segment starting time instant and playout duration respectively.

After presenting how our markup language model interprets the basic hypermedia primitives (content's layout structuring, linking with other documents and the introduction of the time notion in the hypermedia scenario), we show how a multimedia scenario is constructed using the described markup language, and how a coherent playout presentation can be scheduled.

Assume the following hypermedia scenario: Hypermedia document contains a 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 synchronized 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 2 illustrates the graphical presentation of the scenario and its correspondence in terms of playout timelines.

Trying to transform the graphical notation of the scenario in terms of the described markup language, we result to the following part of language rules/expressions:

<TEXT> *text_body* </TEXT>
<IMG> SOURCE=I_1 ID=...STARTIME=0 DURATION=d_{i1} </IMG>
<IMG> SOURCE=I_2 ID=...STARTIME=t_{i2} DURATION=d_{i2} </IMG>
<AU> SOURCE=A_1 ID=...STARTIME=t_{a1} DURATION=d_{a1} </AU>
<AU> SOURCE=A_2 ID=...STARTIME=t_{a2} DURATION=d_{a2} </AU>
<VI> SOURCE=V ID=...STARTIME=t_v DURATION=d_v </VI>

At the client's side, in order to extract all the necessary timing information for the determination of the playout schedule, preprocessing of the received presentation scenario is taking place. In this preprocessing, every media stream S_i is recognized by its corresponding language rule and a structure E_i is informed. This structure contains

the stream's S_i timing parameters like start time t_i and duration d_i, the corresponding data position in the temporary storage mechanisms (media buffers), and other useful information. 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 creating, to play S_i when its deadline expires. The playout algorithm becomes very simple, and can be described as below:

```
for i = 0 to number of structures E_i
   Create a playout thread (i.e. a playout process)
   wait until current relative time = t_i (i.e. wait until presentation time arrives)
   Play incoming stream S_i in nominal rate for duration d_1
end
```

However, the activation of a hyperlink by the user in a time instant th will interrupt the presentation of the scenario, and trigger the operations for the presentation of the linked hypermedia document.

4. The design architecture

Our design effort is concentrated on four major issues concerning a hypermedia/multimedia service. These four issues are:

– the management of user's requests for viewing documents,

– the manipulation of media streams transmission,

– the buffering and synchronization of various media,

– the presentation of the hypermedia document.

In Figure 3, the design approach is depicted. The main ideas are found in [BOU 96, ANT 97b, BOU 98b].

A first issue that arises when dealing with such services, is the invocation, upon connection request, of a connection establishment mechanism in order to provide access to the service. This mechanism evaluates a set of parameters concerning the network and the connection's request options, to decide on connection admission or rejection. Such parameters are the network's condition the specific time the request is sent (e.g. network load, available bandwidth) and the potential load that will be caused due to the new connection. The load that a new connection introduces, is a combination of the resource requirements the data that should be transmitted holds (e.g. bandwidth, interarrival delay, delay jitter, packet loss probability), and the lower thresholds in QoS and Quality of Presentation the user is willing to accept.

The above parameters are evaluated in conjunction with the pricing contract of the specific user (a user who pays more should be serviced, even though it affects the other users), and the connection is either initiated or rejected.

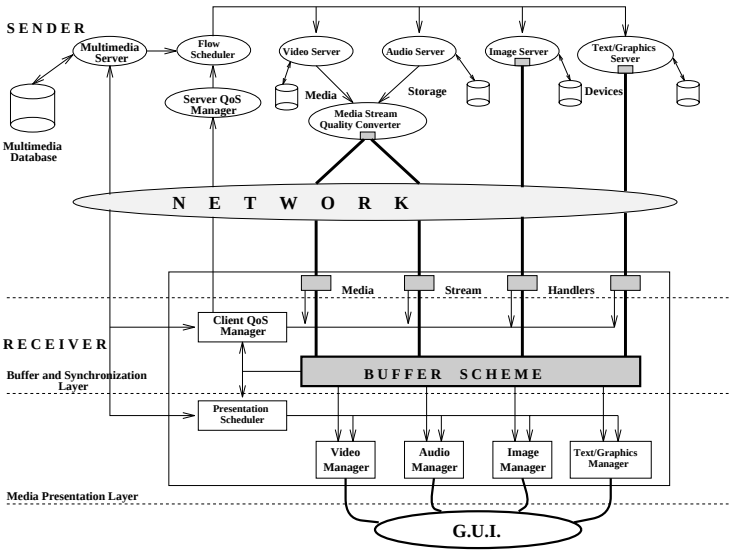

Figure 3. *The general architecture*

Upon connection, the multimedia server is activated to retrieve the requested hypermedia document. The server sends the presentation scenario, that is actually the hypermedia markup language representation of the document, to the *presentation scheduler* at client's side. The presentation scenario, as previously mentioned, specifies the spatiotemporal relationships among the media objects that compose the requested document. The presentation scheduler, by processing the presentation scenario, determines what media streams participate in the multimedia scenario, and when they should be invoked. This triggers the initialization of the corresponding media stream handlers, the associated buffer handlers, and the appropriate media presentation handlers. In addition, the presentation scheduler is responsible for the preservation and satisfaction of the time and presentation constraints of involved and related media, thus responsible for the *inter-* and *intra-media synchronization*.

At the server's site, the *flow scheduler* uses the retrieved from the *multimedia database* presentation scenario to compute a *flow scenario* for each participating media stream. This flow scenario specifies the sending start time instances of the corresponding media streams, as well as other transmission properties (e.g. transmission rates). Furthermore, it activates the appropriate media servers.

In order to deliver the various media data, each activated media server should set up the appropriate network connections. Each of these connections may have its own characteristics, based on the QoS requirements of the sending stream. Setting up a network connection, may be a static task (meaning that once its QoS parameters are determined, upon establishment of the connection, they remain the same throughout its life) or dynamically adjustable; such a protocol is described in [KRI 94].

Network connections are experiencing significant delays, delay variation, and data loss in times of network congestion, which may cause undesirable disruptions in the presentation, especially affecting the time sensitive streams, like audio and video, and leading to gaps in audio/voice, jerky video and/or synchronization failure among related streams.

To face this problem partially, we introduce in our design schema a QoS manager at both the sending and receiving edges. Incoming data packets of a specific stream, besides other information, carry a timestamping indication which is used by the *Client QoS Manager* to carry out conclusions about the connection's condition, e.g. the packet delay, the delay jitter. Based on this information, the client QoS manager, periodically or in specifically calculated intervals, sends feedback reports to the sending side, the *Server QoS Manager*. Using such feedback reports, the service's server possesses knowledge of the overall network performance parameters, and accordingly takes corrective actions to improve the service's operation. Flow scheduler identifies the specific media streams that are not transmitted as desired, and in cooperation with the corresponding *Media Stream Quality Converter* gracefully degrades (upgrades) the stream's quality, e.g. by increasing (decreasing) video compression factor or decreasing (increasing) audio sampling frequency. This results in less network traffic, thus more available bandwidth.

When dealing with the transmission of audio and video streams that should be synchronized, the service first applies the grading technique to the video stream, since audio or voice is considered to be more important to users, meaning that users can tolerate lower video quality rather than "not hear well". Besides, video streams have been turned out to be much more bandwidth consuming. Degrading media quality may be done down to several thresholds, taking into account at the same time the user's desired levels of presentation quality, as have been expressed during the connection request. When falling to the lower threshold, the service may choose to stop transmitting the specific stream. The service should gracefully upgrade the media quality, when the network's condition permits it. With the above described media quality grading technique, we provide the service with a mechanism for *long term* recovery support to synchronization anomalies.

In addition to the above described media degradation mechanism, the existence of a buffering scheme is needed to provide additional synchronization support. The recognition of the need of such a buffering scheme is constituted of the following reasons:

– a place should exist for the temporary storage of incoming data, before they are forwarded to be played/presented,

– this amount of temporarily stored data is used to coarsely smooth the variations of the receiving rates of the various data streams, caused by network probabilistic behavior,

– furthermore, special mechanisms may be applied to deal with synchronization disruptions among related media. Such a mechanism that involves the monitoring of

buffer occupancy with several actions concerning the synchronization of the corresponding media streams, using *dropping* and *duplication* is outlined in [LIT 92].

The described buffer, is a multiple thread queue; each thread is initialized after the establishment of its corresponding media connection.

One basic concept of the buffering layer, is that after the establishment of the parallel media connections, there is a relative delay in the presentation start time of the requested hypermedia object. This initial delay is inserted on purpose in order to feed each involved media buffer with a quantity of data. This quantity is statistically calculated at the buffer's setup time, and depends on the specific transmitted media stream characteristics (frame/sample size, transmission rates, media encoding properties, tolerance to network delays). This length of each media buffer corresponds to a playback time, and we call this time interval, *media time window*.

The media time window is primarily used to smooth delays inserted by the network, the operating system, the transmission/receiving mechanisms. In this way, the experienced delays on data arrival first affect (decrease) the specific media time window (buffer's length) before affecting the quality of presentation and synchronization.

Furthermore, introduced data arrival delay variations and data loss lead to intermedia synchronization corruption, and a combinational algorithm, that involves the buffer's occupancy levels and the presentation scheduler, should be applied to provide synchronization recovery. Transmitted frames/blocks have a presentation deadline but additionally, distinct frames/blocks that belong to related (synchronized) streams have timing constraints corresponding to their playout times with respect to each other.

Media synchronization may be categorized in:

– *intramedia synchronization*, and

– *intermedia synchronization*.

Intramedia synchronization is satisfied when every media object (frame/sample/block) is available (delivered) for playing within its playout deadline. Intermedia synchronization refers to related, due to a presentation scenario, media objects, and is achieved if the temporal requirements among these objects are met, on existence of timely delivery. *Intermedia skew* refers to the difference of the arrival times among media objects that should be synchronized.

When the buffer monitoring mechanism experiences buffer underflow, the presentation scheduler may lead to frame duplication in order to avoid noticeable gaps in presentation. Correspondingly, when buffer's occupancy exceeds some upper threshold, the scheduler should drop frames to decrease the buffer's data. If intermedia skew is introduced among synchronized streams, which is caused by the buffer's underflow or overflow conditions, the scheduler may drop frames from the stream that leads in time or duplicate frames of the lagging stream in order to maintain a better synchronization. In this way, a *short term* synchronization incoherence recovery method

is provided, before the long term synchronization support mechanism in the sending side is activated to provide media encoding grading.

5. Functional description

In the sequel, we describe the actions concerning the service's application protocol, that are the changes in the service's states that occur due to the various users and/or service's interactions/responses.

Initially, the user requests to connect to an existing server. An authentication primitive is invoked at the server side, in order to check that the user has the right to access the service. If the user is not a member of the service, the application prompts the user to fill in a subscription form. This form contains personal data such as name and address, telephone, e-mail, etc. By transmitting the form to the service's server, the user accepts the pricing policy and information content privileges. This form is transmitted to every server of the service, and a database entry of authorized users is updated while the pricing mechanism is initialized. The user may now access the contents of the service.

After the subscription primitive is invoked, the list of available topics (contents of the service) is sent by the connected server. From now on, the user can access the service requesting topics of interest. These topics can be stored in various multimedia servers. When the user requests a topic, this request is forwarded to the particular server where it is stored. For every associated document, the server where this document is stored is specified. The first document of the selected topic is transmitted and presented to the client's desktop. From the application point of view, sequential and explorational links are managed in the same way. If the requested document is stored in another multimedia server, a suspend connection primitive is invoked and a request for a new connection with a new server is performed. The suspended connection remains active for a period of time, in case the user requests to view a previous selected document. When this interval is passed the connection closes and the attached client is informed about the event. At that time client has an active connection with a new multimedia server. The user can issue a disconnect request from the service, at any time. The pricing primitive is informed about the request and the connection closes. In Figure 4, all the above mentioned transitions are depicted.

Interactive operations can be triggered by the user during the presentation of a document. The user can pause the presentation of the document and this request is forwarded to the involved media servers in order to stop transmitting the corresponding stream data. The user can request to resume presentation from the point it was paused. Additionally, the user can request to reload an already selected document or to disable the presentation of a particular media involved in the selected document. The user may also annotate the selected document with his own remarks.

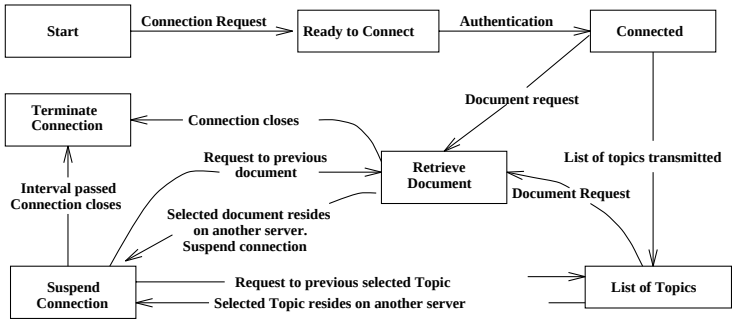

Figure 4. *Application state transition diagram*

6. Application to distance education

The proposed design approach has been followed for the implementation of a prototype hypermedia service for distance education purposes (called Hermes) [ANT 97a].

In the following subsection, an overview of the internal architecture of the service is given. In the sequel, the functional description and samples of the Hermes browser are provided. Particularly, the "connect with the service", as well as the "subscription" operations are discussed. Finally, the "search", the "browsing", which is the core functionality of the service, and the "asynchronous interaction between the tutor and the users", are introduced.

6.1. *Architecture overview*

The architecture of the Hermes service is based on the client-server model. The basic components that comprise it are as follows:

– The *Hermes server* where the lessons are stored. For every lesson a presentation scenario is associated. The presentation scenario of a lesson actually describes the spatio-temporal relationships among various media objects. The inline media objects are stored in the corresponding media servers. When a request for a specific lesson is triggered, the specific server where that lesson resides on, is contacted. The server searches the database of provided lessons and locates the requested lesson. When the presentation scenario is found, the Hermes server activates the involved media servers in order to start the transmission of the corresponding media objects which are referred to the presentation scenario on an on-demand basis.

– *Media servers* in which media objects are stored. For every media object (e.g., text, image, audio, video, etc) a media server is associated with each Hermes server. These media servers may be located in the same host with the Hermes server and each one is responsible for transmitting a certain media type through a parallel connection

which is established between the browser and the corresponding media server. The media objects involved are transmitted from the media servers towards the browser according to the presentation scenario and the presentation constraints. The transmission process of each media object is adjusted according to the feedback reports which are sent by the Hermes browsers and describe the status of the network and the presentation process.

– The *Hermes Browser* is the integrated graphical environment through which the user interacts with the service by retrieving and browsing lessons. The browsers may run on a multimedia PC and facilitate all the provided functionalities of the service. The browsers also manage the synchronized presentation of the requested hypermedia lesson. Internal mechanisms monitor the status of the network and report the network and synchronization anomalies to the corresponding Hermes server and media servers in order to adapt the transmission rates. In that way, the presentation quality is significantly improved while the resources needed are dynamically reserved.

6.2. *Functional description*

Since Hermes is a service designed to be used from users, we focused on simplicity and functionality. It is simple, so as to be used efficiently by users unfamiliar with information technology. Additionally, since Hermes is an on-line service, a lot of perplexing details are hidden from the user. The integrated environment, the service functionalities and the rationale of hypermedia help the user to efficiently interact with the service.

In the sequel, the basic features that make Hermes Service a useful service for educational purposes are introduced.

– Connection-Subscription with the service

– Search for related topics

– Viewing a lesson

– Asynchronous interaction between the tutor and the users

6.2.1. *Connection-Subscription with a Hermes server*

The connection primitive is the first action that the user must follow in order to access the service. Initially, the user must specify the Hermes server that he wishes to connect to. For that reason, a list of available Hermes servers is provided. For every Hermes server, a small description concerning the kind of lessons that are stored in it, is presented. Every Hermes server contains lessons concerning specific and well known thematic units. Therefore, the user may easily access his preferred server.

When the user specifies the Hermes server that he wishes to connect to, an authentication process takes place in order to verify whether the specific user is authorised to use the service or not. If the user is an authorized, he is requested to subscribe to the service filling a subscription form. The subscription form contains a variety of infor-

mation, such as the real name, the address, the postal and e-mail address and telephone number of the user.

A coherent, centralized database of authorized users is updated, and from now on the user may easily access the service. Whenever a user is connected with the service, the database of the authorized users is updated and specific information about the exact time logged into the service, as well as the lessons that are retrieved are captured.

6.2.2. *Search for related topics*

In the event of the user being interested in a certain topic, he is able to search the contents of the whole service by invoking the search primitive. In particular, the user specifies the search token which best describes the topic of interest, and selects the server that is likely to contain lessons on the topic. The user selections are transmitted to the specified server and in the sequel, all the text documents stored in that server are scanned. Additionally, this particular server sends the query to all other Hermes servers for the same reason. In every server the same procedure is performed. The results of the query on every server are forward to the initial server and then directly to the user. It must be noticed that only the lessons which contain the item of interest and the server location are transmitted and presented to the user. In this way, the user may search the contents of the service and retrieve only specific lessons on particular topics, avoiding navigating through all the provided lessons. In addition, the user using a friendly dialog box can specify the search options.

6.2.3. *Viewing a lesson*

After the connection has been established, a list of available lessons which are stored on that server is presented to the user. The user may select a specific lesson from the list by "clicking" on the corresponding hyperlink. In that case, the request is forwarded to the server. The server upon request retrieves the presentation scenario of the lesson and determines the inline media objects as well as the spatio-temporal relationships among them. For every media object referred to in the presentation scenario, the server determines the media server where it is stored, and specifies the transmission requirements for that object. Consequently, all the associated media servers are contacted and the transmission requirements for every media object are explicitly determined. Every media server, builds a parallel connection with the browser, retrieves the media object, transmits the data on on-demand basis based on the transmission requirements. During the transmission of the media objects feedback reports are sent to the media servers involved and the Hermes server in order to adjust the transmission rates or the enconding format.

At the browser site, the presentation scenario of the requested lesson, which is sent by the Hermes server, is scanned and the resources needed are reserved. Every media object is temporarily stored inside the buffers, and is then presented to the user according to the presentation scenario [KRI 94, LI 92]. The browser during the transmission of the hypermedia lesson collects information about the network status and the presentation quality, i.e. the presentation anomalies that may occur.

From the application point of view, the user may follow sequential or explorational links. The sequential links specify a coherent view of the lesson, i.e. the tutor's way, while the explorational links are relative to theme hypermedia documents with additional information on specific topics. In both cases, the browser manages the user selections in the same way. If the selected lesson or topic is stored on another server, a suspend connection primitive is invoked, and a request for a new connection with the new server is triggered. The suspended connection remains active for a period of time in case that the user requests to view a previously selected document. When this interval is passed, the connection closes and the attached client is informed about the event. At that time the client has an active connection with the new server. At any time the user may disconnect from the service by "clicking" on the appropriate button.

Among the several facilities that can be supported by the browser are the following:

– Moving backward and forward in the list of already viewed lessons. This can be achieved with the use of menu buttons.

– Scrolling of the hypertext document.

– On-line help provision.

– Interactive operations can be triggered during the presentation of the lesson. The user can pause the presentation of the lesson and resume later from the point that it was paused via control buttons.

6.2.4. *Asynchronous interaction between the tutor and the users*

In the case of the user wishing to make comments or submit questions concerning a lesson, he is able to use the e-mail service offered by Hermes. The user can send e-mail to the tutor asking for additional information about a theme. In that case, the tutor can send replies to the user prompting him/her to retrieve specific lessons from the service. Using this off-line mechanism an immediate connection between the user and the tutor is achieved and as a consequence, the educational activity is improved. Therefore, the user may avoid the useless navigation through relative, but not specific lessons, of interest.

6.3. *Implementation issues*

A prototype for the Hermes server has been developed under the Unix operating system. The Hermes browser, however, has been implemented both under Windows 95 and Unix. The prototype is used for the delivery of pre-structured multimedia lessons to remote users. The lessons are internally presented using the above mentioned hypermedia markup language [BOU 97]

The implementation of the network support primitives that are needed for the realisation of the on-line operation is based on the TCP/IP protocol suite. The hypermedia presentation scenario and the non time sensitive media objects (e.g., text, images, graphics) are transmitted through TCP connections. For the transmission of time sen-

sitive media objects (e.g. audio, video) we make use of the *Real-time Transport Protocol (RTP)* [SCH 95]. The RTP is an intermediate protocol that provides an application with end-to-end functions for transmission of real time data, such as audio and video. RTP data packets contain, besides pure data, auxiliary information such as:

– a timestamp, indicating the packet's transmission or data sampling time instance,

– packet sequencing information,

– the packet's data payload type (that is the coding format equivalent representation),

– other features.

RTP works over UDP/IP to provide end-to-end delivery of data such as audio and video. The RTP header provides the timing information necessary to synchronize and display audio and video data and to determine whether packets have been lost or it they arrive out of order. Another advantage of RTP is the ability to specify the payload type in the header, thus allowing multiple data and compression types. The RTP itself does not provide any mechanism to ensure timely delivery or other quality-of-service guarantees, but relies on lower-layer services to do so.

RTP is followed by a control protocol *(Real-time Transport Control Protocol, RTCP),* to support several actions, such as monitoring the parameters of an ongoing session. The primary function of RTCP is to provide feedback information. We use this packet's header information to derive statistical measurements concerning network's parameters like packet's transmission delay, delay jitter and packet loss. RTCP feedback packets containing this kind of information/measurements are sent back to the sender, as *receiver's reports*. This feedback information is important for many reasons. First of all the server can decide whether to reduce or increase the transmission rate, by monitoring the delays between the packet arrivals and the delay jitter. Also, using the timestamping information the packets carry, the media skew between the frame's/block's playout and real-time times can be calculated, thus the buffer's monitoring mechanisms can be enabled, as previously described. RTCP also keeps track of the participants in an RTP session and helps in diagnosing distribution faults. We used the RTCP protocol for the transmission of the control data, and the RTP protocol for the transmission of the presentation scenario and the inline media objects.

The interaction between the student and the teacher is implemented via e-mail. The protocols used for this purpose are SMTP and MIME. As can be seen from Figure 5 the supported formats for images are GIF, TIFF, BMP and JPEG. In addition to these standards, PCM, ADPCM and VADPCM are the audio supported standards. Video is displayed at the student's screen in AVI or MPEG format depending on the availability of bandwidth between the client and the server.

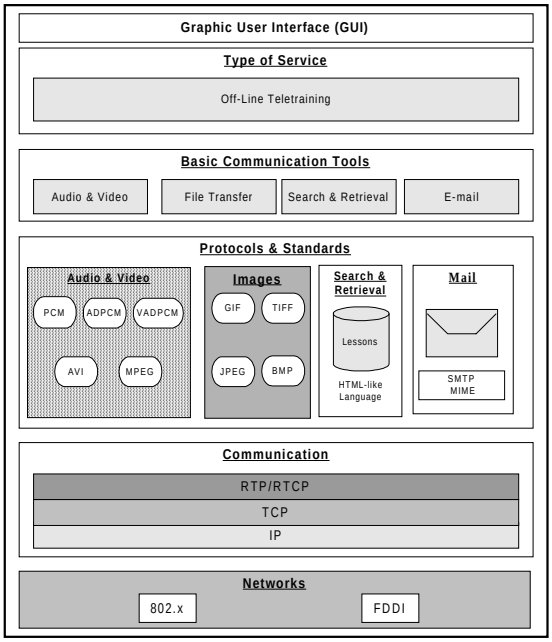

Figure 5. *Protocol stack*

7. Conclusions and future work

The proposed design scheme introduces a method for delivering on-demand hypermedia/multimedia documents over broadband networks. Several issues related to the media streams transmission, buffer and synchronization management and presentation are identified, and solutions are proposed. I particular, we described two closely related methods (long and short term) to deal with the synchronization disruptions in periods of network load.

We also presented a general purpose distance learning service that actually relies on the described design scheme. A prototype has been implemented, tested and validated in various workstation environments and network platforms.

Future work will focus on the improvement of the synchronization method used in conjunction with the buffer's monitoring mechanisms, as well as the implementation of a testbed application on an ATM network. Special effort will be made in the further improvement of the hypermedia markup language to support more complicated presentational features.

8. References

[ANT 97a] ANTONIOU I., BOURAS C., KAPOULAS V., KARAGEORGOPOYLOS D., MOIRAS D., OUZOUNIS V., SPIRAKIS P., "Hermes Service: Distributed Hypermedia Educational Services on Demand", *Proc. of ED-MEDIA/ED-TELECOM 97*, June 1997, p. 19-24.

[ANT 97b] ANTONIOU I., BOURAS C., MIRAS D., OUZOUNIS V., SPIRAKIS P., TATAKIS A., "HY-MOST: Hypermedia Model for Synchronised Presentations", *Proc. of the 3rd Workshop on Open Hypermedia Systems*, April 1997, p. 18-23.

[BAI 97] BAILEY B., "Nsync - A Constraint Based Toolkit for Multimedia", *Tcl Workshop, Boston*, 1997.

[BOU 96] BOURAS C., KAPOULAS V., MIRAS D., OUZOUNIS V., SPIRAKIS P., TATAKIS A., "On-Demand Hypermedia/Multimedia Service over Broadband Networks", *Proc. of the 5th IEEE International Symposium On High Performance Distributed Computing (HPDC-5)*, August 1996, p. 224-231.

[BOU 97] BOURAS C., KAPOULAS V., SPIRAKIS P., TATAKIS A., "An HTML like language supporting time-depended transmission of hypermedia", *Proc. of The Eighth International ACM Hypertext Conference-Hypertext 97*, April 1997.

[BOU 98a] BOURAS C., KAPOULAS V., MIRAS D., OUZOUNIS V., "A Framework for a Distributed Information Service Using Hypermedia/Multimedia Pre-Orchestrated Scenarios", *Proc. of the 10th International Conference on Parallel and Distributed Computing and Systems (Special Session on Communication and Computing for Distributed Multimedia Systems)*, October 1998, p. 226-229.

[BOU 98b] BOURAS C., OUZOUNIS V., SPIRAKIS P., "On Demand Delivery of Multimedia Documents using Distributed Objects", *Proc. of the 2nd International Conference on Parallel and Distributed Computing Networks (PDCN 98)*, December 1998.

[GEO 97] GEORGANAS N., JARMASZ J., "Designing a Distributed Multimedia Synchronization Scheduler", *Proceedings of the 1997 International Conference on Multimedia Computing and Systems (ICMCS '97) , Ottawa, Ontario, Canada*, June 1997.

[HAL 94] HALASZ F., SCHWARTZ M., "The Dexter Hypertext Reference", *Communications of the ACM*, vol. 37, no. 2, 1994, p. 30-39.

[HAR 94] HARDMAN L., BULTERMAN D., ROSSUM G., "The Amsterdam Hypermedia Model: Adding", *Communications of the ACM*, vol. 37, no. 2, 1994, p. 50-62.

[HUA 98] HUANG C.-M., WANG C., "Synchronization for Interactive Multimedia Presentations", *IEEE MultiMedia*, vol. 5, no. 4, 1998.

[KRI 94] KRISHNAMURTHY A., LITTLE T., "Connection-Oriented Service Renegotiation for Scalable Video Delivery", *Proc. of the 1st IEEE Intl. Conf. on Multimedia Computing and Systems (ICMCS '94), Boston, MA*, vol. 5, May 1994, p. 502-507.

[LAM 96] LAMONT L., LI L., BRIMONT R., GEORGANAS N. D., "Synchronization of Multimedia Data for a Multimedia News-on-Demand Application", *IEEE MultiMedia*, vol. 14, no. 1, 1996.

[LI 92] LI L., KARMOUGH A., GEORGANAS N., "Synchronization in Real-Time Multimedia Delivery", *Proc. IEEE ICC '92, Chicago, USA*, June 1992.

[LIT 90] LITTLE T., GHAFFOOR A., "Synchronization and Storage Models for Multimedia Objects", *IEEE Journal on Selected Areas on Communications*, vol. 8, no. 3, 1990, p. 413-427.

[LIT 92] LITTLE T., KAO F., “An Intermedia Skew Control System for Multimedia Data Presentation”, *Proc. 3rd Intl. Workshop on Network and Operation System Support for Digital Audio and Video, La Jolla, CA*, November 1992, p. 121-132.

[LIT 93] LITTLE T., GHAFFOOR A., “Interval Based Conceptual Models for Time-Dependent Multimedia”, *IEEE Transaction on Knowledge and Data Engineering*, vol. 5, no. 4, 1993, p. 551-563.

[NAN 97] NANG J., “A New Multimedia Synchronization Specification Method for Temporal and Spatial Events”, *Proceedings of the 1997 International Conference on Multimedia Computing and Systems (ICMCS ’97) , Ottawa, Ontario, Canada*, June 1997.

[SCH 95] SCHULZRINNE H., ET. AL., “RTP: A Transport Protocol for Real-Time Applications”, March 1995, Internet Draft, Internet Engineering Task Force.

[STE 90] STEINMETZ R., “Synchronization Properties in Multimedia Systems”, *IEEE Journal on Selected Areas on Communications*, vol. 8, no. 3, 1990, p. 401-412.

[STE 93] STEINMETZ R., “Interval Based Conceptual Models for Time-Dependent Multimedia”, *IEEE Transaction on Knowledge and Data Engineering*, vol. 5, no. 4, 1993, p. 551-563.

[W3C98] “Synchronized Multimedia Integration Language (SMIL) 1.0 Specification”, June 1998, W3C Recommendation.

[YAN 99] YANG Z., SUN C., SATTAR A., YANG Y., “A New Look At Multimedia Synchronization in Distributed Environments”, *Proceedings of the Fourth International Symposium on Parallel Architectures, Algorithms, and Networks, Fremantle, Australia*, June 1999.