An Advanced Multimedia Infrastructure for WWW-based Information Systems

Franck Rousseau and Andrzej Duda
LSR-IMAG
BP 72
38402 St Martin d'Hères Cedex, France

Abstract

We present an advanced multimedia infrastructure that integrates continuous real-time media streams within the WWW. This new service architecture focuses on advanced facilities for dealing with multimedia contents. A mediation server is at the heart of the infrastructure. It organizes and structures the multimedia information space. In addition to simple services for accessing and delivering multimedia data, it provides new functionalities for filtering and querying multimedia content. Content servers and on-line sources deliver media streams integrated within synchronized multimedia presentations. We think that the infrastructure will offer new opportunities for WWW-based information systems and electronic commerce.

1 Introduction

Increasingly, information systems take advantage of the World-Wide Web (WWW) framework to provide users with rich multimedia contents delivered over different types of networks. New applications such as information access and electronic commerce require new facilities for managing, structuring, and searching the distributed information space. In addition to that, we need support for seamless delivery of information to different types of client platforms, for example *streaming* multimedia data and *synchronizing* different sources.

The WWW is essentially based on the *pull* paradigm: a client browser requests a desired object (HTML page, GIF image) from a server, the object is then downloaded to a client site before rendering and presentation. There is also another type of emerging multimedia applications based on continuous real-time data. The applications include videoconferencing on the MBONE (vic, vat), Internet radios, and WebTV. They are mainly based on the *push* paradigm in which the provider publish its contents, the client subscribes to a chosen session, which is then delivered by the provider. Currently, each application deals with only one single media stream.

We think that the next step requires seamless integration of continuous real-time media streams within an advanced multimedia infrastructure. This new architecture of services and applications supported by different types of networks (high-speed, wireless) should focus on advanced facilities for dealing with multimedia contents. Both pull and push paradigms should be supported and integrated for more flexibility. In addition to simple services for accessing and delivering multimedia data, we also need new functionalities for structuring, filtering, and querying multimedia content. In a similar way we are looking today for relevant HTML pages using search engines, we would need to index, search for, negotiate the format, and control live multimedia streams that can be delivered from remote servers. Efficient user access may also require caching and storing of relevant contents close to the client's location. Such an advanced infrastructure may provide new opportunities for Web-based information systems and electronic commerce.

In a previous work, we have defined temporal extensions to HTML that allow seamless integration of synchronized multimedia into WWW documents [14]. A document specified using the extensions defines the temporal composition of various media objects, close synchronization between media in terms of master/slave relations, (to specify media priority in case of degradation), and the way in which the media objects are presented on the screen. The extensions are based on three concepts: *hypertime links* for temporal composition, common *time bases* for close lip-sync synchronization between media objects, and *dynamic layout*. In addition to the extensions, we have designed and implemented a flexible execution architecture to support the temporal extensions [13]. The architecture provides a base for a general-purpose player of synchronized multimedia documents.

In this paper, we present an advanced multimedia infrastructure that supports new networked applications. Our architecture provides generic services for negotiating media streams on servers, controlling their delivery, adapting their characteristics to varying network and host conditions, and synchronizing multiple remote streams. Integration of streaming support with our synchronization architecture allows us to de-

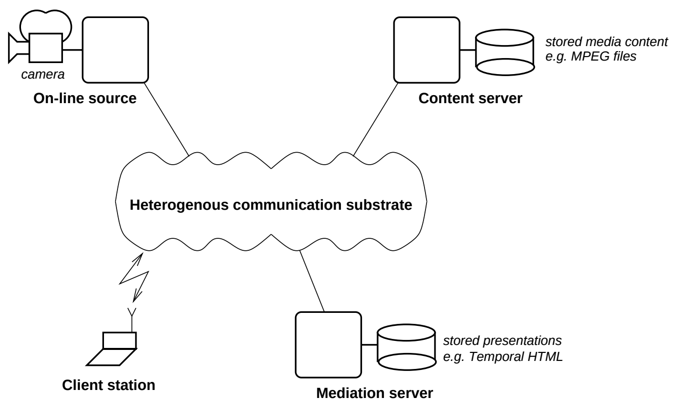

Figure 1: Architecture of the advanced multimedia infrastructure.

fine the concept of QoP (Quality of Presentation) that characterizes in a global manner the quality of document playback in a distributed system.

In the rest of the paper, we present and discuss several elements of the advanced multimedia infrastructure. Section 2 presents the overall architecture. Section 3 introduces temporal extensions to HTML used as a pivot representation of multimedia data. Section 4 presents the execution architecture that supports the temporal extensions. Section 5 discuss the network support for multiple media streams. Finally, we describe the current implementation status (Section 6) and outline conclusions (Section 7).

2 Overall architecture

Figure 1 presents the overall architecture of the advanced multimedia infrastructure. It includes the following elements:

- *Mediation server* organizes and structures the multimedia information space. It is based on *synchronized multimedia presentations* that integrate different multimedia streams within one single document having temporal behavior. The multimedia streams convey the media content stored at content servers or generated on the fly by on-line sources that capture audio or video data. The mediation server provides the traditional WWW server interface (HTTP) and operates according to the pull paradigm. The user can also subscribe to some presentations or media streams that will be delivered when the content becomes available.

- *Content server* provides stored media contents, for example MPEG video or audio files. It supports several basic functionalities: content and format negotiation, delivery of media streams, their control and adaptation. The content server provides an interface for controlling media streams and an interface for data delivery.

- *On-line source* provides media streams coming from the capture of live multimedia data such as audio and video. It supports the same interface for negotiating and controlling media streams as the content server.

- *Client station* connects through different types of communication networks and uses the mediation server to discover the media contents available on the network. It runs a player that downloads a synchronized multimedia presentation and contacts content servers or on-line sources for media stream delivery. The player synchronizes multiple media streams and controls their delivery.

- *Heterogeneous communication substrate* connects the elements of the architecture using various technologies: best effort, networks with differentiated QoS, wireless LANs (this last case is represented in the figure to emphasize the importance of mobile access).

The mediation server may provide sophisticated database features for querying collections of multimedia presentations [21, 1, 9]. It may also manage summaries of the content stored at content servers or available from on-line sources, so that the user can discover resources available on the network in a way similar for example to Discover [18]. As a result, the mediation server acts as a content broker between clients and rich multimedia data managed by content servers and on-line sources. For reasons of efficiency, the mediation server may act as a cache with respect to the content managed by content servers and on-line sources.

Content servers and on-line sources provide the same interface for managing media streams. Content negotiation allows client stations to choose the format of media streams, their bitrate, and possible policies for adaptation to the available bandwidth of the network [6]. Adaptation can be done at two levels. At the level of multimedia data, we can negotiate required formats and their characteristics such as resolution, color depth, compression ratio or perform *distillation* to degrade in a controlled manner the quality of data in order to adapt to the available bitrate. At the level of synchronized presentations, we should be able to adapt a generic presentation to available network resources, required quality of presentation, and even desired contents. For example, a user accessing a presentation over a low-bandwidth link when driving needs a customized version of a presentation different from the version accessed from a wired host. We have defined support for content adaptation of synchronized multimedia presentations based on content descriptions and alternative content markups that allow to specify generic presentations [12]. A generic presentation can be adapted on demand by applying a predicate representing user preferences, predefined profiles or requested contents. The predicate acts like a query that evaluated upon a generic document yields a customized presentation.

Media content is delivered using a chosen transport protocol and the delivery is subject to usual control commands: start, pause, rewind, stop. Content detection functionalities can enrich on-line sources to allow media stream activation in response to some conditions specified by the user. The mediation server, the content server, and the on-line source are logical entities, so that they may be located on the same physical site.

The functional view of the architecture is as follows. The user queries a mediation server to find presentations of interest. A chosen presentation is downloaded and parsed to extract references to content servers or on-line sources. According to the characteristics of the client machine and the current network connectivity, the player of the presentation chooses the adequate format and bitrate of media streams. It then requests the start of their delivery and synchronizes them on reception. If necessary, the player may control the rate of media streams according to the priorities defined in the presentation, for example, it can keep good quality of an audio stream and degrade a video stream when available bit rate decreases. Individual media streams can also be adjusted if needed to re-synchronize streams, for example by skipping over the enhancement layers in MPEG-2.

Several toolkits for manipulating real-time media streams have been developed: *VuSystem* [8], *CMT* [2], and MBone tools [10]. Two of them (*VuSystem* and *CMT*) have the same two-layer architecture: a chain of highly specialized modules to process incoming media data tied together with a scripting language to control the modules. Although the toolkits support flexible development of multimedia applications, they are limited by the fact that their abstractions correspond to a single media stream, for example, the *VuSystem* Puzzle application operates on a single audio/video stream, *vic* can receive several video streams, and *vat* can receive several audio streams, however the streams are independent. The problem is that there is no global control of several streams: if we execute *vic* and *vat* simultaneously, in case of network congestion, the user must adjust manually either of the streams to cope with lower bandwidth.

Our advanced multimedia infrastructure allows leveraging the rich contents of the WWW and mixing them with existing continuous real-time media sources such as MBONE sessions, digital television, Internet radio stations, and stored collections of video or audio. We will discuss below the elements of our architecture in more detail.

3 Synchronized multimedia presentations

The mediation server is based on a pivot representation of multimedia data. Such a representation must specify the temporal composition of different media streams, their layout, and closed lip-sync synchronization constraints. We have defined temporal extensions to HTML that allow seamless integration of synchronized multimedia into WWW documents [14]. A document specified using the extensions defines the temporal composition of various media objects, close synchronization between media in terms of master/slave relations, (to specify media priority in case of degradation), and the way in which the media objects are presented on the screen. Our work is related to the activity of W3C to define a new standard for synchronized multimedia documents: SMIL [20]. SMIL adds to HTML some features related to time behavior: components of a SMIL document may be multimedia streams and support for such objects should be included in SMIL browsers. Our extensions have similar objectives to those of SMIL, however unlike SMIL, they avoid several problems (hybrid approach to the temporal composition that mixes two different abstractions: intervals and time-points; insufficient support for close intermedia synchronization between multimedia objects; static layout).
Our temporal extensions to HTML are based on: *hypertime links*, *time bases*, and *dynamic layout*.

3.1 Hypertime Links for Temporal Composition

A *hypertime link* is a simple functional paradigm derived from temporal point nets to specify temporal composition: a temporal link between an origin and a target. A hypertime link has an explicit temporal semantics: it relates two media samples (e.g. video frames) and assigns the origin's time instant to the target. It can start or skip an object at the specified target position or stop it if the target is the end of the object. Following a hypertime link is automatic in the sense that it does not require any user interaction and consists of skipping in time from the origin media sample to the target.

3.2 Common Time Bases for Close Synchronization

To specify close synchronization between media objects, we define the notion of a *time base*. A time base is a virtual time space in which media objects "live". A time base defines a common time coordinate space for all objects that are related by some relations, for example master-slave dependency. A time base can be seen as a perfect time space in which real world phenomena such as jitter or drift do not exist and media objects behave in a perfect manner. Obviously, such a perfect time space does not exist, however, it can be implemented closely enough using real-time support. If such a support is not available, which is the case of many existing systems, a document should indicate how the quality of presentation is to be maintained and what is the nature of synchronization to be enforced.

We define the nature of synchronization between media segments using the notions of *master* and *slave*. A master-slave relationship defines a master controlling a slave according to its needs. We extend this notion to multiple masters and slaves through the common time base: a master can accelerate time or slow it down, hence slaves and other masters must adjust to time. The master-slave relationship allows the user to easily define the behavior of media segments with respect to synchronization. Another way to control synchronization between media segments is to specify *synchronization points* at which synchronization should be enforced.

The two mechanisms for synchronizing media segments: the master-slave relationship in a time base and synchronization points allow authors to express complex synchronization constraints to ensure that the document will be played back as the authors intended it to be, thus preserving the semantics of the document.

3.3 Media Objects and Dynamic Layout

We suppose that a synchronized multimedia document may include a variety of media objects having temporal behavior. A *media object* defines time evolution of media samples of one type. Media samples must be presented at precise time instants defined by the rate of presentation. The rate may be imposed by the author, adapted to match the duration of another object, or adjusted to synchronize with other objects. A media object schedules presentation of samples within a given time base. In this way, objects in the same time base are synchronized.

In addition to synchronized presentation of media samples, a media object can be controlled by other objects according to temporal composition. A hypertime link activated by another object can change the current presentation of an object and force it to skip to the target sample or to stop.

We define a *dynamic layout* as a special case of a media object. It defines a temporal behavior of the physical layout. The only difference is that layouts are neither masters nor slaves since they do not contain any media samples to be synchronized. It encapsulates *frames*, a means for defining regions of screen in which media objects are presented. Frames can be mixed with static elements such as traditional HTML text paragraphs. Frames can include other layouts to specify nested layouts that provide nested coordinate spaces. Hypertime links define how the layout changes in time. This approach allows seamless integration

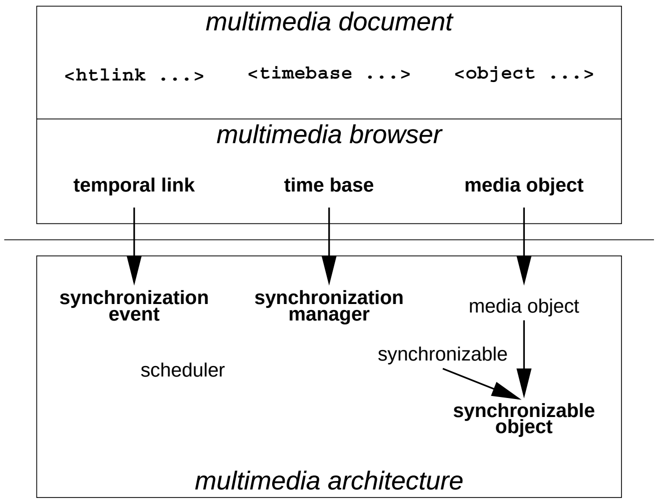

Figure 2: Functional structure of the execution support

of spatial and temporal dimensions into a multimedia document.

3.4 User adaptation of generic presentations

Adaptation can be seen as a transformation problem: the user specifies a predicate that applied to a synchronized multimedia presentation generates a customized view of the presentation. To do so, we have proposed a means of specifying [12]:

- *alternate content:* elements that allow to define alternate media objects, temporal composition, and layout.
- *content descriptions:* a way of associating metadata information with elements of a presentation (e.g. importance, keywords, duration).
- *content predicates:* a condition that selects alternate content or elements that match content descriptors.

4 Execution support for synchronized multimedia documents

We have designed a flexible execution architecture to support the temporal extensions [13]. The architecture provides a base for a general-purpose player of synchronized multimedia documents. Figure 2 presents the global functional view of the support. *Synchronizable objects* implement media objects with synchronization functionalities. *Synchronization managers* take care of controlling time bases, and *synchronization events* support synchronization points and hypertime links.

	Activity time	Idle time	Lifetime	Percentage of total activity
Presentation			370 725 ms	
Synchronization				2.598 %
scheduler	56 ms	370 669 ms	370 725 ms	0.015 %
managers	9 577 ms			2.583 %
Media objects	361 092 ms			97.402 %

Table 1: Synchronization performance

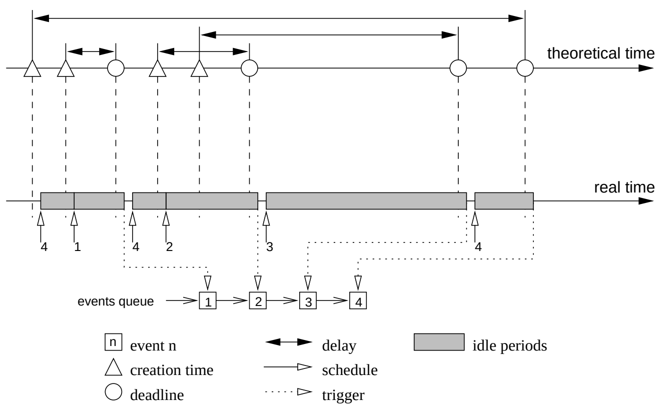

Figure 3: Principle of scheduling

The execution support offers time management functions needed by objects included in a multimedia document. Its role is to guarantee a coherent playback of a document according to its temporal specification. Synchronization of media objects raises three kinds of problems that should be solved: *intramedia synchronization* guarantees that a continuous real-time media will be played at a requested rate; *intermedia synchronization* enforces temporal relations between several media objects played at the same time; *media scheduling* is related to the management of media during a playback to make sure that events defined by the temporal composition of a document happen at a right time, for example activation and termination of a media object.

There are three main components visible at the external interface of the execution support: synchronizable objects, synchronization events, and synchronization managers. A scheduler, an additional internal component, handles synchronization events.

Synchronizable objects integrate media and synchronization: they encapsulate the services needed for a media to be played back and for controlling its execution. For this purpose, a synchronization objects encapsulates a media object and implements a synchronizable interface. Each object has a synchronization role, master or slave.

Media objects are autonomous entities used as building blocks in multimedia documents. They encapsulate media processing functionalities needed to render their corresponding media. They may contain different modules to fetch data, decode, render, and manage quality of service.

Synchronization events convey time information between various entities. A synchronization event defines an action aimed at a target and that has to happen at a given moment defined by a deadline. When its deadline is reached, the event is triggered and sent to the target so that it performs the required action. A hypertime link is implemented using such a synchronization event. The deadline for a hypertime link can be obtained from the relative temporal information associated with the link.

Synchronization managers implement time bases. They manage a pool of synchronizable objects belonging to the same time base. They handle synchronization events on behalf of these objects and enforce synchronization policies defined by time bases through roles and synchronization points. Managers do not handle time themselves — an internal global *scheduler* is in charge of scheduling events.

Figure 3 presents the principle of scheduling. Events are chained in a queue in the order of their instants of activation. The delay to the next event is computed and the scheduler blocks for this period. It wakes up when the delay expires or when a new event is scheduled before. The scheduler is able to adjust events to best fit the temporal constraints. This dynamic scheduling is efficient, because there is only one pending event per media object. Table 1 shows that the overhead of managing events is very small with respect to the total time needed for processing media objects (the measured presentation was composed of some text, five audio tracks, and two video clips).

5 Network support for media streams

In addition to the execution architecture that synchronizes different media objects we need support for managing multiple media streams delivered by remote content servers and on-line sources. Delivery of streams should enforce synchronization constraints specified by the author of a document and adjust the quality of media according to the available network resources.

5.1 Quality of Presentation

We are interested in a single characterization of the global quality perceived by the user of a document presentation. For this purpose, we introduce the concept of *Quality of Presentation* (QoP) that encompasses all the factors and allows the architecture to enforce a given QoP according to a document specification. In the context of our architecture, the global presentation quality depends on three subsystems: the network, the server host, and the synchronization architecture; each of these subsystems contributes in a different way to the global quality: the network through the usual notion of the *Quality of Service* (QoS) that can be negotiated and guaranteed (or not), the server host through the *Quality of Media* (QoM) that can be negotiated with a server and changed if needed, and the synchronization architecture and/or client site through the *Quality of Synchronization* (QoSync). This last factor characterizes the accuracy of synchronization enforced on the host playing a document that may depend on available resources on the client site.

The Quality of Service characterizes the ability of the network to guarantee some parameters such as the *bandwidth*, the *delay*, the *jitter* and the *loss rate*. QoS can vary from best effort IP to deterministic guarantees in AAL1/ATM. We do not assume any underlying QoS support, however we would like our architecture to work in the current best effort Internet, in which case, achieving the highest possible QoP may involve adaptation of QoM and QoSync.

The Quality of Media depends on the encoding of media contents. If we consider encoding with a given compression factor, better quality requires higher data bit rates. Hence the Quality of Media depends directly on the available bandwidth. To fit the available bandwidth of a given network connection, different Quality of Media can be achieved by

- *media scaling* [19] — the dynamic manipulation of media content as it is delivered to the client; for instance, the base layer (BL) and enhancement layers (E1 and E2) in a MPEG-2 video stream require different bit rates,

- *dynamic distillation* [4] — the controlled degradation of quality to adapt to the client platform or network connectivity; for instance, to deliver a video stream to a hand-held device connected via a wireless link, we may reduce the color to grey-scale images with low resolution.

- *software feedback* — the information on instantaneous network load controls the compression factor at the coder [3].

The Quality of Media also depends on the loss rate, because missing information may alter some parts of an encoded media frame.

The Quality of Synchronization depends on the network performance and the media encoding as well as on the accuracy of scheduling on the client site enforced by the execution architecture. Network performance parameters such as the delay and the jitter influence the QoSync, because media samples may arrive too late for being presented. Different types of encoding may influence QoSync as well, if some decoding schemes are computationally expensive. In general, the lack of available resources compromises the Quality of Synchronization.

Finally, the Quality of Presentation comprises the three previous factors. It characterizes how the player of a document conforms to the specification given by the author. Keeping Quality of Presentation the highest possible may require adaptation of transport protocols parameters, media encoding, or resynchronizing of a stream to a given time point.

5.2 Architecture of the support for multiple media streams

To provide a global view of all streams included in a document, we need a layer managing multiple media streams on the client side. Such a layer can control the quality of all components: network transport, media content, and synchronization to provide the highest possible adaptive quality of presentation. The layer should provide a transparent view of media objects to the upper presentation layers and may be used to perform explicit initial negotiation and dynamic re-negotiation of desired media and transport quality: for example, when accessing a document using a hand-held device and a low-bandwidth wireless connection, degraded media quality can be requested by negotiating with the server.

Our architecture includes media objects that encapsulate the required processing on media content: transport, decoding, rendering. The *service manager* deals with the transport performance and network resource reservation. It controls the parameters of QoS if available and adjusts them if needed. The *media manager* is responsible for the content negotiation, the choice of encoding, and the application level control of media content. The *quality of presentation manager* coordinates other managers and media objects. According to the information included in a document, it tries to perform the best using either the service manager or the media manager. If for example, it is notified by the synchronization manager that the presentation slows down, it may ask the media manager to change media resolution or encoding, or resynchronize a stream by skipping to some future media sample. If possible, it may renegotiate network resource reservation via the service manager. The quality of presentation manager receives notifications from decoding and transport operations to react in a similar way. All managers integrate well within the synchro-

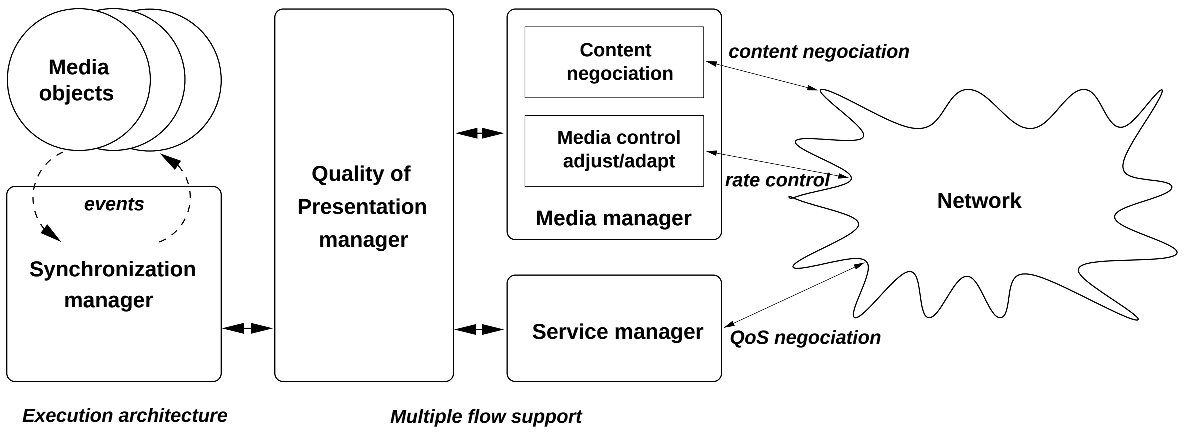

Figure 4: Integration of the network support for multiple media streams with the execution support (the data path is not shown)

nization architecture described in Section 4 by providing control over remote streams. Figure 4 presents the architecture of the support for multiple streams.

The document-wide management of multiple streams provides the best available quality of presentation for a given set of constraints (processing power, bandwidth). According to the explicit or implicit semantics specified in a document, the global management scheme can degrade less important streams to conserve the quality of more important ones. In this way, the presentation quality is balanced over all the streams of a document.

6 Implementation

We are in the initial stage of our implementation and we have already prototyped several elements of our infrastructure. We have developed the execution support for temporal extensions to HTML. The execution support uses Java and SableCC: a parser generated by SableCC parses a document specified in temporal HTML and produces a set of Java components linked with the execution support providing basic synchronization classes. We also use DOM for controlling Java components and synchronization classes.

We are developing Java class libraries that implement the quality of presentation, media, and service managers. We are also prototyping a content server and an on-line source that deliver media streams using existing implementations of RTSP [17, 15], RDF [11], SDP [5], and RTP/RTCP [16]. Network quality of service will be based on DiffServ support for prioritizing network traffic [7]. We have already integrated modules of *vic* and *vat* within the player to receive and synchronize MBONE media sources.

7 Conclusion

In this paper, we have presented an advanced multimedia infrastructure that supports new networked applications. The infrastructure focuses on advanced facilities for dealing with multimedia contents. It provides generic services for negotiating media streams on servers, controlling their delivery, adapting their characteristics to varying network and host conditions, and synchronizing multiple remote streams. Integration of streaming support with our synchronization architecture allows us to define the concept of QoP (Quality of Presentation) that characterizes in a global manner the quality of document playback in a distributed system.

Although we have not yet finished the implementation, so we cannot give any exhaustive evaluation of the design choices, our experience with the execution architecture has been positive and encouraging.

Acknowledgments

This work has been done when Franck Rousseau was with The Open Group Research Institute. He was supported by Bull SA.

References

[1] M. Adiba. Storm: a Structural and Temporal Object-oRiented model for Multimedia databases. In *Proc. International Workshop on Multi-media Database Management Systems*, Blue Mountain Lake, NY, August 1995.

[2] David P. Anderson and George Homsy. A Continuous Media I/O Server and Its Synchronization Mechanism. *IEEE Computer*, 24(10):51–57, October 1991.

[3] Shanwei Cen, Calton Pu, Richard Staehli, Crispin Cowan, and Jonathan Walpole. Demonstrating the Effect of Software Feedback on a Distributed Real-Time MPEG Video Audio Player. In *Proceedings of ACM Multimedia'95*, pages 239–240, San Francisco, CA, November 5–9, 1995.

[4] Fox et al. Adapting to Network and Client Variation via On-Demand, Dynamic Distillation. In *ASPLOS-VII*, Boston, MA, 1996.

[5] Mark Handley and Van Jacobson. SDP: Session Description Protocol. Request for Comments (Proposed Standard) RFC 2327, Internet Engineering Task Force, April 1998.

[6] Koen Holtman and Andrew H. Mutz. Transparent Content Negociation. Request for Comments RFC 2295, Internet Engineering Task Force, March 1998.

[7] V. P. Kumar, T.V. Lakshman, and D. Stiliadis. Beyond best effort : router architectures for the differentiated services of tomorrow's internet. *IEEE Communication Magazine*, (5), May 1998.

[8] Christopher J. Lindblad and David L. Tennenhouse. The VuSystem: A Programming System for Compute-Intensive Multimedia. To appear in *IEEE Journal on Selected Areas in Communications*, 1996.

[9] H. Martin. Specification of intentional presentations using an object-oriented database. In *Proc. Advanced Database Research and Development Series*, volume 8, 1998.

[10] Steven McCanne and Van Jacobson. vic: A Flexible Framework for Packet Video. In *Proceedings of ACM Multimedia'95*, pages 511–522, San Francisco, CA, November 5–9, 1995.

[11] Resource Description Framework (RDF) Model and Syntax Specification. Technical report, World Wide Web Consortium (W3C), 1998.

[12] F. Rousseau, J.A. García-Macías, J.V. de Lima, and A. Duda. User adaptable multimedia presentations for the WWW. In *Proc. Eight International World-Wide Web Conference*, Toronto, mai 1999.

[13] Franck Rousseau and Andrzej Duda. An Execution Architecture for Synchronized Multimedia Presentations. In *Proceedings of the 3rd European Conference on Multimedia Applications, Services and Techniques (ECMAST'98)*, volume 1425 of *Lecture Notes in Computer Science*, pages 42–55, Berlin, Germany, May 26–28, 1998. Springer-Verlag.

[14] Franck Rousseau and Andrzej Duda. Synchronized Multimedia for the WWW. In *Proceedings of the 7th International World Wide Web Conference (WWW7), Computer Networks and ISDN Systems*, volume 30, pages 417–429, Brisbane, Australia, April 14–18, 1998.

[15] Henning Schulzrinne. A comprehensive multimedia control architecture for the Internet. In *Proceedings of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97)*, St. Louis, MO, May 1997.

[16] Henning Schulzrinne, Stephen L. Casner, Ron Frederick, and Van Jacobson. RTP: A Transport Protocol for Real-Time Applications. Request for Comments (Proposed Standard) RFC 1889, Internet Engineering Task Force, February 1996.

[17] Henning Schulzrinne, Anup Rao, and Rob Lanphier. Real Time Streaming Protocol (RTSP). Request for Comments (Proposed Standard) RFC 2326, Internet Engineering Task Force, April 1998.

[18] M.A. Sheldon, A. Duda, R. Weiss, and D.K. Gifford. Discover: A resource discovery system based on content routing. *Computer Networks And ISDN Systems*, 27(6):953–972, 1995.

[19] H. Tokuda and T. Kitayama. Dynamic QOS Control Based on Real-Time Threads. In *Proceedings of the 4th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'93)*, Lancaster, UK, 1993.

[20] Synchronized Multimedia Integration Language. Recommendation REC-smil-19980615, World Wide Web Consortium (W3C), June 15, 1998. Latest version available at `http://www.w3.org/TR/REC-smil/`.

[21] Ron Weiss, Andrzej Duda, and David K. Gifford. Composition and Search with a Video Algebra. *IEEE Multimedia*, 2(1):12–25, Spring 1995.