

A Transport Infrastructure Supporting Real Time Interactive MPEG-4 Client-Server Applications over IP Networks

Haining Liu, Xiaoping Wei, and Magda El Zarki

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92697
{haining, wei, magda}@ics.uci.edu

Abstract. Nearly all of the multimedia streaming applications running on the Internet today are basically configured or designed for 2D video broadcast or multicast purposes. With its tremendous flexibility, MPEG-4 interactive client-server applications are expected to play an important role in online multimedia services in the future. This paper presents the initial design and implementation of a transport infrastructure for an IP based network that will support a client-server system which enables end users to: 1) author their own MPEG-4 presentations, 2) control the delivery of the presentation, and 3) interact with the system to make changes to the presentation in real time. A specification for the overall system structure is outlined. Some initial thoughts on the server and client system designs, the data transport component, QoS provisioning, and the control plane necessary to support an interactive application are described.

1 Introduction

Today, most of the multimedia services consist of a single audio or natural (as opposed to synthetic) 2D video stream. MPEG-4, a newly released ISO/IEC standard, provides a broad framework for the joint description, compression, storage, and transmission of natural and synthetic audio-visual data. It defines improved compression algorithms for audio and video signals, and efficient object-based representation of audio-video scenes [1]. It is foreseen that MPEG-4 will be an important component of multimedia applications on IP-based networks in the near future [4].

In MPEG-4, audio-visual objects are encoded separately into their own Elementary Streams (ES). In addition, the Scene Description (SD), also referred to as the Binary Format for Scene (BIFS), defines the spatio-temporal features of these objects in the final scene to be presented to the end user. Based upon VRML (Virtual Reality Modeling Language), the SD uses a tree-based graph, and can be dynamically updated. The SD is conveyed between the source and the destination through one or more ESs and is transmitted separately. Object Descriptors (ODs) are used to associate scene description components to the actual elementary streams that contain the corresponding coded media data. Each OD groups all descriptive components that are related to a single media object, e.g., an audio or video object, or even an

animated face. ODs carry information on the hierarchical relationships, locations and properties of ESs. ODs themselves are also transported in ESs. The separate transport of media objects, SD and ODs enables flexible user interactivity and content management.

The MPEG-4 standard defines a three-layer structure for an MPEG-4 terminal: the Compression Layer, the Sync Layer and the Delivery layer. The Compression Layer processes individual audio-visual media streams and organizes them in Access Units (AU), the smallest elements that can be attributed individual timestamps. The compression layer can be made to react to the characteristics of a particular delivery layer such as the path-MTU or loss characteristics. The Sync Layer (SL) primarily provides the synchronization between streams. AUs are here encapsulated in SL packets. In case that the AU is larger than the SL packet, it will be fragmented across multiple SL packets. The SL produces an SL-packetized stream i.e. sequences of SL-packets. The SL-packet headers contain timing, sequencing and other information necessary to provide synchronization at the remote end. The packetized streams are then sent to the Delivery Layer.

In the MPEG-4 standard, a delivery framework referred to as the Delivery Multimedia Integration Framework (DMIF) is specified at the interface between the MPEG-4 synchronization layer and the network layer. DMIF provides an abstraction between the core MPEG-4 system components and the retrieval methods [2]. Two levels of primitives are defined in DMIF. One is for communication, between the application and the delivery layer to handle all the data and control flows. The other one is used to handle all the message flows in the control plane between DMIF peers. Mapping these primitives to the available protocol stacks in an IP-based network is still an on-going research issue. Moreover, designing an interactive client-server system for deployment in the Internet world using the recommended primitives is even more challenging [3].

The object-oriented nature of MPEG-4 makes it possible for an end user to manipulate the media objects and create a multimedia presentation tailored to his or her specific needs, end device and connection limitations. The multimedia content resides on a server (or bank of servers) and the end user has either local or remote access to the service. This service model differs from the traditional streaming applications because of its emphasis on end user interactivity. To date, nearly all the streaming multimedia applications that are running on the Internet are basically designed for simple remote retrieval or for broadcasting/multicasting services. Interactive services are useful in settings such as distance learning, gaming, etc. The end user can pick the desired language, the quality of the video, the format of the text, etc.

There are only a few MPEG-4 interactive client-server systems discussed in the literature. H. Kalva *et al.* describe an implementation of a streaming client-server system based on an IP QoS Model called XRM [6]. As such it cannot be extended for use over a generic IP network (it requires a specific broadband kernel – xbind). Y. Pourmohammadi *et al.* propose a DMIF based system design for IP-based networks. However, their system is mainly geared toward remote retrieval and very little is mentioned in the paper regarding client interactivity with respect to the actual presentation playback [7]. In [6], the authors present the Command Descriptor Framework (CDF), which provides a means to associate commands with media objects in the SD. The CDF has been adopted by the MPEG-4 Systems Group, and is part of the version 2 specification. It consists of all the features to support interactivity

in MPEG-4 systems [8-10]. Our transport infrastructure subsumes that command descriptors (CDs) are used for objects in the SD. This will enable end users to interact with individual objects or group of objects and control them.

In order to support an MPEG-4 system that enables end user interactivity as described above, many issues must be considered. To list a few: 1) transmission of end user interactivity commands to the server, 2) transport of media content with QoS provisioning, 3) real time session control based upon end user interactivity, 4) mapping of data streams onto Internet transport protocols, 5) extension of existing IETF signaling and control protocols to support multiple media streams in an interactive environment, etc.

In this paper, we present our initial ideas on the design of a client-server transport architecture which will enable end users to create their own MPEG-4 presentation, control the delivery of the content over an IP-based network, and interact with the system to make changes to the presentation in real time. Section 2 presents the structure of the overall system. Server and client designs are described in section 3. In section 4, data transport, QoS provisioning and control plane message exchange are discussed. In section 5, we conclude and discuss the future work.

2 Overall System Architecture

The system we are proposing to develop is depicted in Figure 1 and consists of 1) an MPEG-4 server, which stores encoded multimedia objects and produces MPEG-4 content streams, 2) an MPEG-4 client, which serves as the platform for the composition of an MPEG-4 presentation as requested by the end user, and 3) an IP network that will transport all the data between the server and the client.

The essence of MPEG-4 lies in its object-oriented structure. As such, each object forms an independent entity that may or may not be linked to other objects, spatially and temporally. The SD, the ODs, the media objects, and the CDs are transmitted to the client through separate streams. This approach gives the end user at the client side tremendous flexibility to interact with the multimedia presentation and manipulate the different media objects. End users can change the spatio-temporal relationships among media objects, turn on or shut down media objects, or even specify different perceptual quality requirements for different media objects dependent upon the associated command descriptors for each object or group of objects. This results in a much more difficult and complicated session management and control architecture [6]. Our design starts out with the premise that end user interactivity is crucial to the service and therefore it targets a flexible session management scheme with efficient and adaptive encapsulation of data for QoS provisioning.

User interactivity can be defined to consist of three degrees or levels of interactivity that correspond to what type of control is desired:

1. Presentation level interactivity: in which a user actually makes changes to the scene by controlling an individual object or group of objects. This also includes presentation creation.
2. Session level interactivity: in which a user controls the playback process of the presentation (i.e., VCR like functionality for the whole session).

3. Local level interactivity: in which a user only makes changes that can be taken care of locally, e.g., changing the position of an object on the screen, volume control, etc.

Throughout our discussion below we will be making references to these three levels as each results in a different type of control message exchange in our system.

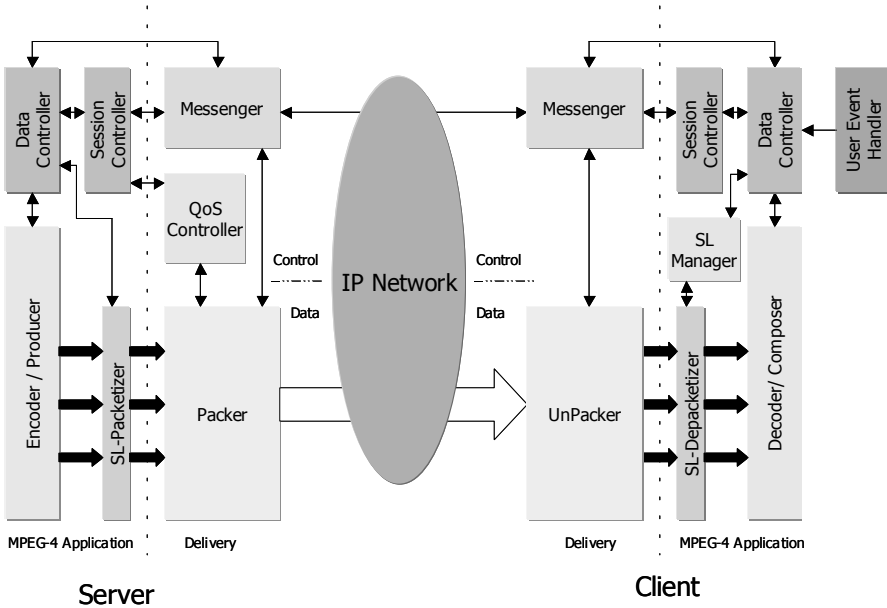


Fig. 1. System Architecture

We assume that the server maintains a database or a list of available MPEG-4 content and provides WWW access to it. An end user at a remote client side retrieves information regarding the media objects that he/she is interested in, and composes a presentation based upon what is available and desired. The system operation, after the end user has completed the composition of the presentation, can be summarized as follows:

1. The client requests a service by submitting the description of the presentation to the Data Controller (DC) at the server side.
2. The DC on the server side, controls the Encoder/Producer module to generate the corresponding SD, ODs, CDs and other media streams based upon the presentation description information submitted by the end user at the client side. The DC then triggers the Session Controller (SC) on the server side to initiate a session.
3. The SC on the server side is responsible for session initiation, control and termination. It passes along the stream information that it obtained from the DC to the QoS Controller (QC) that manages in conjunction with the Packer, the creation of the corresponding transport channels with the appropriate QoS provisions.

4. The Messenger Module (MM) on the server side, which handles the communication of control and signaling data, then signals to the client the initiation of the session and network resource allocation. The encapsulation formats and other information generated by the Packer when processing the “packing” of the SL-packetized streams are also signaled to the client to enable it to unpack the data.
5. The actual stream delivery commences after the client indicates that it is ready to receive, and streams flow from the server to the client. After the decoding and composition procedures, the MPEG-4 presentation authored by the end user is rendered on his or her display.

In the next sections, we describe in some detail the functionality of the different modules and how they interact.

3 Client-Server Model

3.1 The MPEG-4 Server

Upon receiving a new service request from a client, the MPEG-4 server starts a thread for the client, and walks through the steps described in the previous section to setup a session with the client. The server maintains a list of sessions established with clients and a list of associated transport channels and their QoS characteristics.

Figure 2 shows the components of the MPEG-4 Server. The Encoder/Producer compresses raw video sources in real-time or reads out MPEG-4 content stored in MP4 files. The elementary streams produced by the Encoder/Producer are packetized by the SL-Packetizer. The SL-Packetizer adds SL-Packet headers to the AUs in the elementary streams to achieve intra-object stream synchronization. The headers contain information such as decoding and composition time stamps, clock references, padding indication, etc. The whole process is scheduled and controlled by the DC.

The DC is responsible for several functions:

1. It responds to control messages that it gets from the client side DC. These messages include the description of the presentation composed by the user at the client side and the presentation level control commands issued by the remote client DC resulting from user interactions.
2. It communicates with the SC to initiate a session. It also sends SC the session update information as it receives user interactivity commands and makes the appropriate SD and OD changes.
3. It controls the Encoder/Producer and SL-Packetizer to generate and packetize the content as requested by the client.
4. It schedules audio-visual objects under resource constraints. With reference to the System Decoding Model, the AUs must arrive at the client terminal before their decoding time [1]. Efficient scheduling must be applied to meet this timing requirement and also satisfy the delay tolerances and delivery priorities of the different objects.

The SC likewise is responsible for several functions:

1. When triggered by the DC for session initiation, it will coordinate with the QC to set-up and maintain the numerous transport channels associated with the SL packetized streams.
2. It maintains session state information and updates this whenever it receives changes from the DC resulting from user interactivity.
3. It responds to control messages sent to it by the client side SC. These messages include the VCR type commands that the user can use to control the session.

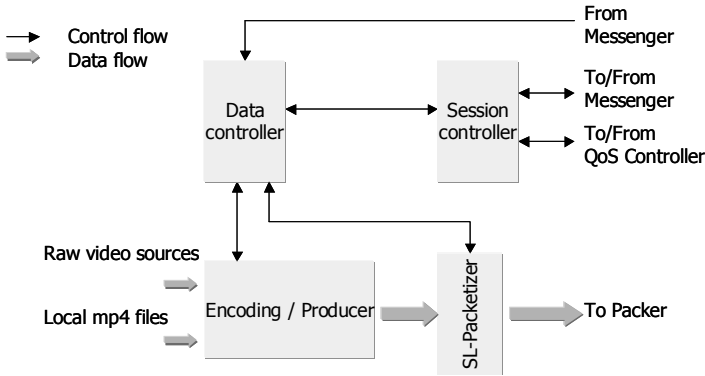


Fig. 2. Structure of the MPEG-4 Server

3.2 The MPEG-4 Client

The architectural design of the MPEG-4 client is based upon the MPEG-4 System Decoder Model (SDM), which is defined to achieve media synchronization, buffer management, and timing, when reconstructing the compressed media data [1]. Figure 3 illustrates the components of the MPEG-4 client.

The SL Manager is responsible for binding the received ESs to decoding buffers. The SL-Depacketizer extracts the ESs received from the Unpacker and passes them to the associated decoding buffers. The corresponding decoders then decode the data in the decoding buffers and produce Composition Units (CUs), which are then put into composition memories to be processed by the compositor. The User Event Handler module handles the user interactivity. It filters the user interactivity commands and passes the messages along to the DC and the SC for processing.

The DC at the client side has the following responsibilities:

1. It controls the decoding and composition process. It collects all the necessary information, e.g., the size of the decoding buffers which is specified in decoder configuration descriptors and signaled to the client via the OD, the appropriate decoding time and composition time which is indicated in the SL packet header, etc., for the decoding process.

2. It manages the flow of control and data information, controls the creation of buffers and associates them with the corresponding decoders.
3. It relays user presentation level interactivity to the server side DC and processes both session level and local level interactivity to manage the data flows on the client terminal.

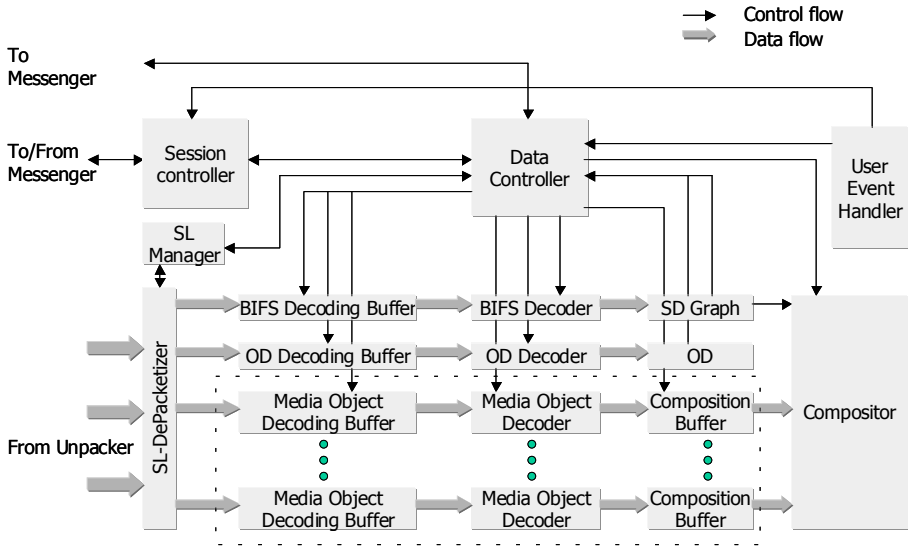


Fig. 3. Structure of the MPEG-4 Client

The SC at the client side communicates with the SC at the server side exchanging session status information and session control data. The User Event Handler will trigger the SC when session level interactivity is detected. The SC then translates the user action into the appropriate session control command.

4 Transport Architecture

The efficient and adaptive encapsulation of MPEG-4 content with regard to the MPEG-4 system specification is still an open issue. There is a lot of ongoing research on how to deliver MPEG-4 content over IP-based networks. Figure 4 shows our proposed design for the delivery layer. The following sections detail the components in this design.

4.1 Transport of MPEG-4 SL-Packetized Streams

Considering that the MPEG-4 SL defines some transport related functions such as timing and sequence numbering, encapsulating SL packets directly into UDP packets seems to be the most straightforward choice for delivering MPEG-4 data over IP-

based networks. Y. Pourmohammadi *et al* presented a system adopting this approach [7]. However, some problems arise with such a solution. Firstly, it is hard to synchronize MPEG-4 streams from different servers in the variable delay environment which is common in the Internet. Secondly, no other multimedia data stream can be synchronized with MPEG-4 data carried directly over UDP in one application. Finally, such a system lacks a reverse channel for carrying feedback information from the client to the server with respect to the quality of the session. This is a critical point if QoS monitoring is desired.

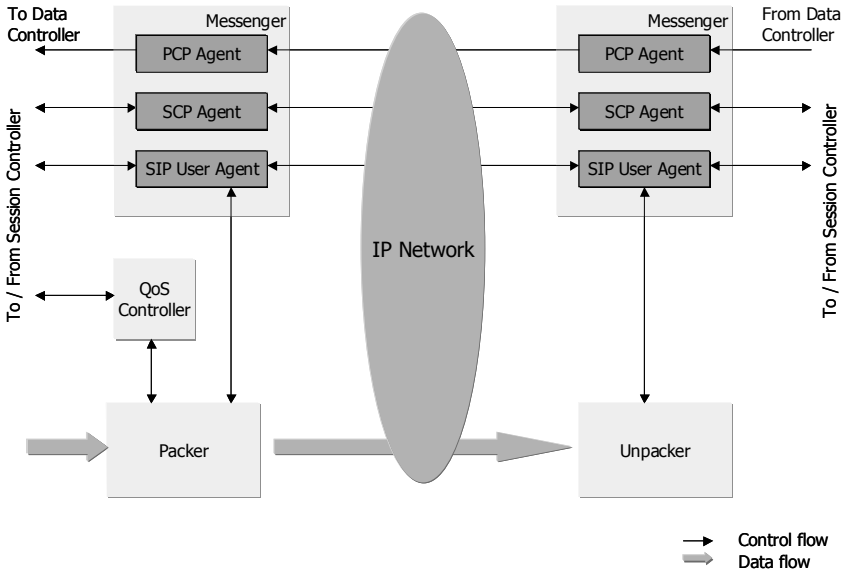


Fig. 4. Structure of the Delivery Layer

Another option is to deliver the SL packets over RTP, a standard protocol providing end-to-end transport functions for real-time Internet applications [11]. RTP has associated with it a control protocol, RTCP, which provides feedback channel for quality monitoring. In addition, the synchronization problems incurred when using UDP directly can be solved by exploiting the timing information contained in the RTCP reports. The problem arising when using RTP is the need to remove the resulting redundancy, as the RTP header duplicates some of the information provided in SL packet header. This adds to the complexity of the system and increases the transport overhead [12].

There are a number of Internet Drafts describing RTP packetization schemes for MPEG-4 data [12-14]. An approach proposed by Avaro *et al*, is attractive due to its solution regarding the duplication problem of the SL packet header and its independence of the MPEG-4 system, i.e., is not DMIF based [12]. The redundant information in the SL packet header is mapped into RTP header and the remaining part, which is called “reduced SL header”, is placed in the RTP payload along with the SL packet payload. Detailed format information is signaled to the receiver via SDP. The MPEG-4 system also defines Flexible Multiplexing (FlexMux) to bundle

several ESs with similar QoS requirements. The FlexMux scheme can be optionally used to simplify session management by reducing the number of RTP sessions needed for an MPEG-4 multimedia session. Van der Meer *et al*, proposed a scheme that provides an RTP payload format for MPEG-4 FlexMux streams [13].

Figure 5 shows the detailed layering structure inside the Packer and Unpacker. In the Packer, the SL-packetized streams are optionally multiplexed into FlexMux streams at the FlexMux layer, or directly passed to the transport protocol stacks composed of RTP, UDP and IP. The resulting IP packets are transported over the Internet. In the Unpacker, the data streams are processed in the reverse manner before they are passed to SL-Depacketizer.

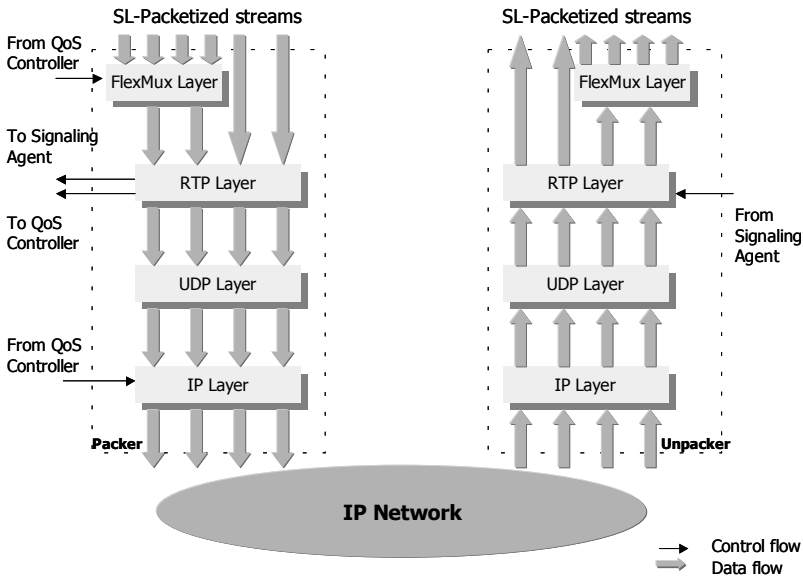


Fig. 5. Layering structure of Packer and Unpacker

In the Packer, the multiplexing of the SL-packetized streams is processed according to the QoS requirements of the streams and as such managed by the QoS controller. At the RTP Layer, the format of the encapsulation is passed to the signaling agent in SDP format to notify the client. The Packer is responsible for the allocation of transport channels, which are defined by port numbers, with the information from the QoS controller. At the IP Layer, certain actions managed by the QoS controller will be passed onto the network layer to meet the IP QoS requirements.

4.2 QoS Provisioning

According to the MPEG-4 system standard, each stream has an associated “transport” QoS description for its delivery [1]. This QoS description consists of a number of QoS metrics such as delay, loss rate, stream priority, etc. It is then up to the implementation of the delivery layer to fulfill these requirements.

In our system, the QoS Controller at the server side takes all the QoS issues into consideration as depicted in Figure 4. It is mainly responsible for managing the transport channel setup according to the required QoS parameters and controlling the traffic conditioning of IP packets. We model the IP network that our system is built on as a Differentiated Services (DiffServ) network, which is also the model for Internet 2. The QoS Controller maps the values of the QoS parameters to the values of the DiffServ Code Point (DSCP), which in turn determines the per-hop forwarding behavior of the IP packets. It then rewrites the DS byte in the IP header. It also controls the policing and rate shaping that takes place at the IP layer of the Packer.

4.3 Exchange of Signaling, Session Control and Presentation Control Messages

There are mainly three kinds of message flows exchanged between the server and the client in our system. Signaling messages are used mainly to locate the client, establish a network session, specify media information, modify, and tear-down an existing network session. Session Control messages contain commands issued by both the server and client to manage the session and provide real time session control to reflect user interactivity. Presentation Control messages are used to relay the presentation composed by the user and the presentation level user interactivity. As shown in Figure 4, three channels are maintained to carry these three distinct message flows between the server and the client.

SIP is a signaling protocol for creating, modifying and terminating sessions with one or more participants [4]. Because of its versatility, SIP fits in well with our system model. We will use SDP to deliver information such as media stream encapsulation format, network resource allocation indication, etc. The SDP messages will be embedded in the SIP message payload. SIP User Agents are placed in the MM in our system for session initiation, termination and transport of SDP messages.

RTSP is a control protocol used for real-time streaming applications [16]. Some papers have proposed schemes for adopting RTSP to provide some basic control and simple signaling for MPEG-4 applications [5]. However, as RTSP was primarily designed for media-on-demand scenarios, it cannot support the sophisticated interactivity required by our system. To reduce the overall system complexity, we have separated the signaling and control functions, and will design a Session Control Protocol (SCP) solely for exchanging control messages to manage the session (e.g., stop, resume, pause, etc.) in real time. Like SIP, the SCP agent will be placed in the MM to handle the message flow.

In our design, presentation control messages are exchanged between the client and the server via the Presentation Control Protocol (PCP). It will be designed specifically to support the presentation level user interactivity. During the presentation playback, the end user is able to interact with, and control what is being displayed at the object level. For example, VCR like functionality, such as stop, pause, resume, fast forward, can be associated with each object or group of objects. These controls will be

implemented initially. More complex controls, such as the texture, dimensions, quality, etc., of an object, will be implemented as the design of the system matures and more detailed CDs are created for the objects. Similar to the other two agents, we will incorporate the PCP agent in the MM to communicate.

5 Conclusions

We presented in this paper a design for a transport infrastructure to support interactive multimedia presentations which enable end users to choose available MPEG-4 media content to compose their own presentation, control the delivery of such media data and interact with the server to modify the presentation in real-time. Detailed structures of the server and client were described. We discussed the issues associated with the delivery of media data and information exchange using a dedicated control plane that supports the exchange of signaling, session control and presentation control messages.

In summary, our work is focused on the following issues:

1. Translation of user interactivity into appropriate control commands
2. Development of a presentation control protocol to transmit user interactivity commands and a session control protocol to enable real time session management and information exchange to support user interactions
3. Processing of the presentation control messages by the server to update the MPEG-4 session
4. Transport of multiple inter-related media streams with IP based QoS provisioning
5. Client side buffer management for decoding the multiple media streams and scene composition

References

1. ISO/IEC JTC 1/SC 29/WG 11, "Information technology-Coding of audio-visual objects, Part1: Systems (ISO/IEC 14496-1)," Dec. 1998.
2. ISO/IEC JTC 1/SC 29/WG 11, "Information technology-Coding of audio-visual objects, Part6: Delivery Multimedia Integration Framework (ISO/IEC 14496-6)," Dec. 1998.
3. ISO/IEC JTC 1/SC 29/WG 11, "Information technology-Coding of audio-visual objects, Part8: Carriage of MPEG-4 contents over IP networks (ISO/IEC 14496-8)," Jan. 2001.
4. D. Wu, Y. T. Hou, W. Zhu, H. Lee, T. Chiang, Y. Zhang, H. J. Chao, " On End-to-End Architecture for Transporting MPEG-4 Video Over the Internet", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, No. 6, Sep. 2000, pp. 923-941.
5. A. Basso and S. Varakliotis, " Transport of MPEG-4 over IP/RTP," Proceedings of ICME 2000, Capri, Italy, June 2000, pp. 1067-1070.

6. H. Kalva, L. Tang, J. Huard, G. Tselikis, J. Zamora, L. Cheok, A. Eleftheriadis, "Implementing Multiplexing, Streaming, and Server Interaction for MPEG-4," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 8, Dec. 1999, pp. 1299–1311.
7. Y. Pourmohammadi, K. A. Haghighi, A. Mohamed, H. M. Alnuweiri, "Streaming MPEG-4 over IP and Broadcast Networks: DMIF Based Architectures," Proceedings of the Packet Video Workshop '01, Seoul, Korea, May 2001.
8. A. Akhtar, H. Kalva, and A. Eleftheriadis, "Command Node Implementation", Contribution ISO-IEC JTC1/SC29/WG11 MGE99/4905, July 1999, Vancouver, Canada.
9. ISO/IEC/SC29/WG11, "Text of ISO/IEC 14496-1/FPDAM 1(MPEG-4 System Version-2)," International Standards Organization, May 1999
10. Akhtar, H. Kalva, and A. Eleftheriadis, "Command Node Implementation in the IM1 Framework", Contribution ISO-IEC JT1/SC29/WG11 MGE00/5687, March 2000, Noordwijkerhout, Netherlands
11. Schulzrinne, Casner, Frederick and Jacobson, "RTP: A Transport Protocol for Real-Time Applications," draft-ietf-avt-rtp-new-08.ps, July 2000.
12. Avaro, Basso, Casner, Civanlar, Gentric, Herpel, Lifshitz, Lim, Perkins, van der Meer, "RTP Payload Format for MPEG-4 Streams," draft-gentric-avt-mpeg4-multiSL-03.txt, April 2001.
13. C.Roux et al, "RTP Payload Format for MPEG-4 Flexmultiplexed Streams," draft-curet-avt-rtp-mpeg4-flexmux-00.txt, Feb. 2001.
14. Y. Kikuchi, T. Nomura, S. Fukunaga, Y. Matsui, H. Kimata, "RTP Payload Format for MPEG-4 Audio/Visual Streams," RFC 3016, Nov. 2000.
15. Handley, Schulzrinne, Schooler and Rosenberg, "SIP: Session Initiation Protocol," ietf-sip-rfc2543bis-02.ps, Nov. 2000.
16. H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April, 1998.
17. M. Handley and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.