Streaming MPEG-4 over IP and Broadcast Networks: DMIF Based Architectures

Y. Pourmohammadi, K. Asrar Haghighi, A. Mohamed, H. M. Alnuweiri

Department of Electrical & Computer Engineering
University of British Columbia
Vancouver, B.C., CANADA V6T 1Z4
Email: yasserp@ece.ubc.ca

***Abstract* -- This paper presents implementations of two MPEG-4 Streaming scenarios (remote retrieval and broadcast) for the delivery of multiple elementary stream presentations containing audio-visual objects. These systems have been tested with the MPEG-4 demonstration software implementation (IM1), but are in fact generic platforms for multimedia presentation delivery. The signalling and delivery layer of these system implementations conform to the recommendations made by part 6 of the MPEG-4 standard, Delivery Multimedia Integration Framework (DMIF). We present the issues involved in designing our streaming server for the remote retrieval and broadcast scenarios. The remote retrieval system enables interactive media streaming across the Internet through its Data and Control Planes. The implementations also include a source-based rate-control system, a Synchronization Layer packetizer, and a signalling mechanism.**

***Index Terms* – Delivery Multimedia Integration Framework (DMIF), MPEG-4, streaming, packetization, source-based rate-control, multimedia delivery, multiplexing, interactive media, broadcast.**

I. INTRODUCTION

Recent advancements in audio-visual scene encoding and presentation have spurred numerous standards. These standards have enabled scene encoding with semantically meaningful objects. MPEG-4, a breakthrough in multimedia technology, is the first standard that addresses presentation content as a set of audio-visual objects [1]. This standard specifies the entire spectrum of tools required for encoding objects within a scene, composing presentations with objects, storing these object-based presentations, and accessing them through a variety of delivery technologies. An MPEG-4 terminal contains three major layers: Compression Layer, Sync Layer, and Delivery Layer as depicted in Figure 1. The layering by the standard adds all the benefits of a modular design and ensures that each layer is only aware of the information that is meaningful to it.

Within the Compression Layer of the MPEG-4 standard, media elements (e.g., audio, background, actors, etc.) are encoded as distinct objects. The resulting objects are available to the Sync Layer (SL) in the form of access units (AUs), the smallest elements that can be attributed individual time stamps (e.g., a frame of video or audio data). In addition to these objects, a scene description is also required to describe the relationship between various objects. The MPEG-4 scene description, also referred to as Binary Information for Scenes (BIFS), provides a spatio-temporal composition of scenes. The advantage of encoding objects separately is that compression can be performed on an individual bitstream based on its characteristics. This will provide higher compression ratios and will allow the use of different quality of service (QoS) service levels. At the receiving terminal, the Compression Layer decodes the incoming AUs from the Sync Layer and performs the complement of the tasks described above.

All information in MPEG-4 is conveyed in a streaming manner. The layer that provides AU packetization within elementary streams and synchronization between the streams is the Sync Layer. The AU may be larger than the SL packet, in which case it will be fragmented across multiple SL packets. At the receiving terminal, the AUs are reassembled and delivered to the Compression Layer. The SL packets contain a header with timing, sequencing and other information to provide synchronization and loss detection. These packetized elementary streams are sent to or received from the Delivery Layer based on the direction of media flow. At the time of this writing, there are ongoing discussions to place numerous short AUs (e.g., low-bitrate audio) in a single SL packet to increase efficiency and reduce overhead [12].

The Delivery Layer provides a means of retrieving MPEG-4 presentations. This layer provides an abstraction layer between the core MPEG-4 systems components and the retrieval method. Unlike its predecessors, MPEG-4 does not target any specific transport technology but instead defines a framework for seamless utilization of these technologies. This framework is referred to as the Delivery Multimedia Integration Framework (DMIF) [2]. It addresses the issues of local file access, broadcast media access, and remote retrieval media access. Due to the exceeding demand in Internet multimedia, DMIF has sought to provide a flexible delivery platform for MPEG-4 and other multimedia technologies.

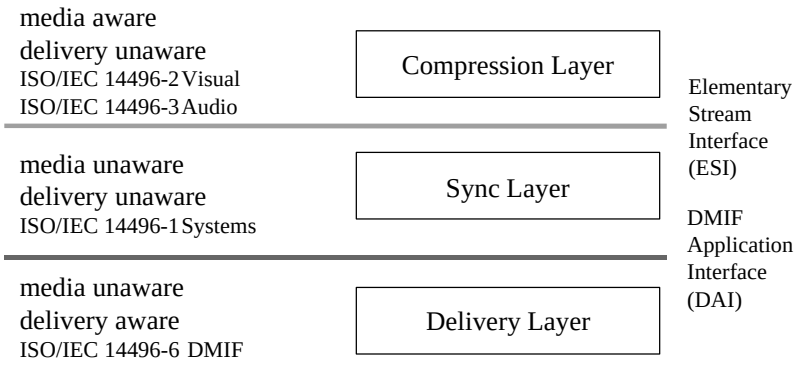

Figure 1 — ISO/IEC 14496 Terminal Architecture

The MPEG-4 standard does not enforce any delivery specifications for media transport across the Internet. There are a number of possible delivery platforms that could be tied into this technology for transport. Some of these platforms are DMIF based, while others do not acknowledge this framework and ignore this part of the standard [4,5].

Although DMIF was not specified solely for MPEG-4, it has been through the effort of the MPEG committee that this flexible delivery framework has been standardized and has become an integrated part of MPEG-4. DMIF has numerous advantages in that it abstracts the media from the delivery technology and enables easy utilization of various media access techniques [3]. These advantages are evident through our implementations of an MPEG-4 streaming system that utilizes the DMIF as its media access framework. Other advantages of DMIF include the QoS provisions and the client/server capability exchange provisions.

This paper presents the issues involved in the realization of the remote and broadcast instances as well as their respective server architectures for MPEG-4 content retrieval. Section II provides an overview of the DMIF communication model and its significance in MPEG-4. The overall system architecture used in our implementations is discussed in section III. In section IV, we discuss the issues in the design and development of a system for the remote retrieval scenario. Section V describes the broadcast scenario design and implementation.

II. THE DMIF COMMUNICATION MODEL

During remote media access, two communication planes are required: a Data Plane for the transport of media data (e.g., video stream), and a Control Plane used for media session management. The DMIF specification adopts out of band signalling, and therefore the Control and Data Planes can use different transport protocols. To ensure the reliability of Control Plane messaging in error prone environments, an error-free transport scheme should be employed.

The architecture used for realizing the remote and broadcast instances of the DMIF correspond to the recommendations made in part 6 of the MPEG-4 standard [2]. The DMIF model is depicted in Figure 2. An overview of the major components of the client/server system and the messaging that takes place

between distributed peers is also seen. The figure shows the local retrieval, broadcast and remote retrieval instances. The latter two scenarios are discussed in further detail in the following sections.

Hypothetically, the DMIF Control Plane can be replaced by any session layer protocol, e.g., Real Time Streaming Protocol (RTSP). However RTSP is not parallel to DMIF but it can be used instead of DMIF or in conjunction with it. One of the characteristics of MPEG-4 media is that it could potentially be composed of a large number of streams. DMIF has been specifically designed to handle these situations, whereas other streaming control protocols, including RTSP, would have to be adapted and greatly extended for such scenarios.

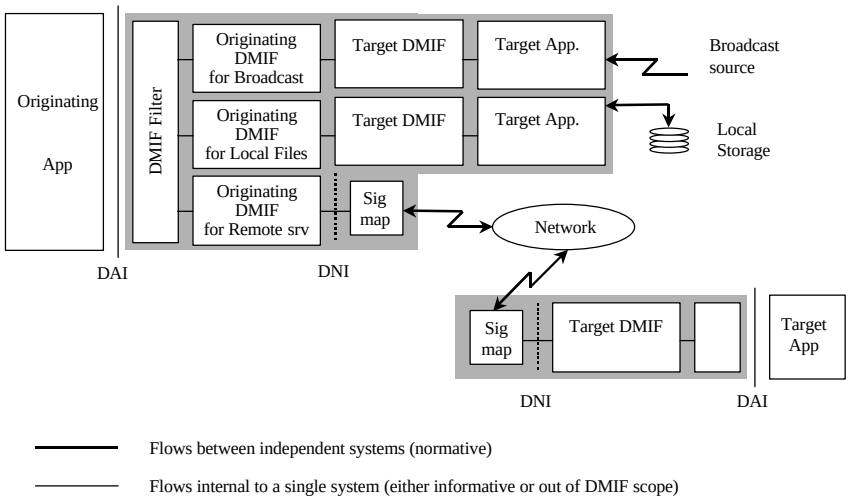

Figure 2. DMIF Communication Model

III. SYSTEM ARCHITECTURE

Our MPEG-4 Streaming System architecture conforms to the recommendations made by the MPEG-4/DMIF standard [2]. As mentioned previously, this results in the existence of a Data Plane and a Control Plane for out-of-band signalling. The system architecture is depicted in Figure 3. This architecture covers both the remote and broadcast retrieval scenarios. Although similar, there are some differentiating factors between the two scenarios that are further discussed in their respective sections. For example, DMIF Default Signalling Protocol (DDSP) is specific to the remote retrieval scenario.

In the remote retrieval scenario, session management requests (initiation, control, termination, etc.) traverse the following modules from the client application to arrive at the server application layer for processing: DMIF layer, underlying transport layer (e.g., TCP), server listening thread, and the server DMIF layer. Through the use of the Control Plane, the client application can request media transmission from the server. This request will be made through the Control Plane of the DMIF layer and sent to the server using a transport channel (e.g., TCP, UDP). During session initiation, the initial object descriptor (IOD) will be sent to the client. A channel is then created to carry the scene descriptor (SD), which describes the relationship between various objects in the presentation. Based on the IOD and the SD, the client determines the number of Data Plane channels (Transmux channels) required to receive the media streams and creates these channels through Control Plane messaging. This is in essence the Data Plane through which the media will arrive at the client. Once the session initiation messaging is completed, the client application will request to ‘Play’ the presentation. When the ‘Play’ request arrives at the server, the server will request the media from the elementary stream provider and send this data to the client through the Data Plane. The flow of media data through the Data Plane from the server to the client can be seen in Figure 3 and is as follows: Elementary Stream provider, packetizer, underlying transmission layer (i.e., RTP, UDP), client input Transmux channels, depacketizer (Sync Layer), decoder at the client application for playback.

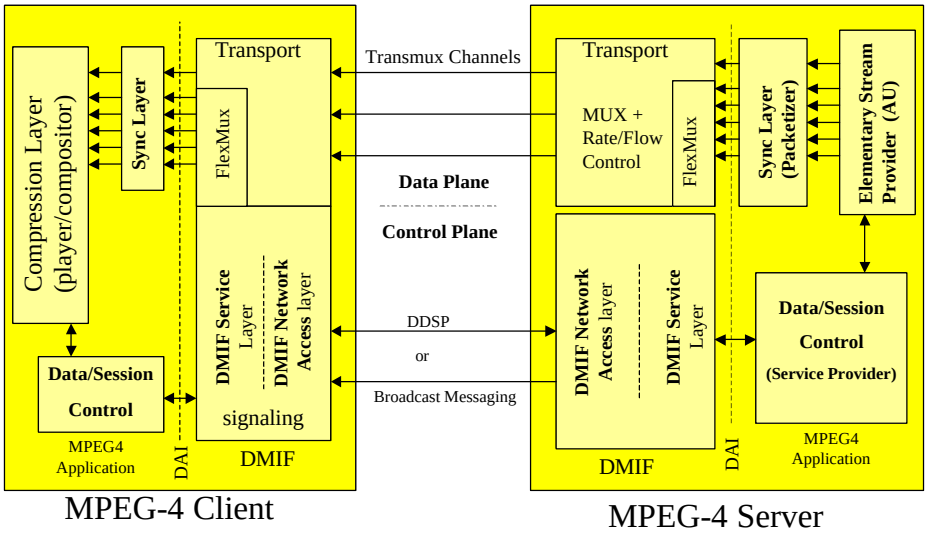

Figure 3. System Architecture

In the broadcast scenario, the server broadcasts session initiation messages periodically, sending the initial object descriptor and the stream map table to all the listening clients. Using this information, clients can locally initiate a session (no interaction with the server) and create data channels. Once data channels are created, the client starts receiving the media streams through these channels. In this scenario, the clients are passive elements receiving media streams and control messages broadcasted by the server. As was the case with the remote retrieval scenario, a network/medium access sub-layer, (i.e. DMIF Medium Access Layer) is responsible for providing a transparent interface for Network connectivity.

IV. REMOTE RETRIEVAL SCENARIO

This scenario has been implemented using a client/server architecture. Since DMIF inherently provides a means for peer-to-peer communication, a peer-to-peer model can easily be realized. The following sections discuss the issues involved in the implementation of a client DMIF instance and an MPEG-4 streaming server for the remote retrieval scenario.

A. MPEG-4 Client

The client implementation is based on the architecture depicted in Figure 4, which represents the instance involved in handling remote media access. The resulting software module, which supports remote access of MPEG-4 content, implements the recommended DMIF Application Interface (DAI). This module interacts with the application through the DMIF client filter. The DMIF standard only describes the semantics of the DAI; defining the syntax is left to the system developers. In our implementation, the DMIF instance for remote retrieval (also referred to as the Remote Instance) is a Dynamically Linked Library, however it could be linked in any other practical form such as a static library. This library was tested using the MPEG-4 reference software, IM1. Interaction between this instance and the DMIF client filter passes through an interface called the DMIF-Plug-in Interface (DPI). The Remote Instance is composed of two main layers as shown in Figure 4. The upper layer, DMIF Service layer (DS), interacts with the DMIF filter and provides the services requested by the application. The lower layer, DMIF Network Access layer (DNA), handles the network control messaging between peers and implements the DMIF Default Signalling Protocol (DDSP). The DNA layer is accessed by the DS layer through the DMIF-Network-Interface (DNI).

Media data is transported across native network transport channels that are referred to as Transmux channels. Creating Transmux channels and managing network sessions between DMIF peers are done using the functionality provided by Network Session objects in the DMIF Network Access layer. The

implemented DNA layer presents its functionality through the DNI primitives regardless of the protocol used for the transportation of the DDSP messages.

The main functionality expected from the DMIF Service layer is to create and manage DMIF services and hide the technology used to transport the control messages and elementary streams. This is done using DMIF Service objects created in this layer.

The separation of the DS and DNA layers facilitates employing a variety of transport technologies. In our implementation of the Control Plane, messages are transported by either TCP or UDP. Since DDSP does not provide error recovery facilities, lost UDP datagrams can halt the system. In order to prevent this problem while UDP is used, a simple error recovery scheme can be applied to the DNA layer. However, UDP is adequate for testing in local intranets, where Ethernet is used as the underlying transport technology, since there is no overhead and little possibility of loss. When using TCP as the transport for the DDSP messages, no error recovery is required. However, TCP requires a connection set-up phase prior to sending the first DDSP message. The reliability that is inherently provided by TCP outweighs the undesirable initial delay. For each network session one TCP connection is established at session-setup time and later DMIF messages are transported by this dedicated connection; therefore only the session-setup message suffers from the TCP initial delay.

The structure used in our implementation, according to the standard, implies that the Data Plane can use any available delivery technology regardless of the delivery scheme used by the Control Plane. In the current implementation, UDP is used for the delivery of MPEG-4 content. TCP is not a good candidate for the transport of time-critical data due to its preference for reliable transmission over timely delivery. RTP would provide additional benefits and is being considered for future development.

Data Plane channels are created in the Transmux channels that have been established between the client and the server. For each Transmux channel a client listening thread is dispatched to receive the packetized elementary stream. At the server, several data channels are multiplexed into one or more Transmux channels; therefore the packets containing multiplexed elementary streams need to be demultiplexed in the client DMIF Instance before being delivered to the Sync Layer. Following the presentation, the client will close the data and Transmux channels before terminating its session with the server.

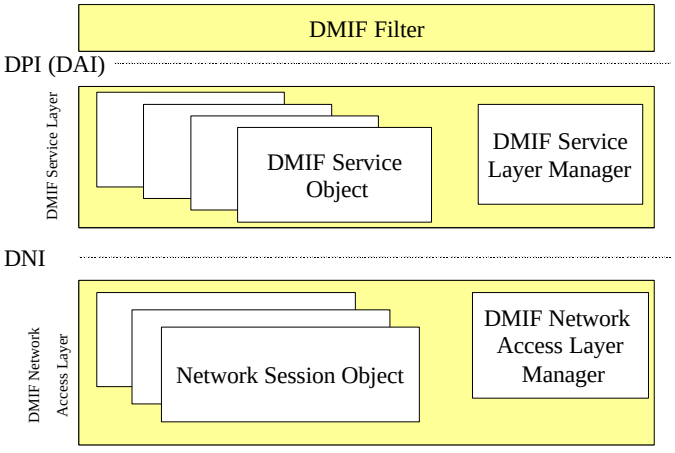

Figure 4. Architecture of DMIF Instance for Remote Retrieval

B. The MPEG-4 Streaming Server

This server implementation is only applicable to the remote retrieval scenario. The Streaming Server implementation consists of two layers, an application layer (service provider layer) and a DMIF layer. The DMIF layer is further divided into two parts, as is the case with the client DMIF layer. The major contribution of this paper entails the server architecture design and implementation, which includes a

multi-client service capability. The server architecture can be seen in Figure 5. Flexibility in exploiting different transport technologies is one of the advantages this architecture offers. The server implementation also supports interactive media transmission.

The Server Application consists of two distinct layers. One layer is in essence the Service Provider layer for the client requests. The other layer consists of one or more Elementary Stream (ES) Provider. The Service Provider layer includes the realization of MPEG-4 Sync Layer. The elementary streams are provided to this layer by the ES Provider, which can be a real-time MPEG-4 encoder, an MP4 file reader, etc. These elementary streams are packetized in this layer considering the transport protocol used by the Data Plane. These stream providers can be added to the implementation as new DLL's and this addition will not oblige any changes in the Server Application layer. To facilitate this media resource abstraction, a Server Resource Interface (SRI) has been defined. At the time of writing this document, only the MP4 file reader instance has been exploited as an ES provider.

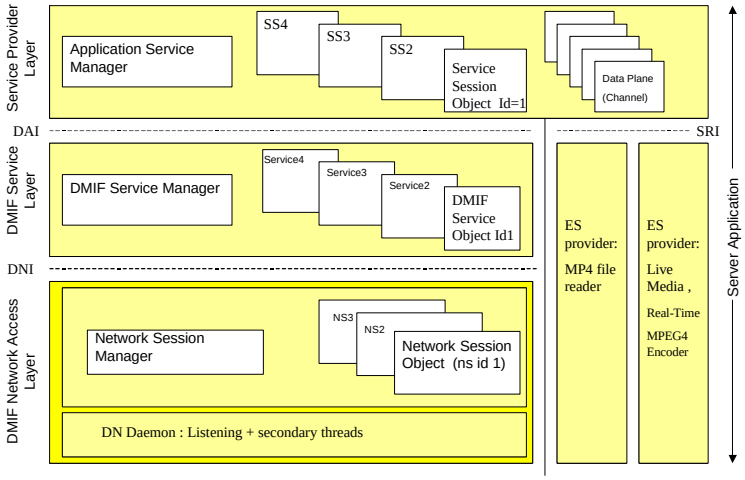

Figure 5. MPEG-4 Streaming Server Architecture

The DMIF layer of the Remote Streaming Server varies slightly from the client-side DMIF layer in that every client request forces the creation of an individual and independent thread. This thread traverses all the DMIF layers as well as the application layer using the appropriate servicing objects. During normal operations, the server only has threads processing the media data for its clients and does not have a thread dedicated to that session awaiting client requests. Whereas with the creation of a single thread per request, once the request has been addressed, the thread will be terminated and the server will be lightly loaded but the servicing objects will still be available for future requests. For example in a scenario where there are 100 streams being sent to 50 clients, our server would need an average of slightly over 100 threads to serve the clients, whereas in the case where each session requires a managing thread, that number would be 150. Each layer in the server architecture consists of a Manager object and a number of Service objects as seen in Figure 5. The DNA layer objects, Network Session objects, handle the DDSP signalling between two peers. All the requests associated with a specific DMIF service are handled by its Service object. Each tuple of Network Session and DMIF Service objects is linked to a Service Session object in the Application (Sync) layer. Thus, each request to the server, as it traverses through the layers, is served by a Network Session object, a DMIF Service object, and a Service Session object. The thread that has been created to service the request is a member of the first servicing object, the Network Session. As the thread traverses the upper layers, it invokes member functions of its associated service objects. This makes the implementation very scalable and enforces separation between various clients since their object handlers are disparate.

The server DMIF layer contains two distinct network access instances for Control Plane messaging; a UDP instance and a TCP instance. It must be noticed that the media transmission method used in the Data Plane is not dependent on the transport protocol adopted by the Control Plane. As

mentioned before, the disadvantage of using TCP, in terms of its slow-start and initial delay, may be outweighed by the reliability it provides. Therefore, applying the error-recovery capability to the UDP instance may be an unnecessary overhead. The flexibility of our architecture facilitated the adoption of both transport methods. The user will then have the ability to choose between them based on network reliability (by specifying it in the requested URL).

C. Data Plane

The Data Plane, as seen in Figure 3, is the transportation medium for multimedia data. This multimedia data can be delivered using any underlying transport protocols, however UDP and RTP are the primary candidates.

Since the MPEG-4 Sync Layer provides some transport related functionalities such as sequence numbering and timing, UDP seems to be the most straightforward option for the delivery of MPEG-4 data over IP. The availability of a multiplexing tool such as FlexMux further encourages the use of UDP, a simple transport protocol. Despite the advantage of simplicity, there are some problems associated with this approach. Inter-media synchronization cannot easily be achieved between MPEG-4 streams coming from different sources. Moreover, other multimedia data streams cannot be synchronized with MPEG-4 data delivered over UDP. Not providing any feedback by UDP, may force the creation of an MPEG-4 backchannel for carrying quality feedbacks.

Exploiting Real-Time Transport Protocol (RTP) in the data plane [4] compensates for some of the aforementioned shortcomings, but adds to the complexity of the system. Quality feedbacks are to some extent provided by the Real-Time Control Protocol (RTCP), a part of RTP. Inter-media synchronization is by nature supported in RTP; therefore synchronization of streams coming from multiple servers becomes possible. Furthermore, non-MPEG-4 data can be synchronized with MPEG-4 data using the timing information available through RTP/RTCP.

There are several approaches to encapsulate SL packets in RTP packets. At the time of writing this document, none of these approaches have been standardized. RTP header contains sequence-numbering and timing information. These fields are redundant since they have already been included in SL packets. This unnecessary redundancy can be removed by modifying the SL headers before transmission. In order to keep the integrity of the Sync Layer, this modification must be accomplished in the DMIF Data Plane and not in the Sync Layer. At the receiver, the DMIF layer rebuilds the original SL packets, and the entire operation remains transparent to the Sync Layer.

Regardless of the underlying transport scheme that may be used, the following holds true for the Data Plane. During channel setup, a number of Transmux channels are created to transport the application data from the server to the client. The Transmux channel has a one-to-one association with a communication channel (e.g., RTP sessions or UDP port). When a client request to initiate the media sequence, i.e. “Play,” arrives at the server, a dedicated thread is dispatched per data channel to send the packetized elementary streams to the DMIF Network Access layer. The DNA layer multiplexes and transmits the packetized elementary streams across the Transmux channel. The Data Plane in our implementation uses the simple mode of the FlexMux recommendations to multiplex the packetized elementary streams for transmission.

The Data Plane at the client needs to de-multiplex the packetized elementary streams and route the packets to their associated data channels. The Sync Layer reassembles the media packets to their primary form of access units. Packetization and reassembly of SL packets are out of the scope of DMIF and are performed in the Synchronization Layer. Other issues the Data Plane deals with are flow and rate control.

The functionality of the Data Plane at the server differs somewhat from that of the client. The ES providers, seen in Figure 5, supply the service provider (application) layer with media access units. The Sync Layer, which is part of the service provider layer, receives and packetizes the access units. Using the information that is provided to the Sync Layer, the SL headers are generated. SL Packets are then multiplexed and passed to the DMIF layer to be transported over Transmux channels. Rate-control is one of the issues that must be addressed at the server. The timing information that is available in the Sync Layer helps in adjusting the rate of transmission. This scheme prevents data underflow or overflow at the client. However, it does not address network bandwidth issues. In order to allow for bandwidth fluctuations, an

adaptive source-coding algorithm must be adopted [7]. Our server rate-control mechanism takes into account the client media buffer size. The client media buffer is used to compensate for the jitter due to the non-deterministic behaviour of the network. The transmission rate of the server is higher at the beginning of a session, in order to fill the client media buffer. Based on the client buffer size, the transmission rate then is gradually slowed to the real rate of the media data.

The rate control scheme may be implemented in any layer depending on the available control information. The more media and network information a layer has, the more efficient it can control the transmission rate. Different QoS constraints will also affect the way each stream is treated [8]. In circumstances where packets need to be queued and buffered before transmission (e.g., heavy server load), real-time scheduling methods must be used.

V. BROADCAST SCENARIO

The broadcast retrieval scenario has been implemented using a client/server architecture. Although our experiments and implementation are based on the broadcast facilities provided by IP, the issues discussed in the following sections will still be valid for any other broadcast technology.

A. MPEG-4 Client

Our client implementation of the broadcast DMIF instance is based on the architecture depicted in Figure 6. The resulting software module implements the recommended DMIF Application Interface (DAI). This module interacts with the application layer through the DMIF client filter. The DMIF broadcast instance is also a dynamic link library (refer to section IV.A), which was tested using the IM1 reference software. Similar to our remote retrieval implementation, the broadcast instance is composed of the DMIF Service layer (DS), which interacts with the DMIF filter to provide the requested services; and the DMIF Medium access layer (DMA), which provides access to the broadcast network as shown in Figure 6.

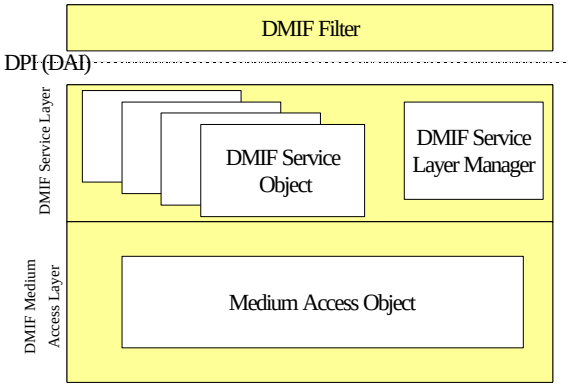

Figure 6. Architecture of Broadcast DMIF instance

In this scenario, both the server and the client perform session initiation and management locally. There is no messaging between the terminals except for the flow of media data, which is delivered through data channels. As described in section IV.C, the client receives the IOD and stream ODs through the DMA layer. These descriptors are used to create the data channels required to handle the incoming streams. The incoming data is then demultiplexed based on the elementary stream identifier and sent to the application sync layer. The sync layer outputs the access units corresponding to the elementary streams and forward them to the application layer for decoding and playback through the composition module. Our experiments have been conducted in a LAN environment using IP delivery facilities. In the broadcast scenario the server receives no feedback from its clients, thus UDP was our primary choice for broadcast emulation.

B. MPEG-4 Server

The DMIF layer of our broadcast server implementation is based on the same architecture depicted in Figure 6. During broadcast, the ES provider feeds the SL with elementary streams. These elementary streams are packetized in the SL, and then multiplexed to one or more broadcast channels in the DMIF layer for transmission.

There are some key issues that need to be addressed in this server implementation. One of these key issues is "client random access" [11]. This concept enables clients to tune in to the broadcasted channels at any time during the presentation. To accommodate this feature, the server has to resend object descriptors periodically (i.e., IOD, BIFS information, and stream object descriptors). These descriptors contain all the information required for creating the data channels at the client and accessing the SL packets in the broadcast channels. This process is depicted in Figure 7. The period of this repetition is derived dynamically throughout the presentation based on the number of available elementary streams.

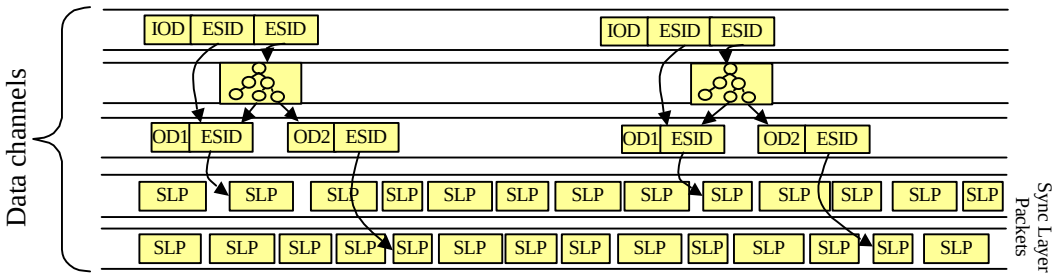

Figure 7. MPEG-4 DMIF broadcast model for client random access

Elementary stream synchronization is another key issue that our server implementation addresses [11]. This enables the client to access the elementary streams at any time during the presentation. We used the timing information from the Sync layer to synchronize the transmission of elementary streams on the server side. This mechanism, however, does not address any network bandwidth fluctuations. Instead, an adaptive coding algorithm must be implemented to resolve these bandwidth changes [7].

VI. CONCLUDING REMARKS

In this paper we have described a scalable architecture for an MPEG-4 Streaming Server as well as a practical design of the DMIF instances for remote retrieval and broadcast scenarios (at the client). These implementations lay the basis for further experiments with the DMIF model. There are various additions that our implementations of the remote retrieval and broadcast scenarios could benefit from.

With the use of the QoS metrics the remote retrieval server and client can implement the channel monitor functionality described in the DMIF standard. This functionality, in addition to a reservation protocol, will enable the client and the server to exchange quality constraints. Also, with the use of RTP for Data Plane media transmission, rate-control could be achieved to meet network bandwidth availability constraints. Another addition to the server would involve transmitting media data based on client capabilities. Therefore, a server could send a reduced set of the available elementary streams to the client to ensure valid media within the client's processing capability threshold. Some server related issues that can be addressed in future versions, are source-based media transcoding and quality assurance through reservation technologies and traffic engineering. This implementation makes use of the DMIF platform that has provisions for such better than best effort technology. Although the system can use QoS metrics in the messaging layer, the required functionality will not be implemented until future revisions. This system

provides the necessary framework for testing stream flow control methods, as well as a platform for exploiting a variety of delivery technologies (e.g., RTP/RTCP, TCP, UDP).

One of the potential enhancements to our broadcast implementation is "push technology" for broadcast/multicast networks [10]. This would provide a client subscription mechanism that will implement a publisher/subscriber model. This model requires the server to broadcast information about all the available channels. Using these broadcast messages, clients can then register to one of the available channels. This may involve using a protocol for announcing the available channels (e.g., session announcement protocol (SAP)) and another protocol for sending channel description information (e.g., session description protocol (SDP)). With this information, clients can subscribe to the broadcast channel.

We have provided our source code for the DMIF instance for remote retrieval and its corresponding server to the MPEG-4 community. A working executable of the Remote Instance and Broadcast Servers as well as the compiled DLL for the remote retrieval instance may be downloaded from the following location: http://www.ece.ubc.ca/MCNL/apadana.html. The required plug-in DLL for the DMIF instance for remote retrieval can also be obtained from this location. This DLL should be used with the IM1 software to conduct remote retrieval experiments. This will allow for performance testing of streamed MPEG-4 media across the Internet with various constraints.

REFERENCES

[1] Coding of Audio-Visual Objects – Part 1: Systems, ISO/IEC 14496-1 International Standard, ISO/IEC JTC1/SC29/WG11 N2501, March 2000.

[2] Coding of Audio-Visual Objects – Part 6: Delivery Multimedia Integration Framework (DMIF), ISO/IEC 14496-6 International Standard, ISO/IEC JTC1/SC29/WG11 N2501, March 2000.

[3] G. Franceschini, "The Delivery Layer in MPEG-4," Signal Processing: Image Communication, vol. 15, pp. 347-363.

[4] Y. Kikuchi, T. Nomura, S. Fukunaga, Y. Matsui, H. Kimata,, "RTP payload format for MPEG-4 Audio/Visual streams," Internet-Draft draft-ietf-avt-rtp-mpeg4-es-04.txt

[5] Dapeng Wu, Yiwei Thomas Hou, Wenwu Zhu, Hung-Ju Lee, Tihao Chiang, Ya-Qin Zhang, H. Jonathan Chao, "On End-to-End Architecture on Transporting MPEG-4 Video over the Internet," IEEE Trans. on Circuit and Syst. for Video Tech., Vol. 10, No. 6, September 2000, pp. 923-941.

[6] Civanlar, Casner, Herpel, "RTP Payload Format for MPEG-4 Streams," Internet Draft draft-ietf-avt-rtp-mpeg4-03.txt

[7] Hung-Ju Lee, Tihao Chiang, and Ya-Qin Zhang, "Scalable Rate Control for MPEG-4 Video," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, No. 6, September 2000, pp. 878-894.

[8] A. Betro, H. F. Sun, and Y. Wang, "MPEG-4 Rate Control for Multiple Video Objects," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, February 1999, pp. 186-199.

[9] W. Ding and B. Liu, "Rate Control of MPEG Video Coding and Recording by Rate-Quantization Modeling," IEEE Trans. on Circuits and Syst. for Video Tech., Vol. 6, February 1996, pp. 12-20.

[10] T. Liao, "Global Information Broadcast: An Architecture for Internet Push Channels," IEEE Internet computing July/August 2000, p16-25.

[11] C. Herpel, A.Eleftheriadis, "MPEG-4 Systems: Elementary stream management" Signal Processing: Image Communication 15 (2000) p299-320.

[12] G. Franceschini, "Improvements to the SL," Proposal to MPEG committee, November 2000.