DELIVERING OBJECT-BASED AUDIO-VISUAL SERVICES

Hari Kalva, Alexandros Eleftheriadis, Javier Zamora

Delivery of audio-visual objects differs from the traditional video on demand systems in the characteristics of the presentations delivered. Video on demand has mainly been about delivering MPEG-2 audio and video[1]. In the case of object based presentations such as MPEG-4 presentations, the media data and the media composition data are transmitted to a client as separate streams, typically with different QOS requirements, in the same session. Furthermore, as the number of objects in a presentation can be quite large, the overhead required to manage the session is large. Interactivity makes this problem more complex as the resources required for a session will now depend on the user behavior, especially when user interaction with objects changes the number of objects in the scene either by adding or deleting objects. In this paper, we present our recent work on delivering audio-visual services based on MPEG-4. The issues covered include object-based audio-visual (MPEG-4) servers, object scheduling, and content creation.

Multimedia, MPEG-4, audio-visual services, object-based representation.

I. Introduction

Image and video encoding has been totally transformed with the advent of new coding and representation techniques [6]. These next generation coding techniques have made possible encoding and representation of audio-visual scenes with semantically meaningful objects. Such a new paradigm of object-based representation of audio-visual scenes and presentations will change the way audio-visual applications are created.

MPEG-4 is specifying tools to encode individual objects, compose presentations with objects, store these object-based presentations and access these presentations in a distributed manner over networks. The main distinguishing feature of object-based audio-visual presentations is the scene composition at the user terminal. The objects that are part of a scene are composed and displayed at the user end as opposed to encoding the composed scenes as is done in the case of MPEG-2. Such object-based representation and presentation has several benefits including compression efficiency and interaction with individual objects. Consider the home shopping channel that is currently available on TV. The information on the screen mostly consists of text, images of products, audio (sales pitch), and sometimes quarter-screen video. All this information is encoded using MPEG-2 video/audio at 30 FPS. However, if this is created using object-based technology, all static information such as text and graphics is transmitted only at the beginning of a scene and the rest of the transmission consists of only audio and quarter-screen video that takes up significantly less bandwidth. In addition to this, the ability to interact with individual objects makes applications such as e-commerce possible.

For all its benefits, object-based presentations bring complexity to both clients and servers. MPEG-2 is asymmetrical in terms of complexity; i.e., encoder is designed to be much more complex in order to keep the decoders simple. While the video streams still follow the same approach, the Systems capabilities make an MPEG-4 player much more complex. MPEG-4 servers are also more complex as they have to handle user interaction in addition to managing individual objects. The complexity depends on the content being delivered. Unlike MPEG-2 streams, MPEG-4 presentations cannot be characterized by a single bit-rate. The complexity actually depends on the number of objects and object playout. The scheduler at the server should take the complexity of presentations into consideration. We are working on algorithms to schedule the delivery of object-based audio-visual presentations in general and MPEG-4 presentations in particular.

This paper is organized as follows. Section 2 gives an overview of MPEG-4 Systems specification. MPEG-4 delivery layer is presented in Section 3. Architecture of the MPEG-4 system we have implemented is described in Section 4, issues in the delivery of object based presentations are discussed in Section 5.

II. MPEG-4 Systems Layer

The MPEG-4 Systems specification provides the glue that binds the audio-visual objects in a presentation [1][5]. The basis for the MPEG-4 Systems architecture is the separation of the media and data streams from the stream descriptions. Stream description, also referred to as BIFS (Binary Format for Scenes), itself is carried in a separate stream. This separation allows for providing different Quality of Service (QOS) for different streams; for example, scene description streams which have very low or no loss tolerance and the associated media streams, which are usually loss tolerant. These individual streams are referred to as elementary streams. These elementary streams are carried across the Systems layer as sync-layer (SL) packetized streams. The sync-layer is configurable and the configuration for a specific elementary stream is specified in the corresponding

H. Kalva and A. Eleftheriadis are with Columbia University, New York and J. Zamora is with Xbind Inc., New York.

Manuscript received July 2, 1999

elementary stream descriptor. Figure 1 shows the data flows between a client and a server.

The data communicated to the client from a server includes at least one scene description. The scene description stream,

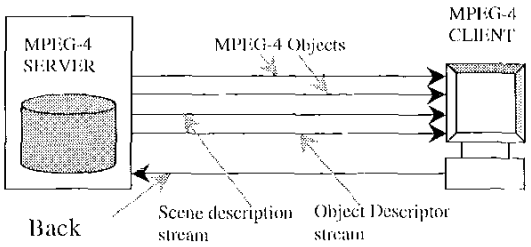


Figure 1. Elementary Streams in an MPEG-4 System

as the name indicates, carries the information that specifies the spatio-temporal composition of objects in a scene. The MPEG-4 scene description is based on the VRML specification. In addition to VRML functionality, MPEG-4 has includes features perform server interaction, polling terminal capability, binary encoding of scenes, and dynamic scene updates. The scene description contains URLs to audio-visual data. The URLs can either point to object descriptors in the object descriptor stream or media data directly at the specified URL. The object descriptor stream contains a sequence of object descriptors (OD) that contain the description of the MPEG-4 objects referred in the scene description. The object descriptor framework is extensible and supports user defined descriptors for custom applications. In addition to the streams shown in Figure 1, an MPEG-4 session could contain an Intellectual Property Management and Protection (IPMP) stream to protect media streams, an Object Content Information (OCI) stream that describes contents of the presentation, and a clock reference stream. All the data flows between a client and a server are SL-packetized.

The sync layer contains the information necessary for inter-media synchronization. Unlike MPEG-2, MPEG-4 Systems does not specify a single clock speed for the elementary streams. Each elementary stream in an MPEG-4 presentation can potentially have a different clock speed. This puts additional burden on a terminal, as it now has to support recovery of multiple clocks.

A. *Bootstrapping an MPEG-4 Terminal*

Since all the data flows to a client are SL-packetized, the question is how does one get the sync-layer configuration of these data flows? When an MPEG-4 session is started, the very first data a terminal receives is the *initial object descriptor* (IOD). The IOD contains the elementary stream descriptors for the scene description stream and possibly an object descriptor stream. The terminal then starts decoding the scene description stream and the OD stream. As the scene graph is constructed, the objects referred to in the scene description are retrieved by establishing channels for the elementary streams, decoded, composed and displayed. The IOD itself is not SL packetized and is usually transmitted to a terminal in a successful response to a session establishment request. The session establishment and channel establishment is done both in the case of local access and networked access. To keep the interface to the underlying transport independent of the transport, MPEG-4 specified a semantic interface to the transport layer. This is described in detail in the next section.

III. The MPEG-4 Delivery Layer

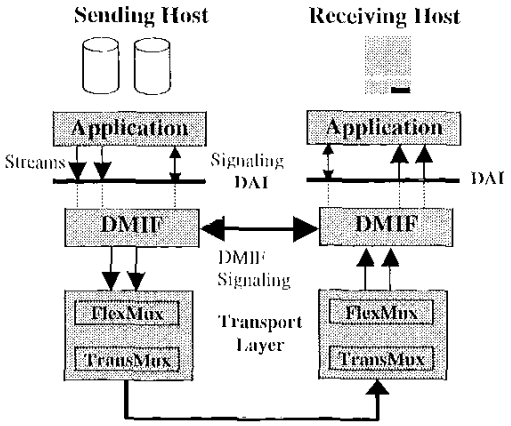


Figure 2. MPEG-4 Transport Layer

The MPEG-4 Systems architecture allows creation of complex scenes with potentially hundreds of objects, thus calling for a high rate of establishment and release of numerous short-term transport channels with the appropriate QOS. Traditional methods of signaling are not adequate to meet this demand because of the high overhead introduced. Furthermore, the applications should not depend on the underlying transport network. The Delivery Multimedia Integration Framework (DMIF) is a general application and transport delivery framework specified by MPEG-4 [4]. DMIF's main purpose is to hide the details of the transport network from the user, as well as to ensure signaling and transport interoperability between end-systems. In order to keep the user unaware of underlying transport details, MPEG-4 defined an interface between user level applications and DMIF, called the DMIF Application Interface (DAI). The DAI provides the required functionality for realizing multimedia applications with QOS support [3].

Through the DAI the user may request service sessions and transport channels without concerns about the selected communication protocol. Furthermore, the DAI shields legacy applications from new delivery technologies since it is the responsibility of the underlying DMIF system to adapt to these new transport mechanisms.

DMIF also specifies an informative DMIF Network Interface (DNI) [4]. This interface highlights the actions that DMIF shall trigger with respect to the network, and the parameters that DMIF peers need to exchange across the network. The actions and parameters of DNI embody the semantics for DMIF Signaling (DS) Messages. DS

Messages define a default session signaling protocol. DNI can also be mapped/piggy-backed on the existing signaling infrastructures.

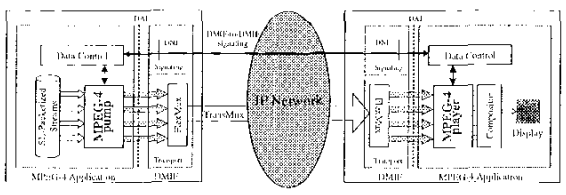
Figure 2 depicts the systems architecture of *XDMIF*, the first reference implementation of DMIF from Xbind [8] and illustrates some of the interactions and components involved during the creation of a multimedia service. The system architecture consists of a signaling and a transport component. These entities or components support the function calls of the DAI. The Control-Plane component might interact with a resource manager that establishes connections and allocates network resources for transport and/or signaling channels. Figure 2 also depicts the FlexMux and TransMux components of the User-Plane.

The user plane DMIF layer is divided into Flexible Multiplexing (FlexMux) and Transport Multiplexing (TransMux) sub-layers. The FlexMux entity models a multiplexing layer which allows grouping of ESs with similar QOS requirements, while the TransMux Layer models the protocol stack that offers transport services matching the requested QOS.

The FlexMux represents a multiplexing layer that provides interleaving of one or more SL-Packetized Streams belonging to different ESs with similar QOS requirements. Each SL-Packet is mapped into a specific FlexMux channel and is provided to the FlexMux layer through the DAI. The FlexMux layer encapsulates one or more SL-Packet(s) into one FlexMux-PDU. The sequence of FlexMux-PDUs interleaved into one stream is called the FlexMux stream. The FlexMux demultiplexer retrieves SPSs from the FlexMux streams and through the DAI invocations provides them to the application.

The FlexMux layer does not provide a reliable error detection or framing of its streams. This functionality is provided via the underlying TransMux layer. The generic term TransMux layer is used to abstract any transport protocol stack that is suitable to transport MPEG-4 data streams and fulfill the end-to-end QOS requirements. The selected transport stack should provide appropriate framing of FlexMux streams, error detection and, if possible, delivery of corrupted data along with an error indication.

IV. System Architecture



MPEG-4 Server MPEG-4 Client

Figure 3. Components of the MPEG-4 System

The MPEG-4 system developed is an end-to-end system consisting of an MPEG-4 server, the DMIF component for transport, signaling, and session management, an IP network, and an MPEG-4 client for media playback/rendering. Figure 3 shows the components of the system.

This system differs from the traditional video on demand systems in the characteristics of the presentations delivered. Video on demand has mainly been about delivering MPEG-2 audio and video. In the case of object based presentations such as MPEG-4 presentations, the media data and the media composition data are transmitted to a client as separate streams, typically with different QoS requirements, in the same session. Furthermore, as the number of objects in a presentation can be quite large, the overhead required to manage the session is large. Interactivity makes this problem more complex as the resources required for a session will now depend on the user behavior, especially when user interaction with objects changes the number of objects in the scene either by adding or deleting objects. The MPEG-4 server consists of an MPEG-4, pump, an object scheduler, and a DMIF instance for IP networks. The server delivers Sync Layer Packets (SL-Packets) to the DMIF layer, which multiplexes them in a FlexMux and transmits them to the client.

The complexity of the player (i.e., client) has grown as a result of the new features and functionality offered by MPEG-4. The player is now also responsible for composing a scene from individual objects in addition to decoding and displaying the objects. A player consists of three logical components, a DMIF instance, elementary stream decoders, and a compositor. The DMIF instance is responsible for managing data access from a network or a file. A player typically contains several decoders each handling a specific elementary stream. Elementary streams are audio-visual streams as well as streams that describe the composition, rendering, and behavior of a presentation. Each object in a presentation is carried in a separate elementary stream. As MPEG-4 presentations can include media objects from several sources, potentially with different clock frequencies, there is additional burden on the client to track multiple clocks. This is typically done using soft Phase-Locked Loops (PLLs). Because of this additional complexity, a player's performance depends on the complexity of the content. Intelligent resource management and usage is necessary to use the resources such as memory efficiently.

The capability to add and remove objects during presentations and interacting with objects differentiates object based audio-visual presentations from traditional audio-visual presentations. DMIF supports this addition and removal of objects in a session efficiently. DMIF also supports network independence by providing a network independent API to the applications called, DMIF Application Interface (DAI). DMIF is also responsible for providing the requested QoS for the applications. Typically, streams with the same QoS requirements are multiplexed into a single channel called FlexMux. These FlexMux packets are transported to the other end on the under lying transport network where they are de-multiplexed by the peer DMIF entity and passed on to the application. DMIF is

also responsible for communicating the user interaction commands from Command Descriptors by means of DAI user command primitives. These commands are transmitted over the DMIF-to-DMIF signaling channel. Our system uses a DMIF instance for IP networks and has been tested over Ethernet and satellite. The media streams are transported using UDP while signaling uses TCP.

V. Content Delivery

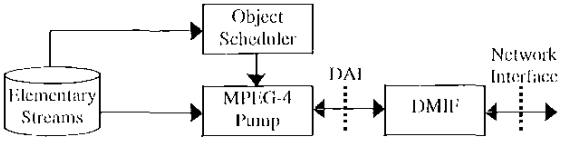


Figure 4. Components of an MPEG-4 Server

Figure 4 shows the components of an MPEG-4 server. An MPEG-4 server typically consists of an MPEG-4, pump, an object scheduler, and a DMIF instance for media transport and signaling. The server delivers Sync Layer Packets (SL-Packets) to the DMIF layer, which multiplexes them in a FlexMux and transmits them to the client. An MPEG-4 server that is transmitting objects should make sure that an access units arrive at the terminal before their decoding time. The server delivers objects in a presentation as scheduled by the presentation scheduler. The scheduler uses the decoding timestamps to schedule the delivery of access units.

The flexibility of MPEG-4 while allowing interactive and complex presentations makes the content creation process non-trivial. Unlike MPEG-2, content creation process involves much more than multiplexing the media streams. The scheduler is also useful during content creation process to determine if the presentation being designed can be scheduled for specific channel rates and client buffer capacity. It may not be possible to find a solution for a given set of resources; i.e, presentation cannot be scheduled with given resources. In order to create a schedulable presentation, some constraints may be relaxed. In the case of scheduling objects, relaxing a constraint may involve increasing the buffer capacity, increasing the channel capacity, not scheduling some object instances, or removing some objects from a presentation.

Network delays and data loss could occur in the system if the content does not request the suitable QoS. A presentation created without the knowledge of target networks and clients could create long startup delays and buffer overflows or underflows. This could cause distortion, gaps in media playback or problems with the synchronization of different media streams. Unlike MPEG-1 and MPEG-2, MPEG-4 presentations are not constant bit rate presentations. The bit rate of a presentation may be highly variable depending on the objects used in the presentation. Further more, there is no notion of bit rate when images are used. These characteristics of MPEG-4 presentations make it difficult to design servers as well as content. A presentation may have to be recreated for different targets or servers have to be intelligent enough to scale a presentation for different networks/clients. Schedulers should be part of content creation process to check the suitability of content for target networks and clients. MPEG-4 has scalable coding tools that allow creation of content that can be adapted to different network and bandwidth conditions.

The complexity of an MPEG-4 presentation is an important factor that influences a server's performance. In case of MPEG-2 content, the average bit-rate and peak bit rate are a good indication of the server resources required to deliver the stream. However, an MPEG-4 presentation cannot be characterized by individual or cumulative bit rates of the objects alone. For example, an MPEG-4 presentation may consist of a sequence of large JPEG images with accompanying audio. Such presentations tend to be very bursty over networks. Since objects may span any arbitrary time period during a presentation, the bit-rate of MPEG-4 presentations can be highly variable depending on the content of presentations. When user interaction is allowed, the resulting asynchronous events affect object scheduling and also the required network and server resources. There is no easier way of handling such events than reserving auxiliary capacity to meet interactivity requirements.

VI. Conclusion

MPEG-4 is an emerging standard with capabilities to support wide-ranging multimedia applications. The nature of the MPEG-4 presentations increases the complexity of clients as well as servers. Furthermore, the complexity involved is a function of the number of objects involved, the object playout, and user interaction that results in local or server events. New algorithms are necessary to optimally schedule the presentation delivery.

VII. References

[1] O. Avaro, A. Eleftheriadis, C. Herpel, G. Rajan, and L. Ward, "MPEG-4 Systems: Overview", *Signal Processing: Image Communication*, Special Issue on MPEG-4, 1999 (to appear).

[2] S.-F. Chang *et. al*, "Columbia's VoD and Multimedia Research Testbed with Heterogeneous Network Support", Journal on Multimedia Tools and Applications, Special Issue on Video on Demand, Kluwer Academic Publishers, Vol. 5, Nr. 2, September 1997, pp. 171-184.

[3] J.-F. Huard *et al.*, "Realizing the MPEG-4 Multimedia Delivery Framework," IEEE Network Magazine, pp. 35-45, *November/December 1998. Special issue on Transmission and* Distribution of Digital Video.

[4] ISO/IEC 14496-6 FDIS "Delivery Multimedia Integration Framework, DMIF," October 1998.

[5] ISO/IEC/SC29/WG11, "Generic Coding of Moving Pictures and Associated Audio (MPEG-4 Final DIS) ISO/IEC 14496-1," International Standards Organization, November 1998.

[6] A. Puri and A. Eleftheriadis, "MPEG-4: A Multimedia Coding Standard Supporting Mobile Applications" *ACM Mobile Networks and Applications Journal*, Special Issue on Mobile Multimedia Communications, Vol. 3, No. 1, June 1998, pp. 5-32 (invited paper).

[7] L. Torres, M. Kunt, eds., "Video Coding : The Second Generation Approach," Kluwer Academic Publishers, 1996.

[8] XDMIF demo software, http://www.xbind.com.