Delivering Object Based Audio-Visual Services

Hari Kalva

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the Graduate School of Arts and Sciences

Columbia University

2000

© 2000

Hari Kalva

All Rights Reserved

ABSTRACT

Delivering Object Based Audio-Visual Services

Hari Kalva

We investigate the different aspects of end-to-end multimedia services: content creation, server and service provider, network, and the end-user terminal. In the first part of the thesis we present the study of system level issues including standardization and interoperability, user interaction, and the design of a distributed video server. In the second part of the thesis we investigate the systems in the context of object-based multimedia services. We present a design for an object-based audio-visual terminal, some of the features of which have been adopted by the MPEG-4 Systems specification. We then present the study of the requirements for a file format to represent object-based audio-visual content and the design of one such format. The design introduces new concepts such as direct streaming that are essential for scalable servers. In the final part of the thesis we investigate the delivery of object-based multimedia presentations and give optimal algorithms for multiplex-scheduling of object-based audio-visual presentations. We show that the audio-visual object scheduling problem is NP-complete in the strong sense. The problem of scheduling audio-visual objects is similar to the problem of sequencing jobs on a single machine. We compare the problems and adapt job-sequencing results to audio-visual object scheduling. We give optimal algorithms for scheduling presentations under resource constraints. The constraints considered are the bandwidth (network constraints) and buffer (terminal constraints). We present algorithms that minimize the resources required for scheduling presentations. We investigate the structure of interactive audio-visual presentations by considering event specifications and event extents. We show that the only way to support interactivity is by reserving channel capacity to deliver the interactive components of presentations. We also present algorithms for computing the minimum auxiliary capacity required to support interactivity in object-based audio-visual presentations.

Contents

Acknowledgements

I would like to express my sincere gratitude to my advisor Prof. Alexandros Eleftheriadis for his support and direction. He provided not only technical and professional support but more importantly the moral support and friendship, without which I could not have possibly succeeded.

I acknowledge with gratitude the support and advice of Prof. Dimitris Anastassiou and Prof. Shih-Fu Chang. I would like to thank them especially for giving me an opportunity to work in the ADVENT project and encouraging me to pursue my PhD.

I would like to thank the following people for their technical discussions and collaboration on a number of projects: Lai-Tee Cheok, Patrick Ngatchou, Aizaz Akthar, Li Tang, Steve Jacobs, and Javier Zamora.

Finally, I would like to express my gratitude to my family for their support and understanding.

Chapter 1

Introduction

1.1 Introduction

Multimedia systems and services are becoming more common with the advances and availability of computer and communications technology. The advances in computing and communications and the emergence of the Internet have spurred the growth of multimedia services into the main stream. Widespread interest in multimedia services came about with the success of Moving Pictures Experts Group (MPEG) technologies and the realization that digital media delivery allows value-added services along with high quality programming over existing cable and telephony infrastructure. This interest resulted in significant research on packet video in general and video-on-demand in particular [32][60][24][31]. The success of multimedia systems and services today can be owed to the standardization of the core technologies: MPEG-1, MPEG-2, most recently MPEG-4, and to certain extent the Digital Audio Visual Council (DAVIC) standards.

1.1.1 MPEG-1

The MPEG-1 standards were the first in the series of standards developed by the MPEG committee. The MPEG-1 standard was intended for video coding at 1.2 Mbps and stereo audio coding at around 250 kbps [5][6], together resulting in bitrates compatible with that of a double-speed CD-ROM player. The typical frame size for MPEG-1 video is 352x240 at 30 frames per second (fps) non-interlaced. Larger frame sizes of up to 4095x4095 are also allowed resulting in higher bitrate video streams. The multiplexing and synchronization of the MPEG-1 audio and video is done as specified in the Systems part of the MPEG-1 standard [7]. Since MPEG-1 was intended for digital storage media such as CD-ROM, the MPEG-1 Systems was not designed to be tolerant to bit errors. Furthermore, to keep the overhead small, the MPEG-1 Systems streams contain large variable length packets. To deliver MPEG-1 streams over the Internet, specialized mapping of MPEG-1 streams on to the payload of Real-time Transport Protocol (RTP) packets has been specified [39][66][109]. MPEG-1 was optimized for applications at about 1.5 Mbps on a reliable storage medium such as CD-ROM and as such not suitable for broadcast quality applications. To address the requirements of broadcast television and high quality applications, the MPEG committee began its work on MPEG-2 in 1992.

1.1.2 MPEG-2

The MPEG-2 standards include significant improvements over MPEG-1. Like its predecessor, MPEG-2 standards have several parts, nine in all [63]. The most important of these are video, audio, and systems. The significant improvements in MPEG-2 video over MPEG-1 video are: support for interlaced and progressive coding, 4:2:2 and 4:4:4 chrominance modes, higher frame sizes, scalability, and many additional prediction modes [10]. While MPEG-1 is optimized for storage-based applications, MPEG-2 is more generic and intended for a variety of applications. The MPEG-2 Systems specification now includes program streams, suitable for applications over a reliable medium, and transport streams with fixed length packets suitable for networked delivery. The part six of the MPEG-2 standard, Digital Storage

Media Command and Control (DSMCC), specifies protocols for session management and application signaling. Application signaling messages enable VCR-like control of MPEG streams as well as features such as file access, application discovery, and application download.

Because of its flexibility and support for a wide range of applications, MPEG-2 has been highly successful. MPEG-2 has been universally adopted for high-quality audio-visual applications including digital broadcast TV, DVD, HDTV, and digital cinema.

1.1.3 MPEG-4

Traditionally the visual component of multimedia presentations was mainly rectangular video, graphics, and text. Advances in image and video encoding and representation techniques [8][9][94] have made possible encoding and representation of audio-visual scenes with semantically meaningful *objects*. The traditionally rectangular video can now be coded and represented as a collection of arbitrarily shaped visual objects. The ability to create object-based scenes and presentations creates many possibilities for a new generation of applications and services. The MPEG-4 series of standards specify tools for such object-based audio-visual presentations [20][106]. While Audio and Video parts of the standard specify new and efficient algorithms for encoding media, the Systems part of MPEG-4 makes the standard radically different by specifies the tools for object-based representation of presentations. The most significant difference in MPEG-4 standards is the scene composition at the user terminal. The individual objects that make up a scene are transmitted as separate elementary streams and composed upon reception according to the composition information delivered along with the media objects. These new representations techniques will need novel ways for delivery, scheduling, and QoS management.

1.1.4 DAVIC

The Digital Audio Visual Council (DAVIC) was formed in 1994 to standardize technology and systems for end-to-end digital delivery services such as video-on-demand (VoD). Because of the large scope of the standard, DAVIC followed a toolbox approach to standardization [48][46][47]. Instead of standardizing a monolithic system, in the toolbox approach, the system is envisioned as a collection of sub-systems and components. The toolbox approach allows highly customized systems built only with a subset of tools in the standard. This model, in addition to allowing scalable systems enables the sub-systems to be used in systems where the system itself is not compliant to the standard. Following a one-functionality one-tool approach, DAVIC defined subsystems, adopted technology for the subsystems, and specified interfaces for the subsystems. The tools specified by DAVIC include all components of the end-to-end digital audio-visual systems including physical interfaces, access networks, transport protocols, information representation, security, and billing.

1.2 Components of a Multimedia System

A general multimedia (audio-visual) system can be segmented into four physical and/or logical components: the content, including representation and management, the server that delivers the content, the network that carries the content, and the user terminal that plays the content. In interactive systems, these components communicate and cooperate to playback the content. There are several international standards bodies specifying and standardizing the technology for end-to-end multimedia services. Figure 1.1 shows the components of an end-to-end multimedia system. The components can be physical and/or logical but have well-defined interfaces between them. The figure shows three such interfaces, with each of the components possibly containing more interfaces. Designing a system as a set of subsystems connected at published interfaces is necessary for the development of systems with such a broad scope. This also allows multiple vendors and service providers to provide components and services.

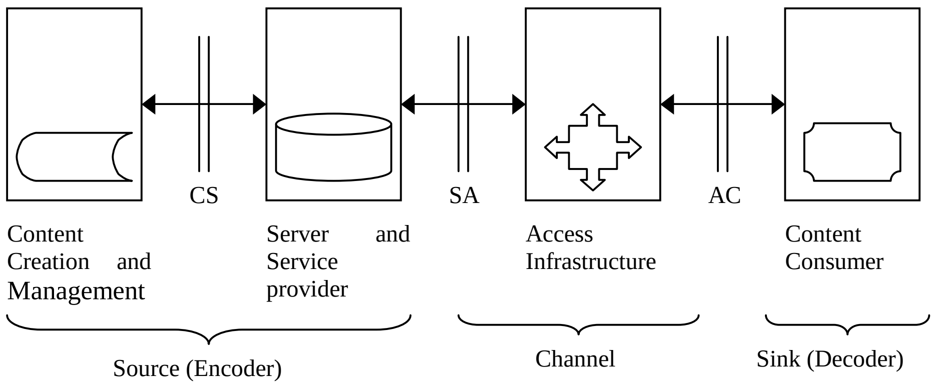

Figure 1.1 Components of a Generic Multimedia System

1.2.1 Content Creation and Management

The content creation process is a creative process akin to creating TV programming. As the amount of content on a system grows, it becomes difficult to manage the content in terms of searching, editing, compiling, and delivering. Content representation techniques should take these factors into consideration. With broadband access becoming common, service providers will be able to deliver high quality audio-visual content to PCs. The major consideration for content creators is the ability to deliver and playback content on multiple platforms (TVs and PCs). Content representation techniques should allow delivery of alternative representations of content based on terminal resources.

Object-based representation of content allows reuse of objects in creating new presentations. As the amount of content grows, the ability to search through the objects and locating the right one becomes difficult. The MPEG-4 Systems layer includes a meta data stream called object content information (OCI) stream, which can be used by content management systems for object location and retrieval. Content management should include tools that allow users to search for content based on the visual or textual description of objects [34][113]. Based on QuickTime, the MPEG-4 content interchange format [23] is a flexible format for representing multimedia presentations. The MPEG-4 file format maps the MPEG-4 Systems layer to the QuickTime architecture creating an efficient format that allows access to MPEG-4 features.

Creating object-based audio-visual presentations is not as straightforward as creating MPEG-2 content for TV broadcasting. The features of object-based presentations results in content with complexity that varies with the structure of the presentation. The content can be as simple as multiplexed audio and video (e.g., MPEG-2 audio and video multiplex) or as complex as a presentation composed of a large number of objects of different types with dynamically changing scenes and user interaction. The complexity of object-based presentations cannot be characterized just by bitrate. The complexity of a presentation depends on the number of objects in the presentation, the type of objects, the dynamics of object addition and deletion, and user interactivity. One important consideration during content creation is the suitability of content for delivery over networks with different capacities and to terminals with different computing, display, and storage resources. During the content creation process, authors should specify alternative representations for objects so that the servers can deliver appropriate objects based on the feedback from the terminals and the network. Determining the alternative representations for presentations dynamically is a difficult problem especially when considering both terminal and network resources.

1.2.2 Server and the Service Provider

The basic function of the servers is to manage sessions between the server and clients and to provide available services. The server is also responsible for publishing the content and services available for clients and to provide support for user interaction. The main consideration in server design is scalability. Columbia's VoD testbed [32] includes a distributed server that can deliver content to a range of client platforms and also scales well. The scalability of a server also depends on the type of content it delivers. With object-based services, servers have to manage multiple object delivery in a single presentation as opposed to delivering a single MPEG-2 transport stream in applications such as video on demand. The complexity of a server increases when interactivity is supported, especially for object-based services. When an object is added to a presentation as a result of user interaction, the server has to locate the object, retrieve it, and then deliver it to the client. The ability of servers to perform these functions efficiently depends on the object scheduling techniques and also the underlying content representation format. For simple cases that do not include interactivity, objects can be efficiently delivered using techniques such as direct streaming that pre-compute transport layer packet headers [23][21][22]. These techniques may not be as useful when objects are added asynchronously as a result of user interaction.

Another important design consideration is the server's ability to support content and service discovery. The application server in Columbia's video-on-demand server is designed as a collection of services that users discover as they browse the content and services available on the server. With object-based representation, and functionality provided by frameworks such as MPEG-4 Systems, the distinction between content and applications is blurring. The content can now include instructions on how to respond to user interaction. Servers will be able to support different applications just by supporting interactive object-based content.

1.2.3 Access Infrastructure

The access network between a server and a client carries the encoded content to the end-user. The configuration of the access network depends on the operating environment. The configuration of the access network varies with available end-user connectivity. Some common access networks include: end-to-end ATM, Internet, satellite, cable, DSL, LMDS, and MMDS. The main problem involving access networks is resource reservation and renegotiation. In networks such as the Internet that do not yet support resource reservations, techniques such as dynamic rate shaping can be used to reduce the object bitrate based on the state of the network [71]. Even in networks supporting QoS, techniques such as media filtering can be used in negotiating and re-negotiating QoS under resource constraints [27][26]. With object-based representation of audio-visual content, adapting content to meet network constraints can be done more efficiently, for example, by structuring presentations to contain objects with priorities proportional to their significance in the presentation.

Delivering object-based services may involve establishing and tearing down connections as objects in presentations are added and deleted. MPEG-4 specifies a delivery framework called DMIF that allows efficient object communication [4]. The DMIF framework also allows development of applications without regard to the underlying network. Not all networks are suitable to carry real-time multimedia traffic. The properties of data networks and their suitability to carry multimedia traffic are summarized in [56][57].

1.2.4 Content Consumer

The content consumer, usually a user terminal with an attached monitor, is the last component of the multimedia delivery chain. These end-user devices are connected to access networks and may have an upstream connection for server interaction and signaling. The digital TV infrastructure uses MPEG-2 transport streams for delivering multiplexed audio, video, and images. For digital TV, the terminals would include MPEG audio and video decoders to playback the content. With object-based audio-visual representation, the presentations can contain many different media types and it is impractical to have a terminal with hardware decoders for all the possible media types. Terminals supporting object-based presentations would have to include software decoders and even programmable processors for efficient decoding. The buffer models have to change to accommodate multiple objects in presentations. With object-based multimedia content and the support for user interaction, the user terminals are becoming more and more complex. Innovative architectures and representation techniques are needed to enable sophisticated applications. The design and architecture of a terminal for object-based audio-visual presentation is presented in [52]. The proposed architecture is a low complexity design and uses a simple format to hierarchically represent multimedia presentations. The MPEG-4 standards define a set of profiles and levels to allow simpler decoders that handle only a sub-set of the media types.

1.3 Contributions of the Thesis

The contributions of this thesis are in the area of audio-visual communications and fall under two categories: 1) Traditional audio-visual systems and 2) object-based audio-visual systems. Traditional here refers to systems using digital audio and video (rectangular) as opposed to object-based audio-visual systems. This classification also highlights the flexibility, and at the same time the complexity, as a result of the object based representation of content. The contributions to the traditional audio-visual systems area are the design and architecture of the VoD application server, the distributed video pump architecture for traffic localization, and contributions to DAVIC standardization and interoperability. Our contributions in object-based audio-visual services area are: 1) contributions to the MPEG-4 Systems standards in the form of terminal and bitstream architecture and server interaction model, 2) scheduling algorithms for object-based presentations, and 3) scheduling algorithms for interactive audio-visual presentations.

1.3.1 Video-On-Demand Systems

Our contributions to the traditional audio-visual services area are in the design and development of Columbia’s VoD testbed. We designed an application server that allows users to browse, select, and playback content available on a server. The application server is based on DSMCC user-user communications model [13] and provides the user interaction support as a set of services. The services are classified based on the functionality supported: directory service allows browsing a server directory, file service provides access to files on the server, and stream service is for controlling media playback. We designed the stream service with an interface to use the services of a video-pump. We specified the CORBA (Common Object Request Broker Architecture) [2] interface on the video pump to be able to launch the video pump service on remote systems, thus making the distributed video-pump possible. Our distributed video-pump design is an effective tool for traffic localization on networks. By replicating the media streams, the video-pump closest to the client can be launched thereby limiting the traffic to that segment of the network. For web-browser-based clients and clients without CORBA support, we designed the system with CORBA services running on the server and with a DSMCC interpreter between the client and the server. This allows the same server to be used for a variety of clients.

The distributed VoD system architecture and design for localization are discussed in detail in Chapter 2.

1.3.2 Object-Based Audio-Visual Services

The main distinguishing feature of object-based audio-visual services is the ability to represent scenes and presentations as a collection of objects composed at the user terminal. Composition at the terminal implies ability to access and interact with individual objects. These features of object-based presentations blur the distinction between applications and content. This paradigm shift in content representation and playback enables the next generation of audio-visual applications and services. New techniques are necessary to deliver these services. Supporting interactive presentations puts additional burden on the system. We investigate object-based audio-visual services considering the complexity, flexibility, authoring, and delivery issues. We investigate the problem of scheduling object-based presentations under network and buffer constraints and present algorithms for multiplex-scheduling object-based presentations.

1.3.2.1 Scheduling Object-based Presentations

Scheduling is a complex problem and has been studied in different application domains including operations management, transportation, flight scheduling, and even video scheduling in VoD systems. The nature of object-based content makes it difficult to deliver compared to MPEG-2 delivery. The complexity of an MPEG-4 content 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. We show that 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 and structure of the presentations.

Our investigation of the scheduling problem in the context of object-based audio-visual presentations addresses the folowing issues: 1) is a presentation schedulable with given resources, 2) what are the additional resources required to schedule a presentation, 3) what is the minimum channel (buffer) capacity required to deliver the presentation, 4) does the presentation remain schedulable when an object is added, and 5) will dropping an object make the presentation schedulable. We investigate the problem and propose a family of algorithms to solve these problems. We formulate the object-scheduling problem by considering individual access units (frames) of audio-visual objects, with precedence constraints on access units within an object. We found similarities with problems in job-shop scheduling. We show that the object-scheduling problem is NP-complete in strong sense. Even though the problem is NP-complete, we use efficient heuristics used in job-shop scheduling to find buffer-optimal schedules.

We first introduce the “FullSched” algorithm to determine the schedulability of a presentation with given resources. This algorithm can be re-purposed to compute the minimum resources required to make the presentation schedulable. We then introduce the “GapSched” algorithm that computes schedules incrementally. We prove the buffer-optimality of both of these algorithms. The GapSched algorithm is especially useful during content creation to quickly determine whether adding an object violates any of the resource constraints. We introduce the concept of *residual data volume* to compute a lower bound on the capacity required to deliver the presentation. Starting with this lower bound on the channel capacity, we use the MinC algorithm to compute the minimum channel capacity required to deliver the presentation. For un-schedulable presentations, if additional resources cannot be acquired, the only alternative is to drop objects or object instances to make the presentation schedulable. Because of the object-based representation of the presentations, dropping an object may compromise the integrity of the content. Like in the case of MPEG-2, dropping frames is still possible for continuous media objects in a presentation. We provide guidelines to structure presentations in ways that makes it easier for systems to deliver alternative representations to meet resource constraints. Determining the alternative representations for presentations is a difficult problem to be considered for future extensions.

1.3.3 Scheduling Interactive Presentations

The nature of object-based audio-visual systems makes it possible to create application with dynamic addition and deletion of objects. This addition and deletion of objects is effected when a user interacts with the presentation. We investigate the characteristics of interactive audio-visual presentations and discuss the issues in delivering interactive presentations. We present algorithms for adding and releasing resources upon object addition and deletion. The interactivity considered here is the user interaction that affects the resources consumed by the server and the network. We present algorithms to determine the schedulability of interactive presentations and the minimum capacity required to deliver an interactive presentation. By computing the resources required for the interactive and core components of a presentation separately we could determine whether a presentation can be fully supported. This allows content creators to design the presentations with relatively independent core and interactive components so that the substance of the presentation is not affected even without the interactive component.

When user interaction is allowed, the resulting asynchronous events affect object delivery and add to the burstiness of the traffic depending on the content. When delivering interactive component presentations, there are two options: 1) reserving the required capacity at session startup 2) acquiring necessary bandwidth after an event happens. The scheduling choice depends on the bandwidth availability, content design, and the applications' tolerance to the events ignored by the server. If bandwidth is available on-demand, the most efficient way is to acquire necessary bandwidth when an event happens. When bandwidth availability cannot be predicted, the best way is to reserve required bandwidth at session startup. Even though the reserved capacity is wasted when the anticipated events do not happen, reserving resources assures the delivery of the interactive component of the presentation. An acceptable compromise would be to prioritize events and reserve bandwidth for higher priority events and request additional bandwidth when a lower priority event happens.

We introduce three new concepts: *event specification*, *event window*, and *event extents* to describe object-based presentations as a set of events and analyze these events to determine the resource requirements for interactive presentations. Resources for interactive components can be allocated by resource reservation or by resource negotiation after an event happens. If resources have to be acquired for every object added as a result of an event, the object addition cannot be guaranteed and depends on the probability of acquiring the required resources. We also propose guidelines to create interactive presentations for efficient use of network resources.

Our contributions in the area of traditional audio-visual services have also been published in [57][56][32][86][80][82][88][87]. The contributions in the area of object-based audio-visual services also appear in: [19][23][21][22][52][53][73][74] [77][78][79][85][84][75][76][83][81][89].

Chapter 2

VoD Testbed: Interaction, Interoperability, and Standardization

2.1 Introduction

Columbia's video on demand testbed has served as a platform for research on audio-visual communications including video servers [105][104], video transmission [120][119], Internet video delivery [71], and standardization and interoperability [54][87][82]. The heterogeneous nature of the testbed gives an opportunity to work with a system consisting of a range of clients, servers, and networks. In this chapter we describe our contributions toward the development of Columbia's VoD testbed.

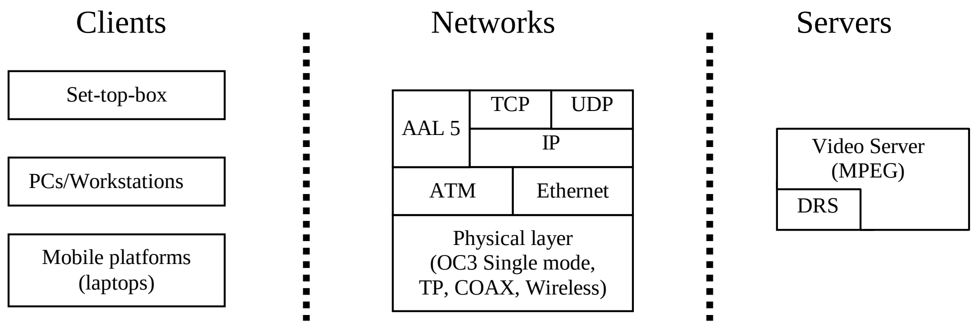

Figure 2.1 Components of the VoD Testbed

The VoD testbed is a client server system with clients and servers connected over a heterogeneous network. The testbed has various types of client platforms including digital set-top-boxes, PCs/Workstations with hardware and software decoders, and mobile computers with wireless connectivity. The network consists of IP and native ATM networks. The wireless network runs IP and Ethernet protocols. The server runs on an SGI Onyx running a general-purpose operating system (IRIX) and delivers

MPEG-2 video and audio multiplexed into MPEG-2 transport streams. The server also includes a dynamic rate-shaping module that scales down the video bitrate in real time. This module is used to reduce the video bitrate when delivering to terminals with fewer resources such as mobile clients. Figure 2.1 shows the components of the VoD testbed in the client, network, and the server domain. The server can deliver streams to the clients on one of the networks the client is attached to. The wide area connectivity of the testbed was used to study the effects of video delivery over wide area networks.

In the rest of this chapter we will discuss our contributions made towards the development of Columbia's VoD testbed. In Section 2.2 we present the system operation and discuss the design and imlementation of the application server. In Section 2.3 we present the distributed server functionality and user interaction support. Finally in Section 2.4 we briefly describe the DAVIC 1.0 efforts in standardizing VoD like services and discuss the design considerations for interoperability.

2.2 System Operation and Application Services

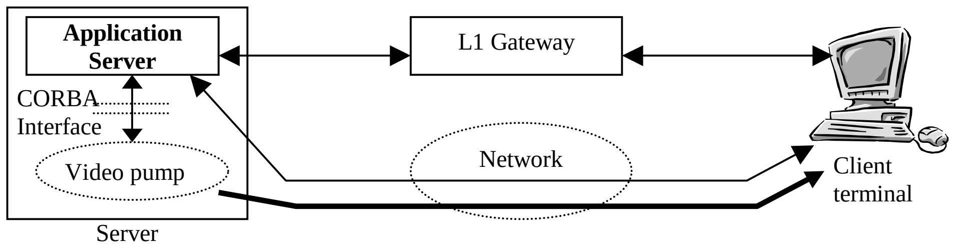

Figure 2.2 High-level System Diagram

Figure 2.2 shows the high-level diagram of the VoD system. It is divided into two domains, the Level 1 and Level 2 domains. The Level 2 domain contains the client (set-top-box) and the server equipment while the rest falls in the Level 1 domain. The Level 1 domain consists of access networks, head-ends, resource managers, and network managers. The Level 1 domain provides a set of gateway services to the components in the Level 2 domain. This Level 1 gateway functions are bundled into a service/system called the Level 1 gateway. The Level 1 gateway can be seen as a directory service that allows the clients to discover servers. This view of the VoD systems follows from the telephony systems and the earlier work on video dial tone. Within a server, there might be one or more services available for clients. The functions that enable the service discovery by the end-user are the Level 2 gateway functions. In Figure 2.2, Level 2 gateway functionality is part of the application server.

The application server provides a set of services to the clients. The services provided in Columbia's testbed are based on DSM-CC user primitives [12]. DSM-CC is a set of network and user primitives specified in part 6 of the MPEG-2 standard. The network primitives allow session setup and management while the user primitives are used for application signaling between a client and a server once a session is established. The user primitives are typically carried over a signaling channel. The application server supports the core DSM-CC services: service gateway, file service, directory service, and stream service. The service gateway is the root service a client first attaches to. Once connected to the service gateway, a client can select file service, to upload or download files (e.g., HTML or text files) from the server. In our VoD testbed, we used the file service to provide news reports in text form to clients. Directory service allows a client to navigate a server and browse through the available content. The stream service is central to VoD applications and delivers videos to clients. The interfaces to these services are specified in Interface Definition Language (IDL) and we used CORBA [2] as a natural choice to implement these interfaces.

The system operation consists of the following steps:

1. A client first connects to its default Level 1 gateway. The Level 1 gateway is the first service clients connect to before session establishment. The Level 1 gateway can reside anywhere on the network and provides a list of servers available. Once the client selects a server, the L1 gateway establishes the initial session and drops out. The client is now connected to the service gateway on the server that provides Level 2 gateway services. The service gateway authenticates the clients' access to the service and provides a client with the root directory service that contain directory, file, or stream services. The services are mapped directly onto a UNIX file system. Directory service is a UNIX directory, file service is available for all the files, and stream service is available for media streams.

2. From the root directory, clients can navigate the server using the directory service discovering the additional directory, file, and stream services. If a directory is selected, the server returns a reference to the directory object. The client can then use the `Directory::List` operation to view the directory contents. When an item from the list is selected (using the `Open` operation), the server returns a reference to the selected object. The directory `List` and `Open` operations are sufficient to navigate the server and discover additional content.

3. If a file is selected, the server returns the selected file to the client. The actual file transfer can be done either on the signaling channel or for large files, a new data connection can be established. In the VoD testbed, the file service was used to provide a sample news service. The news headline is mapped to a file name; if the client selects the file, the server returns a reference to the file object to the client. The client then has the option to either view or save the file. Since the file service is a generic file transfer application, this can be used to download new applications from a server.

4. The server identifies the stream service by examining the file extension. Files with the extension *.inf* contain the meta data about the stream service. The meta data includes alternative video pumps for the stream and enough information to setup a downstream connection and reserve resources for stream delivery. Just like in the case of file and directory service, the server returns a reference to the stream object when a client selects the stream service. The stream object supports an interface to enable VCR-like controls. The process involved in stream delivery is described in the next section.

2.3 Stream Service and the Distributed Video Pump

Figure 2.3 shows the functional diagram of the VoD system. A session starts when the client selects the one of the servers listed by the Level 1 gateway and a bi-directional channel is established between the service gateway and the client. All the clients accessing the same server connect to the same application server and the service gateway. When a client chooses the stream service, an MPEG-2 transport stream is opened for delivery. The application server does not see the actual media data. The application server has access to the meta-data for the media streams and whenever it sees these meta-data files it presents them to the clients as stream objects. The meta-data includes bitrate, title, video type, audio type, primary source URL, alternate sources, preferred video pump, production credits, and even reviews. A client with a reference to the stream object has to invoke the `Open` and `Play` methods to start stream playback. The stream service implementation has a video pump factory that creates video pumps to stream the requested video. The video pump itself has an IDL interface and communicates with the stream service using the IDL interface implemented using CORBA. When a client chooses to pause a stream, it invokes the `Pause` operation on the stream service that, in turn, passes the message along to the pump to pause the stream delivery. The clients do not communicate directly with the video pump for stream control except for receiving the streams. All stream control messages reach the pump through the stream service.

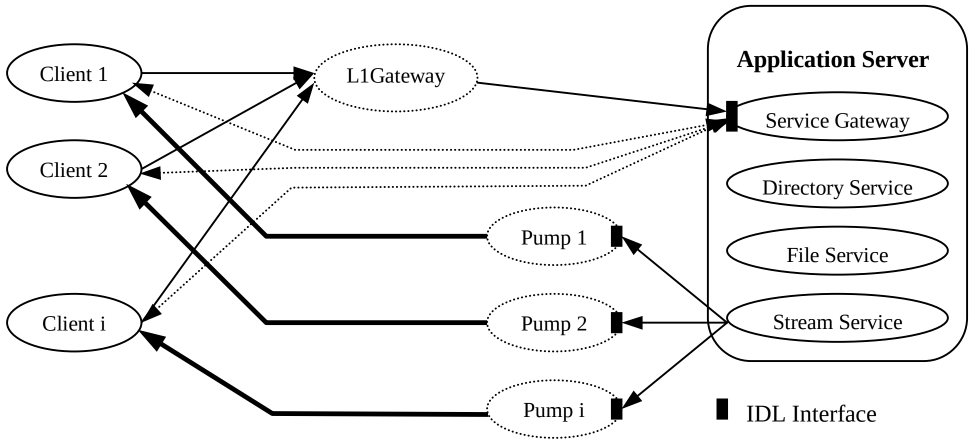

Figure 2.3 Stream Service in the VoD System

The IDL interface supported on the video pump makes the server a distributed VoD server. The meta-data on the application server has the address of the video pump.

The address is nothing but the IP address or machine name the stream service uses to launch the video pump. The video pump factory in the stream service launches a video pump for every stream request it receives. A video pump can be launched anywhere on the network irrespective of the location of the application server. By replicating the content on several locations on the network, network traffic can be localized to certain network segments by launching a video pump closest to the client. This feature can also be used for load balancing on the machines running the video pumps.

2.3.1 Stream Service for Web Clients

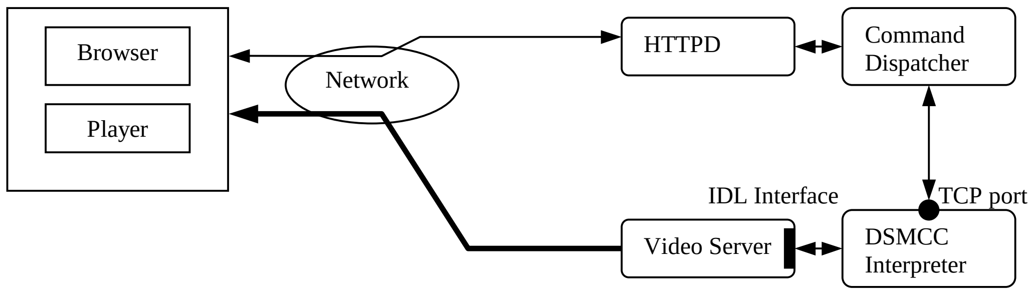

Figure 2.4 Application Signaling in Browser-based Clients

Since a web browser is the most common application/interface people are accustomed to, it is a logical choice for clients running on PCs and workstations. Browser based clients are also easy to deploy on a wider scale. Since web browsers typically do not support CORBA, we developed a *command dispatcher* and a *DSM-CC interpreter* on the server side that acts as an intermediary between a Web server (HTTPD) and the video server. The DSMCC interpreter can be viewed as a client proxy capable of interpreting DSMCC messages and formatting the output with HTML for the Web client. Each client request goes through the HTTP server, which launches the command dispatcher script to communicate with the DSMCC interpreter. The

DSMCC interpreter processes the message by making the appropriate CORBA calls to the application server and passing the results back to the command dispatcher. There is a one-to-one mapping between service interfaces and client command codes. The commands called by the clients include the port number on which the DSMCC interpreter is listening. This browser-based interface is also used with devices that act as passive decoders and do not have an easy way of communicating with the server.

2.4 Standardization and Interoperability

With interest in video-on-demand services increasing and with several planned VoD trials, the companies with vested interests in the success of VoD formed a consortium called the Digital Audio Visual Council (DAVIC) in 1994 to develop standards and specifications for the systems to provide broadband multimedia services. DAVIC defines its purpose as: “to advance the success of emerging digital audio-visual applications and services, initially of the broadcast and interactive type, by the timely availability of internationally-agreed specifications of open interfaces and protocols that maximize interoperability across countries and applications or services”[48]. The DAVIC concept of *Digital Audio-Visual Applications and Services* includes all applications and services in which there is a significant digital audio video component. DAVIC started out mainly in response to the industry interest in video on-demand in the early 90s and apparent lack of standards to ensure interoperability among the various systems and implementations. By the time DAVIC published its first specification in 1995, video on-demand was no longer seen as the *killer application* it was originally thought. However, the technologies and standards DAVIC was developing are the core to support any form of audio-visual services.

Figure 2.5 shows the DAVIC subsystems and the reference points between the subsystems. DAVIC specifies *everything* necessary to support the core audio-visual services it defines; left to right, content provider to the content consumer, and top to bottom, physical layer of the transport to the application layer. Content providers produce content, which is distributed to the consumers by service providers. The delivery system supports the delivery of content to service providers and consumers. DAVIC specifies and/or develops technologies to build these subsystems and defines reference points and interfaces between (and also within) the subsystems. The conformance points A1 – A11 are the reference points and any DAVIC compliant system should be conformant at these points.

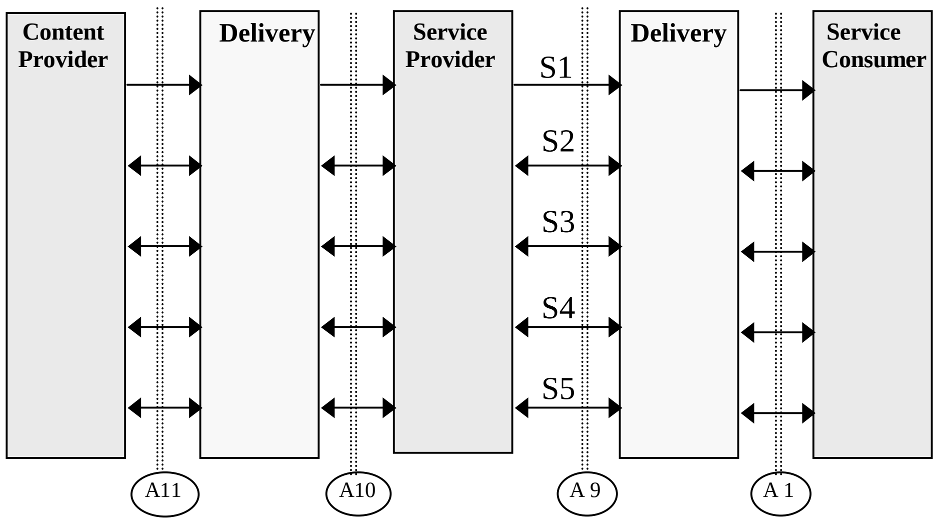

Figure 2.5 Components of a DAVIC System

2.4.1 Information Flows and Reference Points

The subsystems were specified in an evolutionary manner with each new set of DAVIC specifications adding functionality to the previous set. To facilitate this evolutionary approach, reference points, interfaces, and information flows were introduced between the subsystems. The information flow between the subsystems is divided into five logical flows based on the nature of the information. Figure 2.5 shows the information flows and reference points between the subsystems. Only reference points A1, A9, A10, and A11 are shown in the figure. There are also reference points that are internal to the subsystems.

The S1 information flow corresponds to the principal service information, e.g., an MPEG-2 transport stream carrying audio and video. The S2 flow corresponds to the application control information, S3 corresponds to session control, S4 to connection management, and S5 corresponds to billing and other management functions. The S1 flow is unidirectional from the service provider to the service consumer (STU) while the other information flows are bi-directional. Future versions of the specification are expected to include a bi-directional S1 flow supporting applications such as video conferencing. The five logical information flows may use a single physical channel or more than one physical channel.

2.4.2 Interoperability

There are many organizations developing standards for multimedia *tools* including international authorities such as ISO [1] and ITU-T [69]. Multimedia tools refer to components of systems required to enable multimedia presentations and services; MPEG-2 video, MPEG-1 audio, and ATM are some examples of tools for video, audio and networking. Most of the current standards (tools) are developed, for the most part, independently by the standards body concerned. It is rather difficult to ensure that the tools developed independently can be combined to facilitate a single multimedia presentation. DAVIC plays the role of an integrator by providing specifications that bind the various standards enabling multimedia applications and services.

A DAVIC tool is a technology that supports a single DAVIC functionality. For example, ATM is a tool to transport audio-visual data in the core network. Following a one-functionality one-tool policy, DAVIC selects the best technology available to support its functionalities. DAVIC also develops tools to support the functionality that cannot be achieved by the existing technology. Because of the diverse nature of the standards it is dealing with, interoperability of the systems and subsystems was taken up as a part of the standardization process.

To achieve its goal of global deployment of DAVIC systems and services, DAVIC places special emphasis on interoperability. Since the DAVIC specification is made up of a number of independently developed standards, it is important to ensure that these standards work when used together in DAVIC components and systems. The work related to verifying the specification and its interoperability was undertaken by the interoperability sub-group. The charter of the interoperability sub-group was to verify that the specification does not have any ‘gaps’ and to provide guidelines for conformance testing. Towards this end, DAVIC is promoting a series of public interoperability events. We organized the first multi-platform global interoperability event at Columbia University in New York in June 1996 [87] [82].

Eight organizations from around the world participated in this event, cross-connecting their servers and clients on Columbia’s testbed [32]. Even though the implementations were not fully DAVIC compliant, the interoperability experiments gave valuable feedback to DAVIC to clarify and improve the specification [103]. The heterogeneous nature of Columbia's VoD testbed made it an ideal choice for conducting multi-platform interoperability tests. Since the testbed uses open standards for networks and server interfaces, it was relatively easy for connecting the systems from participating organizations. Furthermore, the wide area connectivity of the VoD testbed was essential to test the system functionality over wide area networks. The detailed results of the interoperability experiments can be found in [82]. The feedback from the subsequent interoperability events at the Tokyo Electronics Show [90] and Telecom Interactive were also taken into account in improving the specifications.

2.5 Concluding Remarks

In this chapter we presented the contributions made toward the development of Columbia's VoD testbed. The key contributions made are the design of the application server for VoD services, distributed video pump, application signaling, adapting the system for browser-based clients, and contributions toward the development of DAVIC standards by means of proof of concept implementations and interoperability experiments.

Separating the resource-intensive video delivery using a well-defined interface and its implementation using CORBA is essential for scalable servers and localizing bandwidth-intensive video traffic. With streaming media over the Internet becoming more common, traffic localization together with congestion control mechanisms is necessary to prevent congestion collapse in networks. The ability of a system to support popular platforms, particularly the Internet, is critical for the large-scale deployment of any system. Modular design allowed us to reuse most of the components in adapting the system for browser-based clients. The user interaction support was limited to VCR-like controls. In Chapter 3 we contrast this against the generic user interaction supported in object-based audio-visual systems.

Chapter 3

Object-Based Audio-Visual Services

3.1 Introduction

Image and video encoding has been totally transformed with the advent of new coding and representation techniques [21][8][116]. Over the past several years, image and video compression research has explored new methodologies for high compression using high-level techniques [21][108]. These techniques depart from traditional waveform coding, and attempt to capture more of the high-level structure of visual content. This high-level structure has been referred to as *objects*, and includes components of imagery that are directly associated with visual structures of semantic significance, both simple and complex (e.g., a ball, a table, a man). This next generation of coding techniques has made possible encoding and representation of audio-visual scenes with semantically meaningful objects. This new paradigm of object-based representation of audio-visual scenes/presentations will change the way audio-visual applications are created.

Like any new technology, object-based representation of audio-visual scenes will give rise to many technological challenges while providing feature-rich framework for object-based audio-visual presentations. To appreciate the advantages of object-based systems, we often compare and contrast object-based systems with non-object-based digital audio-visual systems that are widely used today (e.g., digital TV and

HDTV). We refer to such systems as *frame-based systems* to reflect that fact that a user terminal sees the presentation as a sequence of composed and encoded frames.

In this chapter we present some of the issues in the design and deployment of object-based audio-visual services. First we present an overview of object-based presentations and contributions made to the development of the MPEG-4 Systems specification. We then present an overview of the MPEG-4 Systems layer to contrast our proposals to the adopted solutions.

3.2 Components of Object-Based Presentations

3.2.1 The Notion of an Object

An object-based presentation consists of *objects* that are composed to create a scene; a sequence of scenes forms a presentation. There is no clear-cut definition of what constitutes an *object*. When used in the sense of audio-visual (AV) objects, an object can be defined as *something* that has semantic and structural significance in an AV presentation. An object can thus be broadly defined as a building block of an audio-visual scene. When composing a scene of a city, buildings can be the objects in the scene. In a scene that shows the interior of a building, the furniture and other items in the building are the objects. The granularity of objects in a scene depends on the application and context. The main advantage of breaking up a scene into objects is the coding efficiency gained by applying appropriate compression techniques to different objects in a scene. In addition to coding gains, there are several other benefits of object-based representation: modularity, reuse of content, ease of manipulation, object annotation, as well as the possibility of interaction with the objects in a scene.

To appreciate the efficiency of object-based presentations, consider a home shopping channel such as the ones currently available on TV. The information on the screen consists mostly of text, images of products, audio, and video (mostly quarter-screen and sometimes full screen). All this information is encoded using MPEG-2 video/audio at 30 fps. However, if this content 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, video, and text and image updates that take up significantly less bandwidth. In addition to this, the ability to interact with individual objects makes applications such as e-commerce possible.

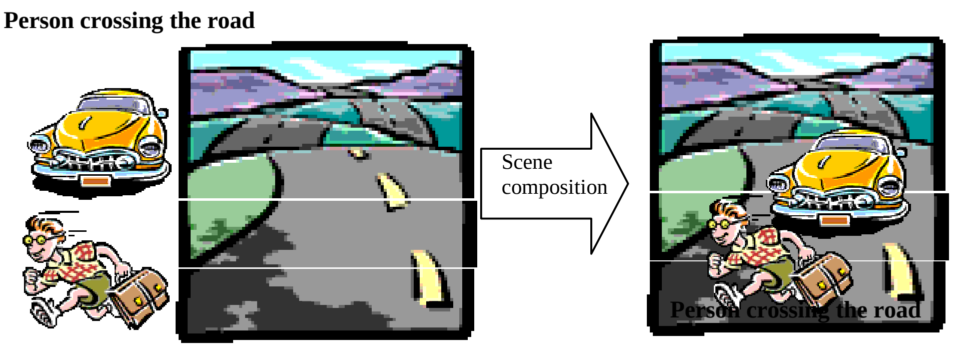

Figure 3.1 Example of an object-based scene

The key characteristic of the object-based approach to audio-visual presentations is the composition of scenes from individual objects at the receiving terminal, rather than during content creation in the production studio (e.g., MPEG-2 video). This allows prioritizing objects and delivering individual objects with the QoS required for that object. Multiplexing tools such as FlexMux [8] allow multiplexing of objects with similar QoS requirements in the same FlexMux stream. Furthermore, static objects such as a scene background are transmitted only once and result in significant bandwidth savings. The ability to dynamically add and remove objects from scenes at the individual user terminal even in broadcast systems makes a new breed of applications and services possible. Frame-based systems do not have this level of sophistication and sometimes use makeshift methods such as image mapping to simulate simple interactive behavior. This paradigm shift while creating new possibilities for applications and services makes content creation and delivery complex. The end user terminals that process object-based presentations are now more complex but also more powerful.

Figure 3.1 shows a scene with four visual objects, a person, a car, the background, and the text. In object-based representation, each of these visual objects are encoded separately in a compression scheme that gives best quality for that object. The final scene as seen on a user terminal would show a person running across a road and the text at the bottom of the scene, just like in a frame-based system. To compose a scene, object-based systems must also include the composition data for the objects that a terminal uses for spatio-temporal placement of objects in a scene. The scene may also have audio objects associated with (or independent of) visual objects. The compressed objects are delivered to a terminal along with the composition information. Since scenes are composed at the user end of the system, users may be given control on which objects are played. If a scene has two audio tracks (in different languages) associated with it, users can choose the track they want to hear. Whether the system continues to deliver the two audio tracks even though only one track is played is system dependent; broadcast systems may deliver all the available tracks while remote interactive systems with upstream channels may deliver objects as and when required. Since even text is treated and delivered as a visual object, it requires far less bandwidth than transmitting the encoded image of the rendered text. However the delivered text object now has to include font information and the user terminals have to know how to render fonts. User terminals could be designed to download fonts or decoders necessary to render the objects received.

3.2.2 Scene Composition

Scene composition can simply be defined as spatio-temporal placement of objects in the scene. Spatial composition determines the position of objects in a scene while temporal composition determines its position over time. Operations such as object animation, addition, and removal can be accomplished by dynamically updating the composition parameters of objects in a scene. All the composition data that is provided to a terminal can itself be treated as a separate object.

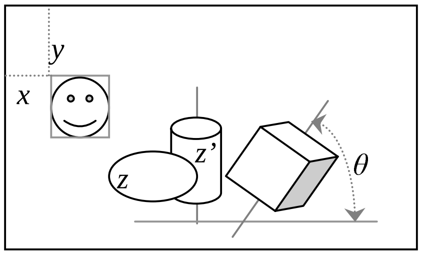

Figure 3.2 Composition Parameters

Since the composition is the most critical part of object-based scenes, the composition data stream has very strict timing constraints and is usually not loss tolerant. Any lost or even delayed composition information could distort the content of a presentation. Treating the composition data as a separate data object allows the system to deliver it over a reliable channel. Figure 3.2 shows the parameters for spatial composition of objects in a scene. The gray lines are not part of the scene; x and y are horizontal and vertical displacements from the top left corner. The cube is rotated by an angle of θ radians. The relative depth of the ellipse and a cylinder are also shown. The ellipse is closer to the viewer ($z < z'$) and hence is displayed on top of the cylinder in the final rendered scene. Even audio objects may have spatial composition associated with them. An object can be animated by continuously updating the necessary composition parameters.

3.3 Terminal and Bitstream Design for Object-Based Audio-Visual Presentations

An object-based presentation is delivered to a terminal with objects and object composition packed in a bitstream. The bitstream should be structured to convey the data and meta data in a way that preserves the semantic separation of objects in the presentation. The bitstream design presented in this section was originally proposed for the MPEG-4 System layer [52]. The bitstream is designed on the basis of separating the object meta-data from the object data and the hierarchical representation of scenes. Composition is the property of a scene and hence should be part of the scene description. In addition to the decoding times, composition times are needed as predictive encoding schemes such as MPEG video have different decoding and composition/display order. Since objects can enter and leave a scene arbitrarily, to increase object reuse, especially for static objects such as images, we use a lifetime time stamp indicating the availability of an object at a terminal. Objects are re-transmitted only if a presentation needs to reuse the object after the objects’ lifetime.

3.3.1 Bitstream Architecture

The structure (syntax and semantics) of the components that make up a bitstream for object-based audio-visual presentations is described in this section. The bitstream consists of two types of data packets, object composition packets (OCPs) and object data packets (ODPs) reflecting the separation of object data and meta-data. The OCPs carry the data necessary to compose the objects in scenes. These include the spatial information for object placement as well as control information that is used to add and remove the objects. Each of these packets has an ID, known as the object ID, that identifies the object the data belongs to.

3.3.1.1 Encoding Scene Composition

The OCPs contain composition parameters necessary for proper placement and orientation of an object in a scene. The OCPs also contain a decoding time stamp, a presentation timestamp, and a timestamp that gives the lifetime of an object at the terminal. Composition update packets (CUP) are used to update the composition parameters of objects. Together with object composition, updates can be used to animate objects or groups of objects in a scene. Compound Composition Packet (CCP) is a container packet that groups the OCPs. Object grouping is necessary in order to be able to apply operations to a set of objects with a single command. This can be used to minimize the amount of composition information sent, as well as to support hierarchical scene composition based on independent sub-scenes. Each sub-scene can be represented as a compound object. A CCP can also contain other CCPs. This structure supports the use of the CCPs to represent presentations hierarchically with scenes and sub scenes. A CCP contains an object id followed by the lifetime

time stamp (LTS), component object count, and composition parameters for the compound object. The compound object is available at the terminal for the duration given by the LTS. Figure 3.3 shows the hierarchical representation of scenes using compound composition packets. The compound object shown in the figure has three component objects; one simple object (ID = 2) and two compound objects (ID = 3 and 4). Each of these compound objects contains two simple objects as shown in the

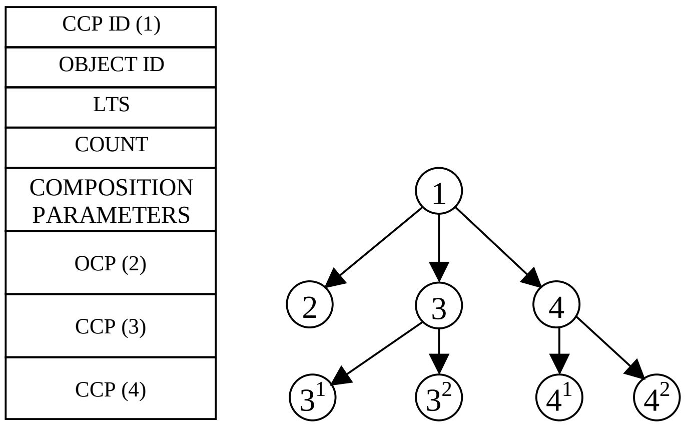

Figure 3.3 Hierarchical Representation of Scenes

Figure 3.3. The object at the top level acts as a grouping node by binding its component objects together. By changing the composition parameters of the top-level object, the composition data for the group can be changed. Representing scenes hierarchically allows efficient navigation and also addressing of scenes and sub-scenes. The object data itself is carried in object data packets (ODP). The ODPs include the timestamps necessary for inter-media synchronization.

3.3.2 Audio-visual Terminal Architecture

An audio-visual (AV) terminal is an end user component used to present (display) audio-visual content. An object-oriented terminal receives information in the form of individual objects, which are placed together to form a scene according to specified composition information. Objects and composition information are transmitted in separate logical channels (LC). When object and composition data are multiplexed, assigning a unique identification number (ID) to the packets creates logical channels in a single multiplexed stream. Such multiplexing can be used to multiplex objects with the same QoS requirements. The H.245 specification [112] specifies a mechanism for the creation and management of such logical channels.

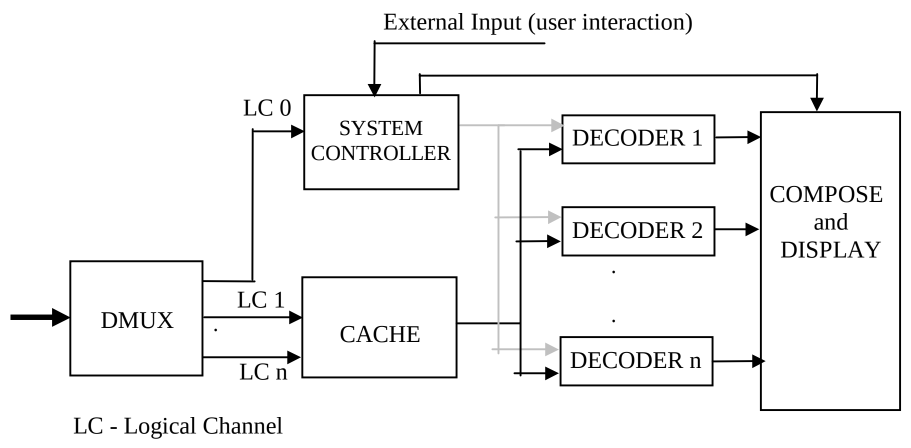

Figure 3.4 Architecture of an Audio-Visual Terminal

Figure 3.4 shows the architecture of an AV terminal. In the AV terminal architecture presented, the AV objects and their composition information are transmitted over a network (or accessed from a local storage device) in separate logical channels. The dmux reads the multiplexed composition and data packets and de-multiplexes it into logical channels. The LC 0 always carries composition information, which is passed on to the System Controller (SC). Composition information consists of object composition information and composition control information. Similarly, the object data is encapsulated in object data packets. The AV objects received on other logical channels are stored in the cache to be acted upon by the decoders. The SC decodes the composition information to compose the object in the scene.

3.3.2.1 Terminal Operation

An object-based bitstream is accessed either from a network interface or a local storage device. The bitstream consists of a series of object composition packets and object data packets multiplexed into logical channels. The DMUX de-multiplexes the packets and passes the composition information to the controller. The dmux also places object data in the cache for use by the decoders. The decoders in the terminal are media specific, i.e., if the object is encoded using MPEG-2, the decoder will be an MPEG-2 decoder or if an object is compressed using JPEG compression, a JPEG decoder will be used. When an object data packet arrives at the terminal, the system controller examines its decoding time stamp and instructs the decoder to decode the object at the decoding time. The decoded object is copied to the presentation buffers at presentation time with appropriate position and orientation as indicated by the composition parameters. An object, if indicated as persistent, remains in the cache until a time given by its expiration time stamp. If such an object is used at a later time (before the expiration time), only the corresponding composition data is sent to the terminal.

3.3.2.2 System Controller

The system controller is the heart of the terminal and controls the decoding and playback of the objects on the AV terminal. The SC first initializes the dmux to read form a local storage device or a network port. This is initiated at startup time either from user interaction or by opening a session at default network address. The dmux reads the input bitstream and feeds the composition data on LC0 to the controller. The composition data begins with the description of the first scene in the AV presentation. This scene can be described as a hierarchical collection of objects using compound composition packets or as a collection of independent object composition packets. A table that associates the elementary streams with the nodes in the scene description immediately follows the scene description. The controller maintains the object IDs (stream IDs) in an object list and also a render list. The render list contains the list of objects that are to be rendered on the display device. An object that is disabled by user interaction is removed from the render list. A node delete command that is sent via a composition control packet causes the deletion of the corresponding object IDs from the object list. The node hierarchy is also maintained and updated whenever a composition update is received.

The SC also maintains timing for each AV object to signal the decoders and decoder buffers of decoding and presentation time. The timing information for the AV objects is specified in terms of its time-base. The terminal uses the system clock to convert an object’s time base into system time. The controller gets the timestamps from the data packets for each object. At the decoding time for an access unit, the data is forwarded to the appropriate decoder, for example, by signaling the decoder to read data from the input buffers.

The terminal design presented is a simple design that illustrates the concepts for a generic audio-visual terminal. Since an object-based terminal should support several media types, it is very likely that such terminals use software decoding for most of the media types. Terminals for generic object-based audio-visual presentations should be based on programmable decoders based on FPGAs [36][117][58] to achieve real-time performance. This gives a terminal the ability to download decoders for media types discovered in the bitstream. This practice is already seen in software media players but is yet to be seen in dedicated media devices such as set-top-boxes.

3.4 Supporting User Interaction in Object-Based Presentations

Interactivity in the case of video on demand systems mainly meant supporting VCR functionality in the system. With object-based video, interactivity is not just stream control but depends on the application and the media types involved. Object based representation of audio-visual objects and scenes lends itself to the design of more sophisticated user interaction with scenes and objects in the scenes. The user interaction mechanisms developed for video-on-demand are not sufficient to support interactivity in object-based audio-visual applications as they were primarily designed to support stream control. In this section we present an architecture to support user interaction in MPEG-4 Systems[1]. This architecture, called the *CommandDescriptor* architecture, integrates well with the MPEG-4 scene description framework and supports fully interactive applications.

[1] A modified version of the CommandDescriptor framework has since been adopted by MPEG-4 Systems for supporting server interaction

3.4.1 MPEG-4 User Interaction Model

Since the MPEG-4 scene description is based on the VRML specification, interactivity supported by MPEG-4 Systems (Version 1) was limited by the capabilities of the VRML event model. VRML's event model is limited to the scene and lacks the mechanism to dynamically control the object delivery from servers or communicate with servers in any way.

As an application's response to user interaction is application dependent, the interaction framework should be generic with ability to support a wide range of applications. Clicking on an object might cause the object to move in one application and hide the object in another application. The ability to interact with a server is necessary in many applications. In an object-based system like MPEG-4, interactivity is more than stream control and is very much application dependent. Since defining an exhaustive list of user interaction messages and system's responses is impossible, a generalized model that supports complete application dependent interactivity is necessary. We designed and implemented the Command Descriptor framework [74]-[79] with all the features to support server interaction in MPEG-4 systems.

3.4.2 Command Descriptor Framework

The Command Descriptor framework provides a means to associate *commands* with media nodes in a scene graph. When a user interacts with the media nodes, the associated commands are processed. The actual mode of interaction is specified by the content creator and may be a mouse click or mouse-over or some other form of interaction. The command descriptor framework consists of three elements, a

CommandDescriptor, a CommandROUTE and a Command. Command descriptor, as the name implies, describes a user interaction command. Command descriptor are attached to media nodes via command ROUTEs. When a user interacts with a node, the associated command descriptors are processed. The processing simply consists of passing the command back to the server over a user command channel, or it may also involve modifying the command parameters before sending the command to the server. More than one command descriptor may be attached to a node and in that case they are all processed when a user interacts with that node.

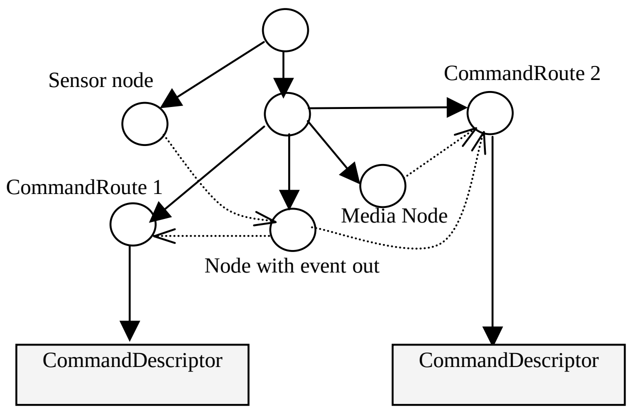

Figure 3.5. Command Descriptors and Command ROUTEs

Figure 3.5 shows an example with command descriptors attached to the media nodes. Two nodes may share a command descriptor if the interaction with the two objects results in identical behavior. Command descriptors are transmitted to the client in a separate elementary stream called *command descriptor stream*. The command routes are similar to the ROUTEs used in the scene description but point to a command descriptor. The command routes are included in the scene graph description and are transmitted in the scene description stream. This separation between commands and command association allows us to modify command syntax without editing a scene graph.

The command descriptors contain a command that determines the interaction behavior of the associated node. The command descriptors can be updated using command descriptor updates allowing a node to support different interaction behavior at different times. The commands can be transmitted to the server using the DAI user command primitives supported by DMIF.

```
class CommandDescriptor: bit(8) commandDescriptorTag = 0x06 {

        bit(16) CommandDescriptorID;

        bit(16) CommandID;

        // stream count; number of ES_IDs
        // associated with this message
        unsigned int (8) count;

        // ES_Id (channel numbers) of the streams
        // affected by the command
        unsigned int (16) ES_ID[count];

        // application-defined parameters
        do {
                unsigned int (8) paramLength;
                char (8) commandParam [paramLength];
        }
        while (paramLength!=0);
}
```

Figure 3.6. Command Descriptor Syntax

Even though the command descriptor framework is generic, and supports application specific user interaction, we need a few standard commands to support consistent application behavior across applications and servers especially for common commands such as stream control. We specify a set of common commands, e.g., stream control commands [67], and the content creators can specify application specific behavior. A server should be aware of these application specific commands in order to process them.

Figure 3.6 shows the syntax of a CommandDescriptor. The command descriptor ID uniquely identifies a command descriptor in a presentation. The command ID indicates the semantics of the command. This simple structure along with command routes in the scene description can be used to support complete application specific interactivity. For example this can be used to support content selection by specifying the presentation in command parameters and the command ID indicates that the command is the content selection command. One way to implement this is to create an initial scene with several images and text that describes a presentation associated with that image. We then associate a content selection descriptor with each image and the corresponding text. When a user clicks on an image, the client transmits the command containing the selected presentation and server starts a new presentation. Command descriptors can be used to support application specific user interaction including complex applications such as electronic commerce and on-line shopping.

3.4.2.1 Advantages of the CommandDescriptor framework

Two other alternatives to command descriptors are using URLs in the nodes (MPEG-4 Systems defines a set of nodes to build scene graphs) to embed commands and creating a node that contains a command (equivalent of embedding a CommandDescriptor in a node). In the first case, a command is coded as a URL field of a node. One advantage of this approach is that it uses the existing node syntax and update mechanisms. The disadvantage of these two approaches is that it requires a server to process BIFS in real-time. BIFS has a complex structure and creating BIFS commands in real-time is more complex when compared with creating

CommandDescriptors that are byte oriented and byte aligned. Furthermore, command descriptors have a very simple structure that is easy to compose and parse.

For MPEG-4 content with server interaction support to run on all servers, the servers should understand all the commands. It is impossible to create interactive content with content/application specific interactivity that can be understood by all servers. The CommandDescriptor framework allows servers to compose the commands (e.g., from an equivalent textual description) in a manner understood by them. A CommandDescriptor provides a well-defined structure for saving data locally in applications that require data persistence (e.g., cookie management). To save the state of a scene when a user terminates a session one can implement and save a command descriptor. URLs and data fields in nodes are not sufficient for efficient implementation of such functionality. In applications with multi user interaction, it may be necessary to exchange the state of a scene among the users. CommandDescriptors are more convenient to exchange data among users. The number of commands a server composes could be quite high in multi-user applications and burden of processing bit-oriented node updates is significant when compared with processing byte-oriented and byte aligned CommandDescriptors.

3.5 Storage Format for Object-Based Audio-Visual Presentations

The ubiquity of the Internet and the continuous increase in the computing power of the desktop computers together with the availability of relatively inexpensive multimedia codecs have made multimedia content generation a readily available functionality in the desktop computers. Creating interactive presentations using object-based presentation techniques requires users to manipulate, compose, edit and transmit multimedia content over packet networks. It is thus necessary to have a multimedia file format that allow fast access to data files, locally or remotely stored, during the process of content creation, composition/manipulation (e.g., file editing and playback), and streaming over packet network.

In this section we will discuss the requirements for storing object-based audio-visual presentations. We will then present our work on multimedia file format in the context of MPEG-4 standardization [23]-[22][53][85]-[76].

3.5.1 Requirements for New Generation Multimedia Formats

The requirements of a file format in terms of access, manipulation, aggregation, editing and streaming vary depending on the media type. It is thus impossible to optimize processing of one type of media without penalizing the other. The file format should not be designed to facilitate a special task (such as, in-place editing, local or remote playback, or to match a specific media) but rather allow a series of effective and flexible tools that allow efficient indexing and data access to perform the aforementioned tasks. In the following sections we will discuss some of the most stringent requirements imposed on multimedia file formats. Our work on the MPEG-4 file format was carried out jointly with AT&T Labs Research.

3.5.1.1 Granularity of the Data Access

Multimedia file formats should be flexible enough to support easy editing and playback of multimedia objects and should provide means to perform editing operations such as copy, cut, and paste to aid users in creating new content easily. Object based presentations usually have separate meta-data and media data. In case of MPEG-4, the meta-data consists of scene description, object description, and content description streams. This implies inter-dependencies among the streams. Any changes to the presentation may result in changes to all the dependent streams. Access to individual elements of these streams is necessary to perform editing operations. This implies support for a proper granularity to access the data of a given media stream.

3.5.1.2 Security

Providing security and protecting intellectual property becomes critical in digital audio-visual systems. Techniques such as digital watermarks exist to ensure authenticity of individual media streams. Such techniques should be extended to MPEG-4 files to ensure authenticity of the presentation. Authenticating a presentation is especially significant in object-based presentation as operations such as adding/removing/replacing objects may affect the content of a presentation and such operations can be accomplished relatively easily with object-based presentations.

3.5.1.3 Playback

The heterogeneity of the multimedia presentation environments requires that the file format offer several ways to access the file data, trading off local terminal resources such as memory and CPU speed for data access speed as appropriate. With universal multimedia access seen as the future direction for multimedia, it becomes important for the next generation of multimedia file formats to enable content re-purposing and delivery to a range of multimedia devices. Playback needs a data access structure that allows access to the segments of different objects that should be played back at a given instance in time. Furthermore the playback may need to be selective, i.e., playback of only some objects or base layers of scalable objects. It is thus important that a file format supports mechanisms to access individual objects and parts of objects as necessary. Playback of multiple views of a presentation should also be supported.

3.5.1.4 Streaming and Media Transport

Early multimedia file formats were designed and tailored for local access at data rates attainable by relatively slow machines. Furthermore they were developed in a PC-centric environment, in contrast to the current trend in distributed networked computing. In today's world, multimedia file formats must be designed for efficient remote random access playback. This capability is often called *streaming*. While no universally agreed-upon definition for the term streaming currently exists, it is generally accepted that the word includes the concept of a remote client rendering multimedia content as it is being received over the network using only a bounded delay buffer. Depending on the type of hardware and software used for streaming, audio-visual presentations can be directly delivered from MPEG-4 files or in case of optimized servers they have to be converted to a server specific format. A file format can therefore be not optimized for streaming but for interchange.

A common format that spans capture, authoring and editing, download and streaming, leads to great flexibility. Material may be reworked after use, or used in multiple ways, without being copied or re-formatted. A desirable solution is to have a set of tools that can be used to create a file optimized for the needs of the applications while allowing easy (computationally inexpensive) conversion between different instances (views) of the file.

3.5.2 IIF – A Format For MPEG-4 Media

The complexity of MPEG-4, owing to the functionality it supports, makes it very demanding in terms of storage and access of multimedia objects that constitute an MPEG-4 presentation. MPEG-4 identified the need for a file format that can cater to the needs of MPEG-4 and possibly the multimedia industry. This resulted in a call for proposals (CFP) [15]. In this section we describe our proposal for the MPEG-4 file format.

The Integrated Intermedia Format is a solution that is designed specifically for MPEG-4 [21][22]. We designed the file format based on segment-based organization of media data, with each segment containing objects of a single scene. An IIF file consists of a header (File Configuration Header (FCH) and File Configuration Extension (FCE)) followed by access tables, and finally one or more segments. Figure 3.7 shows the components of an IIF file in a configuration with access tables in the front. These tables may also be attached at the end by setting appropriate flags in the header, typically in case of recording when access tables are not available at the beginning of a presentation.

3.5.2.1 Indexing Tools

Random access to AUs (frames) of an object is supported by means of indexing tools. Indexing of objects and access units in that object (frames or access units) can be done globally, with the indexing information located in a contiguous space or in a distributed fashion, with indexing information distributed over the media data. A distributed indexing scheme results in lower memory utilization as only a part of the access tables are loaded at a time. Indexing is supported by means of several object access tables that vary in complexity and support distributed or global indexing. The different access tables used in supporting random access are:

Physical Object Table (POT): gives a list of objects present in a file and has pointer to the segment that contains the first access unit of the object. Requires a SOT for accessing an access unit.

Extended Physical Object Table (EPOT): indexes all access units of interest and points to the SOT entry that corresponds to the access unit.

Fat Physical Object Table (FPOT): Expanded version of EPOT. This table indexes

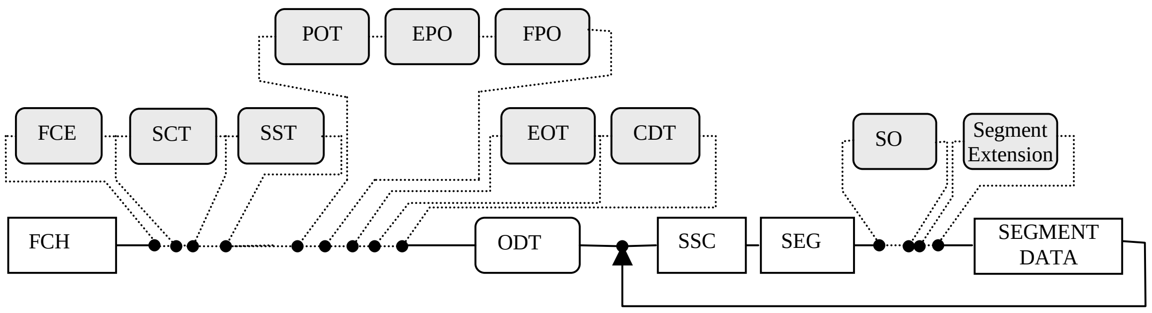

Figure 3.7. Components of an IIF File

all access units and includes their offsets and sizes. This table is sufficient to achieve random access.

Segment Object Table (SOT): The media data, organized into segments, has a table that indexes all the access units in a segment.

Object Descriptor Table (ODT): This table provides direct access to object descriptors. Object descriptor contains all the essential information for a decoder to process the object and is the first piece of information conveyed to a client during session establishment.

Content Descriptor Table (CDT): An object's Object Content Information (OCI) can be directly accessed using this table.

These access tables can be used in different combinations depending on application needs. For example, there might be a need to index one object in an MPEG-4 presentation using FAT while the others are indexed using other tables. This allows flexibility for content creators in indexing the contents depending on the applications.

A BIFS (scene description) stream is the most critical part of an MPEG-4 presentation and special handling might be necessary to communicate it to the user terminal. A BIFS stream is identified in an IIF file by assigning a unique two-byte ID for the BIFS stream (this can also be done by decoding the object descriptors and examining the stream types). This BIFS ID is part of the file header and allows identifying and extracting the BIFS data easily.

One disadvantage of these indexing schemes is the lack of direct time-based indexing. For example it is not possible to access an access units *n* seconds into the presentation without further processing the bitrate and other parameters of the object. This has been identified but can be easily overcome by time-wrapping of the access tables; i.e., associating segments of index tables with presentation time of access units.

3.5.2.2 Segment Based Organization of Media Data

IIF organizes the media data into segments. Segments usually correspond to a scene or some other higher-level construct. Access units in a segment are optionally indexed in a SOT. A segment starts with a unique segment start code that can be used to uniquely identify the beginning of a segment. The segment header has flags that determine the type of the contents in a segment. The segment data could be access units that belong to a single object, multiple objects, object descriptors only, OCI only, or scene description only. This information is useful to prioritize the processing of data in a segment, as some data, for example scene description, are more critical to a presentation than the other.

Another aspect of this segment-based approach is the separation of access tables and actual media data itself. The media data contained in the segments is pure data and can be extracted easily for direct playback. This saves de-packetization time that is necessary if additional information is packed with the access units.

Since all access units are indexed relative to the beginning of a segment, the contents of a segment can be edited with in a segment with changes made to only a single entry in the access table that points to the segments. Another benefit of this approach is the ability to allow un-indexed areas in segments that are treated as free space. This might be a result of editing operation or could also be by design when a content creator decides to leave some free space in segments for later use.

3.5.2.3 Streaming Support

Video playback on the desktop is becoming more and more common today. New generation of file formats are being designed with emphasis on the ability to stream data to desktops. By considering this requirement during the design phase, media can be stored in a way that facilitates streaming and even make it more efficient.

To stream data, a media streamer needs to have access to data units (access units), transport properties (bitrate, max unit size, min unit size, etc) of the objects, and then packetize the access units according to the transport protocol of choice before delivering it over a network. As the number of streams to be streamed increases, the computational power required performing these seemingly insignificant tasks becomes a burden to the streamer reducing it’s capacity. By making the task of access to data units easy, streaming performance is improved. In IIF, an object’s properties such as its average bitrate, peak bitrate, start time, end time, and duration are made available in the stream configuration table (SCT). The overall nature of the MPEG-4 presentation such as average bandwidth, peak bandwidth, and average segment are indicated in file configuration extension (FCE). Both FCE and SCT are optional as indicated in the file header.

To make it even more efficient, IIF supports direct streaming. Direct streaming as the name implies is less work for the streamer. The idea here is to pre-compute the protocol specific packet headers and include them along with the access units. A streamer would then extract the access units, associated pre-computed packet headers and conveys it to the network. This reduces the load and increases streamer efficiency. However, since direct streaming is transport protocol dependent, protocol specific data should be included for each protocol the streamer supports. In IIF, this data is placed in a segment in the optional segment extension. Segment extension contains timestamps and protocol specific information. A drawback in this design is that there was no support for more than one protocol in the same file.

3.5.2.4 External Links

IIF, as mentioned earlier, was designed specifically for MPEG-4. The External Object Table is used to indicate the presence of External objects and/or External links in an MPEG-4 file. External objects refer to AV objects that are referred to in the current file but are present in a different file, which may be located on the current file system or a remote (networked) system. This feature is necessary to support features like local logo or ad insertion in a presentation.

External objects facilitate the use of a set of files to store an MPEG-4 presentation. An EOT shall be present if multiple files are used to store a single presentation or if there are any URLs present in the scene description or elementary stream descriptors. The EOT also lists External links. External links are the URLs used in a presentation that might be activated as a result of user interaction. These are necessary to ensure that the links are available during a presentation and if they are not, the client can be warned prior to the beginning of a session. This is a useful check as some missing links might interrupt the flow of a presentation. It is the responsibility of the server (or player in case of local playback) to ensure that resources are available to access External objects and/or External links during a presentation.

The IIF file format presented is an efficient design specifically created for MPEG-4 Systems. The standardized MPEG-4 file format that was finally adopted is based on QuickTime, one of the competing proposals [41]. The clear separation between media data and metadata in the QuickTime format was an important reason for the choice as well as the existing user base. Even though QuickTime is a good design and is well suited for multimedia presentations, the efforts to customize the QuickTime architecture to MPEG-4 Systems resulted in significant overhead in the MPEG-4 file format.

3.6 A Comparison with MPEG-4 Systems

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 [93][106]. The MPEG-4 Systems specification provides the glue that binds the audio-visual objects in a presentation [8][20]. Some of the concepts of the object-based bitstream and terminal presented in the previous section can also be seen in the MPEG-4 Systems layer. The basis for the MPEG-4 Systems architecture is the separation of the media and data streams from the stream descriptions. The scene description stream, also referred to as BIFS (Binary Format for Scenes), describes a scene in terms of its composition and evolution over time and includes the scene composition and scene update information. The other data stream that is part of the MPEG-4 systems is the object description or OD stream, which describes the properties of data and media streams in a presentation. The description contains a sequence of object descriptors, which encapsulate the stream properties such as scalability, QoS required to deliver the stream and the decoders and buffers required to process the stream. The object descriptor framework is an extensible framework that allows separation of an object and the object's properties. This separation allows for providing different 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 at the system level. The separation of media data and meta data also makes it possible to use different media data (MPEG-1 or H.263 video) without modifying the scene description.

An elementary stream is composed of a sequence of access units (e.g., frames in an MPEG-2 video stream) and is carried across the Systems layer as sync-layer (SL) packetized access units. The sync-layer is configurable and the configuration for a specific elementary stream is specified in its elementary stream (ES) descriptor. The ES descriptor for an elementary stream can be found in the object descriptor for that stream which is carried separately in the OD stream. The sync layer contains the information necessary for inter-media synchronization. The sync-layer configuration indicates the mechanism used to synchronize the objects in a presentation by indicating the use of timestamps or implicit media specific timing. Unlike MPEG-2, MPEG-4 Systems does not specify a single clock 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. In addition to the scene description and object description streams, an MPEG-4 session can contain Intellectual Property Management and Protection (IPMP) streams to protect media streams, or Object Content Information (OCI) streams that describe the contents of the presentation, and a clock reference stream [20][64][110]. All the data flows between a client and a server are SL-packetized.

The data communicated to the client from a server includes at least one scene description stream. The scene description stream, 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. VRML was intended for 3D modeling and is a static representation (a new object cannot be dynamically added to the model). MPEG-4 Systems extended the VRML specification with additional 2D nodes, a binary representation, dynamic updates to scenes, and new nodes for server interaction and flex-timing [91]. A scene is represented as a graph with media objects associated with the leaf nodes. The elementary streams carrying media data are bound to these leaf nodes by means of BIFS URLs. The URLs can either point to object descriptors in the object descriptor stream or media data directly at the specified URL. The intermediate nodes in the scene graph correspond to functions such as transformations, grouping, sensors, and interpolators.

The VRML event model adopted by MPEG-4 systems has a mechanism called *ROUTEs* that propagates events in a scene. This event model allows nodes such as sensors and interpolators to be connected to audio-visual nodes to create effects such as animation. This mechanism however is limited to a scene; there are no routes from a server to a client to propagate user events to a server. Our work on server interaction described in Section 3.4 specifies an architecture to propagate user events to a server [67][79]. This has been adapted to fit tightly in a scene graph by encapsulating the server command functionality in a new node called Command Node [19].

In addition to VRML functionality, MPEG-4 includes features to perform server interaction, polling terminal capability, binary encoding of scenes, animation, and dynamic scene updates. MPEG-4 is also specifying a Java interface to access a scene graph from an applet. These features make possible content with a range of functionality blurring the line between applications and content.

Figure 3.8 shows the binding of elementary streams in MPEG-4 Systems. The Figure shows a scene graph with a group node (G), a transform node (T), an image node (I), an audio node (A), and a video node (V). Elementary streams are shown in the Figure with a circle enclosing the components of the stream. The scene description forms a separate elementary stream. The media nodes in a scene description are associated with a media object by means of object IDs (OD ID). The object descriptors of the objects in the scene are carried in an object descriptor stream. An object descriptor is associated with one or more elementary streams. The elementary streams are packetized and carried in separate channels.

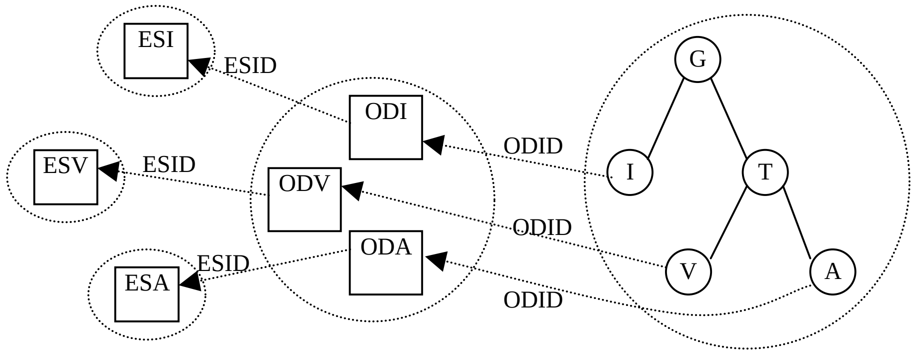

Figure 3.8 Stream Association in MPEG-4 Systems

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, 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 called Delivery Multimedia Integration Framework (DMIF) [12].

3.7 Delivering Object-Based Presentations

Delivering MPEG-4 presentations differs from traditional video on demand delivery in the characteristics of the presentations delivered. VoD applications primarily involve delivering MPEG-2 transport streams. 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 a 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.

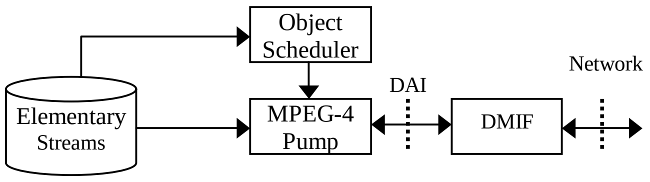

Figure 3.9 Components of an MPEG-4 Server

Figure 3.9 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 access units arrive at the terminal before their decoding time. The pump retrieves the objects from the disk and delivers them as scheduled by the presentation scheduler. The scheduler uses the decoding timestamps to schedule the delivery of access units.

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. The structure and nature of an MPEG-4 presentation determines the complexity of the content. When user interaction is allowed, the resulting asynchronous events affect object delivery and add to the burstiness of the traffic on the network and the complexity on the content.

When delivering interactive components of presentations, there are two options: 1) reserving the required capacity at session startup 2) acquiring necessary bandwidth after an event happens. The scheduling choice depends on the bandwidth availability, content design, and the applications' tolerance to the events ignored by the server. If bandwidth is available on-demand, the most efficient way is to acquire the necessary bandwidth when an event happens. When bandwidth availability cannot be predicted, the best way is to reserve the required bandwidth at session startup. Even though the reserved capacity is wasted when the anticipated events do not happen, reserving resources assures the delivery of the interactive component of the presentation. An acceptable compromise would be to prioritize events and reserve the bandwidth for higher priority events and request additional bandwidth when a lower priority event happens.

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 bitrate of a presentation may be highly variable depending on the objects used in the presentation. Presentations 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.

3.8 Concluding Remarks

In this chapter we presented an overview of object-based audio-visual services. The key contributions in this area are the bitstream design, terminal architecture, user interaction framework, and file format for object-based presentations. The bitstream is design is based on the premise of separating meta-data from the media data and hierarchical representation of object-based presentations. The bitstream is also designed to dynamically update the scenes using composition updates, node addition, and node deletion. The terminal architecture is an illustrative example of a terminal for audio-visual presentations. It introduces the concept of persistent objects and reuse of persistent objects from local cache. We proposed the original architecture for user interaction and file format in MPEG-4 Systems. Even though the final form of these components in the MPEG-4 standard differ from the proposed versions, the contributions formed the underlying basis. We then presented an overview of MPEG-4 Systems layer to highlight the similarities to the bitstream and terminal architecture presented earlier in the chapter and the contributions made in the development of the MPEG-4 Systems standard. Finally, we briefly considered delivery of object-based presentations with respect to content representation and scheduling. In the next two chapters, we will consider the delivery-related problem of scheduling interactive object-based audio-visual presentations.

Chapter 4

Scheduling Object-based Audio-Visual Presentations

4.1 Introduction

The MPEG-4 Systems specification [20][64] [8][110] defines an architecture and tools to create audio-visual scenes from individual objects. The scene description and synchronization tools are at the core of the systems specification. The MPEG-4 architecture allows creation of complex presentations with wide-ranging applications. As the complexity of the content increases, so does the complexity of the servers and user-terminals involved. The servers now have to manage multiple streams (objects) to deliver a single presentation.

The flexibility of MPEG-4 enables complex interactive presentations but makes the content creation process non-trivial. Unlike MPEG-2, the content creation process involves much more than multiplexing the media streams. Determining the schedulability of a presentation is also important during the 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 schedule a presentation with a given set of 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.

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 delivery and add to the burstiness of the traffic depending on the content. When delivering interactive components of presentations, there are two options: 1) reserving the required capacity at session startup, and 2) acquiring necessary bandwidth after an event happens. The scheduling choice depends on the bandwidth availability, content design, and the applications' tolerance to the events ignored by the server. If bandwidth is available on-demand, the most efficient way is to acquire the necessary bandwidth when an event happens. When bandwidth availability cannot be predicted, the best way is to reserve the required bandwidth at session startup. Even though the reserved capacity is wasted when the anticipated events do not happen, reserving resources assures the delivery of the interactive component of the presentation. An acceptable compromise would be to prioritize events and reserve bandwidth for higher priority events and request additional bandwidth when a lower priority event happens.

In this chapter we discuss the problem of scheduling audio-visual objects and present algorithms for optimal scheduling of audio-visual objects. We present new algorithms, based on job sequencing on a single machine proposed by Carlier [29], for scheduling objects in a presentation. This chapter paper is organized as follows: the general problem of scheduling audio-visual objects and related earlier work is presented in Section 4.2. Complexity of object-based audio-visual presentations is discussed in Section 4.3. The characteristics of startup delay and terminal buffer are discussed in Section 4.4. In Section 4.5 we present several algorithms to schedule audio-visual presentations. We conclude the chapter in Section 4.6.

4.2 Scheduling Audio-Visual Objects

Scheduling and multiplexing of audio-visual (AV) objects in a presentation is a complex problem. Scheduling of audio-visual objects has been the subject of study in [25][100][111]. In [100] Little and Ghafoor present synchronization of multi-object presentations using Petri-net models to describe timing relations in multimedia presentations. They present network-level and application-level synchronization protocols for multi-object presentations. The problem considered is delivering objects from multiple sources to a single destination. The problem we are considering is the network-independent scheduling of interactive audio-visual objects on the server side. We assume the use of underlying network services for establishing connections for data transport. We also show that scheduling objects jointly results in bandwidth savings. In the Firefly system [25], the issue addressed was scheduling a set of local objects to ensure synchronization by adjusting the duration of the media objects involved. The authors address the synchronization problem by adjusting the play-rate of objects (speeding up or slowing down playback) but do not consider network delivery issues. In [111] Song *et. al*, describe the JINSEL system that uses bandwidth profiles to reserve bandwidth for media objects on a delivery path. The JINSEL system computes the bandwidth required on the network segments on the delivery path using the amount of buffer available on the switch/component. Disk scheduling for structured presentations was studied in [55].

In the following, the problem is explained in the context of MPEG-4 Systems. MPEG-4 Systems specifies an architecture to describe scenes and communicate audio-visual data that corresponds to the objects in a scene [20]. A scene consists of one or more audio-visual objects with each of these objects associated with an elementary stream that carries the corresponding data. All the elementary streams are typically multiplexed in a transport multiplex. A server that is transmitting objects (elementary streams) should make sure that an *access unit* (access unit is the smallest data entity to which timing information can be attributed; e.g., frames in an elementary stream) arrives at the terminal before its decoding time. The constraints on the server transmission are the channel capacity and buffer capacity at the receiving terminal. This problem has similarities with VBR scheduling [104], where the goal is to maximize the number of streams supported by a server. The difference is that in VBR scheduling discussed in [104] and references therein, the assumption is that the video data being handled is periodic (e.g., 30 fps). In a general architecture such as MPEG-4, such assumption is not valid as the data can consist of only still images and associated audio. Furthermore, the multiple streams in MPEG-4 presentations are synchronized at the same end-user terminal using a single clock or possibly multiple clocks whereas there are no inter-dependencies when scheduling multiple VBR video streams. This puts tighter restrictions on the scheduling of an AV presentation. In such cases the decoding times of individual access units have to be considered for efficient scheduling. Furthermore, the delay tolerances and relative priorities of objects in an audio-visual presentation can be used to schedule objects for delivery. To make a presentation schedulable, objects of lower priority could be dropped. Even different instances of an object may be assigned different priorities (e.g, higher priority for I and P frames and a lower priority for B frames in an MPEG video stream). These characteristics of the audio-visual services can be used to efficiently schedule a presentation with minimal resource consumption.

4.2.1 System Model and Assumptions

We discuss the scheduling of audio-visual objects in the context of a system consisting of client (end-user), server, and network components as shown in Figure 4.1.a. Figure 4.1.b shows the server model. The server delivers objects in a presentation as scheduled by the scheduler. The scheduler uses the decoding timestamps to schedule the delivery of access units. A decoder is assumed at the far end that decodes the objects for real-time playback. On the client side, data is retrieved from the network and provided to the decoders at decoding time of that access unit. Any data that arrives before its decoding time is buffered at the terminal.

The terminal buffer model is not considered to keep the schedule independent of terminal designs. However we need the minimum buffer size for a class of terminals to compute object schedules. The data delivered from the server is transported on the channel established between the client and the server. The following assumptions are made about the content, decoders, network, and the server.

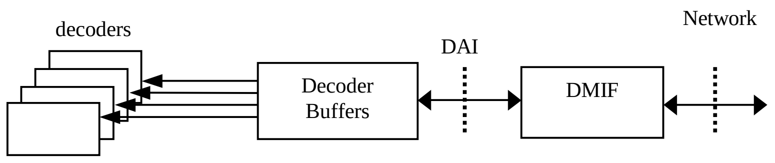

Figure 4.1.a. Terminal Model

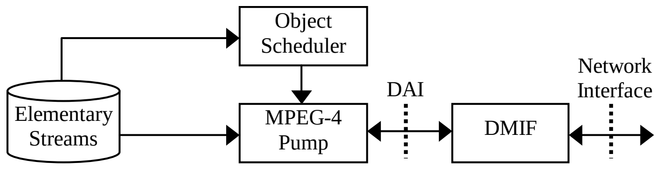

Figure 4.1.b. Server Model

Content:

- An audio-visual presentation is composed of one or more objects (AV Objects).
- An access unit (AU) is the smallest piece of data that can be associated with a decoding time.
- An audio-visual object contains one or more access units.
- Objects and their access units may be assigned relative priorities.

Terminal/Decoders:

- The decoders have given, limited memory for receiving and decoder buffers.
- The object data is removed instantaneously from the buffer at the decoding time given by the object's decoding timestamp.

- An object/instance that is received before the decoding time is buffered in the decoder-input buffers until its decoding time.
- More than one object instance may be present in the decoder-input buffers.

Channel/Network:

- End-to-end delays from the server to the player (including the transmission delay) are assumed to be constant.
- The capacity required for the signaling channel is assumed to be negligibly small.
- The transport layer is work conserving, and delivers the packets to the network instantaneously.

Server:

- Audio-visual objects are available at the server in the form of time-stamped access units.
- All the access units of an object are delivered in their decoding order.
- A server presents an access unit to the transport layer at the send time determined by the scheduler.

4.2.2 Notation

The following notation is used in this discussion.

$\mathscr{N}$ — set of objects to be scheduled

N — number of objects to be scheduled

n_i — number of access units per object ($\leq i \leq N$)

$A_j(k)$ — access unit k of object j

$A \equiv \bigcup_{j,k} A_j(k)$ — set of all access units in the presentation

$T_j^d(k)$ — decoding time of $A_j(k)$.

$T_j^s(k)$ — send time of $A_j(k)$.

$\sigma \equiv \bigcup_{j,k} T_j^s(k)$ — the send-time schedule

C — a transmission channel of capacity C.

$s_j(k)$ — size in bytes of $A_j(k)$

$d_j(k)$ — duration (channel occupancy) of access unit k on the wire; $d_j(k) = C \div sizeof(A_j(k))$;

$B_{\max}$ — terminal buffer capacity assuming a single demultiplexing buffer.

$B(t)$ — buffer occupancy at time t.

T_s — startup delay.

$T_s^{\max}$ — max startup delay.

$T_s^{\min} = \sum_k d_k(0) \quad \ni T_k(0) = 0$ — time to transmit AUs of all objects with DTS/CTS of zero.

4.2.3 Problem Formulation

Given a set of *N* objects that comprise an audio-visual presentation, with each object containing n_i access units each with a decoding time $T_j^d(k)$, of k^{th} access unit of object *j*, a transmission channel of capacity *C*, terminal buffer of size *B*, allowed startup delay of $T_s^{\max}$, and duration (channel occupancy) of each access unit on the channel, $d_j(k)$: is there a schedule σ that satisfies the following constraints?

$$T_j^s(k) \le T_j^d(k) - d_j(k) \tag{1}$$

$$T_j^s(k+1) \ge T_j^s(k) + d_j(k) \tag{2}$$

$$\text{if } i \in A - \{j\}, \text{then} \quad \text{either} \quad T_i^s + d_i \le T_j^s \quad \text{or} \quad T_i^s \ge T_j^s + d_j \tag{3}$$

$$B\ (t) = \sum_{j,k} C * d_j(k) \qquad\qquad j \ni T_j^s(k) + d_j(k) \le t, T_j^d(k) \ge t \tag{4}$$

$$T_s \ \le T_s^{\max} \tag{5}$$

Constraints (1)-(5) represent the conditions for transmission and playback of object based audio-visual presentations. Constraint (1) enforces the on-time delivery of access units. Ignoring the constant end-to-end delays, (1) gives the latest time an access unit can be transmitted. Constraint (2) imposes intra-object synchronization by enforcing precedence constraints among the access units. Access units are never transmitted out of order; they are transmitted in their decoding order. Since a single channel is used for transmission, channel occupancy of any two access units cannot overlap. Constraint (3) ensures that data is delivered on a single channel between a server and a client. Equation (4) gives the buffer occupancy at the end-user terminal at time *t*. Constraint (5) gives a bound on the startup delay for the given presentation. If the problem cannot be solved, i.e., a schedule that satisfies the given resource constraints cannot be found, some of the constraints could be relaxed in order to find a schedule. The constraints can be relaxed by: reducing the number of objects, increasing the startup delay, or increasing the channel capacity.

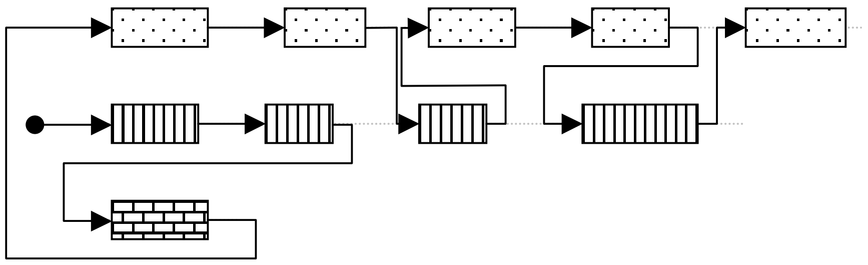

Figure 4.2. Sequencing of access units in a 3-object Presentation

EXAMPLE:

Consider the scheduling of a presentation with three objects as shown in Figure 4.2. Object 1 has five access units, object 2 has three, and object 3 has one access unit. The AUs are shown with increasing decoding time stamps from left to right. We have to find a schedule, if one exists, that sequences the AUs starting with the first AU of one the three objects and satisfying the constraints. Figure 4.2 shows one such sequence. The general problem of determining the existence of such a sequence is NP-complete. We prove that in *Theorem 1* below.

Scheduling is a complex problem and has been widely studied [30][37][42][107]. Many of the scheduling problems are NP-complete and a number of approximation algorithms are developed trading off optimality for tractability [61][116]. The scheduling problem closest to the audio-visual object scheduling is job shop scheduling on a single machine. There has been earlier work on scheduling on single machines. Complexity of machine scheduling problems is studied in [98]. Carlier proved the NP-hardness of one-machine sequencing problem in [29] and some approximation algorithms are discussed in [61]. Another problem with similarities to audio-visual scheduling is job scheduling with temporal distant constraints. NP-completeness results and polynomial time algorithms for a restricted instance of the problem are given in [72]. In spite of the similarities to the current problem, the approximation results for single machine scheduling problems cannot be applied to audio-visual object scheduling because of an entirely different problem domain and additional constraints on AV-object scheduling. Approximation algorithms are based on heuristics and domain knowledge is essential to develop good designs. Even though the results of single-machine scheduling are not directly applicable to audio-visual presentations, some of the results can be used in scheduling individual objects on a channel. The results of single machine scheduling problem as formulated by Lawler in [95][96] may be used to determine the schedulablity of individual objects.

4.2.4 Complexity of audio-visual object scheduling

Theorem 1: Scheduling of access units in Audio-Visual presentations (SAV) is $\mathcal{NP}$- Complete in the strong sense.

Proof: We prove this by transforming the problem of SEQUENCING WITHIN INTERVALS (SWI), proven to be $\mathcal{NP}$-Complete in the strong sense [107].

We restate SWI below.

INSTANCE: A finite set T of tasks *and, for each $t \in T$, an integer* release time $r(t) \geq 0$, *a* deadline $d(t) \in Z^+$, *and a length $l(t) \in Z^+$.*

QUESTION: Does there exist a feasible schedule for T, i.e., a function $\sigma : T \rightarrow Z^+$, such that, for each $t \in T, \sigma(t) \geq r(t), \sigma(t) + l(t) \leq d(t)$, and if $t' \in T - \{t\}$, then, either $\sigma(t') + l(t') \leq \sigma(t)$ or $\sigma(t') \geq \sigma(t) + l(t)$

The basic units of the SWI problem are the tasks $t \in T$. The local replacement for each $t \in T$ is a single access unit $A_j(k)$ with $r(A_j(k)) \geq T_j^s(k-1)$,

$d(t) = T_j^d(k)$ $l(t) = d_j(k)$. *We disregard the buffer and startup delay constraints. It is easy to see that this instance can be created from SWI in polynomial time. Since SWI can be transformed to SAV, SAV is at least as hard as SWI.*

Since SAV is $\mathcal{NP}$ - Complete in the strong sense, it cannot be solved by a pseudo-polynomial time algorithm. We present several polynomial-time algorithms based on heuristics and constraint relaxation and evaluate their performance with respect to speed and efficiency.

4.3 Characterizing Object-Based Audio-Visual Presentations

The complexity of an MPEG-4 presentation is an important factor that influences terminal design (in terms of support for content playback), the channel capacity required and also a server's performance. In case of MPEG-2 content, the average bit-rate and peak bit rate are good indicators of the resources required to process 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. Furthermore, the result of combining CBR streams on a single channel is shown to be VBR [59] and object schedules generated based on CBR reservations [111] may not work.

The complexity of an MPEG-4 presentation mainly depends on the number and type of objects involved, object playout, and interactivity (local and remote). Complexity of a presentation may be functionally expressed as:

$$C_p = f(N, OT(i), P_i, I_{l,} I_r) \qquad 1 \le i \le N$$

C_p is the complexity of the presentation, N is the number of objects in the presentation
$OT(i)$ is the object type
P_i is the object playout time (the length of time an object is present in a presentation)
I_l and I_r are possible local and remote events respectively.

$$C_p = \sum_{i=1}^{N} OT(i) * P_i + f(I_l) + g(I_r) \qquad 1 \le i \le N$$

The above formula can be used to estimate the complexity of object-based presentations.

On the server side, while the size of an object may indicate the complexity, on the player side, a number representative of the resources needed to decode an object is needed. This differentiation gives rise to different notion of content complexity at the server end and player end of a system. Duration of an object in a presentation is representative of the playout complexity. *OT(i)*P(i)* is a linear relationship indicating that an object that is played for a longer duration needs more resources and hence is more complex. When user interaction is allowed, the resulting asynchronous events consume more resources, may affect object scheduling, and also the required network and server resources. While local interaction with a presentation only affects a player, the user interaction resulting in a server interaction affects both the client and a server. The function $g(\cdot)$ gives the complexity as a result of server interaction and the function $f(\cdot)$ gives the complexity as a result of local interaction. This may be a negative value if it results in reduced complexity; e.g., when an object is removed from a presentation.

4.4 Startup Delay and Terminal Buffer

An MPEG-4 terminal has a finite buffer to store the received data until they are decoded. The amount of buffer capacity required depends on the type and number of elementary streams being buffered. Since there are usually no limits on the number of objects in audio-visual presentations, it is not practical to have sufficient buffer for *all* presentations. A terminal should be designed to support a *class* of presentations. The amount of buffer available also determines the upper bound on the startup delay for a session. The higher the startup delay, the higher the buffer capacity required (with channel capacity remaining the same). When scheduling presentations, a scheduler should assume the minimum allowable buffer for terminals in order to support all terminal types. Even though the knowledge of the buffer occupancy at a terminal may help improve the schedule, it makes the schedule dependent on the buffer model used by the terminals. Since the buffer model and management in a terminal depends on terminal design, we designed the scheduler to be buffer model independent.

Startup delay can be defined as the time a user has to wait from the time a request is made until the time the presentation starts. A startup delay of T_s is *not* equal to buffering T_s seconds of the presentation. The amount of startup delay varies from presentation to presentation and even for the same presentation, it may vary with varying resources (e.g, bandwidth and buffer). Startup delay can be viewed as preloading the beginning of a presentation so that the presentation is played back continuously once the playback starts. The amount of startup delay required for the smooth playback of a presentation depends on the channel capacity. For any channel, the minimum startup delay is the time needed to transmit (buffer) access units that are presented at time 0 (AUs with timestamp 0).

Consider a presentation composed of several images displayed on the first screen, followed by an audio track. The images to be displayed on the first screen should reach the terminal before the presentation starts, resulting in a startup delay. If the channel bandwidth reserved for the presentation is allocated based on the low bitrate audio stream that follows the images, the startup delay will be higher. On the other hand, if the higher bandwidth is reserved to minimize the startup delay, the capacity may be wasted during the remainder of the presentation when low bitrate audio is delivered. The tradeoff depends on resource availability and startup-delay tolerance of the application.

Given a startup delay, T_s, the buffer required is equal to the size of the objects that can be loaded (transmitted to the client) in time T_s. The minimum buffer required for this delay is $T_s * C$. The minimum startup delay for any presentation is equal to the time required to transmit (load) the objects/ instances to be displayed at time 0. We refer to this time as T_s^0. T_s^0 is the optimal startup delay for startup delay-optimal schedules and is the lower bound on startup delay for bandwidth-optimal schedules.

4.4.1 Residual Data Volume

We introduce the notion of *data volume* to quickly compute the minimum startup delays needed for a presentation and determine the non-schedulability. Data volume (V_d) is the amount of data (in bits) transferred during a session. The amount of data that can be carried by a channel during a session is the data pipe volume ($_{p} = C * D_p$). The amount of data volume exceeding the data pipe volume is the residual data volume ($_{res} = V_d - V_p$). A positive V_{res} gives the lower bound on the amount of data to be loaded during startup and hence determines the lower bound on the startup delay for a presentation. A negative value of V_{res} indicates unused channel capacity during the session. We prove the lower bound on channel capacity required in Theorem 2 below.

Theorem 2: For a presentation of duration $D_{p,}$ *the lower bound on channel capacity required for a startup delay-optimal schedule is:*

$C \geq C_{\min}$ *where* $C_{\min} = \dfrac{V_d}{D_p}$*, and the bound is tight.*

Proof:

$$_{d} = \sum_{j,k} s_j(k)$$

$$D_p = \max_j\{T_j^d(n_j)\} - \min_j\{T_j^d(n_j)\}$$

For a presentation of length D_p, the data pipe volume at the given pipe capacity is

$$_{p} = C * D_p$$

Assuming that the buffers are filled up at a rate C, the startup delay due to V_{res} *is*

$$T_s^{res} = (V_d - V_p) / C$$

To minimize the startup delay,

$$T_s^{res} = (V_d - V_p) / C = 0 \Rightarrow V_p = V_d$$

Since ${}_p = C * D_p$, substituting ${}_p$ we get the lower bound on the channel capacity

$$C_{\min} = \frac{V_d}{D_p}$$

From constraint *(1)*

$$T_j^s(1) \le T_j^d(1) - d_j(1) \quad \Rightarrow \quad T_j^s(1) \le T_j^d(1) - s_j(1)/C$$

From constraint *(2)*

$$T_j^s(1) \ge T_j^s(0) + s_j(0)/C$$

Assuming that the presentation starts at time *0*

$$T_j^s(0) = 0, \quad \Rightarrow \; T_j^s(1) \ge s_j(0)/C$$

From constraints *(1)* and *(2)*,

$$T_j^s(2) \ge T_j^s(1) + s_j(1)/C \quad \Rightarrow \quad T_j^s(2) \ge s_j(0)/C + s_j(1)/C$$

$$T_j^s(2) \le T_j^d(2) - s_j(2)/C \quad \Rightarrow \quad T_j^d(2) \ge \; T_j^s(2) + s_j(2)/C$$

$$\Rightarrow \quad T_j^d(2) \ge \; s_j(0)/C + s_j(1)/C + s_j(2)/C$$

Similarly,

$$T_j^d(n_j) \ge \; s_j(0)/C + s_j(1)/C + \ldots + s_j(n_j)/C$$

Since the AUs are transmitted on a single channel,

$$D_p = \max\{T_j^d(n_j)\} \ge \; \sum_{\mathrm{j}} s_j(0)/C + s_j(1)/C + \ldots + s_j(n_j)/C$$

$$D_p \ge \frac{1}{C}\sum_{j,k} s_j(k)$$

$$\Rightarrow \; D_p \ge \frac{1}{C} * C_{\min} * D_p$$

$$\Rightarrow C \ge C_{\min}$$

We can show that the bound is tight by considering the example as shown in the Figure 4.3.

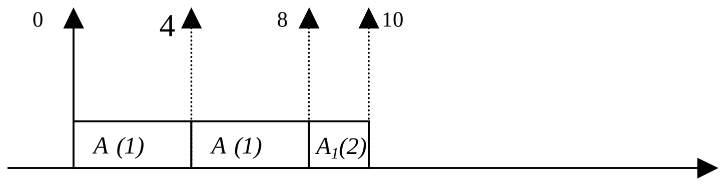

Figure 4.3. Example to prove the tightness of the bound

Object 1 has 2 AUs and object 2 has 1 AU. The decoding times and sizes of AUs in bytes are*:*

$T_1^d(1) = 4,\ T_1^d(2) = 10,\ T_2^d(1) = 8,\ s_1(1) = 10,\ s_1(2) = 5,\ s_2(1) = 10$.

With these values the send times and channel capacity are*:*

$T_1^s(1) = 0, T_1^s(2) = 8, T_2^s(1) = 4,\ \mathit{and}$

$C = C_{\min} = 2.5\ \mathit{bytes}/\sec$.

The actual channel capacity required to minimize startup delay may be higher depending on the timing constraints of access units. Note that irrespective of the channel capacity, the minimum startup delay remains non-zero and is equal to T_s^0.

Thus for any given presentation with resource constraints:

the minimum startup delay is, $T_s^{\min} = \max\{T_s^{res}, T_s^0\}$

the minimum buffer capacity required is, $B_{\min} = T_s^{\min} * C$

the presentation is schedulable only if the available buffer is at least equal to $B_{\min}$.

4.5 Scheduling Algorithms

In this section we describe a family of scheduling algorithms for AV object scheduling. Given an AV presentation, the scheduling algorithms compute a delivery schedule according to the selected criteria. We assume that the terminal buffer is fixed and compute startup delay-optimal or bandwidth-minimizing schedules. The algorithms can also be re-purposed to compute the minimum terminal buffer required for a given channel capacity. Figure 4.4 shows the flowchart for selecting an appropriate scheduling algorithm. The choice of the algorithm depends on the applications, resource availability, and constraints.

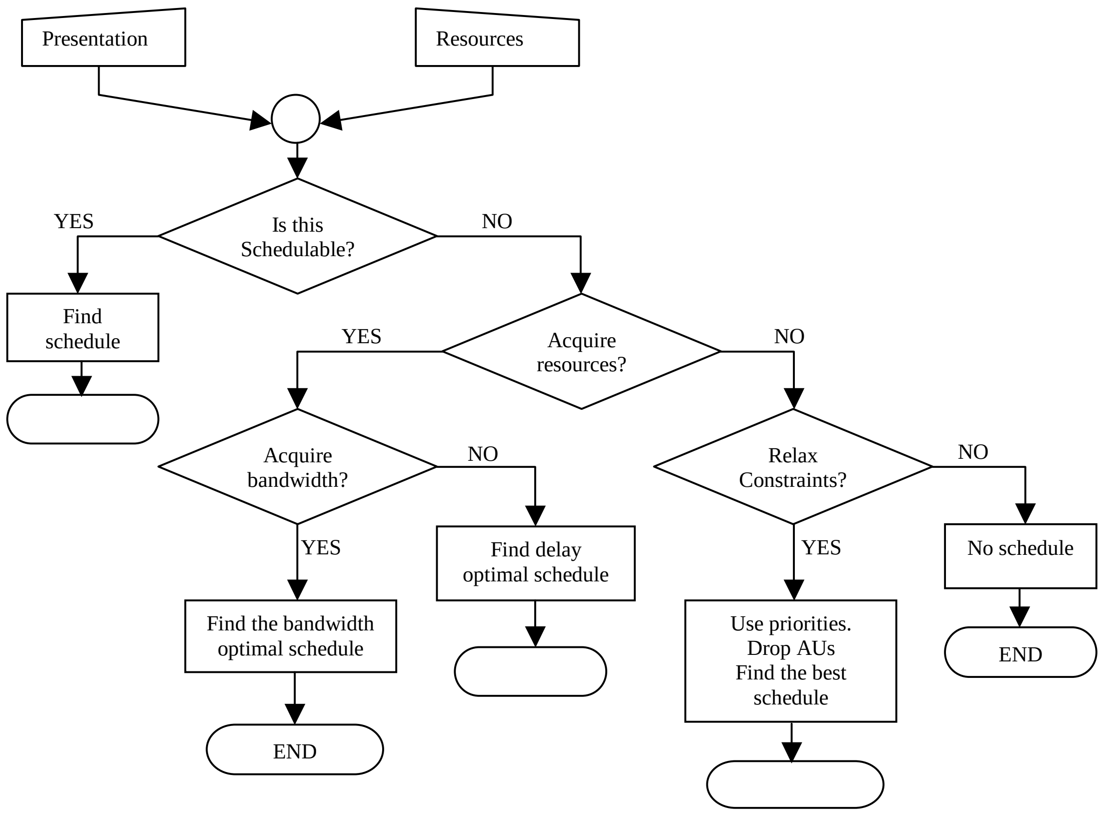

Figure 4.4. Determining the Schedulability of a Presentation

4.5.1 Algorithm FullSched

This algorithm is based on the last-to-first idea mentioned in [96] for scheduling jobs on a single machine. The main principle behind this algorithm is scheduling an AU with latest deadline first and scheduling it as close to the deadline as possible. The algorithm computes the schedule starting with an AU with the latest decoding time in the presentation. This algorithm computes the schedule, the required startup delay, and any channel idle times. The channel idle times computed are used in the gap-scheduling algorithm described in Section 4.5.2.

Let *S* be the set of current AUs to be scheduled. Initialize *S* to contain the last AU of each of the objects to be scheduled. Let x_j be the index of the next AU of object *j* to be scheduled.

$x_j = n_j, \qquad 1 \le j \le N$

Initialize $S = \{A_j(x_j)\}, \qquad 1 \le j \le N$

$S(j)$ is the AU of object *j* to be scheduled next.

S contains at most one AU for every object *j*.

G is the set of channel idle times. Idle time is given by a tuple <t, d>, i.e, the channel is idle for duration d starting at time t. Initialize $G = \{\phi\}$

Set current time $i = \infty$

Sort AU of objects in the decreasing order of their decoding times.

```
BEGIN
    while ($S \ne \phi$){
        $i = \min\{i, \max\{T_j^d(k)\}\}, \quad T_j^d(k) \ni A_j(k) \in S$
        $T_j^s(x_j) = i - d_j(x_j)$; //send time for $A_j(x_j)$
        // Update $i$
        $i$ -= $d_j(x_j)$;
        $x_j$--;
        // Update $S$ by removing $S(j)$ from S
        $S$ -= $S(j)$;
        // add $A_j(x_j)$ to $S$
        if( $x_j \ne 0$)
            $S$ += $AU(j, x_j)$
        if ($i > \max\{T_j^d(k)\}$), $\qquad T_j^d(k) \ni A_j(k) \in S$
            // there is a gap on the channel
            $G$ += $(\{\max\{T_j^d(k)\}, i - \{\max\{T_j^d(k)\}\})$
        }
        if $T^s_{first} < 0$
        then $T_s = |T^s_{first}|$
        $T^d_j(k) \mathrel{+}= T_s$,        $\forall j,k$
    END
```

The process begins with *S* initialized with the last AU of each of the objects in the presentation and *G* initially empty. In each of the iterations, the AU with the latest decoding time is scheduled as close to the decoding time as possible. Ties are broken arbitrarily. Once the AU of an object is scheduled, the next AU of that object is added to *S* as long as there are AUs to be scheduled. The current time is given by *i*. A value of *i* greater than the largest decoding time of AUs in *S* ($\max\{T^d_j(k)\}$) indicates idle time on the channel (gaps or slots). The channel is idle because nothing can be scheduled between $\max\{T^d_j(k)\}$ and *i*. This is illustrated in the example below.

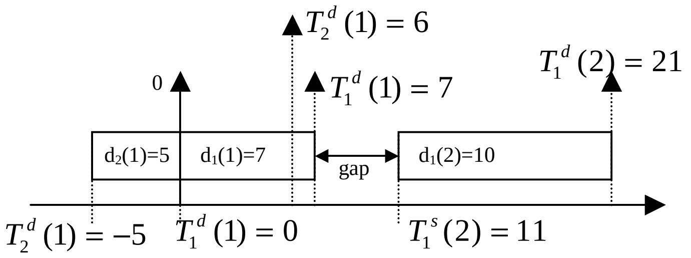

Figure 4.5. Applying FullSched to a two-object Presentation

EXAMPLE: Consider two objects O_1 with two AUs and object O_2 with one AU with duration on channel, d, and decoding time stamp, T given as a set of tuples <d,T>. O_1 ={<7,7>, <10, 21>} and O_2 = {<5, 6>}. After $A_1(2)$ is scheduled, at time $T^s_1(2) = 11$, nothing can be scheduled between $T^d_1(1) = 7$ and current time *i*=11 resulting in a gap on the channel. A negative value of the send time indicates that the AU has to be transmitted before the presentation starts giving rise to a startup delay.

When S becomes empty, i.e., all AUs are scheduled, a negative value of i indicates the required startup delay for the presentation and G gives the set of gaps on the channel. Since the decoding times, $T_j(k)$, are all non-negative, once i becomes negative, there are no gaps on the channel indicating that the AUs are tightly packed. A gap is not an indication of the sub-optimality of the schedule. However, it may indicate the sub-optimality of the bandwidth-optimized schedule; i.e, it may be possible to schedule the presentation at a lower bandwidth. When N=1, this algorithm can be used to determine the schedulability of individual objects and determine the un-schedulability of a presentation. This is especially useful during the content creation process where objects are added to create presentations. When an object is added during an editing operation, it is faster to determine the un-schedulability of a presentation by computing the independent schedules of objects and adding the startup delays of the independent schedules. However, a full schedule should still be computed after the editing operations to determine the schedulability of the presentation under given resource constraints. This algorithm is not efficient in computing the schedulability during the content creation process, as the full schedule needs to be re-computed every time an object is added. We next present a gap-scheduling algorithm that computes incremental schedules to determine the schedulability of a presentation and is well suited for the content creation process. We also prove that FullSched and gap-scheduling algorithm compute startup delay optimal schedules.

Theorem 3: Algorithm FullSched produces a startup delay-optimal schedule.

Proof: The algorithm selects an AU with the latest decoding time and schedules it as close to the deadline as possible. i.e., the algorithm schedules the AUs in non-increasing order of their decoding times. On a conceptual timeline, with time increasing from left to right, we are stacking the AUs as much to the right as possible. Gaps occur only when there is nothing to be scheduled in that gap. Any (or part of) AUs that appear to the left of the origin (time = 0) give the startup delay. Since the algorithm always moves the AUs to the right whenever possible, the startup delay is minimized. A smaller startup delay is not possible because, it would mean moving the AUs to the right implying that there is a usable gap on the channel. This cannot be the case because the algorithm would have scheduled an AU in that gap!

4.5.2 The GapSched Algorithm

The gap-scheduling (GapSched) algorithm schedules AUs in the available gaps on a channel. It starts with available gaps on a channel and tries to fit an access unit or a partial AU using the `SplitAndSchedule` procedure. The initial set of gaps may be obtained by using FullSched to schedule a single object. The algorithm looks for the first available gap starting at a time less than the decoding time of the AU to be scheduled. Since G is already sorted in the decreasing order of gap-times, the look up can be done very efficiently. If the gap duration is not long enough to fit an AU, the AU is split, with one part scheduled in the current gap and the other added to S to be scheduled next. The AUs in the presentation are iteratively scheduled until *S* becomes empty.

```
S contains all the AU of the object $j$
$S = \{A_j(k)\}, \qquad 1 \le k \le n_j, j \in \{N\}$
Sort AUs in $S$ in the decreasing order of their decoding times
$G$ = set of available slots $\ne \{\phi\}$.
$G(l)$, is the $l$ th tuple in $G$ with start time $G(l).t$ and duration $G(l).d$.
$k = n_j$
BEGIN
    while $(S \ne \phi)\{$
        find a slot $l$, $G(l)$, such that $T_j^d(k) > G(l).t$
        if $(G(l).d \ge d_j(k))\{$
            $T_j^s(k) = G(l).t - d_j(k)$  //send time for $A_j(k)$
            k--;
            // update the gap
            if $(G(l).d - d_j(k) > 0)$
                $G(l).d = G(l).d - d_j(k)$
            else
                $G -= \{G(l)\};$
            // remove AU from the set
            $S -= A_j(k);$
        }
        else{
            PROCEDURE SplitAndSchedule ($A_j(k)$,$G(l)$);
        }
    }
END
```

Split the AU into two parts, one part that is scheduled in *G(l)* and the other that is placed back in *S*.

```
PROCEDURE SplitAndSchedule ( $A_j(k)$ ,G(l)){
        Create a sub - AU of length $G(l).d$ containing the last $G(l).d * C$ bytes of the AU
        $t'_j(k) = G(l).t;$
        $d_j(k) = d_j(k) - G(l).d;$
        $G -= \{G(l)\};$
}
```

4.5.3 The IncSched Algorithm

The incremental scheduling (IncSched) algorithm computes the schedule for a presentation by considering one object at a time. This is a typical content creation scenario where objects are composed to create a presentation. Instead of re-computing the full schedule with FullSched algorithm each time an object is added, this algorithm computes the schedules incrementally by scheduling the AU in the available gaps. Note that not all the gaps are schedulable. A gap is un-schedulable if there are no AUs with decoding times greater than the gap time. An un-schedulable gap indicates unused bandwidth, which is either due to the structure of the presentation or due to a sub-optimal schedule. The IncSched algorithm uses FullSched and GapSched algorithms to schedule a presentation. This algorithm appears to be more efficient than FullSched as it schedules parts of AUs and fills all the gaps. However, this is only as efficient as FullSched as far as startup delay is concerned. Splitting the AUs in order to pack the gaps is not going to decrease the startup delay, as the available channel capacity is the same. The startup delay, like in other cases, is given by the send-time of the first AU transmitted.

OBJ is the set of objects in the presentation.

```
BEGIN
    Apply FullSched and compute schedule for object 1.
    //LRG is a sufficiently large number to accommodate startup delay.
    $G += \{< -LRG, i - LRG >\}$
    for $j \in OBJ - \{1\}$, apply gap scheduling GS to $j$.
    for $j \in OBJ$, find $t_{first}$, the send time of the first AU to be transmitted
(smallest $t_j(k)$)
    if $T^s_{first} < 0$
    then $T_s = \left|T^s_{first}\right|$
    $T^d_j(k) \ += T_s, \qquad \forall j, k$
END
```

Theorem 4: The IncSched algorithm is startup delay-optimal.

Proof: The first object is scheduled using FullSched producing a startup delay-optimal schedule for that object. GapSched, when applied iteratively to the remaining objects, packs the AUs tightly; i.e., an access unit is scheduled if the gap time is less than the decoding time for that AU. The resulting startup delay is optimal because the algorithm would reduce the startup delay by moving the AU to the right on the timeline if any schedulable gap is available.

EXAMPLE: Consider two objects O_1 with two AUs and object O_2 with one AU with duration on channel, d, and decoding time stamp, T given as a set of tuples <d,T>. O_1 ={<7,7>, <10, 21>} and O_2 = {<5, 6>}. The top part of the Figure 4.6 shows the schedule computed using FullSched and the bottom half shows the schedule computed with IS, with object 2 scheduled first using FullSched. The figure also shows un-schedulable gaps in both the schedules.d

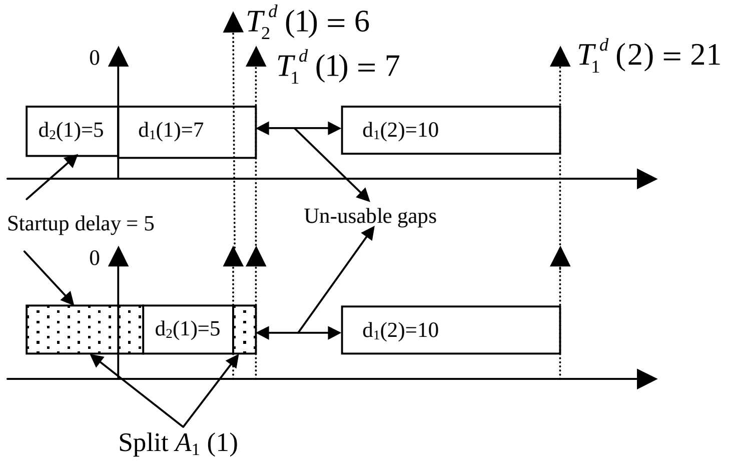

Figure 4.6. Schedules Computed Using FullSched and IncSched

There may be cases where splitting the AUs is necessary, for example, the underlying transport layer may not be able to handle large AUs. This result shows that AUs can be split while maintaining the optimality of the schedule. Although the IncSched algorithm produces an optimal schedule and is useful in determining the schedulability of a presentation in applications such as content creation, the schedule generated by FullSched may be more efficient when the overhead due to splitting and packetizing is significant.

4.5.4 Algorithm MinC

In scheduling audio-visual objects, we have so far answered two questions: 1) is the given presentation schedulable under the given resource constraints and 2) what is the minimum startup delay required for this presentation. If the answer to question 1 is negative (or if bandwidth consumption needs to be minimized), the question we need to address is: what are the minimum amounts of resources required to schedule the presentation. Since we cannot make assumptions about decoder buffers in order to keep the schedules player-independent, the only resource that can be acquired is the bandwidth (*C*). We next present the MinC algorithm that computes the minimum bandwidth required (CBR) to schedule a presentation.

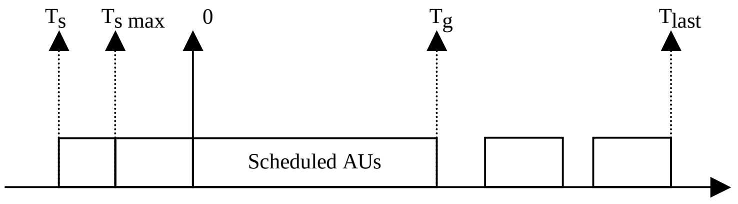

Figure 4.7. First gap-time and startup delay of a presentation

This algorithm is based on the premise that there is a gap on the channel only when everything else *after* the gap-time has been scheduled. Otherwise an un-scheduled AU would have taken up the gap. The presentation is not schedulable because there is not enough channel capacity until the first gap time, T_g (smallest gap time). Consider the case in Figure 4.7. T_g is the first gap-time, $T_s^{\max}$, is the maximum allowable startup delay with the current channel capacity, and T_s is the current start-up dealy. The channel capacity should be increased to accommodate T_s -$T_s^{\max}$ in the duration T_g - $T_s^{\max}$. The new value of C then is $C_{\text{new}} = ((T_g + T_s) + (T_s - T_s^{\max})) / (T_g + T_s) * C$. The algorithm also outputs the bandwidth profile (*BP*) for the presentation in the form of 3-tuples <capacity, start, end>. Note that the increased channel capacity is not going to affect the schedule from T_g to T_{last}. A finer bandwidth profile can be obtained by initializing C with $C_{\min}$ and increasing C by a small value in each iteration.

```
BEGIN
    $G_c = |G|$ = gap count, number of gaps on the channel.
    $BP = \{\phi\}$
    $T_{last}$ is the decoding time of the first AU scheduled (= duration of the presentation)
    $C = C_{min}$, computed using the results of theorem 2.
    SCHEDULE: compute schedule using FullSched
    If schedulable goto END
    if ($G_c == 0$)
        $T_g = T_{last}$
    else {
        $T_{g\text{-}old} = T_g$
        Find the smallest gap time, $T_g$
        $BP$ += {<$C$, $T_g$, $T_{g\text{-}old}$>}
    }
    $C = ((T_g + T_s) + (T_s - T_s^{\max})) / (T_g + T_s) * C$
    goto: SCHEDULE
END
```

The channel capacity output by the algorithm is in the form of a set of 3-tuples forming a bandwidth profile. The minimum CBR channel required is given by the maximum value of C in the bandwidth profile. This profile may also be used to reserve session bandwidth efficiently. Since the schedule is computed from last to first (right-to-left on the timeline), the bandwidth profile will always be a step function with possible steps (decreasing) from left to right. Figure 8 shows some sample profiles. This algorithm does not give the best profile to reserve variable session bandwidth since the algorithm does not reduce the bandwidth when it is unused. Consider the example shown in the Figure 4.8. At T_g, a capacity increase is necessary. Suppose the increase in C at T_g is sufficient to schedule the presentation. It is possible that the presentation from 0 to T_b could have been scheduled with a much smaller capacity.

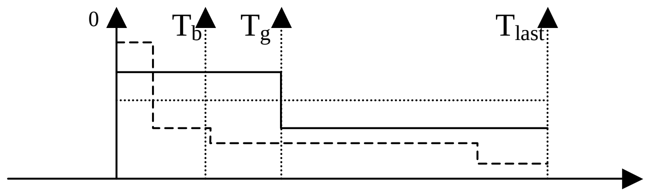

Figure 4.8. Typical shapes of bandwidth usage generated by MinC

4.5.5 Algorithm BestSched

When a presentation cannot be scheduled with the given resources, and additional resources cannot be acquired, the only way to schedule the presentation is to drop some access units. AUs cannot be dropped arbitrarily as they have different effects on the presentation. Content creators should assign priorities to objects and possibly AUs of objects to help a scheduler in determining the AUs to be dropped. The following algorithm schedules a presentation by dropping lower priority objects.

```
BEGIN
    SCHEDULE: Compute schedule using FullSched.
    if ($B <= T_s * C$) {
        Remove $A_j(k)$ of lower priority objects such that,
        $R = \{A_j(k)\}$    ∍   $C * \sum d_j(k) \geq T_s * C - B$
```

```
            $A -= \{R\}$
            goto: SCHEDULE
    }
END
```

4.6 Discussion and Results

Determining the schedulability of a presentation is of *O(n)* complexity, where n is the number of AUs in the presentation. Both FullSched and GapSched fall under this category. These algorithms are used to determine the schedulability, compute an optimal startup delay for the given channel capacity, and for computing incremental schedules. The MinC algorithm, used to compute the minimum channel capacity required to schedule the presentation, calls FullSched iteratively with channel capacity incremented in each iteration. The number of iterations depends on the structure of the presentation and the initial value of C. The complexity of this algorithm is *O(Kn) = O(n),* where K is a constant determined by the structure of the presentation and the initial channel capacity. The proposed algorithms are fast enough to determine the schedulability of the presentations in real-time.

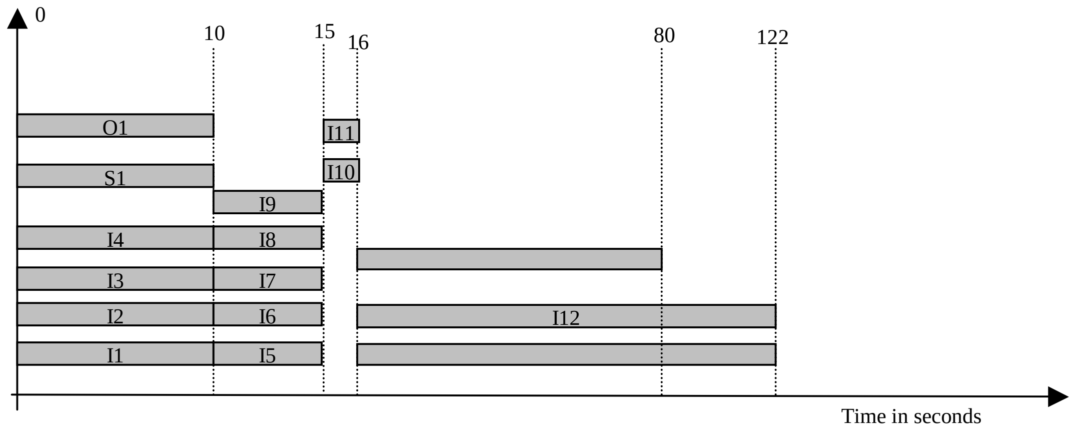

Figure 4.9. Structure of the presentation in the example

The structure of the presentation has significant impact on the performance of MinC algorithm. To aid the discussion, we consider a relatively complex MPEG-4 presentation with structural overview as shown in Figure 4.9. The properties of the objects in the presentation are tabulated in Table 4.1.

Table 4.1: Properties of objects in the example

Object FileName (ID)	Size (KB)	Start Time	AU Count
Scene.od (O1)	0.5	0	1
Scene.bif (S1)	1 (1025 Bytes)	0	4
Main1.jpg (I1)	25	0	1
Main2.jpg (I2)	22	0	1
Main3.jpg (I3)	19	0	1
Main4.jpg (I4)	20	0	1
main_ui.jpg (I5)	39	10	1
Advent_logo.jpg (I6)	7	10	1
CU_logo.jpg (I7)	9	10	1
Lm_logo.jpg (I8)	6	10	1
Xbind_logo.jpg (I9)	8	10	1
geo_pict.jpg (I10)	23	15	11
dance_pict.jpg (I11)	29	15	1
next_page.jpg (I12)	51	16	1
clip01.h263 (V1)	552	16	974
clip01.g723 (A1)	64	16	3233

In the following discussion we refer to objects by the codes shown in the first column of the table. The presentation is made up of 16 objects including scene description, object description, images, audio, and video. The presentation is composed of three scenes. Before the scenes are loaded, the scene description and object description streams are received and decoded by the terminal. The first scene consists of four jpeg images (I1 - I4) animated to give a breakout effect. The scene is encoded to animate the images for 10 seconds and then load the second scene. The second scene consists of a background image (I5), four logos with animation effects (I6 - I9), and two images (I10 and I11) with descriptive text of the following audio-visual scene. The last scene consists of a background image, an audio stream, and a video stream. The temporal layout of the presentation is shown in Figure 4.9. The times indicated are the decoding times of the first AUs of the objects starting at that time. Thus the first four images (I1-I4), the scene description (S1) and the object descriptor stream (O1) should reach the decoder before anything is displayed on the screen. This amounts to the minimum startup delay for the presentation. The objects I5 - I9 should reach the decoder by the time t = 10, I10 and I11 by 15, and the first AU of V1 and A1, and the object I12 should reach the terminal by the time t = 16. The video ends at t = 80 while the audio stream continues until the end of the presentation. The total length of the presentation is 122 seconds. This temporal ordering of objects in the presentation results in higher data rates toward the beginning of the presentation (object data to be delivered in the first 16 seconds: 261 KB ~= 130 Kbps).

4.7 Startup Delay and Capacity Computation

Figure 4.10 shows the plot of the minimum channel capacity required for a given startup delay. This is a scenario with variable buffer at the terminal. We assume a work-conserving transport layer that delivers the objects at the minimum required capacity. The amount of buffer available at the terminal should be at least sufficient to store the data during the startup. For a startup delay of T_s, if $C_{\min}$ is the min capacity required, then the buffer at the terminal $B_{\min} > T_s * C_{\min}$. This curve is useful to determine the amount of buffer (delay) required based on the available network capacity, especially with terminals such as PCs with sufficient memory. As mentioned earlier in the discussion of the MinC algorithm, the MinC algorithm also computes the bandwidth profile for presentations. Figure 4.11 shows the bandwidth profile computed for a startup delay of 5 seconds. The minimum capacity in the profile is 59 Kbps even for the segment (80 - 120 secs) that only has low bit rate audio (6 Kbps). This is because MinC starts with an initial value of C computed using the residual data volume as described in Section 4.4. This starting point is acceptable for computing a CBR channel required; for a profile to be used in reserving variable network resources, a lower initial value of C should be selected. The final bandwidth jump in the profile gives the minimum channel capacity required for the given delay or buffer.

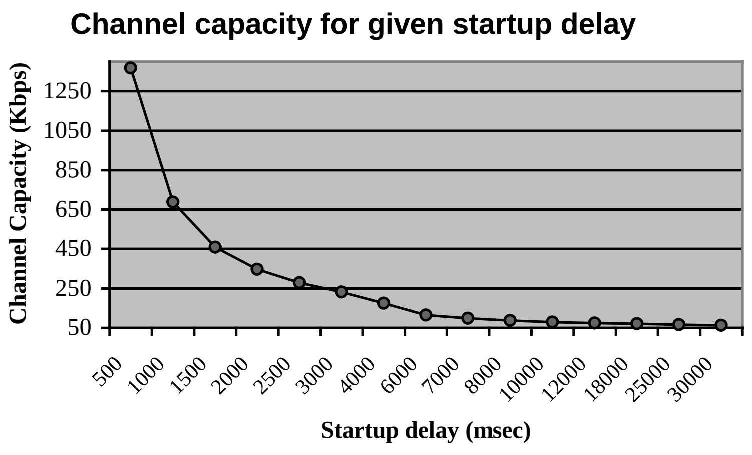

Figure 4.10. Computing min capacity using MinC

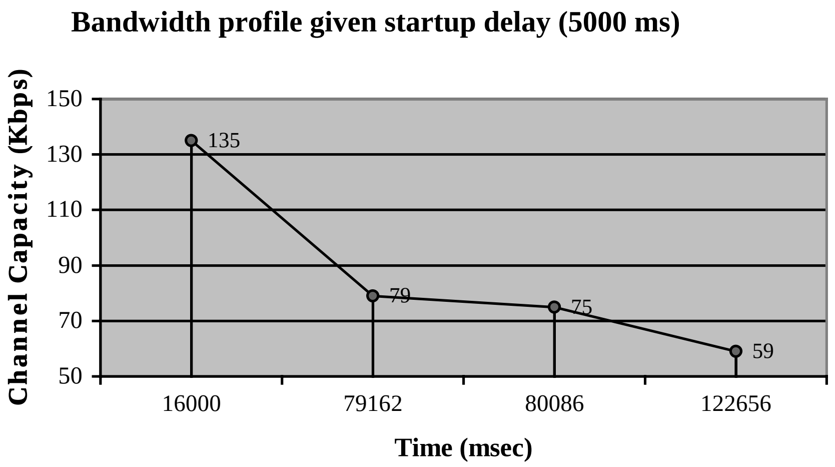

Figure 4.11. Computing the bandwidth profile using MinC

4.8 Buffer and Capacity Computation

The available buffer at the terminal determines the amount of startup delay a terminal can support. The available channel capacity imposes a lower limit on the buffer required. Lower channel capacity implies higher startup delays and hence larger required buffer. Figure 4.13 gives the required buffer at various channel capacities. This can be directly converted to the startup delay at that capacity. Computing the capacity for a given buffer is bit more computationally intensive. Unlike the previous case where we assumed enough capacity to support the required startup delay, the buffer capacity is fixed in this case. This typically the scenario when using dedicated devices such as set-top-boxed with fixed receiving buffers. Since the buffer is fixed, the supported startup delay decreases as channel capacity increases. For MinC algorithm to complete, the reduction in startup delay due to increased channel capacity should be greater than the reduction in startup delay supported by the buffer.

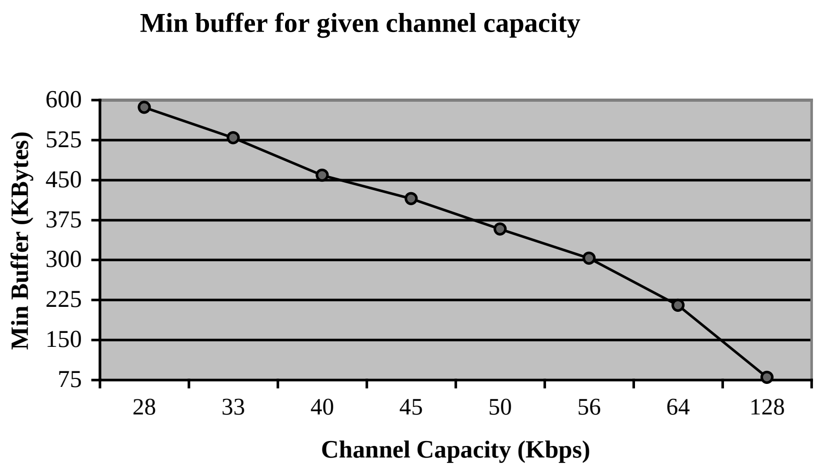

Figure 4.12. Minimum required buffer

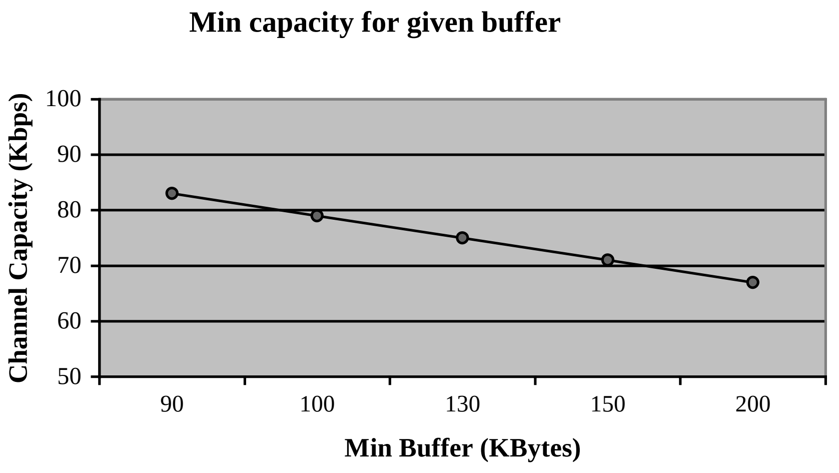

Figure 4.13. Min Capacity for a given buffer

Figure 4.14 shows a case with terminal buffer $< B_{\min}$. For the given presentation, $B_{\min} = 85KB$, the size of objects to be decoded at time 0. As discussed in Section 4.4, for a presentation to be schedulable, the available buffer should be greater than $B_{\min}$, the lower bound on the required buffer capacity for the presentation. Since the terminal buffer is less than the required buffer, at any given capacity C, the supported startup delay $\frac{B_{term}}{C} < \frac{B_{\min}}{C}$, the required startup-delay. The presentation is hence un-schedulable with terminal buffer capacity of 84 KB. This is depicted in Figure 4.14, which shows the presentation is un-schedulable even at 3323 Kbps. The plot shows that no matter how much the channel capacity is increased, the presentation cannot be scheduled because of limited terminal buffer. To avoid infinite loops in MinC, the scheduler should first examine the available and required buffer capacities.

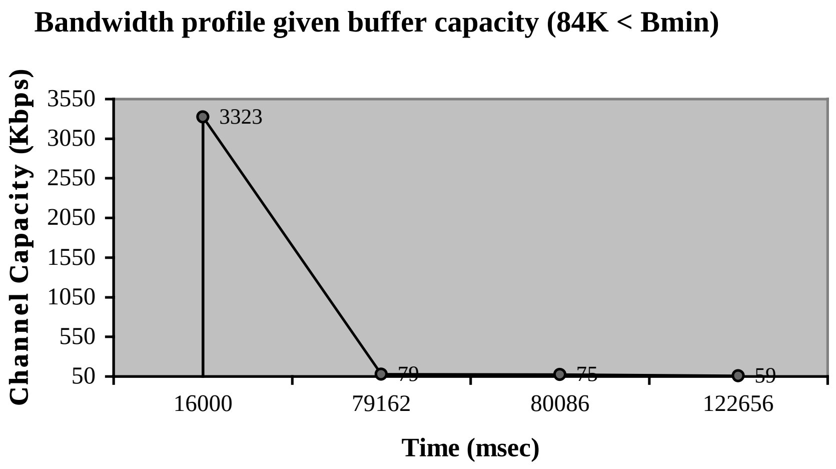

Figure 4.14. Partial profile for low terminal buffer

4.9 Conclusion

We presented the problem of scheduling audio-visual objects under resource constraints. The problem is $\mathcal{NP}$-complete in the strong sense. We explored similarities with the problem of sequencing jobs on a single machine and used the idea of last-to-first scheduling to develop heuristic algorithms to determine schedulability and compute startup delay-optimal schedules. We introduced the notion of residual data volume to compute lower bounds on buffer, channel capacity and startup delay. Determining the schedulability of presentations online is important for applications like content creation where an additional object may make the presentation un-schedulable. We presented an algorithm that computes incremental schedules and produces a startup delay optimal schedule. Starting with a lower bound on the channel capacity for scheduling a presentation, the MinC algorithms minimizes the CBR channel capacity required to schedule the presentation. The proposed algorithms are of low complexity and can be implemented efficiently.

Chapter 5

Scheduling Interactive AV Presentations

5.1 Introduction

Interactivity in the context of audio-visual presentations suggests the features enabling the users to interact with the presentation. In the case of object-based presentations, this includes user interaction with individual objects and interaction among the objects themselves. Interaction with the objects in a presentation results in *events* that may alter the presentation or may just process some information without affecting the objects in the presentation. These events can be synchronous, happening at a predetermined time or asynchronous, happening anytime, usually within a time window during a presentation. Synchronous events can be classified into two types: *certain events*, happening at a predetermined time, and *user events*, happening at a predetermined time but only if the user interacts with an object (e.g., user interaction changing the ending of a movie). Asynchronous events, as the name indicates, happen when a user interacts with the objects and hence timing of such events is not known until the event happens. Of significance to the delivery-scheduling problem are the events that alter the presentation by adding and removing objects.

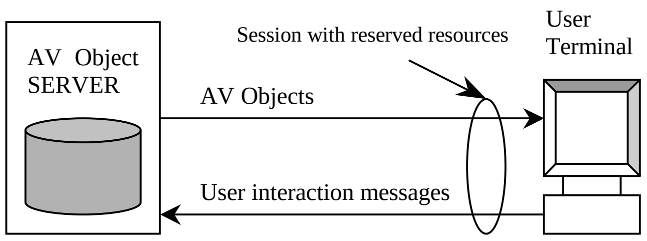

Figure 5.1. Message Exchange in User Interaction

Figure 5.1 shows the exchange of messages in interactive audio-visual presentations. If an event results in the addition of objects to a presentation, new resources will be required in order to schedule the altered presentation. A new schedule is needed only when the objects are added to a presentation. When objects are removed from a presentation, it does not affect the rest of the schedule, however rescheduling may save some network resources.

When a user is presented with a choice to interact with the presentation, the number of objects delivered changes only when the user chooses to interact with the presentation; i.e., for every possible user event, there are two possible schedules. With each event resulting in an altered schedule, the possible number of schedules grows exponentially with the number of events (2^n). A brute-force way of determining the schedulability by computing the feasibility of each of these schedules becomes impractical as the number of events grows. Figure 5.2 shows the exponential growth in the number of possible schedules due to user events. Consider the case depicted in Figure 5.3. When an event e1 happens, the objects are scheduled according to the schedule S_{12} and scheduled according to the schedule S_{11} if the event does not happen. It is easy to see that the options increase exponentially with the number of events (2^n, where n is the number of events). To determine the schedulability of interactive presentations, we have to examine if the presentation is schedulable under all the possible combination of events. The events have to be analyzed with respect to the resources required to support the events and the resources available when the events happen. To make the problem of determining the schedulability of interactive presentations tractable, a presentation can be viewed as consisting of a core and an interactive component. The core component of the presentation contains objects that are delivered synchronously while the interactive component contains objects that may be added/removed upon user interaction. To determine the schedulability of a presentation, we can compute the schedulability of the core presentation and the interactive components of the presentation separately.

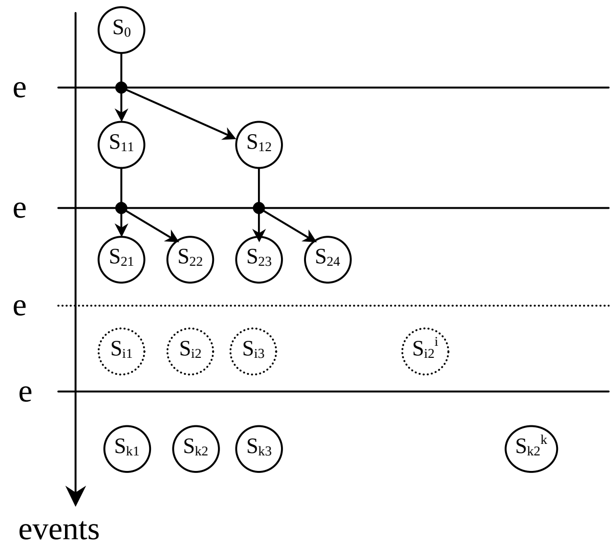

Figure 5.2. Possible Schedules as a Result of User Events

Typically events happen within specific time windows during a presentation. These event windows are designed by a content creator during the content creation process. Figure 5.3 shows the event window, the start time T_S, and the end time T_E, for an event that adds an object of duration *D*. Since the event that adds objects to a presentation can happen anytime from T_S to T_E, the object playout could start as early as T_S and last as long as $T_E + D$. The time interval $[T_S, T_E + D]$ is referred to as the event extent. The resources required to satisfy an event should be reserved at least until the end of the event window and can be released at the end of the event window if the event doesn't happen. For overlapping extents, it is not possible to compute the schedules of the objects together without knowing when the events are going to happen. If there are two or more possible events overlapping in time, the resources required to complete the events should be computed separately. Since non-overlapping events could use the same channel bandwidth, all the events should be considered when computing the amount of capacity required for the interactive component. Furthermore, if the required capacity is computed using MinC algorithm, the capacity cannot be reserved according to the resulting bandwidth profile since there is no way of determining when the event happens. The channel capacity required to complete an event is the CBR channel required for the objects involved. A combined object schedule for the objects added because of interaction should be computed only for those objects that are added as a result of the same event. Even though reserving resources that may not be used seems unreasonable, there is no other way to guarantee proper response to events in interactive presentations. The algorithms presented in the next section minimize the auxiliary capacity required to support interactive presentations.

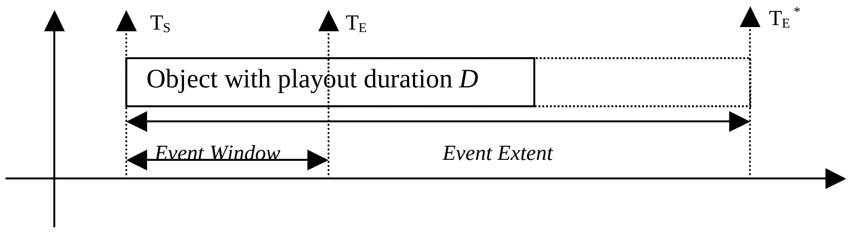

Figure 5.3. Event Window and Event Extent

5.2 Event Specification

We specify an event as a 4-tuple $< T_S, T_E, D, \pm O >$. T_S and T_E are the beginning and end of the time window during which the event may happen, D is the duration of the object added (0 if removed), and O is the object list with + indicating addition and - indicating removal of the object. With this notation, we can represent interactive audio-visual presentations as two sets of such 4-tuples, one with certain events corresponding to the objects in the core presentation (with $T_S = T_E$) and the other is the interactive component of the presentation, corresponding to the objects that are added or removed as a result of user interaction. An interactive presentation is schedulable only if the core and the interactive components are schedulable. We can determine the schedulability of the core and the interactive components of a presentation separately using the algorithms presented in Chapter 4. We also need algorithms that consider the event overlap and compute the minimum capacity required to support the interactive component.

To support user interaction that involves adding objects to a presentation, it would be necessary to reserve some channel capacity to support anticipated user interaction. The issue here is to determine the minimum channel capacity required to support user events. It is necessary to compute the capacity required to complete each event separately for cases where only some of the events are supported because of resource constraints.

5.3 The MinC-I Algorithm

The algorithm MinC-I computes the minimum capacity necessary for the interactive component of a presentation. The algorithm first computes the minimum capacity required for each of the events separately using the MinC algorithm. In the next step, the algorithm finds non-overlapping time segments and the capacity required for each of the segments. The event extents are *linearized* into a sequence of start-times and end-times of segments in a non-decreasing order. The set I contains 3-tuples, each formed with a start-time or an end-time, a boolean value indicating if the time is an end-time, and the channel capacity required for the event the 3-tuple belongs to. The elements of I are now 3-tuples <T, B, C>, where T is the start time or end time, the boolean B is true if the time is an end time of a an event, and C is the capacity required for that event. SEG is a set of 3-tuples, each 3-tuple representing a time segment with start-time, end-time and channel capacity for that segment. < T_S, T_E, C>. In each iteration, the next non-overlapping segment and the channel capacity required are computed. Since the segments are non-overlapping, the largest capacity segment gives the maximum capacity required to support the interactive presentation. The operation of the algorithm is illustrated in the example following the algorithm.

S is the set of 4-tuples representing the interactive component of a presentation resulting in object additions

$S(i)$ is the ith 4-tuple of the set.

```
BEGIN
        // find the min C required for each event
        $while(S \neq \emptyset)\{$
                MinC ($S(i)$);
                $S \ -= S(i)$;
                $i$++;
        }
```

```
//for every non-overlapping segments on the channel
//find SEG(i).<st, et, C>
$N = |I|;$
$C_{prev} = 0;$
$C_{\max} = 0;$
$for\ (i = 0; i < N; i++)\ \{$
        $if\ (i \neq 0)\{$
          $SEG(i-1).et\ =\ I(i).t;$

        $SEG(i).st\ =\ I(i).t;$
        $if\ (I(i).et)\{$
          $C_{prev} -= I(i).C;$
          $SEG(i).C\ = C_{prev};$
         $else\{$
          $SEG(i).C\ = C_{prev} + I(i).C;$

        $C_{prev} = SEG(i).C;$
        $if\ (C_{\max} < SEG(i).C)$
           $C_{\max} = SEG(i).C;$

$\}$
END
```

EXAMPLE: The example shows 5 event extents (start-time until the latest possible end time). The capacity required for each of the events is C_i, $1 \le i \le 5$. The start-time and end-time of the events are st_i, et_i, $1 \le i \le 5$.

$$I = \{<st_1, F, C_1>, <st_2, F, C_2>, <et_1, T, C_1>, <et_2, T, C_2>, <st_3, F, C_3>, \\ <st_4, F, C_4>, <et_4, T, C_4>, <et_3, T, C_3>, <st_5, F, C_5>, <et_5, T, C_5>\}$$

The table shows the segments and the channel capacity for each segment. Note that the algorithm outputs spurious segments of zero length, segments with same start-time and end-time (marked with an * in the table). Such segments are discarded.

Table 5.1: Example of the MinC-I Algorithm

SEG(i)	ST	ET	C
1	st_1	st_2	C_1
2	st_2	et_1	$C_1 + C_2$
3	et_1	et_2	$^*C_2$
4	et_2	st_3	*0*
5	st_3	st_4	$^*C_3$
6	st_4	et_4	$C_3 + C_4$
7	et_4	et_3	C_3
8	et_3	st_5	**0*
9	$st5_1$	et_5	C_5

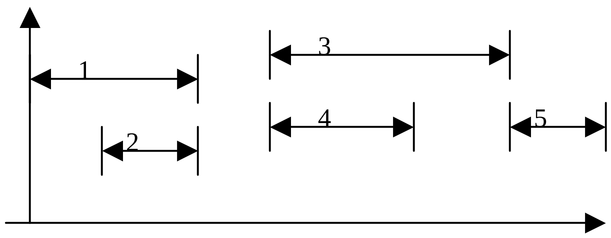

Figure 5.4. Interactive components of a presentation showing the event extents and event numbers

5.4 Dynamic Scheduling and Resource Re-negotiation

When objects are added to or removed from a presentation, the channel capacity needed for the presentation changes. For additions, additional capacity should be acquired and for deletions channel capacity can be released. Additional channel capacity may not be available when an object is added to a presentation. In order to guarantee the support for interactive presentations, additional channel capacity required for added objects should be reserved at the beginning of the presentation. However, for objects that are deleted form a presentation, excess capacity can be released upon user interaction. The capacity to be released depends on the structure of the objects in the presentation and the structure of the presentation itself.

Theorem 5: $C \leq \sum_j C_j \quad 1 \leq j \leq N$ *, i.e., the capacity required to schedule a set of objects together is at most equal to the sum of the capacities required to schedule the objects individually.*

Proof: Consider the worst case, $C = \sum_j C_j$ *, this is apparently true since C can be split into N separate channels to schedule the presentation. When the objects are scheduled together any* gaps *on the channel when an object is scheduled individually may be used for scheduling other objects thus reducing the capacity required when scheduling the objects together. This result is useful in releasing resources when objects are removed upon user interaction.*

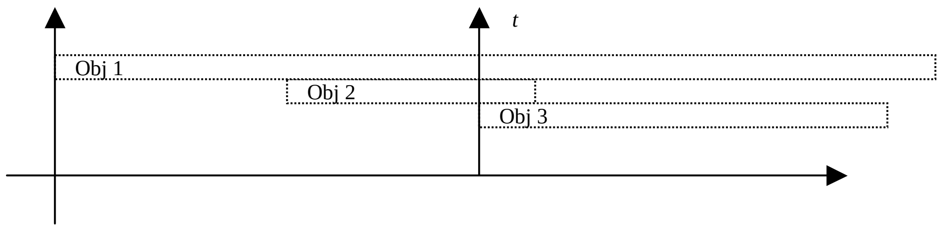

Figure 5.5. Overlapping events

This result cannot be used to release resources in generic object-based presentations. Specifically, this result is useful only when the objects involved are periodic audio-visual objects (such as audio and video). Consider the case depicted in Figure 5.5. Suppose that the structure of object 1 is such that, it does not have any access units to be scheduled during the period when object 2 and object 3 overlap. During that period, only objects 2 and 3 occupy the channel. The resulting presentation has a structure that results in a channel capacity roughly equal to the capacity required to deliver 2 objects. Further suppose that user interaction results in the removal of object 2 at time *t*. Upon such a user event, the capacity cannot be reduced by an amount corresponding to object 2 as it will render the remaining presentation un-schedulable.

When objects are removed from a presentation, the channel capacity cannot always be reduced by releasing resources. The way resources are consumed is completely dependent on the structure of the objects and time-relationships among objects. Theorem 5 can be applied to reduce the reserved channel capacity only when the objects involved are continuous and periodic media streams. A server's response to user interaction, in terms of renegotiating the channel capacity, thus becomes content dependent.

For continuous media objects that are removed as a result of user interaction, the unused channel capacity can be released. The send-time of the remaining AUs should be updated to reflect the new reduced channel capacity. Algorithm *AdjustSchedule*, given below, re-computes the send time to reflect the changes.

5.5 Algorithm AdjustSchedule

Object *k* is removed from the presentation because of user interaction.

C = current channel capacity

C_k = channel capacity required to schedule object *k*.

$C^{'} = C - C_k$ = reduced channel capacity

n is the total number of AUs (from all objects) in the presentation. The AU are indexed in the increasing order of their send times.

m is the index of the first AU to be scheduled after removing object *k*.

```
BEGIN
        $T_n^s = T_n^d - S_n / C^'$
        $while(n \neq m)\{$
           $if\,(T_{n-1}^s \notin AU(k,\cdot)\{$
              $T_{n-1}^s = \max\{T_n^d + S_n / C^', T_{n-1}^d\} - S_{n-1} / C^';$
           $\}else\{$
              $remove\ the\ AU\ from\ the\ schedule;$
           $\}$
           $n_{--};$

END
```

Starting with the last AU to be scheduled, the algorithm adjusts the send time of the AUs. Since the AUs are already scheduled in the order of their decoding times, we traverse the schedule from the last to the current AU. If the AU belongs to the deleted object, it is removed from the schedule.

5.6 Content Creation

Creating object-based audio-visual presentations is not as straightforward as creating MPEG-2 content for TV broadcasting. The features of object-based presentations allow the creation of content that varies in complexity. The content can be as simple as multiplexed audio and video (e.g., MPEG-2 audio and video multiplex) or as complex as a presentation composed of a large number of objects of different types with dynamically changing scenes and user interaction. The complexity of object-based presentations cannot be characterized by a single bitrate. The complexity of a presentation depends on the number of objects in the presentation, the type of objects, and dynamic object addition and deletion due to user interactivity.

One important consideration during content creation is the suitability of content for delivery over networks with different capacities and to terminals with different resources. During the content creation process, authors should specify alternative representations for objects so that the servers can deliver appropriate objects based on feedback from the terminals and the network. Determining the alternative representations for presentations dynamically is a difficult problem especially when considering both terminal and network resources. Content creators need the feedback from the scheduler during authoring in order to specify alternative representations for objects and/or their composition in order to make the presentation schedulable under different resource constraints. An un-schedulable presentation can be made schedulable by removing/replacing certain objects or by decreasing an object's size (e.g., by decreasing the resolution) or by changing the object playout, i.e., the times when an object enters and leaves a scene. Assigning priorities to objects and access units for servers to drop certain objects and/or access units to deliver the presentation under resource constraints. The content creator is in the best position to assign priorities so that the integrity of the presentation does not suffer when objects are dropped.

When creating interactive presentations, the core and interactive components should be created as independent components with interactive component only enhancing the core component. This *separable* design of interactive presentations does not force users to interact with the presentation. Furthermore, when interactive component cannot be delivered because of low network or terminal resources, the presentation will still be meaningful to the user.

5.7 Conclusion

We presented the problem of scheduling interactive audio-visual presentations under resource constraints and discussed the issues in delivering such presentations. User interaction may result in asynchronous or synchronous events. The events that affect the delivery are those events that add or delete objects in a scene. While adding requires more resources, both network and server resources, deleting objects reduces resource consumption. Since it may not be possible to acquire resources during a session, resources must be reserved during session setup to support user interaction that results in object addition. We introduced a notation to represent interactive presentations in terms of its core and interactive components. This allows us to determine the schedulability of core and interactive components separately. For the interactive component, we compute the resources required to complete each event. In the face of resource scarcity, only the most important events or the events that can be supported with the available auxiliary capacity can be supported. We presented algorithms to compute the additional capacity required for interactive events that result in object addition and to compute the amount of resources that can be released when objects are deleted.

Chapter 6

Conclusions and Future Work

We presented our work in the area of audio-visual communications. Our contributions follow the natural evolution of audio-visual services from delivering digital audio-visual content to delivering object-based audio-visual services. Our initial contributions in this area are toward the development of Columbia's VoD testbed. The key contributions made are the design of the application server for VoD services, distributed video pump, application signaling, adapting the system for browser-based clients, and contributions toward the development of DAVIC standards by means of proof of concept implementations and interoperability experiments.

We designed the application server by separating the resource intensive audio-visual content delivery from data and service delivery using. These well-defined interfaces and their implementation using CORBA is essential for localizing bandwidth intensive video traffic to network segments. With streaming media over the Internet becoming more common, traffic localization together with congestion control mechanisms is necessary to prevent congestion collapse in networks. We designed the server interfaces to support different client platforms with the same server. We showed that implementation and experimentation is essential to standardizing systems with a very broad scope such as DAVIC.

The natural extension of our work in audio-visual services is object-based audio-visual services that allow a finer grain of control on the delivery and presentation of audio-visual content. We contributed to the development of MPEG-4 Systems which specifies tools for representing object-based audio-visual presentations. Our contributions to the development of the MPEG-4 Systems standard are in the areas of bitstream design, terminal architecture, user interaction framework, and file format. We designed the bitstream architecture based on the premise of separating meta-data from the media data and hierarchical representation of object-based presentations. The bitstream is also designed to dynamically update the scenes using composition updates, node addition, and node deletion. We proposed the original architecture for user interaction and file format in MPEG-4 Systems. Even though the final form of these components in the MPEG-4 standard differ from the proposed versions, the contributions formed the underlying basis and helped in the final definition of the MPEG-4 Systems specification.

We continued our work on object-based audio-visual services by considering the implications of object-based representation on scheduling and delivery over networks. The structure and nature of an MPEG-4 presentation determines the complexity of the content both for delivery and presentation. We presented the problem of scheduling audio-visual objects under resource constraints. We showed that the problem is NP-complete in the strong sense. We explored similarities with the problem of sequencing jobs on a single machine and used the idea of last-to-first scheduling to develop heuristic algorithms to determine schedulability and compute startup delay-optimal schedules. We introduced the notion of residual data volume to compute lower bounds on buffer, channel capacity and startup delay. Determining the schedulability of presentations online is important for applications like content creation where an additional object may make the presentation un-schedulable. We presented an algorithm that computes incremental schedules and produces a startup delay optimal schedule. Starting with a lower bound on the channel capacity for scheduling a presentation, the MinC algorithm minimizes the CBR channel capacity required to schedule the presentation. The proposed algorithms are of low complexity and can be implemented efficiently.

We also discussed the issues in delivering interactive audio-visual presentations. User interaction may result in asynchronous or synchronous events. The events that affect the delivery are those events that add or delete objects in a scene. While adding requires more resources (both network and server), deleting objects reduces resource consumption. Since it may not be possible to acquire resources during a session, resources must be reserved during session setup to support user interaction that results in object addition. We introduced a notation to represent interactive presentations in terms of their core and interactive components. This allows us to determine the schedulability of core and interactive components separately. For the interactive component, we compute the resources required to complete each event. In the face of resource scarcity, only the most important events or the events that can be supported with the available auxiliary capacity can be supported. We presented algorithms to compute the additional capacity required for interactive events that result in object addition and to compute the amount of resources that can be released when objects are deleted.

The natural extension of this work is delivering object-based presentations to terminals with varying resources (computational and bandwidth resources). The MPEG-4 Systems framework has a simple way for a presentation to query terminal resources. The problem of determining an alternative representation for object-based presentations is very complex. The alternative representations depend on the type of the content, type (encoding) of the objects in the content, and the resource availability at the terminal. The difficult part is finding a representation that does not compromise the integrity of the content. Content creators can provide general guidelines on scaling the content and prioritizing the objects in the content. Such guidelines would be helpful but cannot be exhaustive and cannot cover all the terminal and resource constraint scenarios. Determining alternative representations dynamically based on resource availability is a difficult problem but it also has valuable applications.

As object-based content moves to resource constrained specialized devices such as set-top-boxes and hand-held/mobile devices, resource management on the terminal becomes critical. The resource management policies would depend on the type and structure of the content and the device playing back the content. Functionality provided by the frameworks such as MPEG-J, the Java extensions to MPEG-4 Systems, are required for resource management. With these features, the content delivered to a terminal would now include a Java applet programmed to manage resources during content playback.

References

[1] www.iso.ch, International Organization for Standardization.

[2] OMG – Object Management Group, “Common Object Request Broker: Architecture and Specification,” CORBA Revision 2.0 (July 1995) (OMG CORBA 2.0)

[3] ISO/IEC/SC29/WG11, “Call for Proposals for the MPEG-4 Intermedia Format,” N1919, International Organization for Standardization, October 1997.

[4] ISO/IEC/SC29/WG11, “Delivery Multimedia Integration Framework, DMIF (ISO/IEC 14496-6),” International Organization for Standardization, April 1999.

[5] ISO/IEC/SC29/WG11, “Generic Coding of Moving Pictures and Associated Audio (MPEG-1 Video) ISO/IEC 11172-2,” International Organization for Standardization, November 1991.

[6] ISO/IEC/SC29/WG11, “Generic Coding of Moving Pictures and Associated Audio (MPEG-1 Audio) ISO/IEC 11172-3,” International Organization for Standardization, November 1991.

[7] ISO/IEC/SC29/WG11, “Generic Coding of Moving Pictures and Associated Audio (MPEG-1 Systems) ISO/IEC 11172-1,” International Organization for Standardization, November 1991.

[8] ISO/IEC/SC29/WG11, “Generic Coding of Moving Pictures and Associated Audio (MPEG-4 Systems) - ISO/IEC 14386-1,” International Organization for Standardization, April 1999.

[9] ISO/IEC/SC29/WG11, “Generic Coding of Moving Pictures and Associated Audio (MPEG-4 Video) - ISO/IEC 14386-2,” International Organization for Standardization, April 1999.

[10] ISO/IEC/SC29/WG11, “Generic Coding of Moving Pictures and Associated Audio (MPEG-2 Video) ISO/IEC 13818-2,” International Organization for Standardization, November 1994.

[11] ISO/IEC/SC29/WG11, “Generic Coding of Moving Pictures and Associated Audio (MPEG-2 Audio) ISO/IEC 13818-3,” International Organization for Standardization, November 1994.

[12] ISO/IEC/SC29/WG11, “Generic Coding of Moving Pictures and Associated Audio (MPEG-2 Systems) ISO/IEC 13818-1,” International Organization for Standardization, November 1994.

[13] ISO/IEC/SC29/WG11, “Generic Coding of Moving Pictures and Associated Audio (DSM-CC)- ISO/IEC 13818-6,” International Organization for Standardization, 1996.

[14] ITU-T, “ITU Recommendation M.3010: Principles for a Telecommunications Management Network”.

[15] ISO/IEC/SC29/WG11, "Requirements for the MPEG-4 Intermedia Format," N1886, International Organization for Standardization, October, 1997.

[16] ISO/IEC/SC29/WG11, "Text of ISO/IEC 13818-1/PDAM7," MPEG document N2664, March 1999.

[17] ISO/IEC/SC29/WG11, "Verification Model, MPEG-4 Version 2," ISO/IEC/SC29/WG11 MPEG98/N2224, International Organization for Standardization, March 1998.

[18] ISO/IEC/SC29/WG11, "Working Draft, MPEG-4 Version 2," ISO/IEC/SC29/WG11 MPEG98/N2211, International Organization for Standardization, March 1998.

[19] A. Akhtar, H. Kalva, and A. Eleftheriadis, "Implementation of CommandDescriptor and CommandDescriptorNode", Contribution ISO-IEC JTC1/SC29/WG11 MGE99/4431, March 1999, Seoul, Korea (47th MPEG meeting).

[20] 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).

[21] A. Basso et. al., "MPEG-4 Integrated Intermedia Format (IIF): Basic Specification," ISO/IEC/SC29/WG11 MPEG98/M2978, International Organization for Standardization, February 1998.

[22] A. Basso et. al., "MPEG-4 Integrated Intermedia Format (IIF): Extension Specification," ISO/IEC/SC29/WG11 MPEG98/M2979, International Organization for Standardization, February 1998.

[23] A. Basso, H. Kalva, A. Puri, A. Eleftheriadis, and R. L. Schmidt, "The MPEG-4 File Format: An Advanced Multifunctional Standard for New Generation Multimedia Content", IEEE Trans. on Circuits and Systems for Video Technology, 1999 (to appear).

[24] C. Blank, "The FSN challenge: Large-scale interactive television," IEEE Computer, Vol. 28, No. 5, May 1995, pp. 9-12.

[25] M. C. Buchanan and P. T. Zellweger, "Scheduling Multimedia Documents Using Temporal Constraints," NOSDAV 92, pp. 223-235.

[26] A.T. Campbell and G. Coulson, "A QOS Adaptive Multimedia Transport System: Design, Implementation and Experiences", Distributed Systems Engineering Journal, Special Issue on Quality of Service, Vol. 4, pg. 48-58, April 1997.

[27] A.T. Campbell, G. Coulson, and D. Hutchison, "Transporting QoS Adaptive Flows", ACM/Springer Verlag Multimedia Systems Journal , Special Issue on QoS Architecture, Vol. 6 No. 3, pg. 167-178, May 1998.

[28] J. Carey, "Interactive Television Trails and Marketplace Experiences," Multimedia Tools and Applications, Special Issue On Video-on-Demand, Trials, and Interoperability, Vol. 5, No. 2, Sept. 1997, pp. 207-216.

[29] J. Carlier, “The One Machine Sequencing Problem,” European Journal of Operational Research, No. 11, 1982, pp. 42-47.

[30] T.L. Casvant and J.G. Kuhl, “A Taxonomy of Scheduling in General Purpose Distributed Computing Systems,” IEEE Transactions on Software Engineering, Vol. 14, No. 2, Feb 1988, pp. 141-154.

[31] Y.-H. Chang *et. al.*, “An open systems approach to video on demand,” IEEE Communications Magazine, Vol 32, No. 5, May 1994, pp. 68-80.

[32] S.-F. Chang, A. Eleftheriadis, D. Anastassiou, S. Jacobs, H. Kalva, and J. Zamora, “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.

[33] S.-F. Chang, A. Eleftheriadis, and D. Anastassiou, “Development of Columbia's Video on Demand Testbed”, Signal Processing: Image Communication, Special Issue on Video on Demand and Interactive Television, Vol. 8, Nr. 3, April 1996, pp. 191-207.

[34] S.-F. Chang, Q. Huang, T. Huang, A. Puri, and B. Shahraray, “Multimedia Search and retrieval,” in Advances in Multimedia: Systems, Standards, and Networks, A. Puri and T. Chen (eds.). New York: Marcel Dekker, in press, 1999.

[35] P. Chou et. al., “The MPEG-4 Intermedia Format (MIF) as an Extension of ASF,” ISO/IEC/SC29/WG11 MPEG98/M2969, International Organization for Standardization, February 1998.

[36] H.A. Chow, H. Alnuweiri “An FPGA-based transformable coprocessor for MPEG video processing,” SPIE-Int. Soc. Opt. Eng. Proceedings of SPIE - the International Society for Optical Engineering, vol.2914, 1996, pp.308-20. USA.

[37] P. Chretienne, E.G. Coffman Jr., J.K. Lenstra, and Z. Liu, *editors*, “Scheduling Theory and its Applications,” John Wiley & Sons, 1995.

[38] R. Civanlar, A. Basso, and C. Herpel, “RTP Payload Format for MPEG-4 Streams,” draft-ietf-avt-rtp-mpeg4-01.txt, Internet Draft, IETF, February 1999.

[39] M. Civanlar, G. Cash, and B. Haskell, “RTP Payload Format for Bundled MPEG,” RFC 2343, The IETF, May 1998.

[40] Apple Computer , “Bento Specification,” Bento Specification Revision 1.0d5, July 15, 1993.

[41] Apple Computer Corp, “Quicktime Specification,” May 1996.

[42] R.W. Conway, W.L. Maxwell, and L.W. Miller, “Theory of Scheduling,” Addison-Wesley Publishing Company, 1967.

[43] Microsoft Corporation, “ASF Specification,” February 1998.

[44] D. Crocker, “To be on the Internet,” RFC 1775, IETF, March 1995.

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

[46] The Digital Audio-Visual Council (DAVIC), "DAVIC 1.2 Specifications," DAVIC 1.2 specification, Geneva, Switzerland, December 1996.

[47] The Digital Audio-Visual Council (DAVIC), "DAVIC 1.3 Specifications," DAVIC 1.3 specification, Geneva, Switzerland, December 1997.

[48] The Digital Audio-Visual Council (DAVIC), "Part 1 – Description of davic Functionalities," DAVIC 1.0 specification, Geneva, Switzerland, 1995.

[49] The Digital Audio-Visual Council (DAVIC), "Part 9 – Information Representation," DAVIC 1.0 specification, Geneva, Switzerland, January 1996.

[50] The Digital Audio-Visual Council (DAVIC), "Part 9 – Information Representation," DAVIC 1.1 specification, Geneva, Switzerland, November 1996.

[51] The Digital Audio-Visual Council (DAVIC), "Part 3 – Service Provider System Architecture and Interfaces," DAVIC 1.1 specification, Geneva, Switzerland, November 1996.

[52] A. Eleftheriadis and H. Kalva, "A Proposed Architecture for an Object-Based Audio-Visual Bitstream and Terminal", Contribution ISO-IEC JTC1/SC29/WG11 MPEG97/1619, February 1997, Seville, Spain (38th MPEG meeting).

[53] A. Eleftheriadis, H. Kalva, A. Puri, and R. Schmidt, "Stored File Format for MPEG-4 (Rev. 2.0)", Contribution ISO-IEC JTC1/SC29/WG11 MPEG97/2536, July 1997, Stockholm, Sweden (40th MPEG meeting).

[54] A. Eleftheriadis, "Architecting Video-on-Demand Systems: DAVIC 1.0 and Beyond," Proceedings, International Symposium on Multimedia Communications and Video Coding, Brooklyn, New York, October 1995.

[55] M.L. Escobar-Molano, "Management of Resources to Support Coordinated Display of Structured Presentations," Ph.D. Dissertation, Graduate School, University of Southern California, 1996.

[56] B. Furht and H. Kalva, "Multimedia Networks," Multimedia Systems and Techniques, B. Furht, Ed., Kluwer Academic Publishers, 1996.

[57] B. Furht and H. Kalva, "Network Architectures for Interactive Television," Third orsa (now informs) Telecommunications Conference, Boca Raton, Florida, 20-22 March 1995.

[58] J. Greenbaum, and M. Baxter, "Increased FPGA capacity enables scalable, flexible CCMs: an example from image processing," Proceedings of the 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 1997, pp.211-17.

[59] L. Grossglauser and S. Keshav, "On CBR Service," Proceedings of the INFOCOM, March 1996.

[60] H.M. Vin and P.V. Rangan, "Designing a multi-user HDTV Storage Server," IEEE Journal on Selected Areas in Communications, Vol. 11, No. 1, 1993.

[61] L.A. Hall, "Approximation Algorithms for Scheduling," Approximation algorithms for NP-hard problems," D. S. Hochbaum edts., PWS Pub. Co., c1997, pp. 1-45.

[62] G.G. Hartwick, "From Interactive Television to Internet Applications," Multimedia Tools and Applications, Special Issue On Video-on-Demand, Trials, and Interoperability, Vol. 5, No. 2, Sept. 1997, pp. 217-222.

[63] B. Haskell, A. Puri, and A. Netravali, "Digital Video: An Introduction to MPEG-2," Chapman and Hall, 1997.

[64] C. Herpel and A. Eleftheriadis, "MPEG-4 Systems: Elementary Stream Management", Signal Processing: Image Communication, Special Issue on MPEG-4, 1999 (to appear).

[65] P. Hoddie et. al., "Quicktime File Format as the Basis for MPEG-4 Intermedia Format," ISO/IEC/SC29/WG11 MPEG98/M2980, International Organization for Standardization, February 1998.

[66] D. Hoffman *et. al.*, "RTP Payload Format for MPEG1/MPEG2 Video," RFC 2050, Networks Working Group, The IETF, 1998.

[67] J.-F. Huard and A. A. Lazar, "A Programmable Transport Architecture with QOS Guarantees," IEEE Communications Magazine, Vol. 36, No. 10, October 1998, pp. 54-62.

[68] J.-F. Huard, A. A. Lazar, K.-S. Lim and G. S. Tselikis, "Realizing the MPEG-4 Multimedia Delivery Framework," IEEE Networks, Vol 12, No 6, 1998, pp 35-45.

[69] www.itu.int, International Telecommunications Union.

[70] S. Jacobs and A. Eleftheriadis, "Streaming Video using TCP Flow Control and Dynamic Rate Shaping", Journal of Visual Communication and Image Representation, Special Issue on Image Technology for Word-Wide-Web Applications, Vol. 9, No. 3, September 1998, pp. 211-222

[71] S. Jacobs and A. Eleftheriadis, "Streaming Video using TCP Flow Control and Dynamic Rate Shaping", Journal of Visual Communication and Image Representation, Special Issue on Image Technology for Word-Wide-Web Applications, Vol. 9, No. 3, September 1998, pp. 211-222.

[72] C.-C.Han, K.-J.Lin, J.W.-S. Liu, "Scheduling Jobs with Temporal Distance Constraints," SIAM Journal on Computing, Vol. 24, N0. 5, October 1995, pp. 1104-1121.

[73] H. Kalva and A. Eleftheriadis, "Delivering MPEG-4 Content", Packet Video 99, New York, NY, April 26-27 1999.

[74] H. Kalva and A. Eleftheriadis, “MPEG-4 Interaction Model and Required Normative Support”, Contribution ISO-IEC JTC1/SC29/WG11 MPEG97/2888, October 1997, Fribourg, Switzerland (41st MPEG meeting).

[75] H. Kalva and A. Eleftheriadis, “Requirements for MPEG-4 File Format”, Contribution ISO-IEC JTC1/SC29/WG11 MPEG97/2886, October 1997, Fribourg, Switzerland (41st MPEG meeting).

[76] H. Kalva and A. Eleftheriadis, “Software Implementation of the MPEG-4 Intermedia Format Proposal from Columbia University and AT&T Research,” Contribution ISO-IEC JTC1/SC29/WG11 MPEG98/3190, February 1998, San Jose, CA (42nd MPEG meeting).

[77] H. Kalva and A. Eleftheriadis, “Syntax and Semantics of Control Messages for User Interaction”, Contribution ISO-IEC JTC1/SC29/WG11 MPEG98/3189, February 1998, San Jose, CA (42nd MPEG meeting).

[78] H. Kalva and A. Eleftheriadis, “Use of CommandDescriptors and CommandROUTES to Support User Interaction “, Contribution ISO-IEC JTC1/SC29/WG11 MPEG98/M3506, March 1998, Tokyo, Japan (43rd MPEG meeting).

[79] H. Kalva and A. Eleftheriadis, “Using Command Descriptors”, Contribution ISO-IEC JTC1/SC29/WG11 MPEG98/4269, December 1998, Rome, Italy (46th MPEG meeting).

[80] H. Kalva and B. Furht, “Techniques for Improving the Capacity of Video-on-Demand Systems,” Proceedings of the 29th Hawaii International Conference on System Sciences (HICSS-29*)*, Vol 2, Jan 3-6 1996.

[81] H. Kalva, *et. al.*, “Implementing Multiplexing, Streaming and Server Interaction for MPEG-4,” IEEE Trans. on Circuits and Systems for Video Technology, Special Issue on Object Based Video and Description, Vol. 9, No. 8, Dec 1999, pp. 1299-1312.

[82] H. Kalva, S.-F. Chang, and A. Eleftheriadis, “DAVIC and Interoperability Experiments”, Journal on Multimedia Tools and Applications, Special Issue on Video on Demand, Kluwer Academic Publishers, Vol. 5, Nr. 2, September 1997, pp. 119-132.

[83] H. Kalva, L-T. Cheok, A. Eleftheriadis, “MPEG-4 Systems and Applications,” Demonstration, ACM Multimedia '99, Orlando, FL.

[84] H. Kalva, A. Eleftheriadis, A. Basso, R. Schmidt, and A. Puri, “File Format for MPEG-4 (Rev. 3.0)”, Contribution ISO-IEC JTC1/SC29/WG11 MPEG97/2873, October 1997, Fribourg, Switzerland (41st MPEG meeting).

[85] H. Kalva, A. Eleftheriadis, A. Puri, and R. Schmidt, “Stored File Format for MPEG-4”, Contribution ISO-IEC JTC1/SC29/WG11 MPEG97/2062, April 1997, Bristol, UK (39th MPEG meeting).

[86] H. Kalva, A. Eleftheriadis, and S.-F. Chang, “Columbia's Video on Demand Testbed”, Proceedings, International Conference on Communications, Montreal, Canada, June 1997 (invited presentation).

[87] H. Kalva, “DAVIC New York Interoperability Experiments: Report and Results,” DAVIC Contribution No. DAVIC/TC/SYS/96/09/008, Geneva, Switzerland, 1996.

[88] H. Kalva, H. Okuda, A. Eleftheriadis, and S.-F. Chang, “DAVIC Standard for Multimedia Applications,” Handbook of Multimedia Computing, B. Furht, Ed., CRC Press, 1998.

[89] H. Kalva, J. Zamora, and A. Eleftheriadis, “Delivering Object-based Audio-Visual Services,” International Conference on Consumer Electronics, Los Angeles, CA, June 22-24 1999 (invited presentation).

[90] H. Kasahara, “Report of the DAVIC Interoperability Event in Tokyo Electronics Show ’96,” DAVIC Document, DAVIC/358, Hong Kong, December 1996.

[91] M. Kim, P. Westerink, and W. Belknap, “MPEG-4 Advanced Synchronization Model (FlexTime Model),” Contribution ISO-IEC JTC1/SC29/WG11 MPEG99/5307, December 1999, (50th MPEG meeting).

[92] L. Kleinrock and A. Nilsson, “On Optimal Scheduling Algorithms for Time-Shared Systems,” Journal of the ACM, Vol. 28, No. 3, July 1981, pp. 477-486.

[93] R. Koenen, “MPEG-4: Multimedia for our time,” IEEE Spectrum, Vol. 36, No. 2, pp. 26-33, February 1999.

[94] M. Kunt, A. Ikonomopoulos, M. Kocher, “Second-generation image coding techniques,” IEEE Proceedings, Vol. 73, No. 4, pp. 549–574, April 1985.

[95] E.L. Lawler, “A Functional Equation and its Application to Resource Allocation and Sequencing Problems,” Management Science, Vol. 16, No. 1, September 1969, pp. 77-84.

[96] E.L. Lawler, “Optimal Sequencing of a Single Machine Subject to Precedence Constraints,” Management Science, Vol. 19, No. 5, January 1973, pp. 544-546.

[97] A. Lazar, “Programming Telecommunication Networks,” IEEE Network, pp. 8-18, September 1997.

[98] J.K. Lenstra, A.H.G.R. Kan, and P. Brucker, “Complexity of Machine Scheduling Problems,” Annals of Discrete Mathematics 1, 1977, pp. 343-362.

[99] Z. Lifshitz, “APIs for System Software Implementation,” Contribution no. ISO/IEC JTC1/SC29 MPEG97/M3111.

[100] T.D.C. Little and A. Ghafoor, “Multimedia Synchronization Protocols for Broadband Integrated Services,” IEEE Journal on Selected Areas in Communications, Vol 9, No. 9, Dec 1991, pp. 1368-1382.

[101] T.D.C. Little and A. Ghafoor, “Synchronization and Storage Models for Multimedia Objects,” IEEE Journal on Selected Areas in Communications, Vol 8, No. 3, Dec 1990, pp. 413-427.

[102] M. Milenkovic, “Delivering Interactive Services to Home Using Digital Video Broadcasting Infrastructure,” IEEE Multimedia v 5 n 4 Oct-Dec 1998, p 34-43.

[103] H. Okuda, M. Morinaga, H. Kasahara, and K. Shimamura, “An Interoperability Testbed and Test Results for DAVIC 1.0 specification”, Journal on Multimedia Tools and Applications, Special Issue on Video on Demand, Kluwer Academic Publishers, Vol. 5, Nr. 2, September 1997, pp. 147-160.

[104] S. Paek and S.F. Chang, “Video Server Retrieval Scheduling and Resource Reservation for Variable bit Rate Scalable Video,” To be published in IEEE Transactions on Circuits and Systems for Video Technology.

[105] S. Paek, P. Bocheck, S-F. Chang, “Scalable MPEG2 Video Servers With Heterogeneous QoS on Parallel Disk Arrays,” 5th IEEE Workshop on Network and Operating System Support for Digital Audio & Video. New Hampshire, April 1995.

[106] 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).

[107] M. R. Garey and D. S. Johnson, “Computers and intractability: a guide to the theory of NP-completeness,” W. H. Freeman, San Francisco, 1979.

[108] N. S. Jayant, “Signal compression: Technology targets and research directions,” IEEE Journal on Selected Areas in Communications, Special issue on speech and image coding, June 1992.

[109] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications, “ Internet Engineering Task Force, Jul. 1999.

[110] J. Signes, Y. Fisher, and A. Eleftheriadis, "BIFS Technical Description", Multimedia Systems, Standard, Networks, A. Puri and T. Chen, Editors, Marcel Dekker, 1999 (to appear).

[111] J. Song, A. Dan, and D. Sitaram, “Efficient Retrieval of Composite Multimedia Objects in JINSIL Distributed System,” ACM SIGMETRICS, June 1997.

[112] ITU-T Study Group 16, “Recommendation H.245: Control protocol for Multimedia Communication,” ITU, 1997.

[113] H. Sundaram, S.F. Chang “Efficient Video Sequence Retrieval in Large Repositories,” Proc. SPIE Storage and Retrieval for Image and Video Databases VII, San Jose CA, Jan 23-29 1999.

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

[115] D. Trietsch, “Scheduling Flights at Hub Airports,” Transportation Research, Vol 27, No. 2, 1993, pp 133-150.

[116] J.D. Ullman, “NP-Complete Scheduling Problems,” Journal of Computer and System Sciences, No. 10, 1975, pp. 384-393. O. Avaro, P. Chou, A. Eleftheriadis, C. Herpel, and C. Reader, “The MPEG-4 System and Description Languages: A Way Ahead in Audio Visual Information Representation”, Signal Processing: Image Communication, Special Issue on MPEG-4, Vol. 9, No. 4, May 1997, pp. 385-431.

[117] K. Y. Yu, J-H. Lee, and J-H. Park, “Design of audio processing unit for multipoint video conferencing system,” Proceedings of the ICT '98. International Conference on Telecommunications. vol.1, 1998, pp.241-5.

[118] J. Zamora, “Cell Delay Variation Performance of CBR and VBR MPEG-2 Sources in an ATM Multiplexer,” in Proceedings VIII European Signal Processing Conference, Trieste, Italy, September 10-13, 1996.

[119] J. Zamora, S. Jacobs, A. Eleftheriadis, S.-F. Chang, and D. Anastassiou, “A Practical Methodology for Guaranteeing QoS for Video on Demand”, IEEE Trans. on Circuits and Systems for Video Technology, 1999 (to appear).

[120] J. Zamora, “Video-on-Demand Systems and Broadband Networks: Quality of Service Issues ,” PhD thesis, Columbia University, New York, NY, 1998.