INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Université d'Ottawa · University of Ottawa

Synchronization Control of Coded Video Streams: Algorithms and Implementation

By

Mourad Daami, B.A.Sc.

A Thesis submitted to the
School of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

MASTER OF APPLIED SCIENCE

Ottawa-Carleton Institute of Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
University of Ottawa

May 1997

© Mourad Daami, Ottawa, Canada, 1997

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-26314-2

Canada

Abstract

One of the key design issues in distributed multimedia systems is synchronization. Multimedia synchronization can be disrupted by a number of factors. The transmission of various data streams, with different traffic characteristics and possibly from multiple servers over a network introduces inevitable random delays and delay variations both within and among different streams. The processing loads and context switching overheads on the servers and the clients introduce further disturbances. To compensate for these disruptions, MCRLab adopted a two tiered approach. First, a prevention mechanism pre-computes delivery schedules for the different media using negotiated Quality of Service (QoS) parameters. Second, an error recovery mechanism known as the Stream Synchronization Protocol (SSP) maintains synchronized playback at the client. Within the SSP, special attention is given to time-dependent coded video streams due to the presence of a hardware/software decoding entity and special characteristics of coded time-dependent video streams. A Predecoder Synchronization Controller (PSC) is needed to protect from buffer overflow and underflow conditions and maintain stream continuity at the client.

In this thesis, we propose to blend the intra-media synchronization control of coded video streams with the decoding process. We present a client based application level buffer control scheme for coded video to maintain stream continuity, avoid data overflow and underflow conditions and compensate for possible clock drifts at the client. The control policy, referred to as the PSC, uses buffer occupancy as its main control parameter. Based on high and low buffer occupancy thresholds, overflow threats and starvation threats are detected. Time contraction (selective frame drop) and expansion concepts (selective frame repeat) are used to prevent these data overflow and starvation threats to become real conditions, respectively. PSC algorithms are specified for both Motion JPEG and MPEG. In the case of starvation or overflow threats, the PSC was found to smoothen the stream, avoiding long frame freeze and abrupt scene change situations, by creating localized variations in the rate of displayed stream.

A prototype implementation of both MJPEG and MPEG PSCs was also carried out. For this purpose, a stream structure for transmission from the server to the client was developed. Based on this structure and based on a bounded delay network model, buffering delay (for starvation prevention) and buffer size (for overflow prevention) values were derived. In addition to the stream structure, the prototype implementation showed the impact the nature of the decoder has on the architecture of the synchronization scheme. In the case of a hardware decoder, the PSC had to be combined with the Media Synchronization Controller (MSC) (responsible for inter-stream synchronization) just before the decoder. In the case of a software decoder, the PSC lies before the decoder and the MSC had to be positioned after the decoder.

Acknowledgments

I would like to thank my supervisor, Professor N. D. Georganas, for his continuous support , patience and encouragement during my stay at the Multimedia Communications Research Laboratory at the University of Ottawa.

I also gratefully acknowledge the financial support of the Canadian Institute for Telecommunications Research, the University of Ottawa and my country Tunisia.

I would also like to thank the members of MCRLab for their friendliness and their helpfulness. My special thanks go to, Mr. Jeff Brinskelle for his help with the prototype and the Lab Equipment, Mr. Wayne Robbins, for the insightful scientific and philosophical discussions we had, Mr. Keith Smith, Mr. Michel Ouellett for being outstanding colleagues. A special mention goes to Prof. Moshe Krieger for his continuous moral support and for making my graduate stay an interesting experience.

I would like to express my gratitude towards the support staff of the Electrical Engineering Department for their kindness. In particular, I would like to thank, Mme. Lepage for her assistance and availability to provide me with necessary administrative documents and Mme. Michele Roy for her inexorable sense of humor and help with departmental equipment.

Finally, my deep and sincere gratitude goes to my family for their love, support and patience during my long foreign stay.

Table of Contents

Abstract ii
Acknowledgments iii
Table of Contents iv
List of Figures vii

INTRODUCTION 1
1.1 MULTIMEDIA SYSTEMS 1
1.2 MULTIMEDIA SYNCHRONIZATION 3
1.3 SYNCHRONIZATION CONTROL OF CODED VIDEO STREAMS 5
1.4 THE CITR BROADBAND SERVICES PROJECT 6
1.5 THESIS OUTLINE 9
1.6 PUBLICATIONS 10

MULTIMEDIA SYNCHRONIZATION 11
2.1 MULTIMEDIA SYSTEMS 13
2.1.1 The Application Domain *13*
2.1.2 The System Domain *14*
2.1.3 The Network and Device Domain *14*
2.1.4 The Cross Domain *15*
2.1.5 Multimedia Systems Requirements *16*
2.2 NOTION OF SYNCHRONIZATION 18
2.3 INTER- AND INTRA- MEDIA SYNCHRONIZATION 19
2.3.1 Types of Media *19*
2.3.2 Intra- vs. Inter- Media Synchronization *20*
2.3.3 Live vs. Synthetic Synchronization *21*
2.3.4 Synchronization Granularity *22*
2.4 MULTIMEDIA SYNCHRONIZATION REFERENCE MODEL 23
2.5 SYNCHRONIZATION IN A NETWORKED ENVIRONMENT 25

2.5.1 Synchronization Transport 25
2.5.2 Types and Causes of Synchronization Anomalies 28
2.5.3 Synchronization Strategies 34
2.5.4 Additional Considerations 38
2.6 SYNCHRONIZATION SPECIFICATION 39
2.6.1 Synchronization Specification Methods 40
2.6.2 Synchronization and Quality of Service 46
PREDECODER SYNCHRONIZATION CONTROLLER: ALGORITHMS 54
3.1 A MULTIMEDIA NEWS-ON-DEMAND TELESERVICE 54
3.1.1 The Media Production Center 55
3.1.2 The Media Distribution Center 57
3.1.3 The Communication Subsystem 57
3.1.3 The User End 58
3.2 SYSTEM ARCHITECTURE AND SYNCHRONIZATION MECHANISMS 59
3.2.1 Stream Delivery Scheduling 61
3.2.2 The Stream Synchronization Protocol 64
3.3 PREDECODER SYNCHRONIZATION CONTROLLER 65
3.3.1 Characteristics of Coded Video Streams 67
3.3.2 Need For A PSC 72
3.3.3 Design Objectives 74
3.3.4 Control Parameters 75
3.3.5 MJPEG PSC 78
3.3.6 MPEG PSC 83
PREDECODER SYNCHRONIZATION CONTROLLER: IMPLEMENTATION 89
4.1 HARDWARE AND SOFTWARE SETUP 90
4.2 THE ORIGINAL VERSUS THE MODIFIED PROTOTYPE 91
4.3 STREAM STRUCTURE 95
4.3.1 Buffering Delay For Starvation Threats 97
4.3.2 Buffer Dimensioning For Overflow Threats 98
4.4 MJPEG PSC ARCHITECTURE 101
4.5 MPEG PSC ARCHITECTURE 104

CONCLUSION **110**

REFERENCES **114**

List of Figures

Figure 1.4.1 General Architecture of The CITR NoD Prototype 8
Figure 2.4.1 Synchronization Reference Model 24
Figure 2.6.1 Types of Temporal Relations Between Two Time Intervals 42
Figure 2.6.2 Time-axis Based Synchronization Specification 43
Figure 2.6.3 Types of Transition Nodes 45
Figure 2.6.4 TFG Model of Basic Temporal Interval Relations 46
Figure 2.6.5 QoS Abstraction in a Networked Multimedia System 48
Figure 2.6.6 QoS For Inter-media Synchronization 50
Figure 3.1.1 Architecture of a System Supporting a Multimedia News-on-Demand Teleservice 56
Figure 3.2.1 Synchronization Architecture of MCRLab's NoD Prototype 60
Figure 3.2.2 Delivery Scheduling Operation 62
Figure 3.3.1 Basic Image Encoding Steps 68
Figure 3.3.2 Buffer Occupancy For PSC Buffer 76
Figure 3.3.3 Stream Continuity For Starvation Threats Using PSC 81
Figure 3.3.4 Stream Continuity For Overflow Threats Using PSC 82
Figure 3.3.5 MPEG Video Bit Stream 85
Figure 4.2.1 Client Architecture For Original Prototype 93
Figure 4.2.2 Client Architecture For Modified Prototype 94
Figure 4.3.1 Stream Structure 95
Figure 4.3.2 Worst Case Scenario For Starvation at Client Buffer 97
Figure 4.3.3 Important Time Lines For Client Buffer 99
Figure 4.3.4 Worst Case Scenario For Overflow Threats At Client Buffer 100
Figure 4.4.1 MJPEG PSC Architecture 101
Figure 4.4.2 MJPEG Shared Memory Buffer Structure 103
Figure 4.5.1 MPEG PSC Architecture 105
Figure 4.5.2 Auxiliary Frame Buffer Structure 107

Chapter One

Introduction

1.1 Multimedia systems

With computing power and storage capacity exponentially increasing over the last few years coupled with recent advances in multimedia compression standards [PEN93] and [LEG91], the capability to process non traditional information such as audio and video became a desktop reality. In addition, the deployment of new broadband telecommunications technologies, such as the Asynchronous Transfer Mode (ATM) [LEB92], permits the transfer of such bulky information across computer communication networks. As a result, multimedia systems emerged as the offspring of the ever converging computing and telecommunications worlds. As defined in [STE95.2], multimedia systems are characterized by the integrated computer controlled generation, manipulation, presentation, storage and communication of independent discrete (time-independent) and continuous (time-dependent) media. In a less formal manner, distributed multimedia systems involve two or more computer nodes interconnected

through a broadband communication network capable of exchanging and processing a multitude of media including continuous data streams such as audio and video. Depending on the communication mode, multimedia applications can be divided into three categories:

- Conversational services are carried out in a live mode. Conversational multimedia applications are of interest to the institutional and business communities with such applications as: distance learning, telemedicine, video conferencing, computer supported collaborative work environments and so on.

- Presentational services are carried in real time but do not involve human to human interactions. Presentational multimedia applications can be further divided into: retrieval services such as Video-on-Demand, News-on-Demand, Yellow Pages, Digital Libraries; and broadcast services such as HDTV.

- Messaging services which are multimedia mailing applications based on a store and forward communication mode. A perfect example would be multimedia email [CAR94] and multimedia fax [PAT94].

Multimedia systems present several requirements that distinguish them from traditional information processing systems. These requirements mainly stem from the real time nature of continuous media streams as well as the variety of media involved in the application. First, multimedia systems need to integrate the processing, the handling, the storage and the rendering of a variety of media objects in a seamless manner in order to fully exploit the potential of the application and the resources

involved. Second, multimedia systems should provide a flexible and dynamic resource management in order to accommodate variable system conditions and conflicting media characteristics. Third, the processing of media objects should be in real time. Fourth, User Interactivity should be supported in an inherent way. Finally, and perhaps most importantly, multimedia systems should be able to maintain synchronization among the media objects involved during the course of a multimedia session.

1.2 Multimedia Synchronization

One of the key design issues in distributed multimedia systems is synchronization. In the context of multimedia communications, synchronization refers to the preservation of the temporal relationships within and between various media [GEO96]. Intra-stream synchronization refers to maintaining the continuity of an isochronous stream such as video and audio. These streams consist of small presentation units that adhere to strict temporal constraints to ensure acceptable user perception at playback. Inter-stream synchronization refers to maintaining temporal relationships between different media. These could be tight, as in lip-sync between audio and video, or loose, as in text captions and audio.

In a distributed multimedia system synchronization can be disrupted by a number of factors. The transmission of various data streams, with different traffic characteristics and possibly from multiple servers over a network introduces inevitable random delays and delay variations both within and among different streams. The processing loads and context switching overheads on the servers and the clients introduce further disturbances. To compensate for these disruptions a complete stream synchronization

protocol must be specified. Although such a protocol must span all the layers involved in the distributed multimedia system, we feel that the most significant synchronization effort must lie within the application services layers i.e. closer to the user to ensure the best rendering quality.

The Multimedia Communications Research Laboratory (MCRLab) at the University of Ottawa has been conducting synchronization research for a multimedia News-on-demand application within the Canadian Institute for Telecommunications Research's (CITR) broadband services project. A News-on-Demand application typically involves the retrieval of a news article, composed of video, audio, text captions and graphics from multiple servers over a broadband network. While such an application is quite tolerant to start-up delay, the rendering should be carried in real time to avoid huge buffering requirements at the client. MCRLab adopted a two tiered approach to the synchronization problem without the use of a global clock [GEO96]. First, a prevention mechanism pre-computes delivery schedules for the different media using negotiated Quality of Service (QoS) parameters. Second, an error recovery mechanism known as the Stream Synchronization Protocol (SSP) maintains synchronized playback at the client. Within the SSP, special attention is given to time-dependent coded video streams due to the presence of a hardware/software decoding entity and special characteristics of coded time-dependent video streams. A Predecoder Synchronization Controller is needed to protect from buffer overflow and underflow conditions and maintain stream continuity at the client.

1.3 Synchronization Control of Coded Video Streams

In multimedia systems, continuous media streams, such as audio and video, require huge storage space and transmission bandwidth. For example, the speeds of broadcast quality NTSC video and studio quality NTSC video are about 120 Mbits/s and 216 Mbits/s respectively [LYL92]. The use of appropriate compression techniques is needed to reduce storage and transfer rate requirements, making the integration of continuous media services economically viable with current storage and network technologies. Fortunately , recent advances in the area of multimedia compression during the last few years has made this a reality. Most notably are the Join Photographic Experts Group (JPEG),[JPE93] and [PEN93], for still images and video, and the Moving Pictures Experts Group (MPEG), [MPE93] and [LEG91] for video and audio.

Coded video streams are expected to become the predominant type of multimedia traffic transmitted and are one of the most demanding in terms of bandwidth, delay and processing requirements [TS96]. For this reason we will focus our work on coded video streams. Time-dependent coded video streams are very sensitive to delay and delay variation. Moreover, they are more vulnerable to data loss than non coded streams. In addition, they may be of a variable bit rate (VBR) nature making it difficult to accurately estimate their buffering requirements. furthermore, they may have a different presentation, transmission and decoding orders which is the case with MPEG video. Also, coded video streams have a high throughput, which is even orders of magnitude higher after decompression, necessitating fast real time processing at the level of the coded stream in order to avoid huge buffering needs at the client.

Finally, coded video streams require a decoding entity (hardware/software) at the client. These distinctions dictate special provisions in terms of synchronization control of these streams at the client.

We propose to blend the decoding of coded streams with the synchronization mechanism. The resultant Predecoder Synchronization Controller (PSC) (due to the fact that it sits before the decoder at the client) ensures the intra-media synchronization (continuity of the video stream) while protecting the client buffer from overflow and underflow conditions. The PSC, based on buffer occupancy as a main parameter, employs time expansion and time contraction policies to avoid underflow and overflow conditions, respectively. The PSC functionality was specified and implemented for both MJPEG and MPEG video bit streams.

1.4 The CITR Broadband Services Project

The context of this research lies within the confines of the Canadian Telecommunications Research (CITR) institute Multimedia News-on-Demand project which is part of a major project denominated "Broadband Services". The aim of this project is to develop a thorough understanding of the hardware, software, networking and database issues related to the development of future presentational multimedia applications. The project has been split into six complementary projects, each of which is being investigated by a different university:

- Multimedia Data Management examines issues in relation to storage and retrieval of multimedia objects over high speed networks (Prof. T. Oszu, University of Alberta).

- Scalable Video Encoding for Database Storage is concerned with the flexible representation of compressed images and video (Prof. E. Dubois, INRS Telecom.).

- Quality of Service Negotiation and Adaptation studies methods for QoS management, negotiation, monitoring and renegotiation (Prof. G. Bochmann, University of Montreal).

- Distributed Continuous-Media File System looks into the development of a file system that supports time sensitive multimedia data such as high quality video streams (Prof. G. Neufeld, University of British Columbia).

- Synchronization of Multimedia Data for Presentational Applications has as goal to develop a complete software synchronization solution for distributed environments (Prof. N. D. Georganas, University of Ottawa).

- Project Integration focuses on coordinating the different projects and the integration of the software modules developed by each project (Prof. J. W. Wong, University of Waterloo).

Figure 1.4.1 Shows the general architecture of the CITR NoD prototype:

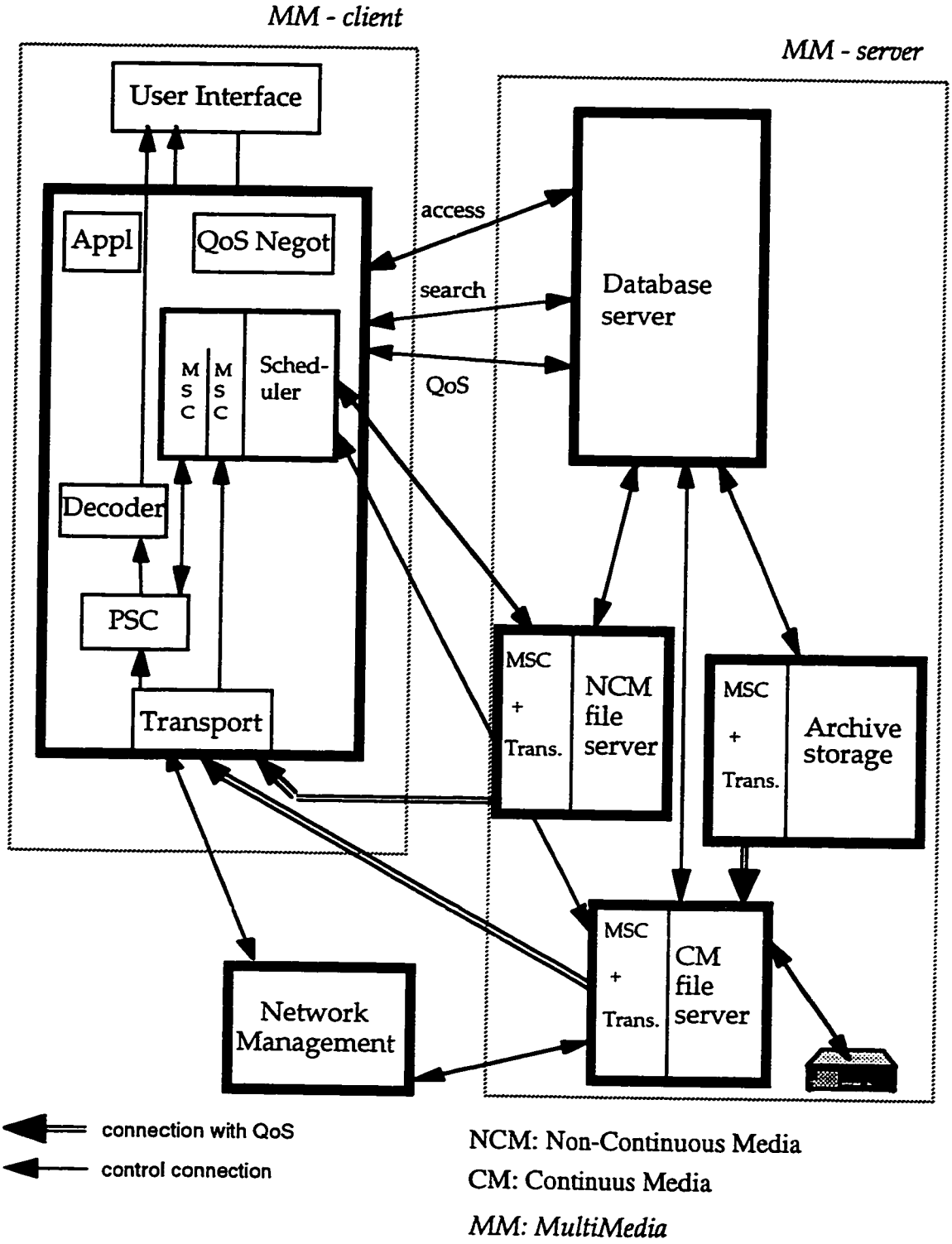

Figure 1.4.1 General Architecture of The CITR NoD Prototype

1.5 Thesis Outline

The thesis is organized as follows:

Chapter 2 presents a general overview of multimedia systems and a comprehensive review of multimedia synchronization. The focus of the synchronization review is on synchronization in a distributed environment. First, the notions of synchronization and synchronization reference model are discussed. Second, the types of synchronization errors and their causes are detailed. Third, the various synchronization schemes discussed in the literature are presented. Finally, synchronization specification is elaborated upon, both from the point of view of specification methods and QoS requirements.

Chapter 3 details the Predecoder Synchronization algorithms developed. First, the concept of a News-on-Demand teleservice is outlined. Second, the MCRLab Synchronization architecture is reviewed. Third, video compression and coded video streams are discussed. Finally the PSC algorithms , along with the need for such entity, the design goals and the control parameters used, are specified for MJPEG and MPEG video bit streams.

Chapter 4 focuses on the implementation issues encountered in the development of the PSC entities. We begin by comparing the original prototype and the modified one augmented with the PSC functionality. Then, a stream structure used at the server side, is developed and used to compute buffering delay and buffer size for the prevention of starvation and overflow conditions at the client. Finally, the PSC implementation architecture and mechanisms are detailed.

Finally, chapter 5 provides a summary of the work done, the major contributions of the theses and ponders on future considerations as an extension of this work.

1.6 Publications

One publication arose from this research:

- M. Daami and N.D. Georganas, " Client Based Synchronization Control of Coded Data Streams," Proc. IEEE Multimedia Systems '97, Ottawa, Ontario, June 1997.

Chapter Two

Multimedia Synchronization

The advent of microprocessor based computer systems in the early sixties revolutionized the way we handle traditional information like text and graphics. Killer applications included database systems and spread sheets which metamorphosed the way information is stored, retrieved and manipulated. With computing power and storage capacity exponentially increasing over the years, the capability to process non traditional information such as audio and video became a desktop reality. In addition, the deployment of new broadband telecommunications technologies, such as ATM, permits the transfer of such bulky information across computer communication networks. As a result, multimedia systems emerged as the offspring of the ever converging computing and telecommunications worlds. As defined in [STE95.2], multimedia systems are characterized by the integrated computer controlled generation, manipulation, presentation, storage and communication of independent discrete (time-independent) and continuous (time-dependent) media. With capabilities of handling and sharing all modes of information, multimedia systems are leading the way into a new

information revolution. This time, not only information processing is being revolutionized, but also the way we work and collaborate around such information.

This chapter provides a contextual background for the work conducted in this theses. Section 2.1 briefly discusses the aspects and requirements of multimedia systems. The subsequent sections provide an in-depth discussion of one of these requirements, namely, multimedia synchronization. The focus in this discussion will be on synchronization in the distributed environment.

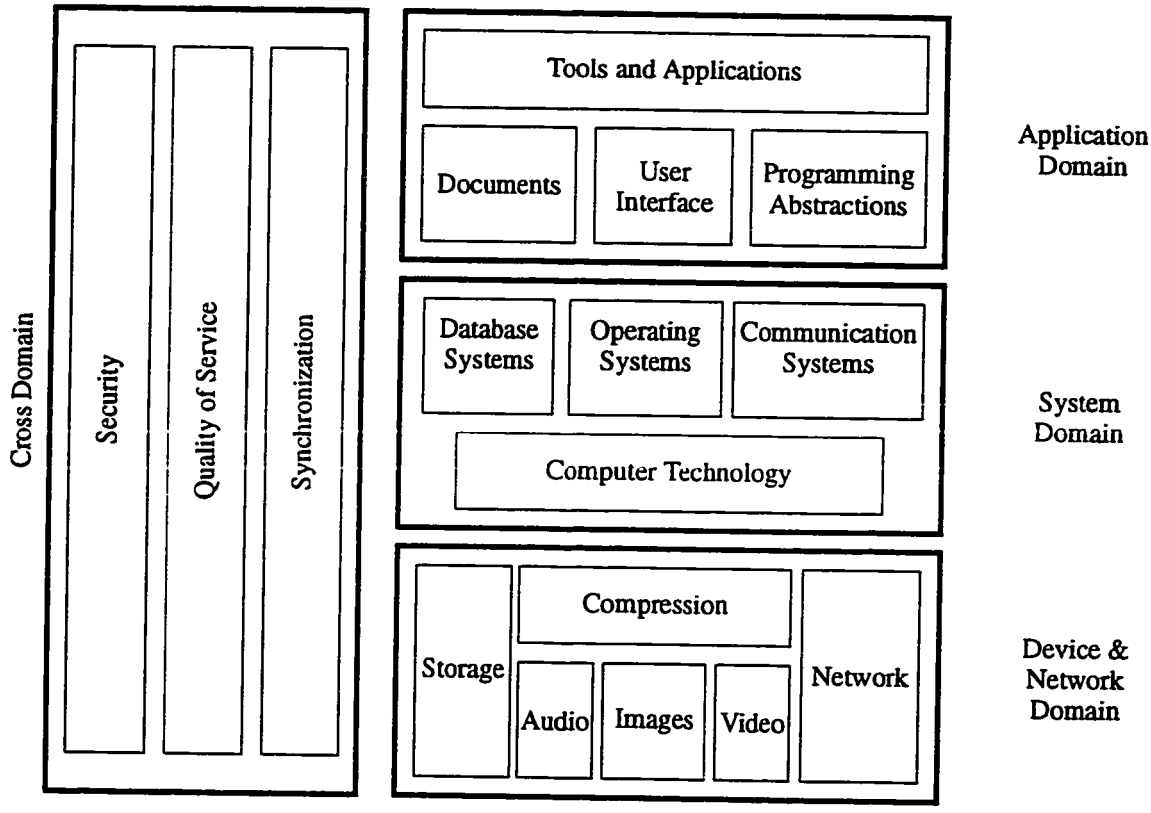

Figure 2.1.1 Complete View of Multimedia Systems

2.1 Multimedia Systems

Distributed multimedia systems are characterized by the integrated computer controlled generation, manipulation, presentation, storage and communication of independent discrete and continuous media. This very definition suggests a layered model of distributed multimedia systems. Figure 2.1.1 depicts a dissection of the various aspects and topics involved in multimedia systems grouped in three layers: application, system and network and device domains [STE95.2] (with few modifications).

2.1.1 The Application Domain

The application domain is the highest level of multimedia systems. The application domain encompasses the software applications, tools, abstractions, interfaces and formats necessary in the development of a multimedia service. For example, the MHEG (Multimedia and Hypermedia Expert Group) standard [ISO93], which defines the representation and encoding of final form multimedia documents and hypermedia information objects to be exchanged within or across applications and services, is an application domain multimedia document format. Multimedia applications can be divided into three categories:

- Conversational multimedia applications involve multimedia conferencing systems and computer supported collaborative work (CSCW) environments. Conversational multimedia is carried out live and requires causality constraints.

- Presentational multimedia applications are retrieval based applications such as News-on-Demand or Video-on-Demand teleservices. Presentational multimedia requires distribution servers which supply prerecorded stored multimedia documents to the users' workstations.

- Multimedia mailing applications are store and forward applications with no specific time constraints during the delivery of multimedia data. Examples of such applications are Multimedia Fax [PAT94] and Multimedia Mail [CAR94].

2.1.2 The System Domain

The system level domain provides the run time environment for multimedia applications and services. Most notably are the operating system [STE95.1] and the communication system [DUP92] aspects. The key word at the system level is real time, due to the time constraints of continuous media. The computer technology aspect defines the hardware platform on which the operating system and the transport system are executed. The data base system allows structured access as well as management of large amounts of multimedia data [WOE86].

2.1.3 The Network and Device Domain

The network and device level domain is the lowest level in multimedia systems. The device aspect involves multimedia data rendering and capturing technologies such as audio boards/speakers, video adapters and digital cameras. Moreover, the various digital formats of the involved media are specified at this level. The multimedia compression standards such as JPEG [WAL91] and MPEG [LEG91]. Digital storage technology is an important enabling factor for multimedia systems. The advent of optical storage technologies, such as CD-ROMs, diminished significantly the cost of disk space allowing for the storage of large multimedia files. Figure 2.1.2 gives a glimpse of the storage requirements of multimedia data [FUR94]. Finally, the networking aspect describes high speed networking and access technologies used to transfer media data between client and server workstations. These include the Fiber Distributed Data Interface (FDDI) [ROS89] and the Asynchronous Transfer Mode (ATM) [LEB92] technologies.

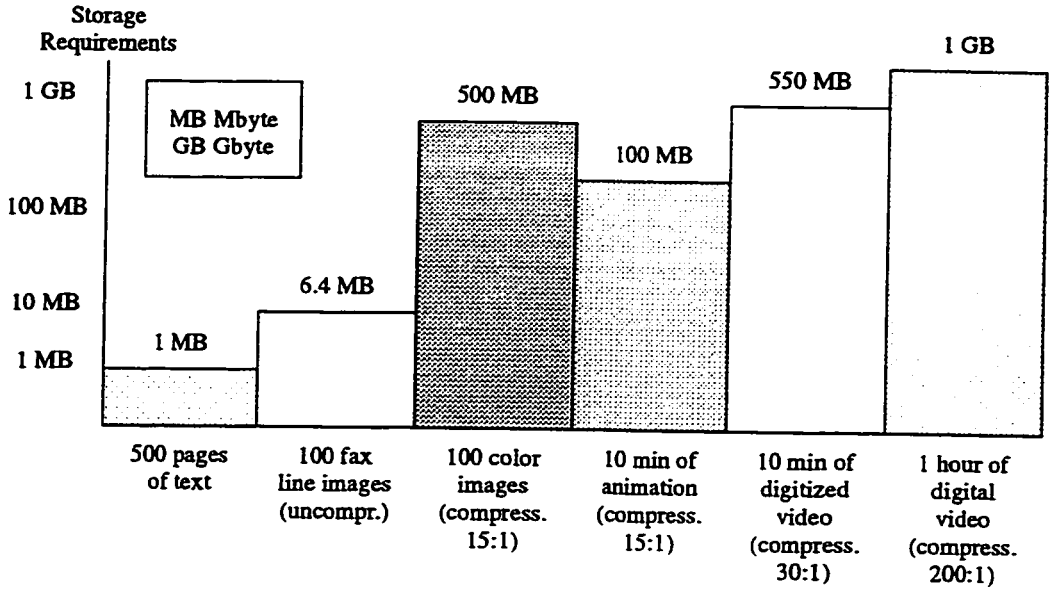

Figure 2.1.2 Storage Requirements For Multimedia Data

2.1.4 The Cross Domain

The cross domain involves aspects of the multimedia system which is not confined within the logical bounds of one of the previous domains. The security aspect in multimedia system ranges from copyright infringements to user and design access. Security is a key issue in multimedia systems and is yet to be investigated in a thorough manner. Another important aspect of multimedia systems that spans the user level to the device level is the Quality of Service (QoS) aspect. QoS defines qualitative and quantitative metrics to describe the degree of performance required from the multimedia system to meet the desired user perception quality. More on the QoS issue is detailed in section 2.6.2. Finally, another key issue is multimedia synchronization. Multimedia synchronization aims at the preservation of the temporal relations within and among multimedia objects during the rendering process. The rest of this chapter will be dedicated to this critical issue.

2.1.5 Multimedia Systems Requirements

The handling of isochronous media such as video and audio imposes a new set of requirements on multimedia systems compared to traditional computational systems. Some of the most significant requirements are outlined below:

Integration: Multimedia systems involve the computer controlled manipulation of several information types with often different processing requirements. A key challenge in multimedia systems is to provide a general framework for information modeling, integrating different media types [SCH90]. Such an "information model" will provide mechanisms for structuring existing and future information types for easier handling and decentralized processing. Moreover, it will provide services for the generation, processing, retrieval and presentation of supported media types.

Dynamic and flexible resource management: Multimedia systems are characterized by the wide variety of services due to the multitude of media involved. Each of these services is subject to a range of user perception requirements and choices. Moreover, the distributed nature of multimedia systems creates highly dynamic environments in terms of network and server loads and information location. Therefore, multimedia systems should provide flexible resource allocation and management in order to accommodate the dynamic nature of the underlying system. For instance, this includes the negotiation of network QoS parameters at the connection setup and allows for renegotiation as the presentation progresses.

Real Time: The time related semantics of continuous media types impose real time constraints on the underlying system hardware/software. Increased CPU power is required for timely compression/decompression of multimedia data. The computer architecture must provide fast

and efficient I/O and high throughput bus architectures for maximum CPU to memory bandwidth. The operating system should support new data types, real time scheduling and threading and fast interrupt processing. Storage devices must sustain high transfer rates and have fast access speeds. Network and transport protocols should provide high bandwidth, low latency and low jitter for timely and continuous delivery of multimedia data.

Interactivity and Coordination: Multimedia applications require various degrees of user interactivity depending on the presentational or conversational nature of the application. For example, a Video-on-Demand application should provide common VCR like functions for the user. However, in a joint editing conferencing system, the interactivity level becomes more complex and some coordination is necessary to preserve causality and data integrity throughout the session. Coordination can be divided into two main areas. First, we have the coordination of those participating in the conferencing session by assigning roles to different users such as conference leader, arbitrator and so on. Moreover, this type of coordination handles the joining and leaving of the session. Second, we have the coordination of the data and information being processed, making sure that the joint windows are updated in a timely and correct way.

The final requirement of multimedia systems in this condensed list is multimedia synchronization which will be discussed in depth throughout the rest of this chapter.

2.2 Notion of Synchronization

To synchronize, according to the English language dictionary, is to occur at the same time or simultaneously. Therefore, any notion of synchronization is inherently related to time. For example, synchronization in distributed database systems refers to database access

and database content. Synchronized access refers to simultaneous read and write operations by two or more users. Synchronized content refers to having up to date information across a distributed database. Both of theses synchronization activities, while different in nature, are tied to factor time. Synchronization in multimedia systems is no exception to this rule. It refers to the temporal relations between media objects in a multimedia document or system. This notion can be further extrapolated to include the content and the spatial relations among the multimedia objects. The following three paragraphs elaborate on each of these synchronization properties.

Spatial relations define the space used for the presentation of a specific media object or unit at a certain point of time in the multimedia presentation. Spatial relations depend on the nature of the output device. If the device is two dimensional, such as a monitor, spatial relations define the area in which the media object is to occur with respect to other media objects. Such an area is commonly referred as a layout frame. If the device is three dimensional, such as in virtual reality systems, spatial relations define the position in space where the object is to occur. Finally spatial relations can be extended to audio content where the placement of audio sources helps simulate a natural and live environment.

Content relations describe the content dependency among media so that a change in the content of one media object will trigger a simultaneous change in the related media objects. The concept here is to separate the view of the media object from the content of the object itself. For example, the curve in a graph will depend on the data content of the related spread sheet. A more elaborate example is the composition of a news cast document. The initial logo and music can be tied the type of the news cast: a news flash,

morning or evening news, thereby avoiding the trouble of maintaining three separate news documents.

Temporal relations describe the temporal dependency between media objects. For example, the video of a singer and its corresponding voice must be played together (as recorded) in order to be comprehensible. Unlike spatial and content synchronization, temporal synchronization is fundamentally tied to the rendering of multimedia objects. In fact temporal relations are the very result of the involvement of continuous media which allows us to call a specific system a multimedia one. Therefore, the notion of multimedia synchronization will refer to the temporal synchronization of multimedia objects in all of the subsequent discussions.

2.3 Inter- and Intra- Media Synchronization

2.3.1 Types of Media

In multimedia synchronization, we distinguish between two types of media, depending on their temporal requirements. First, there are time-independent (discrete) media objects which can be any type of traditional media, like text and still images. When isolated, the semantic contribution of such media types does not depend on time. Second, there are time-dependent (continuous or isochronous) media objects, such as video, animations and audio. Time-dependent media objects are composed of several consecutive media units which are related in time. The semantic contribution of such media objects depends on the continuous presentation of their media units with the specified time durations. Because of this continuous presentation in time, time-dependent media objects are also referred to as continuous media streams or simply media streams.

2.3.2 Intra- vs. Inter- Media Synchronization

There are two types of synchronization operations depending on the temporal relations within or among different media objects:

Intra-media synchronization refers to the temporal relations between media units within a time-dependent media object. For instance, the frames of a video object with a rate of 30 frames per second must be presented in consecutive intervals of 33 ms each (figure 2.3.1). Intra-media synchronization is also known as continuity to denote the strict timing relations between the media presentation units.

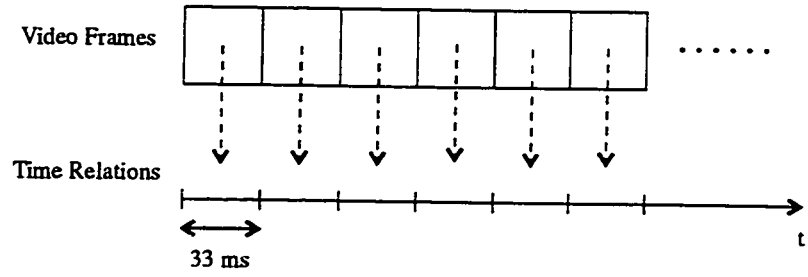

Figure 2.3.1 Intra-media Synchronization Relations Within a 30 f/s Video Object

Inter-media synchronization refers to the temporal relations linking the presentation of various media objects within a multimedia document. This includes the relative starts of the rendition of different objects and synchronization points during the rendition of such objects. For example, figure 2.3.2 shows a multimedia document scenario which depicts the relative start times of each object (t_0, t_1, t_2, t_3, t_4). Moreover, during the rendition of each object, there are further inter-media synchronization points as is the case between text and the images.

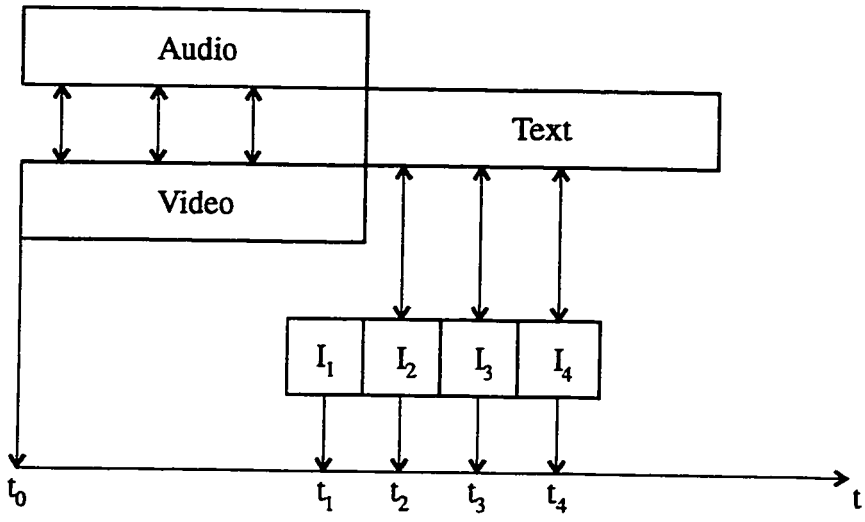

Figure 2.3.2 Inter-media Synchronization Relations in a Multimedia Scenario

2.3.3 Live vs. Synthetic Synchronization

Live and synthetic synchronization refer to the way temporal synchronization relations between media objects are defined.

Live synchronization implies an intrinsic time relation between media objects. In other words, live synchronization refers to the temporal relations between media objects as they existed during the capturing process. For example, the synchronization relation between a singer's voice and his/her concert's video is a live synchronization relation. Both media objects must be displayed together as they were captured for synchronous playback. Another example of live synchronization is the movement of the mouse pointer with the voice of the presenter in a telecollaborative conferencing application. In general, live synchronization is more apparent in conversational multimedia, where media capturing and media rendering are carried in real time. However, live synchronization must also be preserved in presentational multimedia, despite the temporary storage phase of the media objects.

Synthetic synchronization refers to the artificial specification of temporal relationships between independently captured or created media objects. An example of synthetic synchronization relations are text captions that translate a conversation to another language. The display of text annotations must be "manually" coordinated with the corresponding scenes. Synthetic synchronization allows for flexible and creative specification of multimedia documents.

2.3.4 Synchronization Granularity

Synchronization granularity refers to the level of synchronization relations between and within media objects. The notion of synchronization granularity is tied to time-dependent media streams. Continuous media streams consists of a sequence of information units called Logical Data Units (LDUs) by Steinmetz [BLA96]. The interpretation and size of an LDU depends on the level of synchronization relation. However, the smallest possible LDU is that which is the smallest possible presentation unit of a media stream. For example, figure 2.3.3 shows the different levels of granularity of a video stream (in digital format). The frames could be seen as LDUs for intra-media synchronization specification. Moreover, a clip could be seen as an LDU for inter-media synchronization between the video and its corresponding audio stream.

In general synchronization granularity implies a hierarchical decomposition of media objects into:

- a content hierarchy as is the case of a video stream
- or a coding hierarchy as is the case in a audio stream, where an LDU could be specified as fixed duration of sound samples.

This hierarchical view of synchronization granularity suggests that synchronization operations are to be carried at different levels. This is the topic of the next section which describes a reference model for multimedia synchronization.

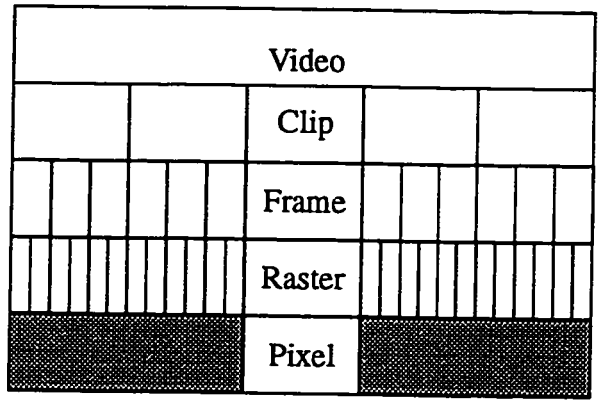

Figure 2.3.3 Granularity of a Video Stream

2.4 Multimedia Synchronization Reference Model

As illustrated in the previous section, multimedia synchronization can be viewed in a layered way depending on the application, synchronization operation and media granularity. A reference model is therefore needed to classify, identify and provide synchronization structures and mechanisms for each synchronization layer. Several synchronization reference models were suggested in the literature. We adopt and elaborate on that first introduced by Steinmetz et al. in [MEY93].

The reference model, shown in figure 2.4.1, consists of four layers, namely, specification, object, stream and media layers. Each layer provides synchronization mechanisms as well as an interface of services to the layer on top of it. The following paragraphs discuss the main characteristics of each layer in the reference model.

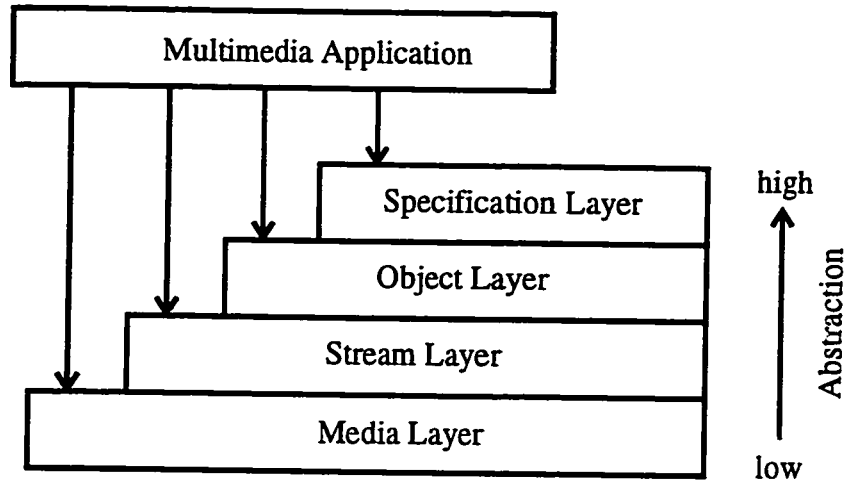

Figure 2.4.1 Synchronization Reference Model

Media Layer: The media layer operates on a single continuous media stream. The media stream is treated as a sequence of LDUs such as frames for video. Mechanisms provided at the media layer ensure continuity of a media stream through proper resource management and process scheduling. Moreover, network data connections are handled at this layer in the case of a distributed environment.

Stream layer: The stream layer operates on a collection of continuous media streams which are to be presented in parallel. The stream layer delegates stream continuity to the media layer; however, it maintains inter-stream synchronization between different media. The stream layer execution must be in a real time environment so that timing constraints, specified as QoS parameters (see section 2.6.2 for more details) by the application, are respected or met.

Object layer: The object layer operates on all types of media, thereby similarly abstracting both time dependent and time independent media. The object layer is responsible for synchronized rendering of a particular multimedia document. This is achieved through a common schedule for all media objects. The object layer hides the complexity of handling continuous media

streams. Furthermore, the object layer ensures inter-object synchronization between continuous and discrete media objects.

Specification layer: The specification layer does not conduct any particular synchronization operation. Instead, the specification layer provides applications and tools to create synchronization specification of a multimedia document. Such tools include multimedia editors and authoring systems. Moreover, the specification layer includes an interface to convert the specification to an object schedule for use by the object layer. Finally, the specification layer maps the QoS requirements at the user level to those at the object layer necessary to maintain the desired presentation quality.

2.5 Synchronization in a Networked Environment

A distributed computing environment consists of one or several multimedia data sources connected through a broadband communication network to one or several multimedia data sinks. This physical separation between the multimedia data storage and playback platforms introduces a multitude of issues and constraints that further complicate the synchronization problem. While the network is a centric factor behind such restrictions, the processing efforts' impact on synchronization, both at the source and sink nodes, is not to be ignored. In this section, we discuss the different aspects of synchronization in a distributed environment.

2.5.1 Synchronization Transport

Synchronization information is needed for proper rendering of multimedia data at the client site. In a distributed environment, multimedia data storage resides on remote network nodes. Therefore, a mechanism for delivering multimedia synchronization specification is needed.

Depending on the nature of the multimedia application and the level of synchronization specification, four different transport mechanisms are described:

Complete delivery of synchronization information before start of presentation: This approach, used in [LAM94.1] and [LIT90.1], is more of an object layer technique mainly used to deliver synthetic synchronization information of the multimedia document. As such, it does not apply to Computer Supported Cooperative Work (CSCW) or conversational multimedia applications. Typically, in a presentational multimedia applications , the synchronization specification, referred to as a scenario, of an edited document is stored in a meta-information database. When a document is requested by a user, the scenario is retrieved and sent to the client before the presentation begins. This induces an inevitable delay associated with the delivery of the scenario. Nevertheless, the simplicity of implementation of this technique compensates for such a drawback and therefore provides an attractive and effective solution, especially when dealing with several multimedia sources.

Using synchronization markers: the synchronization marker concept is a stream layer technique that is mainly used to synchronize multiple streams over multiple channels. The idea is to insert Synchronization Markers (SMs) into the LDUs of the data streams at the sender [SHE90]. The data streams experience different delays and delay variations when they arrive at the client. Therefore, the transport system of the receiving end needs to buffer the data streams until all LDUs corresponding to adjacent SMs are received. It then passes the aligned LDUs to the user application for synchronized playback. The synchronization markers could well be time stamps generated by the capturing devices as in the case of the Logical Time System (LTS) introduced in [AND91]. While the SM scheme requires minor modifications to the transport system, it necessitates extensive buffering at the receiver end to align LDUs with

corresponding SMs. Moreover, the insertion of SM into coded data streams may prove difficult if coding is carried in real time. Finally, the SM scheme supports only parallel synchronization constructs, making it unsuitable for more intricate synchronization relationships.

Using a separate synchronization channel: This synchronization mechanism, described in [SHE90], uses a independent communication channel to convey synchronization information. This allows the different data streams to be transmitted within their own channels without any modifications by the transport system, except for necessary packetizing. The synchronization channel approach is useful when no synchronization information is know in advance, as is the case in live video and audio transmission. Besides, it allows for sophisticated synchronization mechanisms to be transmitted, enabling complex multimedia sessions. In addition, the synchronization channel presents a higher bandwidth overhead than that of the SM scheme. Moreover, it requires far more complex processing at the client. For instance, the synchronization specification must be decoded at the client at the same time as the display of multimedia data. This increases the overall CPU load at the receiving node and may cause loss of synchronization at times. Furthermore, the synchronization channel itself may experience delays and data losses which temporarily disrupt or invalidate the multimedia applications. Finally, this mechanism may prove difficult to implement in the case of multiple multimedia source nodes and clients.

Multiplexing media streams at the source: This mainly stream layer approach merges the LDUs of different media streams together on the same virtual communication channel. As a result, the related synchronization information is transferred with the media units. Therefore, this method eliminates the need for an additional synchronization channel. Moreover, multiplexing the LDUs nullifies any additional delays associated with synchronization

specification delivery. The main disadvantage of this mechanism is the difficulty to select an appropriate QoS which matches all media involved. In fact, the stringent requirements of all media should be combined for such a multiplexed stream. This might prove a waste in communication resources given the benefits of statistical multiplexing in B-ISDN. Furthermore, this approach is not suitable in the case of multiple distributed media sources, which is highly likely in widely deployed multimedia services. While the multiplexing referred to in this section is carried out by the transport system, stream multiplexing could occur at the coding level. For example, MPEG transport and program stream specifications define a template for multiplexing several video and audio streams along with their synchronization information. In this case, the coded stream could be regarded as a single media type allowing the transport system to further multiplex it with other media objects.

2.5.2 Types and Causes of Synchronization Anomalies

The presentational of multimedia objects requires the adherence to some form of temporal constraints. These define relations between granules of different media as well as relations between granules within the same media stream. Asynchronies or synchronization anomalies occur when these temporal constraints are not met during the rendering of multimedia data. While strictly speaking these errors disrupt the synchronization of multimedia data, the human perceptibility to such anomalies is subject to some tolerance measures. This will be discussed in section 2.6 dealing with presentation quality and synchronization specification methods. The emphasis in this section is to identify the types of synchronization errors. In fact, the presentation of such issue in the context of distributed synchronization seems very appropriate. The reason being that a networked environment provides a complete picture of

what can disrupt the synchronization of multimedia data. This leads to the discussion, further on, on the main causes of synchronization errors.

The first type of asynchronies concerns the temporal relations linking granules of different media streams. This referred to as *skew*. Skew, as Little puts it in [LIT90.2], is defined as the average difference in presentation times between two synchronized objects (or streams) over *n* synchronization points at their synchronization instant. Based on this definition, skew depends on the time interval taken to observe the media streams. For instance, in figure 2.5.1, the skew between Streams A and B in the period of T_0 and T_1 is ½ unit of time over 3 synchronization points. However, between T_0 and T_2, the skew adds up to 7 units of time over 5 synchronization points. Stream B therefore is said to be skewed. The skew can also be negative. This is the case when the presentation of the media units takes less time than expected. One major cause of skew is clock drift and is discussed later on.

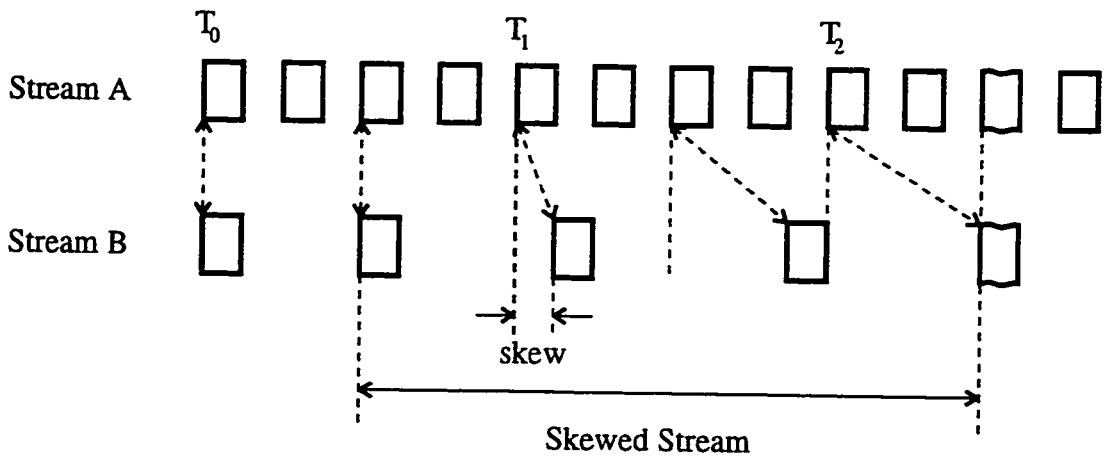

Figure 2.5.1 Inter-stream Skew

Another definition of skew is the time difference between related LDUs (or media units) of two or more media streams [STE96]. This definition could be referred to as instantaneous

skew, if contrasted to the earlier definition. Both definitions, therefore, are complementary and might be used interchangeably.

The second type of synchronization errors concerns the intra-stream temporal relations. This is referred to as *jitter*. Jitter is defined as the difference between the actual presentation time of a media unit and its nominal presentation time (we should distinguish here between stream jitter and network jitter. The latter implies variation in delays experienced by packets). In figure 2.5.2, both media units 2 and 4 experience a jitter of 1 and -1 time units, respectively. Stream jitter implies local variations in the rate of presentation of LDUs. This may be perceived as glitches in the case of an audio stream. Both the perception of jitter and skew are subject to some tolerance measures and are discussed in section 2.6.

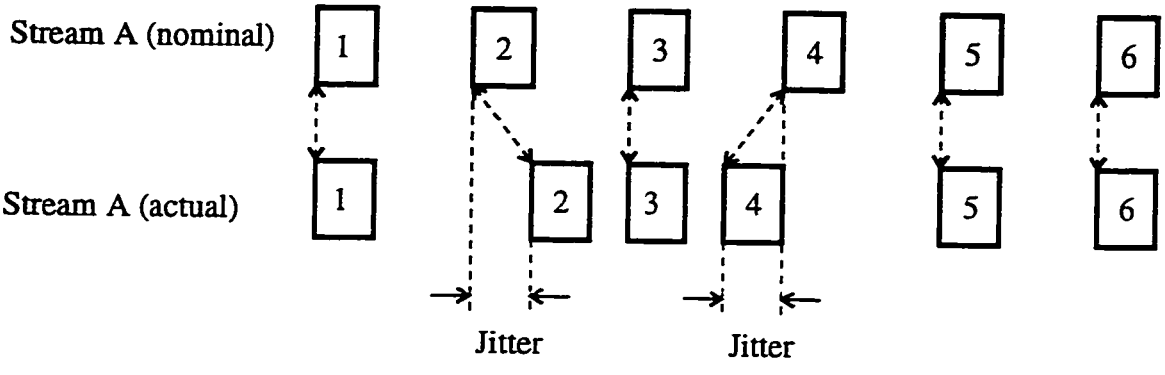

Figure 2.5.2 Intra-stream Jitter

By closely examining the asynchronies that can occur, we can conclude that except for synchronized start up of media rendering it is sufficient or necessary to maintain intra-stream synchronization (i.e. prevent stream jitter) in order to ensure inter-stream synchronization (i.e. prevent skews). Having described the two main synchronization errors that media streams could suffer from, we now discuss the main causes of such errors within the frame of a distributed multimedia system.

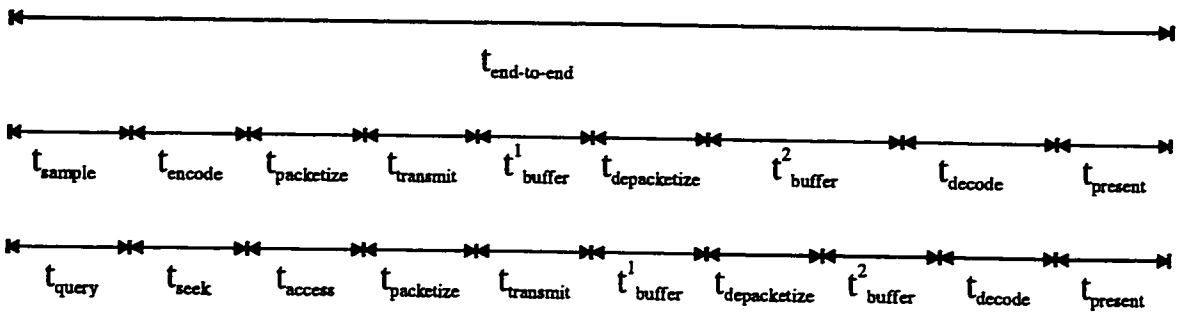

Figure 2.5.3 End-to-end Delay Canonical Decomposition

End-to-end delay: The physical separation between the multimedia data sources and multimedia data sinks imposes an end-to-end delay experienced by media units before their eventual presentation at the client site. Figure 2.5.3 depicts a canonical decomposition of the various components of the end-to-end delay. For live and conversational multimedia applications there is a media sampling (t_{sample}) and media encoding (t_{encode}) delays associated with the data acquisition from live sources. an equivalent delay, composed of a query (t_{query}) and access ($t_{seek} + t_{access}$) delays, is associated with presentational multimedia applications. These delays mainly affect the startup stage of media rendering and require some buffering at the client site. The remaining delays are a packetizing delay ($t_{packetize}$) introduced by the transport layer, a propagation delay ($t_{transmit}$) due to network transmission, a reassembly delay ($t^1_{buffer} + t_{depacketize}$) caused by the sink's transport system, a decoding delay ($t^2_{buffer} + t_{decode}$) in case the stream is a compressed one and finally a presentation delay ($t_{presentation}$) that includes processing at the client for the final rendering of a media unit. The delays depicted above are all random delays that depend on several factors. For instance , the transmission delay ($t_{transmit}$), which essentially models the communication network effect, depends on the available bit-rate, the network structure, the switching fabric, the switching mode (circuit, packet, ATM), the network load, the Medium Access (MAC) layer's processing and the queuing delays in the network nodes [HUI90]. While estimated

bounds for these delays can be used for calculation purposes, in real life, these delays follow a random probability distribution. This randomness induces a variation in delays in media streams at two levels: at the packet level to form what is known as network jitter and at the media unit level as a partial result of the network jitter. These delay variations prevent the continuous rendering of the media units as they arrive at the sink. Moreover it will create localized media variations in the media stream being presented. Therefore, the intra-stream synchronization is disrupted and stream jitter arises. Delay variations can be overcome or minimized by buffering at the client. This will smooth out the media stream for continuous presentation.

Clock synchronization and drift: All synchronization schemes require some measurement of time at the local system. Some of these schemes require knowledge of global timing relations in order to properly carry the rendering process. Therefore, in a distributed environment, the synchronization accuracy between clocks at the sink and source nodes should be considered. Lack of source and sink clocks synchronization manifests itself in two different forms. First, clock offsets may exist between source and sink nodes. This has a direct effect on the transmission start times of different sources. Clock offsets must be taken in account in order to make sure that different media streams make it to the client site at relatively the same time. This helps avoid large buffering at the client. Second, differences in clock frequencies at the source and the client may result in clock drifts over time. On one hand, if the source's clock is faster than the sink's clock, then buffer overflows at the client may result. On the other hand, if the source's clock is slower than the sink's clock, then starvation may result. Both conditions will disrupt stream continuity and eventually inter-stream synchronization. The clock synchronization problem becomes more difficult in the case of multiple sources and sinks. Moreover, clock offsets and drifts cannot be known in advance, leading to maximum estimate

strategies. One way to ensure clocks are synchronized is to use clock synchronization protocols such as the Network Time Protocol (NTP) [MIL92]. However, this may not desirable because workstations belonging to different entities may not accept to synchronize. Furthermore, the accuracy of such protocols depends on deterministic channel delays which are very hard to satisfy. Finally mechanisms not using global clock synchronization are simpler and more light weight [COR95].

Processing loads: While network effects on the media stream are centric to any distributed synchronization effort, we should not overlook the effects of various data processing activities that these same streams undergo at the source and sink systems. The major factors in play at this point are the hardware platforms, providing raw processing power, and the operating system, featuring the runtime environment for different activities composing the multimedia applications. On the hardware side, desktop multimedia power is now a reality with increasingly powerful CPUs. The same can be said about mass storage technologies in terms of access and transfer rates. The operating system side leaves less to be desired when compared with its hardware counterpart. Different activities at the server (disk reads scheduling, transport layer processing and information caching) and at the client (buffer management, transport layer processing, media decoding and media rendering) require hard real time runtime environments to meet strict constraints imposed by continuous and bulky multimedia data. Unfortunately, such operating systems are yet to be available off the shelf. Jitter at the stream level can be introduced due to inappropriate CPU and device scheduling, data copying overhead, inefficient thread management and context switching and the lack of fast intertask communication primitives [BUR94]. Moreover, variability of CPU loads can cause jitter or discrete losses in the stream.

2.5.3 Synchronization Strategies

Having discussed the different synchronization errors and their common causes we now present the different synchronization schemes developed in the literature.

Feedback approach to synchronization: Server feedback was suggested as a mechanism for maintaining client/server synchronization in [ROW94], [ROW92], [CEN95], [VEN95] and [HUI96] among others. The technique aims at reporting the temporal state of the presentation of different streams at the client to the server. The server, therefore, will either adjust the transmission rate, delay certain streams or simply drop media units. Various approaches with different complexity and control parameters are outlined in the literature. We detail the work proposed by Venkat in [VEN95]. In this paper, light weight feedback messages, transmitted periodically by the remote user station back to the multimedia server, are used by the server to estimate the playout instants of the media stream at the client. These estimates allow the server to detect impending buffer overflow or underflow at the remote site and so can readjust the rate of the data stream to avoid loss of data and synchronization at the client. For inter-stream synchronization, the server may either have to speed up some streams or slow down others, causing breaks in their presentation. The playback of a master stream can be maintained continuously, while other slave streams may be subject to discontinuities.

Server feedback may be suitable for distributed video playback applications involving a single server and a single client; however, in widely deployed multimedia on demand services with multiple servers typically handling hundreds of connections, server feedback may be hard to implement. Moreover, server feedback imposes an additional load on an already loaded

server. Finally, feedback messages might suffer from jitter introduced during transmission, affecting the accuracy of the synchronization mechanism.

Network resource controlled synchronization mechanisms: Synchronized delivery of multimedia data can also be ensured by intelligently controlling network resources to guarantee delays and delay variations. At least, such is the claim of several papers in the literature, [ZNA95], [FER92] and [BAQ96] among others. The basic tenet of this approach is to guarantee the availability of resources on-demand or in a static mode so that media streams are not constrained at the time of transmission. In [ZNA95], Znati proposes a network architecture to support the timing and synchronization requirements of multimedia calls. The architecture is based on a network level abstraction called (Φ,Δ)-stream. An instance of a (Φ,Δ)-stream is a simplex end-to-end communication stream associated with a multimedia call. The parameter Φ represents the synchronization interval and Δrepresents the end-to-end delay specified by the call. The duration of a synchronization interval, Φ, is determined by the persistence duration of the various data. The data generated by the (Φ,Δ)-streams within a synchronization interval constitutes a data frame or a playback unit. In order to achieve synchronization, delay caused by the network delivering the data frame should not exceed the end-to-end delay, Δ, specified by the application. To map this synchronization model onto the network model, Znati includes schemes to guarantee support for on-time delivery of packets, jitter control, efficient network admission control parameters and efficient network load based per node delay assignment.

In general, network resource controlled synchronization schemes depend on whether network resources are allocated statically or dynamically. Static allocation usually implies worst

case scenarios and, therefore, are a potential waste of communication resources at network nodes. Dynamic allocation is more efficient, but imposes additional loads that might impair the performance of switching nodes or yet might not be accommodated by the network. In order for such schemes to succeed, a balanced approach must be reached.

Scheduling based synchronization: To preserve the temporal relations between and within media-streams during their rendering, the LDUs composing each stream need be available at the client before their playout deadlines. LDUs belonging to different streams have different sizes and experienced if random delays over the network, due to different characteristics of the channels assigned to each stream. Therefore, LDUs should be prescheduled at the source in order to make it before playout deadline at the receiver end. Prescheduling takes in account the random delays and diverse transmission times of each media stream. This scheduling based synchronization approach is described in [LIT91] and [LAM96]. Due to random delays, scheduled LDUs may arrive earlier than their playout deadlines and may necessitate buffering at the client. The amount of buffering depends on the scheduling approach which also depends on the network delay distribution. Since multiple concurrent streams with different sizes and channel characteristics are considered, a common playout or presentation schedule may translate into different control times or delivery schedules and different buffering needs. We elaborate here on the approach proposed in [LIT91.2].

Little in [LIT91.2] provides several algorithms to generate a playout schedule, translate it into control-schedules for stream delivery and compute the required buffering at the sink node. The idea is to decompose the playout schedule into subschedules for each traffic or resource class. The derived subschedules can be used by the server to determine the time instance at which objects need to be put on the network. The receiver can then buffer the incoming

objects until their playout deadline. A second way, described in [LIT91.2], is to use the derived skew to determine the worst skew between any two objects. Each object may then be delayed by that factor to achieve synchronization at the client. Obviously, the tradeoff with respect to the first approach is increased buffer requirements and delay versus a simpler implementation. In general, schedule based synchronization schemes are adequate for presentational multimedia applications. However, conversational multimedia applications do not have a predetermined scenario making it difficult to generate control schedules.

Synchronization recovery at the client: All the strategies discussed so far concentrate on either the server (feedback and scheduling) or the network (resource controlled) sides of the multimedia system. An alternative is to recover synchronization at the client with proper buffering and process management. Synchronization recovery could be combined with synchronization scheduling as is the case in [LAM96] through the concept of intentional delays and presentation rescheduling. Other approaches an be completely client based as is the case in [ROT95] and [CHE96]. In [CHEN96], Chen proposes a media synchronization model to solve asynchronies due to server overload. The approach works by assigning higher priority to more important media and adopting a delay or drop policy to treat the lower priority ones. In [ROT95], an Adaptive Stream Synchronization Protocol (ASP) is proposed by Rothermel. ASP works by incorporating buffer level control mechanisms to immediately react to changes in network conditions. A stream becomes critical when it runs the risk of buffer underflow or overflow. ASP, therefore, adapts the critical stream's playout rate accordingly. ASP also supports the notion of a master stream which acts as a reference for inter-stream synchronization. Although ASP needs no prior knowledge on the network conditions except for the average rates of the different streams, it still face few shortcomings. ASP handles

all streams in the same way consequently it fails to grasp the different characteristics of different streams which might affect visual perception at playback. Moreover, in the case of frequent network fluctuations, the synchronization overhead might become intolerable by the application itself, causing more asynchronies to occur.

2.5.4 Additional Considerations

Having described in depth synchronization anomalies, their causes and strategies to prevent them, we realize that synchronization in the distributed environment is a multi-step multi-layered process. A good synchronization protocol, therefore, is the one that covers most of these steps in a comprehensive way, depending on the application involved. These steps as listed in [BLA96] are:

- Synchronization during object acquisition
- Synchronization during object retrieval and access.
- Synchronization during delivery of media streams in the network.
- Synchronization during the transport process.
- Synchronization at the sink node for proper playback.

Additionally, in a distributed environment, the types of communications relations should be considered during the synchronization process. One to one, one to many, many to one and many to many relations between source and sink nodes have different characteristics in terms of delays, clock synchronization and interaction. And speaking of interaction, another important aspect of multimedia systems, that directly relates to synchronization, is user

interaction. The ability of the user to have access to common VCR functions must be accommodated. Moreover, in conversational multimedia applications, causality must be preserved to ensure comprehensible rendering.

2.6 Synchronization Specification

Synchronization specification aims at describing the temporal dependencies among and within the multimedia objects so that the user application can properly render the multimedia presentation. As previously seen with Steinmetz's synchronization reference model, synchronization specification is closest to the user application; however; the information contained within impacts all synchronization layers as well as those of the networked multimedia system. Synchronization specification comprises two types of information:

- Temporal relations between and within the multimedia data possibly including user interaction.

- Qualitative and quantitative metrics describing the desired degree of synchronization performance.

Temporal information, commonly referred to as a scenario, requires a specification method or language capable of capturing all possible relations among different media, whether continuous or discrete. Subsection 2.6.1 delves a little more on specification methods' requirements. Moreover, it summarizes different approaches to such a problem. Finally, we elaborate on the Time Flow Graph (TFG) method developed at MCRLab [LIK94.1],[LIK94.2].

The qualitative and quantitative metrics describing generally presentation and more specifically synchronization performance are referred to as Quality of Service (QoS) parameters. Subsection 2.6.2 investigates the QoS issue at different levels of the multimedia system. While the emphasis was on multimedia synchronization, the context was stretched to include the whole networked multimedia system. The main reason for doing so is the tight relation that exists between both areas as was illustrated in section 2.5.

2.6.1 Synchronization Specification Methods

Synchronization specification methods provide a formal way of describing synchronization information. This, on one hand, defines an interface between the multimedia presentation and the multimedia application for proper rendition. On the other hand, it allows inconsistencies to be detected early enough to prevent runtime errors. Among the many requirements for such formal description discussed in the literature, we outline five we feel are most relevant:

- The specification method must be able to accommodate both time-independent and time-dependent media. The accommodation should be transparent in the sense that media are to be similarly abstracted at the object layer; while their peculiarities are to be preserved at the stream and media layers.

- The specification method should allow for the inclusion of QoS information, at least those immediately tied to synchronization performance.

- The specification method must span all possible types of temporal relations between and within media objects. This implies coverage of both inter- and intra- media synchronization relations. This aspect should be flexible enough to specify composite

relations involving more that two media objects as is usually the case in a real multimedia document.

- The specification method should provide constructs to describe complex and hierarchical scenarios. Often, multimedia documents are composed of other smaller multimedia documents. These can be combined in a linear, hierarchical or any plausible way. In fact, it is desirable to handle each document as one entity somehow providing for an encapsulation of synchronization information.

- Finally, and possibly optionally, the specification method might support user interaction during rendition as is the case in conversational multimedia. This incorporates an element of decision making in the specification. As a result, dynamic scenarios might arise as the rendition progresses.

To summarize the previous points, one needs to point out the following observation: The specification methods must span the object, stream and media layers as defined in Steinmetz's reference model. In [BLA96], Steinmetz and Blakowski classify synchronization specification methods into three categories:

Interval based specification: This is based on the abstraction of time as intervals. Therefore, the specification methods based on this notion use the interval duration of an object as a synchronization element. There are thirteen possible ways to synchronize two time intervals as suggested in [HAM72]. These can be reduced to seven, if we exclude those which are inversible, as shown in figure 2.6.1 [LIT90.1]. By complementing these relations with measures of time, any type of synchronization constraint between two objects can be described.

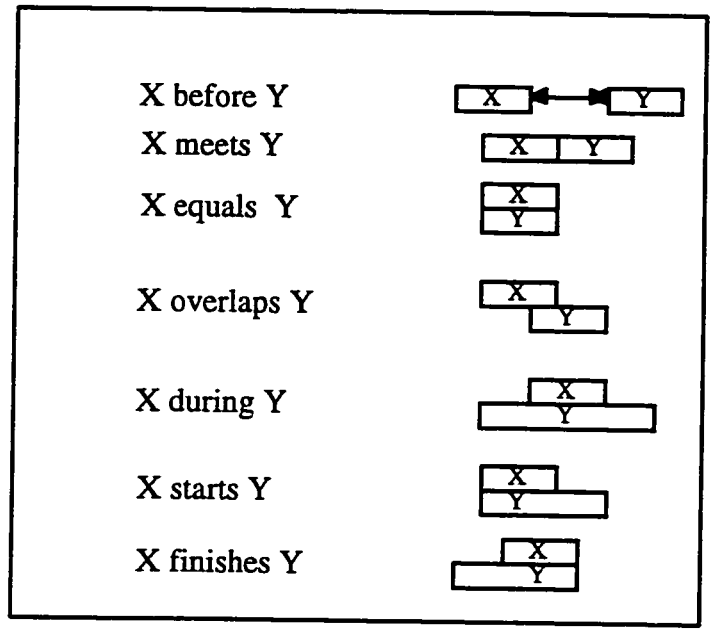

Figure 2.6.1 Types of Temporal Relations Between Two Time Intervals

Axes-based synchronization: This is based on the time point as the abstraction or the synchronization primitive. The synchronization events are mapped independently onto points in time on a shared time axis (figure 2.6.2). This allows for the incorporation or removal of media objects without disrupting synchronization between other objects comprised in the presentation. The time axis could be the actual real time, suggesting some sort of global timer based synchronization. Additionally, the time axis could be generalized to specify user defined time units. This virtual axis approach is useful to control the rendition rate of a presentation at runtime. For instance, a piece of music synchronized with its musical notes could be played at different speeds depending on the mapping of a virtual time axis onto a real one.

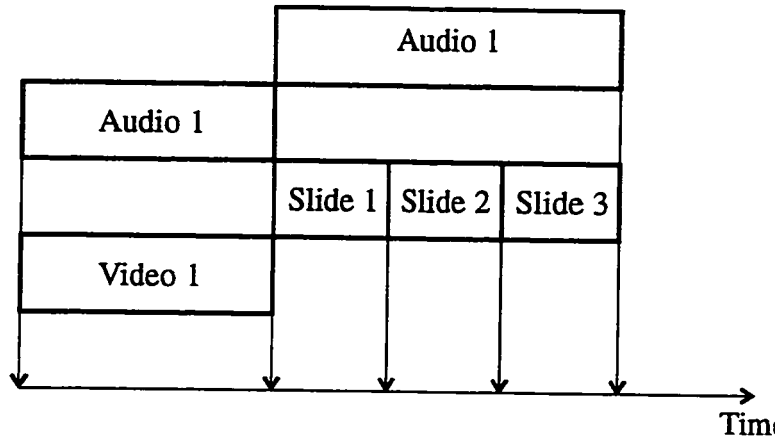

Figure 2.6.2 Time-axis Based Synchronization Specification

Control flow based specification: Unlike the axes-based specification where synchronization points are instants in time, in the control flow based specification, synchronization points are defined and triggered by the concurrent flow of different presentation threads. This allows for flexible specification of synchronization by somehow loosening (stretching) the relation between time and the flow of the presentation. Among the control flow based methods we find the reference point method [STE90] and the timed petri net one [LIT91.1].

As an example of a specification method, we illustrate the Time Flow Graph (TFG) method, introduced by Li in [LIK94.1] and developed at MCRLab. Based on the above classification the TFG method, while based on temporal intervals, falls more into the control flow class. A TFG is defined to be a directed graph represented by $\{\Delta_N, N_t, E_d\}$ tuples where:

- Δ_N is a set of nodes for the interval vectors or objects of the scenario, with each node having the duration of one interval object.

- N_t is a set of transit nodes, with zero duration, used to link different interval objects.

- E_d is a set of directed edges connecting all nodes and describing the sequential flow of the presentation.

As a result of this definition, an interval object $\vec{n}_x$ (or vector in Li's words) can be specified as a node N_x of Δ_N with duration equal to that of $\vec{n}_x$ and a number of δ_x nodes of Δ_N specifying the flexible intermissions between $\vec{n}_x$ and other object intervals. Therefore, $\vec{n}_x$ can be represented by one of the following forms:

$(\delta_x^{(p)}, N_x, \delta_x^{(f)})$, $(N_x, \delta_x^{(f)})$, $(\delta_x^{(p)}, N_x)$ or N_x.

Moreover, an additional node N_τ of Δ_N having a duration τ is needed to describe fixed intermissions between interval objects. This is the case in interval relations before and overlap.

Transition nodes, N_t, are employed for the sequential specifications of relations between activities (sub-graphs in the TFG). There are two such relations:

- {A} B: Interval(s) in activity B will start after all the intervals in activity A are finished.
- <A> B: Interval(s) in B will start when one of the intervals in A is finished

corresponding to these relations a square-node and a triangular node are used respectively (figure 2.6.3). based on these there are two important nodes in N_t: N_s to delimit the start of an activity and N_e to delimit the end of it.

In addition to the definition of the TFG, there are a number of properties or constraints that are to be enforced when constructing TFGs in order to ensure the consistency and correctness of the model being developed. The interested reader is referred to [LIK94.1] for details.

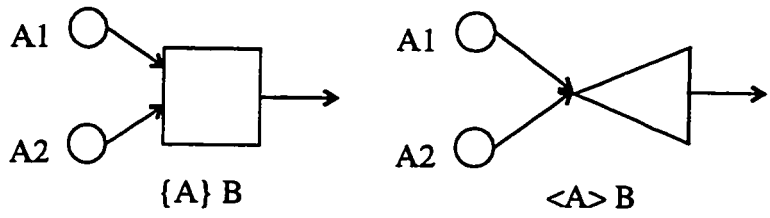

Figure 2.6.3 Types of Transition Nodes

Using the above definition , figure 2.6.4 shows the seven basic interval temporal relations and their corresponding TFG model.

As a modeling language, the TFG presents a unique characteristic that distinguishes it from a number of control flow methods. Often times, the duration of objects composing a multimedia scenario is not known apriori. Using flexible intermissions, the TFG method is able to specify the scenario in what is called the relative synchronization domain. Later, before or during the rendition of the presentation, this relative scenario can be translated into the absolute synchronization domain, when exact durations are obtained. Because of the properties of the TFG, the synchronization relations are preserved in the transition. Consequently, the TFG allows the specification of *fuzzy* scenarios. Moreover, it allows for the incorporation, at a later stage, of synchronization QoS parameters in terms of skews. The idea of relative and absolute synchronization domains imposes a dialogue between different entities involved in the multimedia system.

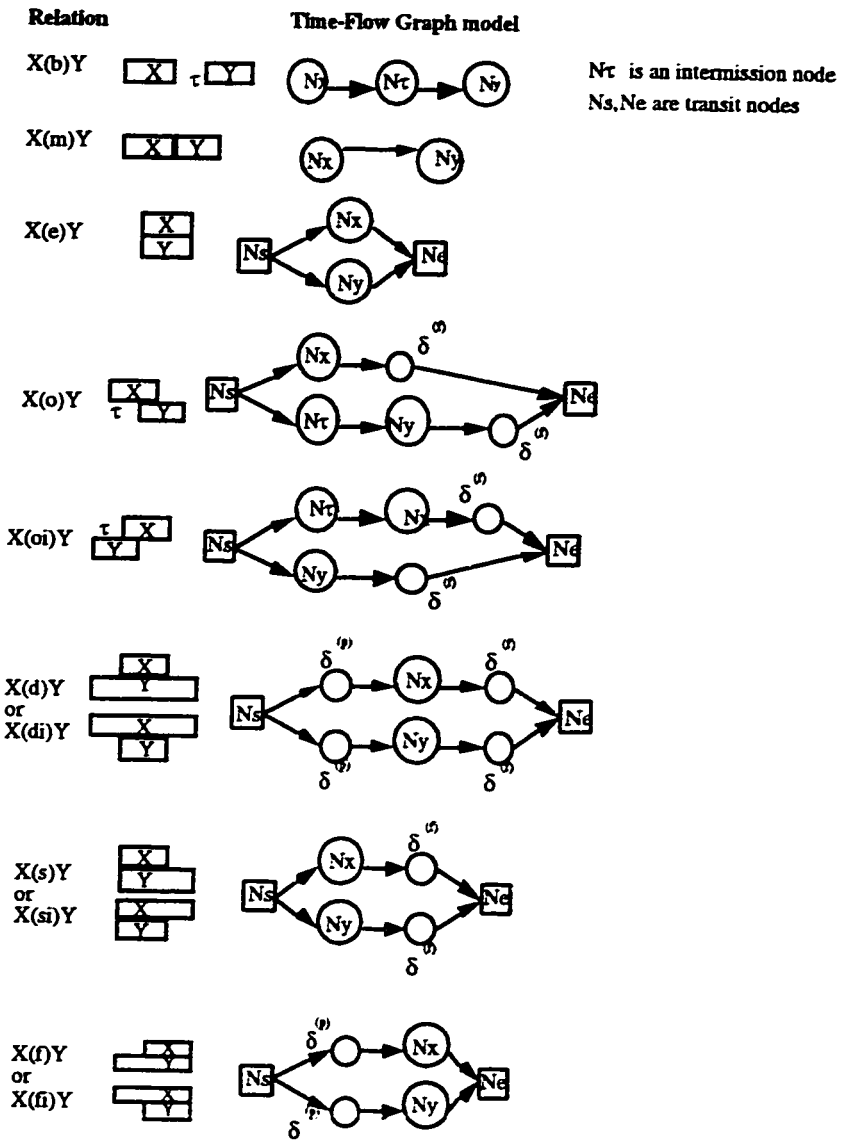

Figure 2.6.4 TFG Model of Basic Temporal Interval Relations

2.6.2 Synchronization and Quality of Service

Part of the synchronization specification are qualitative and quantitative metrics to describe the desired rendering performance for satisfactory user perception. These metrics or characteristics are more commonly known as Quality of Service (QoS) parameters. The notion of QoS originated in the communication field to describe technical characteristics, such as error rate, which were not meant to be verifiable or even observable by the user

application. As time-dependent data became prevalent in multimedia applications, this QoS view evolved into one that requires all levels of the entire distributed system to participate in guaranteeing the desired performance. Under this new approach, the user application provides a set of QoS requirements, which are then conveyed to the various components of the multimedia system in the form of QoS parameters. Based on this view, Vogel et al, in [VOG95], propose the following definition of QoS:

> "Quality of service represents the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of the application".

QoS parameters can be dissected into five different categories as shown in table 2.6.1 [VOG95]. Most notably is the synchronization oriented parameters, which as will be discussed later are timing related parameters, possibly under full control of the presentation environment. Note that, while parameters belong to different categories, they might still be interdependent and possibly contradictory. For instance, a high resolution video will require a high transmission bit rate, skyrocketing thereafter the connection's cost.

Category	Example Parameters
Performance-oriented	End-to-end delay and bit rate
Format-oriented	Video resolution, frame rate, storage format, compression scheme
Synchronization-oriented	Skew between audio and video
Cost-oriented	Connection and data transmission charges and copyright fees
User-oriented	Subjective image and sound quality

Table 2.6.1 Categories of QoS Parameters

Issues related to QoS can be broadly divided into three related activities [VOG96]:

- Assessing QoS in terms of user subjective wishes or satisfaction with the quality of the application performance, synchronization cost, etc.
- Mapping results of the assessment onto QoS parameters for various system components.
- Negotiation between system components or layers (embedded protocols) to ensure that all system components can meet the required QoS parameter values consistently.

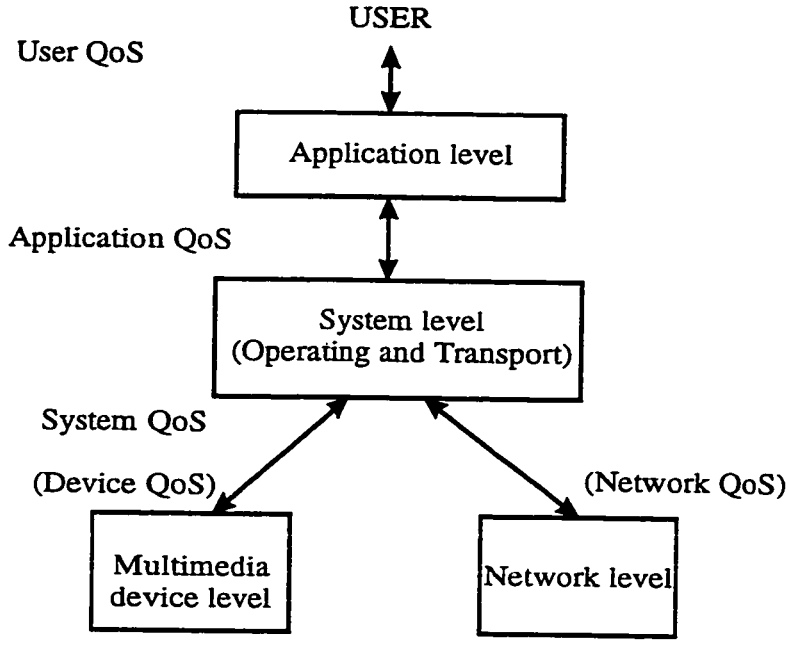

Figure 2.6.5 QoS Abstraction in a Networked Multimedia System

These activities can be put into perspective if we consider the three abstraction layers below the user composing a networked multimedia system as outlined in figure 2.6.5 [NAH95]. Each of these three layers provide services which are parametrized with corresponding QoS metrics. The first activity relates to the user and application layers, which in many ways can be blended together. The second and the third activities correspond to the arrows linking different layers allowing for mapping (downward direction) and renegotiation (upward direction) of different

QoS parameters. In the next three subsections, we elaborate on each of these layers from a QoS point of view. We mostly focus on the application layer, where the emphasis will be on synchronization.

Application QoS parameters: Application QoS can be regarded as the quantitative description of qualitative user QoS. Therefore, both can be regarded as two faces of the same coin. In general, application QoS are usually specified in terms of media quality and media relations [SMI95]. Media quality involves quantitative metrics of media characteristics. For example, sample size and sample rate are two important measures of audio quality. Similarly, frame width, frame height, color resolution and frame rate describe important characteristics of video objects. Media relations quantify synchronization constraints between and within different media. Therefore synchronization QoS parameters involve timing relations that can be controlled by the presentation environment. In this context, Steinmetz uses media jitter and media skew, as defined earlier in this chapter, as the main synchronization QoS parameters [STE96]. At the IBM European Networking Center in Heidelberg, he conducted an important experiment to assess the tolerable levels of media drifts in human perception of media synchronization. His results show that skews between related media streams may still give the effect that the data is still "in-sync" and give some constraints under which jitter may be tolerated. For example, "lip synchronization", referring to the temporal relation between audio and video during human speech, could tolerate skews of +/- 80 ms, depending on which media is ahead of which. Moreover, pointer synchronization with audio in CSCW application can tolerate a skew of up to 750 ms. In terms of media jitter, video was found to be more tolerable than audio because of the higher sensitivity of the human ear compared to the human eye. In fact, at the change of a video scene, we can easily drop up to 15 frames without

disturbing the viewer. Figure 2.6.6 summarizes the findings of Steinmetz. While his experiments considered only skew between related media or jitter within a particular medium, Steinmetz points out the complex nature of multimedia scenarios involving several media objects at the same time. In this regard, he proposed an aggregation method to derive tolerable skews between several media and between two media using their tolerable skews to another reference medium. Other work on the human perception of synchronization can be found in [KUR94] and [APT95]. Also, more detailed QoS metrics and values can be found in [MMC95].

Media		Mode, Application	QoS
video	animation	correlated	+/- 120 ms
	audio	lip synchronization	+/- 80 ms
	image	overlay	+/- 240 ms
		non overlay	+/- 500 ms
	text	overlay	+/- 240 ms
		non overlay	+/- 500 ms
audio	animation	event correlation (e.g. dancing)	+/- 80 ms
	audio	tightly coupled (stereo)	+/- 11 us
		loosely coupled (dialog mode with various participants)	+/- 120 ms
		loosely coupled (e.g. backgroung music)	+/- 500 ms
	image	tightly coupled (e.g. music with notes)	+/- 5 ms
		loosely coupled (e.g. slide show)	+/- 500 ms
	text	text annotation	+/- 240 ms
	pointer	audio relates to showed item	- 500 ms + 750 ms

Figure 2.6.6 QoS For Inter-media Synchronization

A different approach to stream continuity and synchronization QoS specification is found in [WIJ96]. While the authors do not present values relating to human perception of synchronization, as in [STE96], they present a more rigorous framework of QoS for continuous media. The authors qualify continuity of a continuous media stream with sequencing, display rate and drift profiles. The sequencing profile consists of tolerable aggregate and consecutive frame miss ratios. The rate profile specifies the average rendition rate and its variation. Given a rate profile, the ideal time unit for frame display is determined as an offset from the start of the stream. The drift profile specifies the average and bursty deviation of schedules for frames from such fixed display points in time. Once each media continuity profile is specified, synchronization requirements of a collection of continuous media streams are specified by mixing rate and drift profiles of each media stream. The resultant are vectors of frames that can be displayed simultaneously. Consequently, sun specification itself consists of rate and drift profiles. Synchronization rate profiles consists of average rates and permissible deviations. Synchronization drift profiles specify permissible aggregate and bursty time drifts between schedules of simultaneously displayable frames. The advantage of this approach is that it presents a flexible and unified interface to specifying continuous media QoS requirements independently of the application involved. Furthermore, it "faithfully" (to replicate the authors' words) quantifies qualitative description of continuous media streams. This approach can be seen as complementary to the work done on human perception of synchronization and constitutes another milestone towards a comprehensive specification of synchronization QoS.

System level QoS parameters: System QoS parameters describe transport and operating system performance aspects that will guarantee the application level QoS parameters.. The operating

system provides the runtime environment for both the application and the transport protocol. The transport system handles all processing related to data and stream connections. Therefore, system level QoS quantify and qualify the processing requirements at the end systems, in order to meet the desired performance of the application for the specified user perception quality.

QoS constraints on operating systems mainly relate to heir real time behavior. This is due to the often stringent timing requirements of continuous media processing. One difference, however, with traditional real time systems is that there is more tolerance to errors in multimedia processing. Operating system QoS can be low level or high level. Low level parameters include CPU time, bus speed, available memory, decoder buffer space, etc. High level parameters relate to process scheduling and resource reservation [STE95].

Transport QoS constraints aim at maintaining a steady delivery of multimedia data and streams through network connections to the user application. This is quite important in the case of presentational multimedia applications because irregularities in the data flow will cause degradation of the audio and video quality. Examples of transport QoS parameters are transport unit maximum size, throughput, end-to-end transient delay and transport buffer size. Other qualitative transport QoS parameters include the ordered delivery of data and error recovery mechanisms. Guaranteeing some of these parameters requires resource reservation in the end system and at the network level. Issues related to transport QoS are detailed in some approaches to multimedia transport protocols such as the ST-II protocol [DEL94] and the Resource Reservation Protocol (RSVP) [ZHA93].

Network QoS parameters: Network QoS parameters describe the performance of the network connecting the end multimedia systems. Typical network QoS parameters are bandwidth, bit

error rate, packet loss, end-to-end delay and packet delay variation. Other more complex parameters are peak data rate and burst length. Different value combinations of network QoS parameters will result in different network service classes relating to the nature of the data being transmitted. The ATM network protocol provides explicit facilities for handling QoS within this signaling protocol making it an ideal choice for distributed multimedia applications.

Device QoS parameters: Device QoS parameters describe the rendering multimedia devices such as audio and video hardware. Typical QoS parameters in the device domain are timing and throughput demands for multimedia data units.

Chapter Three

Predecoder Synchronization Controller: Algorithms

3.1 A Multimedia News-on-Demand Teleservice

News, as defined in the English language dictionary, refers to recent events and happening, information about such events or the presentation of such information. In reality, news is a blend of all three definitions revolving around the word information. News play an important role in our contemporary society, providing a basis for decision making, culture, entertainment, etc. Earlier throughout history, news were conveyed mainly through speeches and oral conversations. With advances in mass printing technologies, news papers and magazines became the premier medium of news exchange providing a richer content of text, graphics and pictures. Later on, the radio brought audio content to news casting and live information to the public. The advent of television lead to video , coupled with audio, becoming the main news medium to

date. Although these different technologies were and are still being used to convey news, they are all of a broadcast nature. In other words, they do not allow a person a full control over the news he wants to receive and when to receive it. In an increasingly fast paced society, this is about to change. In fact, the convergence of media production, computer and networking technologies is paving the ground for a new breed of news, a fully interactive Multimedia News-on-Demand (NoD) service. Such a service will allow people to custom tailor their news digest, with a rich content of video, audio, animation, text and graphics, at any time they feel convenient. Figure 3.1.1 depicts the overall architecture and composition of a system that supports such a service.

3.1.1 The Media Production Center

The media production center is responsible for providing the news content for the system. Its main functionalities involve:

- The capturing and the reception of news from different sources, such as a camera, a satellite link or a microphone.

- The authoring and editing of multimedia news stories, which will adhere to common multimedia document structure and use compression standards for storage and playback.

- The publishing of the created content by transferring it via the broadband network to the multimedia distribution center, where it is stored in the various media servers.

Other responsibilities of the media production center will be to update and discard stories based on time. Different media production centers might exist as is the case with different news agencies, providing multiple views of information and richer diversified content.

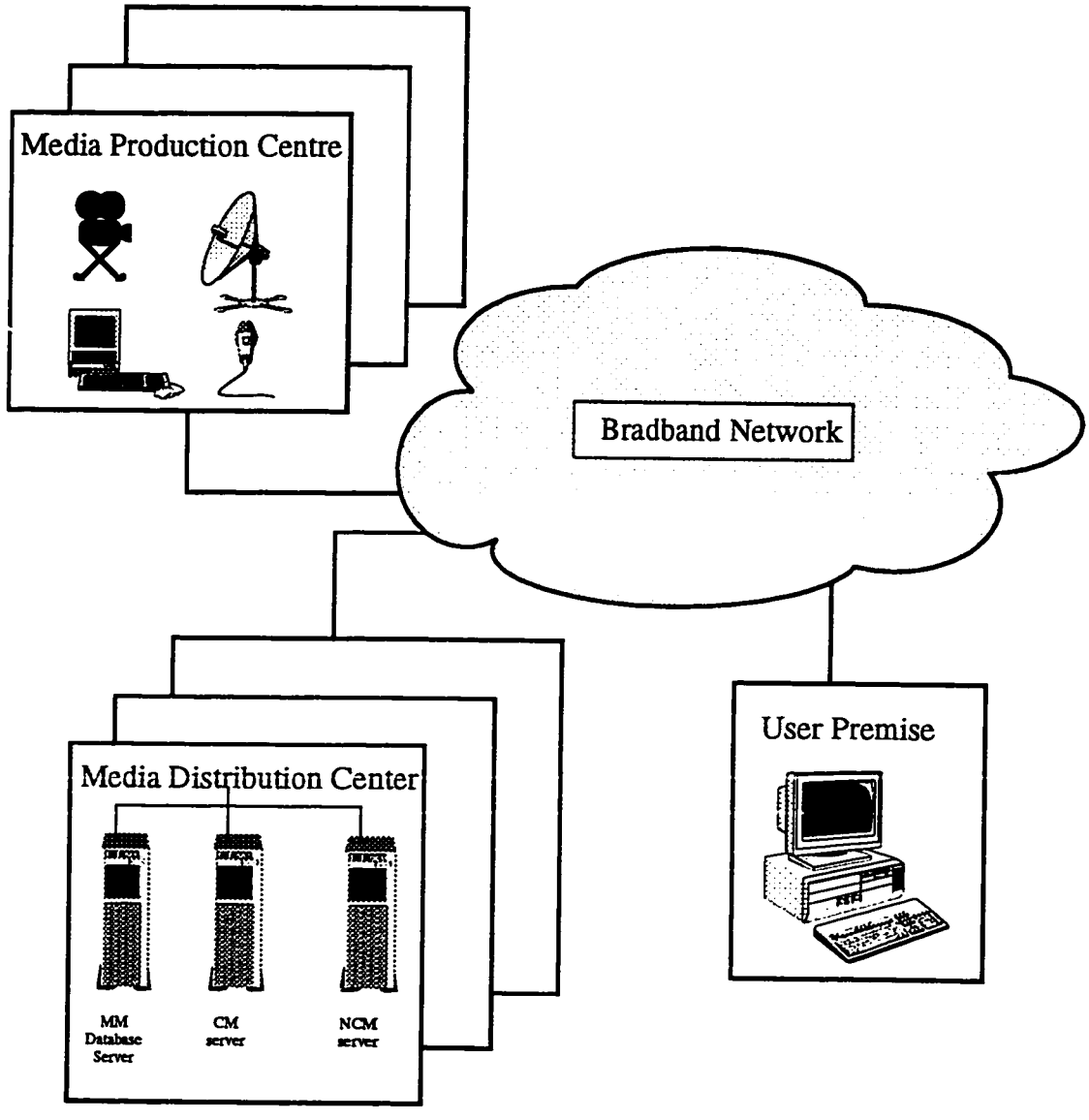

Figure 3.1.1 Architecture of a System Supporting a Multimedia News-on-Demand Teleservice

3.1.2 The Media Distribution Center

The media distribution center acts as a storage for the multimedia news articles published by the media production center. In addition, it provides transmission and session management services for users of the multimedia NoD service. The media distribution center consists of a multimedia database, continuous media and non-continuous media servers. The multimedia database keeps meta information on the multimedia news articles such as the rendering scenario of the article and the location of its components. It also provides browsing and search capabilities to the user. Furthermore, the users will have personalized profiles outlining their preferences and access levels. The continuous media servers (CMS) are used to store continuous media components of the multimedia articles, namely, audio and video. Due to the real time nature of video and audio the CMSs feature advanced scheduling capabilities for handling multiple connections and requests. Moreover, the CMS supports interactive functionalities, such as fast forward or backward. Finally, the non-continuous media servers hold non continuous media, such as text and graphics. The different servers composing a media distribution center need not be in the same location and could be scattered at different nodes across the network. Moreover, multiple servers holding the same articles could exist at the same time to prevent server overload conditions.

3.1.3 The Communication Subsystem

To support the high bandwidth requirements of a multimedia NoD service, suitable transport and access technologies must be used for the network interconnecting various components of the system. The Asynchronous Transfer Mode (ATM) technology seems

to have established itself as the transport of choice for broadband networks. ATM combines advantages of both circuit and packet switched networks. Moreover, it supports resource reservation and different traffic characteristics through statistical multiplexing, allowing for guaranteed QoS parameters. In terms of access technologies, NoD sessions require continuous transfer of huge amounts of data. For instance an NTSC quality MPEG stream requires 1.5 Mbits/s. Several signaling technologies are currently available for home access at such high speeds. The Asymmetrical Digital Subscriber Loop (ADSL) provides users with a 1.536 Mbits/s downstream connection and 16 Kbits/s upstream connection (ideal for control and interactive signaling) on existing twisted copper-pair wiring. The Cable TV (CATV) technology allows enough bandwidth for hundreds of simultaneous connections and is widely deployed in North America. However, it requires some adaptation to allow bi-directional signaling. The multimedia NoD service could also be accessed through local area networks, in which case Local ATM (LATM) and Fast Ethernet are the most likely candidates.

3.1.3 The User End

The user is a centric entity in the design and specification of a multimedia NoD service. Therefore, the user equipment must be capable of rendering multimedia articles in an acceptable manner. A modern mid-range personal computer system with a sound card, speakers, an MPEG capable video card and a high speed cable or ADSL modem should be sufficient. Another alternative is a TV like appliance (much like recent WebTVs) capable of connecting to NoD servers and executing necessary software for rendering multimedia articles. On top of these hardware platforms, multimedia NoD

application software is required. The application must be easy to use with a trivial user interface. Its functionalities should be accessible with easy to use controls, such as a mouse or a remote control. Aside from the interface, the application must include the ability to handle multiple network connections with different input streams. The rendering of multimedia data, although tolerant for some startup delay, must be carried in real time. In fact, a store and forward approach needs huge, and therefore impractical, buffering at the client. In support for the just in time rendering of the multimedia presentation, a synchronization architecture must be specified in order to avoid and recover from possible asynchronies, introduced by the network, the server or the client processing itself. While this synchronization scheme should span all layers of the multimedia NoD system, the most important synchronization effort is that performed at the application level. The application being the closest layer to the users, the quality of the presentation is more sensitive to how well synchronization is maintained at this layer. In the next section, a synchronization architecture developed at MCRLab to support multimedia NoD applications is detailed.

3.2 System Architecture and Synchronization Mechanisms

This section outlines the system and synchronization architecture developed at MCRLab for the CITR NoD prototype. The work presented here was first introduced in [LAM94.1] and [LAM94.2].

Figure 3.2.1 depicts the overall system architecture of the multimedia NoD system. The user's workstations are connected through an ATM network to a meta-database

server and a number of independent continuous media file servers (audio, video) and non-continuous media file servers (text, images, graphics). The meta-database server stores the different listings of multimedia articles along their presentation scenario. The presentation scenario describes the temporal relations of the different media objects composing the multimedia article. The presentation scenario spans the object, stream and media layers of Steinmetz's synchronization reference model (section 2.4). The various media servers store their related media objects.

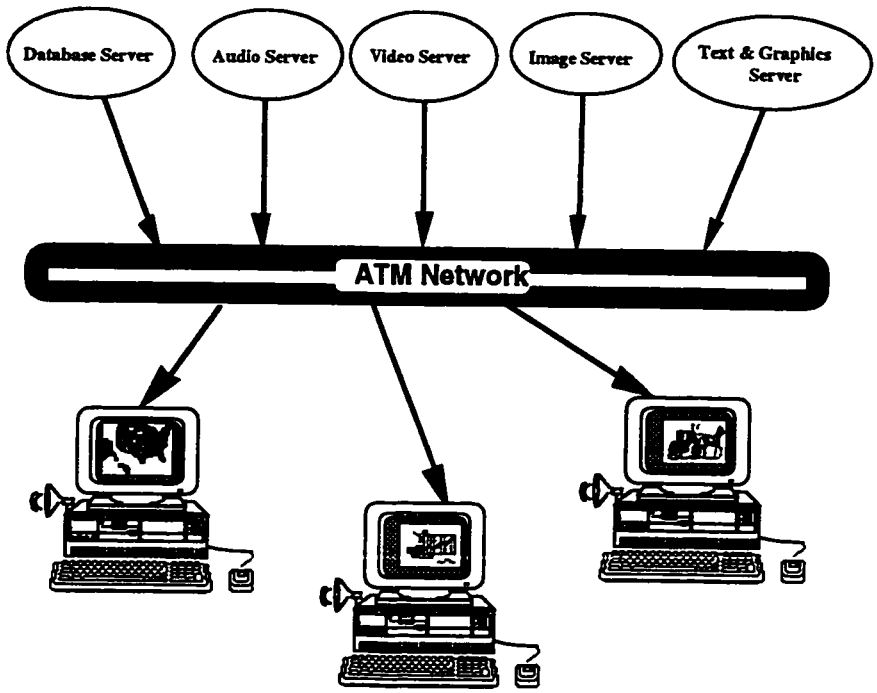

Figure 3.2.1 Synchronization Architecture of MCRLab's NoD Prototype

Typically, the user retrieves the scenario of the desired article by querying the meta-database server. Information about the media object IDs and their location is extracted from the scenario and connections are then opened from the various media servers to the client workstations, without the use of a global clock mechanism.

Different connections are used because it is desirable to transmit streams independently [LAZ90] for two main reasons: different media have different traffic characteristics; and for bandwidth gain using statistical multiplexing of the different traffic classes. The various media streams are delivered and rendered in real time at the client workstations to avoid huge buffering requirements at the client. The processing at the servers and the transmission of the data across the network, during delivery to the user workstations, introduces inevitable random delays and jitters among the various media objects. As a consequence, the continuity of isochronous media streams may be interrupted, introducing gaps in the presentation. Moreover, the different delays experienced on different connections may create mismatching in the presentation, causing a loss of inter-object synchronization. Therefore a synchronization protocol is needed to maintain the intra- and inter- media synchronization in the system. MCRLab's approach to the preservation of temporal relations among and within multimedia streams is two fold:

- An error prevention mechanism is first used. A stream delivery schedule is precomputed for each media server, taking in account the mean delay and delay variance of each connection

- An error recovery mechanism, known as the Stream Synchronization Protocol (SSP), performs resynchronization operations at the client when necessary.

3.2.1 Stream Delivery Scheduling

The stream delivery scheduling aims at minimizing the synchronization errors that might occur during the delivery of the various media objects. This achieved

by computing the optimal media transfer times for each server given the network connection's QoS parameters and other delays that a medium will experience, such as buffering and decoding delays, before its presentation. Figure 3.2.2 shows the delivery scheduling operation.

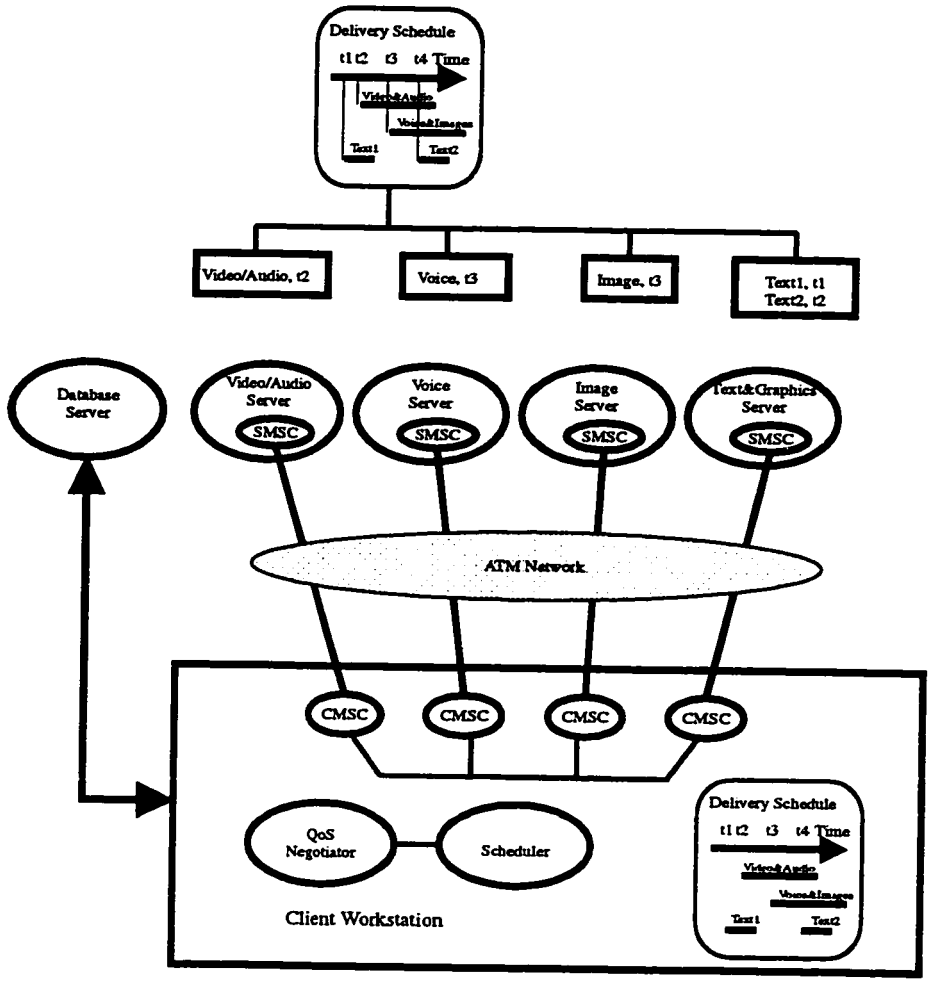

Figure 3.2.2 Delivery Scheduling Operation

A user at a client station will issue a request for a multimedia article through a graphical user interface (GUI). Moreover, the user would have selected, using

the GUI, application QoS parameters such as frame playback rate and picture quality. A scheduler process then retrieves the presentation scenario of the selected article, through the client interface to the meta-database server. The location of the media (both continuous and discrete) servers and the identifiers for the media objects composing the article are also supplied. At this point, the scheduler enters into a QoS parameters negotiation phase, using the QoS negotiation module. The negotiations, involving the client, the severs and the network, will produce QoS parameters needed by the synchronization mechanisms, namely, average delay, jitter and throughput. Once the path is clear for a presentation to begin, the scheduler uses the QoS parameters and estimates for the decoding delays to precompute a delivery schedule for each media stream, based on the Time Flow Graph (TFG) specification method (see section 2.6.1). The delivery schedule specifies the times the media servers should start transmitting data once the start signal is received. The computations aim at compensating the random delays and jitters that different streams might experience during their delivery. The model assumes an underlying network delay distribution with an average end-to-end delay μ and delay variance σ^2. The delivery times, $T_{delivery}$, of each media are obtained as follows:

$$T_{presentation} = T_{delivery} + D_{total}$$

D_{total} is the total delay encountered by the media units and is composed of three components:

$$D_{total} = D_{e\text{-}e} + D_{decoding} + D_{buffering}$$

Where $D_{buffering}$ is a buffering delay to smooth out network and processing jitter and is computed through a failure probability P(fail) as follows:

$$P((D_{e\text{-}e} - \mu - D_{buffering}) > 0) < P(fail)$$

After deriving the delivery schedules, the scheduler sends the delivery schedule to all Server Media Synchronization Controllers (SMSCs) which are responsible for the data connection from each media server to the client.

While the distribution based computations are generic enough, it is often the case that the network delay distribution is not available or hard to predict. Moreover, the decoding delay (in the case of software and possibly hardware decoders -due to network drivers) are not fixed and can vary according to the frame type (as in MPEG) and the client load. In order to overcome these two shortcomings, we propose an alternative derivation of buffering delay based on the bounded delay model in Chapter 4. In addition, our work on coded data streams will minimize the involvement of the decoding activity in the stream delivery scheduling, avoiding the difficult task of decoding delay estimation.

3.2.2 The Stream Synchronization Protocol

Scheduling and traffic prediction are not sufficient to maintain stream synchronization because of random network delay and delay variations and because media servers can start transmitting data relatively at different times due to the lack of a global clock. This results in the disruption of both inter- and intra- media synchronization of the media objects. The scheduler process, therefore, enters phase two of the synchronization mechanism: synchronization error recovery to compensate

for the random disturbance encountered. For each media stream, the scheduler spawns a Client Media Synchronization Controller (CMSC). The concurrent CMSCs share a common presentation schedule based on the TFG model. The CMSCs are required to start their respective media presentation according to the shared schedule. During playback, the CMSCs keep track of the arrival times of the data segments. The CMSCs then use application defined skew tolerance parameters to detect any asynchronies. The skew tolerance parameters refer to synchronization errors tolerated by human perception and are derived from experiments conducted by Steinmetz at IBM in Heidelberg [STE96]. If asynchronies occur, the CMSCs will communicate together via the shared schedule in order to reschedule the current presentation in real time, using the notion of intentional delays. In this fashion, the CMSCs maintain inter-stream synchronization. This error recovery scheme is referred to as the Stream Synchronization Protocol (SSP).

Within the SSP, special attention is given to time-dependent coded data streams. The presence of a decoder at the client and special characteristics of coded time dependent data streams require proper handling at the client to ensure stream continuity and avoid data overflow and underflow conditions. For this purpose, a Predecoder Synchronization Controller (PSC) is specified. The next section describes the PSC mechanism in detail.

3.3 Predecoder Synchronization Controller

In multimedia systems, continuous media streams, such as audio and video, require huge storage space and transmission bandwidth. For example, the speeds of broadcast

quality NTSC video and studio quality NTSC video are about 120 Mbits/s and 216 Mbits/s respectively [LYL92]. Multiply that by the duration of a movie in seconds and you get a feel for the storage space needed to hold such data. The economic accommodation of these requirements with current storage and network technologies is near impossible. The use of appropriate compression techniques is needed to reduce storage and transfer rate requirements, making the integration of continuous media services economically viable with current storage and network technologies. Fortunately , recent advances in the area of multimedia compression during the last few years has made this a reality. Most notably are the Join Photographic Experts Group (JPEG),[JPE93] and [PEN93], for still images and video, and the Moving Pictures Experts Group (MPEG), [MPE93] and [LEG91] for video and audio. For instance, MPEG video coding is capable of providing compression ratios of 100:1 with a picture quality that resembles that of a VCR [FOR93].

Coded video streams are expected to become the predominant type of multimedia traffic transmitted and are one of the most demanding in terms of bandwidth, delay and processing requirements [TS96]. For this reason we will focus our work on coded video streams. In addition to their special characteristics compared to non coded media objects (which we will illustrate in the next section), coded video streams require a decoding entity (hardware/software) at the client. These distinctions dictate special provisions in terms of synchronization control of these streams. While the MPEG standard provides some synchronization mechanisms at the level of the transport stream, this is not the case for individual video bit streams. Moreover, we only consider individual video bit streams because the Stream Synchronization Protocol (SSP) adopted

individual independent connections for different media objects. In addition, the MPEG standard has no protection mechanisms against overflow or underflow conditions at the decoder level relegating it to the implementer's discretion [MPE93].

We propose to blend the decoding of coded streams with the synchronization mechanism. The resultant Predecoder Synchronization Controller (PSC) (due to the fact that it sits before the decoder at the client) ensures the intra-media synchronization (continuity of the video stream) while protecting the client buffer from overflow and underflow conditions. In the next subsections, we investigate the characteristics of coded video data streams and the need for the PSC. Then, the design goals and the control parameters for the PSC are presented. Finally, we specify PSC algorithms for two types of coded video streams: Motion JPEG and MPEG video bit streams.

3.3.1 Characteristics of Coded Video Streams

Compression techniques take advantage of special characteristics (or limitations) of the human visual system to achieve lossy, yet visually lossless, compressed images and video. In fact, the human eye is more sensitive to changes in brightness than to color changes. As a result an image could be divided into a luminance component and two chrominance signals. The chrominance signals can then be sampled at a lower resolution yielding some compression gain. Further compression can be achieved by using other techniques such as entropy and source coding. On one hand, entropy coding is a lossless process which considers the stream as a simple digital sequence and the semantics of the data are ignored. An example of entropy coding is the runlength coding technique.

Source encoding, on the other hand, is a lossy process that take into account the semantics of the data. Source coding might make use of the spatial redundancies in the picture or might transform and process the image from spatial domain to a two dimensional frequency domain. In general, image and video compression use a hybrid of source and entropy encoding as is shown in figure 3.3.1, which depicts the basic image encoding steps. The image processing and the quantization steps represent the source coding part and the last stage further compresses the resulting data using entropy coding [GEO94].

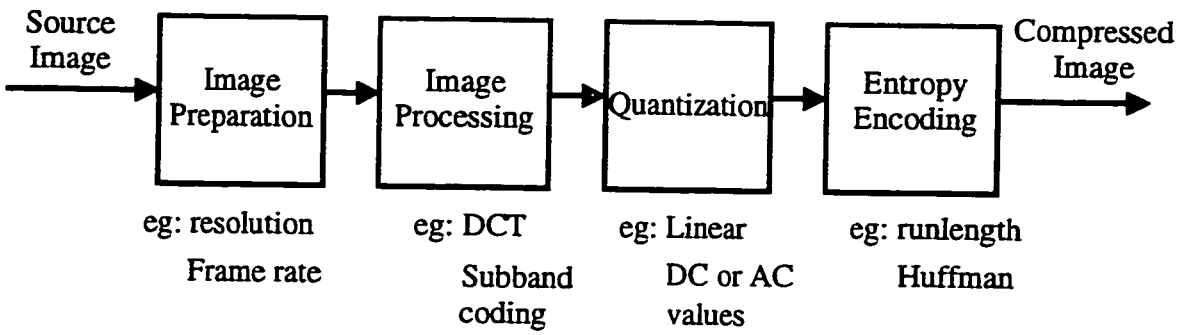

Figure 3.3.1 Basic Image Encoding Steps

After the compression stage, the resultant coded video stream differs significantly from the initial uncompressed video stream in several key communication aspects. These characteristics are outlined below:

Throughput: The throughput of the coded stream is usually orders of magnitude lower than the original stream. For example, a 100 Mbits/s full motion uncompressed video could result in a 2 Mbits/s MPEG-2 coded stream or a 10 Mbits/s MJPEG coded stream (assuming a 320x240 frame for MJPEG). While the resultant bit rate is still high, The this allows the real time transmission of the video streams over current ATM networks.

throughput of the coded video stream is directly related to the compression ratio achieved. The compression ratio depends on the desired quality of the decoded image and the coding technique being used. For instance MPEG achieves a higher compression that MJPEG by taking advantage of inter-frame redundancy. The compression ratio only holds as a measure of performance if the output image is comparable in size and quality to the original uncompressed image. A more objective way to specify the compression achieved is the number of bits per displayed pixel needed in the compressed stream.

Nature of the coded video stream: There are two different types of video bit streams depending on the variation of the compressed bit rates: Constant Bit Rate (CBR) and Variable Bit Rate (VBR). The content of the input video stream significantly influences the bit rate of the coded video sequence. High action scenes, the frequency of scene changes, pans, zooms and the picture quality lead to changes in the bit rate of the coded video stream over time. Therefore the resultant video stream is of a VBR nature. In order to transmit the coded stream over fixed rate communication channels, a buffer is used to smooth the output of the encoder. If the buffer tends to overflow or underflow, the encoding parameters (such as the quantization table) are adjusted accordingly to maintain a CBR stream at the output. Therefore, CBR coded video streams have a variable picture quality compared to VBR coded video streams. VBR video streams also offer the advantage of allowing statistical multiplexing over communication channels making it a more attractive choice than CBR video for future applications. Both VBR and CBR coded video streams have variable frame sizes resulting in different decoding and display rates. However, buffering requirements for VBR streams are more difficult to predict due to the bursty nature of the stream.

Structure of the stream: While the input uncompressed video stream might be seen as a sequence of appended fixed size frames, the output coded video stream could be structured differently. This is the case in the MPEG coding standard where three different types of frames are defined: I (Intra-coded), P (Predictive coded) and B (Bidirectionally interpolated) frames. Furthermore, each of these frames is subdivided into smaller macro-blocks which are further divided into blocks. This results into a sort of syntactical hierarchy in the coded stream that could be extended further by considering groups and sequences of frames as higher layers. Moreover, the fact that some types of frames depend on others for decoding leads to different transmission, decoding and display orders, further complicating the buffering at the client. The advantage of stream structuring is to allow for easy browsing and information extraction from the stream. Other advantages include the provision for scalability to different resolutions within the same stream.

Robustness: Robustness refers to the vulnerability of the data stream to data loss. In general uncompressed video streams can tolerate a small amount of loss provided that it meets its stringent delay requirements as shall be discussed later. For instance, a 30 frames/s video could easily tolerate a 5 frames loss resulting in a 25 frames/s video with little perceptible change to the human eye. However, due to the high compression ratios achieved, the coded stream will have stringent data loss requirements. Important data might be lost that not only could damage one frame but also could affect the quality of a number of frames in the sequence. Certain coding standards, such as MPEG, could recover from data losses by having the decoder estimate or substitute the data lost using

data from neighboring frames. This could prove costly in processing time at the client while not necessarily yielding the derived picture quality.

Delay sensitivity: Coded video streams, like their uncompressed counter parts, are extremely delay sensitive due to the continuous nature of the stream. Strict timing relations must be met during the transmission of the stream in order to avoid gaps in the presentation. For example, in conversational multimedia, a maximum end-to-end delay of 150 ms must not be exceeded [STE94]. However, in presentational multimedia applications, a coded video stream could be a little more tolerant to startup and end-to-end delays, relegating the issue to a fundamental tradeoff between increased delay and buffering versus reduced bandwidth. Nevertheless, in all multimedia applications, coded video streams are very sensitive to delay variations (or jitter), which must be bounded or guaranteed by the network. Part of the end-to-end delay encountered by coded video streams is the compression (in the case of conversational) and decompression delays. In conversational multimedia, both delays must be real time, i.e. smaller than the time required for the presentation of one media unit (frame). For instance, in a conferencing application with video streams of 15 frames/s, the decompression/compression delays must be lower than 66 ms. In presentational multimedia applications, only decompression must be carried in real time, allowing for off-line compression with probably a higher quality for the pictures. Current hardware technology allows for both real time compression and decompression of MJPEG and MPEG video streams. However, the compression boards are still prohibitively costly for widespread use. With increasing CPU power and memory bandwidth, real time software decompression of coded video streams is now feasible.

In [PAT93], Rowe et al. reported that their software implementation of a MPEG-1 decoder was able to decode an MPEG-1 30 frames/s (320x240 pixels) video in real time over a HP-750 RISC workstation.

3.3.2 Need For A PSC

All media streams handled by their respective MSCs need application level buffering at the client. This buffer is different from those required at the network socket which are typically allotted by the operating system or the networking hardware. The main purpose behind application level buffering is to have sufficient media units available for synchronized startup of the rendering processes. Although these buffers are allocated in bytes, one can look at them as pools of media units, where a media unit is the smallest presentable granule such as a frame for video.

As discussed in the precious section, Time-dependent coded video streams are very sensitive to delay and delay variation. Moreover, they are more vulnerable to data loss than non coded streams. In addition, they may be of a variable bit rate (VBR) nature making it difficult to accurately estimate their buffering requirements. furthermore, they may have a different presentation, transmission and decoding orders which is the case with MPEG video. Also, coded video streams have a high throughput, which is even orders of magnitude higher after decompression, necessitating fast real time processing at the level of the coded stream in order to avoid huge buffering needs at the client. Finally, the presence of a decoder entity be it software or hardware further distinguishes continuous coded video streams from the others. Consequently, a proper buffer control and management mechanism, referred to as PSC, must be specified to:

- Isolate the decoder's internal buffer from network fluctuations by smoothing out jitter and random delays. This is particularly important in the case of hardware decoders, which might crash if their internal buffer is not filled at a regular basis.

- Avoid data starvation or data overflow conditions at the client's buffer. Data starvation causes gaps in the presentation, while data overflow can cause the loss of important data necessary to decode not only one but several upcoming frames.

- Prevent network data readers from stalling due to filled buffers. This may cause data loss at the network interface.

- Compensate for clock drifts (in frequency) between the server and the client which might result in underflow or overflow situations due to different rates of production and consumption in the system.

- maintain stream continuity (intra stream synchronization). In fact, an occasional frame drop might not be as noticeable as a frame freeze due to starvation.

- And finally, communicate with the MSC (in case of software decoder) to adapt to decoder processing load conditions. The decoded stream may be as much as 20 times faster than the original coded stream. Therefore, it is inconvenient for the MSC to drop decoded frames or to have huge buffers to contain the decoded bit stream. If the decoder is not capable of matching the playback rate , media units should be dropped by the PSC to save on processing time and buffering space.

3.3.3 Design Objectives

In order for the PSC to fulfill the functionalities specified in the previous section, several design goals must be met. The design objectives which were considered are enumerated below:

- The PSC mechanism must be easily adaptable to work with both hardware and software decoders. This may involve slight architectural changes in terms of the positioning of the PSC with respect to the decoding entity. However, this should not violate the main functionalities of the control mechanism, namely, protecting from overflow and underflow conditions and maintaining stream continuity. More on the architectural constraints are discussed in the next chapter detailing the implementation of the PSC.

- The PSC is a client-based control scheme. Typically, media servers will be handling hundreds of connections simultaneously. Therefore server feedback would not be a viable solution in our case because of the extra load and implementation complexity on the server. Should conditions deteriorate at the client due to the degradation of network conditions or due to insufficient processing power at the client, a safer solution would be to renegotiate the quality of service parameters of the connection through the QoS negotiation module.

- The PSC must be light-weight: the algorithm and its implementation should be far from being CPU and resource strains. The PSC should not be a rendering bottleneck, rather it should act as a rendering enabler. This is especially valid for the case where the decoding entity is implemented in software. The actions

undertaken by the PSC must be fast in order to free the CPU for the decoding process.

- Finally, the PSC should be adaptable to conversational multimedia applications. Despite the current focus being on a presentational multimedia application, the functionality of the PSC should be easily extensible to conversational multimedia. Moreover the PSC should be portable, in the sense that the algorithms should be incorporated readily and easily in the design of other presentational multimedia applications, such as Video-on-Demand.

3.3.4 Control Parameters

Any control scheme is bound to have a number of control parameters upon which the mechanism acts or reacts. The PSC is no exception to this rule. We identify three control parameters: the Buffer Occupancy, the Presentation Time Stamp and the Buffering Delay. Each of these is discussed in depth in the following paragraphs.

Buffer Occupancy: The main control parameter used by the PSC is buffer occupancy. Buffer occupancy refers to the ratio of filled buffer space in bytes to the total buffer space allocated, at any point in time during the multimedia presentation. We define an upper and lower thresholds for buffer occupancy (figure 3.3.2). Buffer occupancy high (BOH) refers to the higher threshold. Buffer occupancy low (BOL) refers to the lower one. The following relation holds true between BOH and BOL:

$$BOL = \text{total buffer space} - BOH$$

This merely states that both buffer regions below BOL and above BOH are equal in size. If at a certain point in time during the presentation the buffer occupancy falls below the BOL line, then there is a threat of buffer starvation. If buffer occupancy crosses the BOH line, then there is a threat of buffer overflow (figure 3.3.2). Both these conditions trigger appropriate actions detailed in the following subsections.

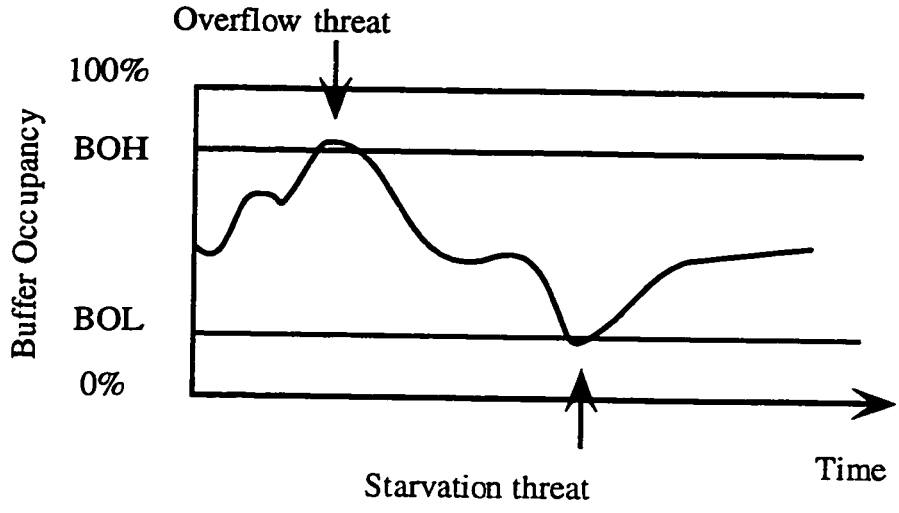

Figure 3.3.2 Buffer Occupancy For PSC Buffer

Note that a distinction must be made between starvation or overflow threats and real starvation or overflow conditions. In fact, the threat might lead into a real condition or it might not. The width of the buffer region lying between the BOH and BOL lines has a direct impact on the probability of real buffer starvation or overflow. On one hand, the wider the region the less overflow and underflow threats detected. Therefore, should one threat occur it might be too late to trigger appropriate actions to avoid the real starvation or overflow. This means that the wider the region between BOH and BOL, the higher the probability of real starvation or overflow. On the other hand, the narrower the region, the more underflow or overflow threats detected. Therefore,

the PSC will have a larger pool of media units to handle and avoid the actual threat becoming a real condition. This means that the narrower the region between BOL and BOH, the lower the probability of real starvation or overflow occurring. These observations dictate a proper dimensioning of BOH and BOL for the smoothest user perception at playback.

Presentation Time Stamp: The presentation time stamp refers to the ideal time instant a media unit or a frame should be rendered. The presentation time stamp of a video frame is derived based on the frame display rate of the video stream and the relative start time of the presentation of the stream. For example, if the presentation started at time t_0 seconds and the frame rate of the video is R frames/s, then the presentation time stamp of frame number n is $(t_0 + n/R) + t_p$ seconds. The t_p components accounts for the time period required by the display process to actually render the image and is usually a measurable constant value.

The MSC in order to catch up with the presentation, might instruct the PSC to drop some late media units, irrespective of buffer occupancy. This saves on the processing by the decoder (software and hardware alike). Moreover, if the decoder is ahead of the presentation schedule, the MSC might instruct the PSC to slow down the input to the decoder (software case) in order to avoid large buffering of the uncompressed stream.

The Buffering Delay: Finally, the buffering delay is a startup delay to smooth out the jitter introduced by the network and the operating system between media units, thus disrupting inter-media continuity. In section 3.2, a computation of the buffering delay

by Li based on network delay probability distribution model was presented. Due to the limitations of this model, we present an alternative derivation, based on the bounded delay model, in the next chapter. The concern in here is that the buffer space for initial startup delay must be higher than the BOL threshold and lower than the BOH threshold. This will ensure normal commencement of the presentation.

3.3.5 MJPEG PSC

JPEG (Joint Photographic Experts Group) is an international standard, jointly developed by the International Organization for Standardization (ISO) and the International Telecommunication Union-Telecom. Sector (ITU-TS), for continuous-tone still image compression, for both color and gray scale images. The JPEG compression algorithm is a lossy/lossless coding technique based on the Discrete Cosine Transform (DCT). It can compress still images to ratios of 10:1 up to 50:1 without visibly affecting image quality. The JPEG standard defines four compression modes:

- Sequential: In this mode each image is encoded in a single left-to-right, top-to-bottom scan. This lossy sequential DCT based mode is the simplest and must be supported in both hardware and software implementations.

- Progressive: In this mode, the image is encoded in multiple scans. This is helpful for applications in which transmission time is too long and the viewer prefers to watch the image building in multiple coarse-to-clear stages.

- Lossless: This mode guarantees exact recovery of every sample value of the source image. However, the compression efficiency is significantly

lowered. This mode may be helpful in applications where the exact image should be recovered, such as medical imaging applications.

- Hierarchical: In this mode, the image is encoded at multiple resolutions. The lower resolution images are accessible without having to decompress the image at full resolution.

A lower resolution (typically 320x240 pixels) fast coding and decoding sequential mode version of the JPEG algorithm is used for the coding of video sequences. This is known as Motion JPEG (MJPEG). MJPEG video streams have no temporal coding dependencies between successive frames, since all frames are coded in JPEG. Consequently, MJPEG allows for fast indexing and easy access of video frames. Moreover, MJPEG video streams can be compressed and decompressed in real time using hardware boards, making it an attractive solution for teleconferencing applications.

For the purpose of the PSC, we first considered MJPEG video coded streams. Since there are no dependencies between MJPEG frames (as described above), the frames are presented in the order they are received. The PSC constantly monitors the buffering occupancy each time a frame is ready for presentation. If the buffer occupancy exceeds BOH (overflow threat), a time contraction policy is used; namely, the PSC will drop frames in an alternate manner until the buffer occupancy falls to within normal bounds. The alternation of frame dropping ensures a smoother transition for visual perception. if the buffer occupancy falls short of BOL (starvation threat), a time expansion scheme is used; namely the PSC will repeat each frame twice until the buffer regains its normal level. Again this scheme avoids a total screen freeze due to real starvation. The PSC

keeps track of the number of frames dropped or repeated during the presentation of a data segment. When the buffer occupancy falls back to normal, the PSC will compensate for those frames by either repeating or dropping selected frames to help the MSC recover inter-stream synchrony. The MJPEG PSC is outlined below in pseudo-algorithmic notation:

```
Get frame
Get Buffer occupancy (BO)
if (MSC signals frame late)
  drop frame
if (BO > BOH)
  if (previous frame dropped)
     decode frame
  else
     drop frame
if (BO < BOL)
  if (previous frame repeated)
     decode frame
  else
     hold current and display previous frame
if (BO normal)
  if (frames repeated not nil)
     if (previous frame dropped)
       decode frame
     else
       drop frame
  if (frames dropped not nil)
     if (previous frame repeated)
       decode frame
else
       hold frame and display previous
```

```
if (frames dropped/repeated nil)
      decode frame
```

By using the time contraction and expansion policy, which amounts to localized variations in the average display rate of the stream, not only does the PSC maintain the stream continuity but also it compensates for possible clock drifts between the client and the server. This is so because clock drifts tend either to overflow or underflow the client buffer. By protecting against such conditions the PSC limits the effect of clock drifts to a minimum, hence enforcing the no global clock policy of the Stream Synchronization Protocol. It is also to be noted that the PSC is a prevention mechanism, i.e. it is the exception not the rule. In fact, if no disturbances occur the action of the PSC is reduced to a minimum.

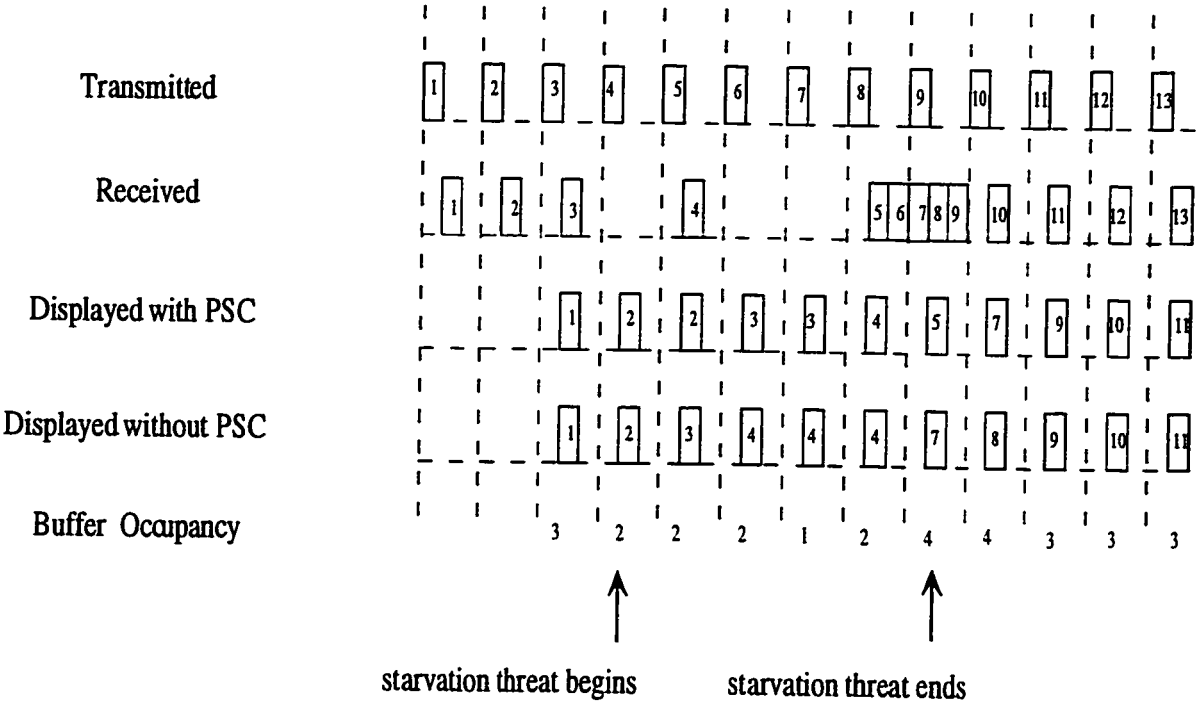

Figure 3.3.3 Stream Continuity For Starvation Threats Using PSC

An example of the PSC avoiding starvation conditions is shown in figure 3.3.3. We assume the buffer at the client to be six frames and BOL to be two frames. The rendering starts with the reception of frame 3. Frames 4 and 5 encounter random delays during their transmission causing a starvation threat to occur at the client. Without the PSC mechanism, the display sequence will freeze at frame 4 for three continuous frames then will abruptly switch to frame 7 due to the discarding of late frames 5 and 6. With the PSC, however, the picture freezing is avoided by repeating frames 2 and 3 during the starvation threat. The abrupt change is avoided by selectively dropping frames 6 and 8 to recover inter stream synchronization.

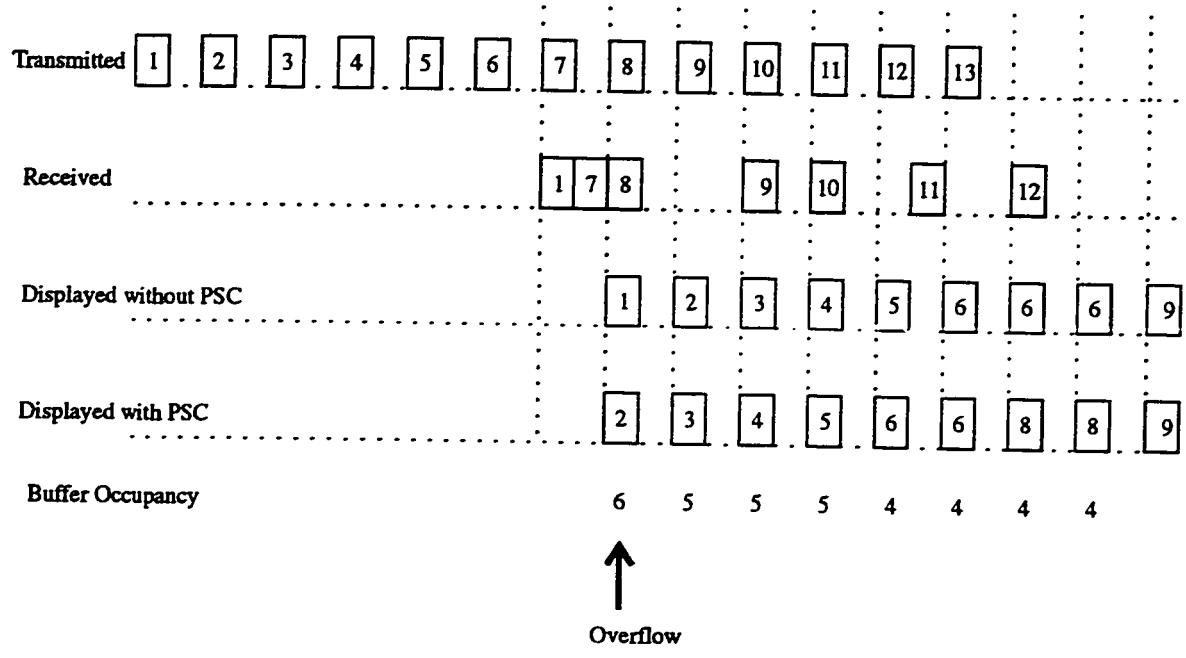

Figure 3.3.4 Stream Continuity For Overflow Threats Using PSC

An example of the PSC avoiding overflow conditions is shown in figure 3.3.4. We assume the buffer at the client to be six frames and BOL to be one frame. The first

frame of the stream experiences a significant unusual delay (about six frames) causing a burst to occur at the client. The burst is frame 1 to 7. For this reason an overflow occurs at the client and frame 7 is lost. The rendering starts one frame unit after the first frame is received causing frame 8 to also be lost when the PSC is not activated. The result is that frame 6 will freeze at frame 6for three time units and then an abrupt change to frame 9 is noticed. When the PSC is activated, frame 1 is dropped to accelerate the stream, hence allowing frame 8 to be buffered. When the buffer occupancy stabilizes to 4, the PSC compensates for frames 7 (lost to overflow) and frame 1 (dropped) by repeating frame 6 and frame 8. This allows inter-stream synchronization to be recovered as well. As can be seen, the PSC avoids a long freeze of the screen and an abrupt change of frames, thereby smoothing the effect of the overflow condition that occurred over the time period affected.

3.3.6 MPEG PSC

After developing a PSC algorithm for MJPEG video streams, we consider MPEG (Moving Pictures Experts Group) video bit streams. MPEG, an ISO/IEC/JTC1/SC29/WG11 standards group, was formed in 1988 to develop a standard for sorting video and associated audio on digital media. MPEG-1, the first standard released in 1993 [MPE93], defines the encoding of video and audio streams for up to 1.5 Mbits/s bandwidth. MPEG-2, the second standard released in 1995 [MPE95], defines the encoding of video and audio streams for up to 100 Mbits/s. The MPEG standards consist each of several documents. The main three are the system, video and audio. The system document defines a syntax for the transport and the multiplexing of video and audio streams. The video document describes

the syntax and the semantics of the video coding algorithm. Finally, the audio document describes the syntax and semantics for the audio coding algorithm.

We concentrate in this work, as explained earlier in the PSC subsection, on MPEG video bit streams. The following discussion holds true for both MPEG-1 and MPEG-2 video bit streams and is not affected by the minor encoding differences of both standards. This is so because MPEG-2 is backward compatible with MPEG-1.

MPEG video coding uses three techniques to compress video data:

- Transform coding, which exploits the insensitivity of the human eye to high frequency visual information, is based on the DCT and allows the image to be represented by fewer values.

- Motion compensation exploits the fact that a frame in the video sequence is likely to be similar to its predecessor and, therefore, can be reconstructed from it.

- Entropy coding is finally used in the form of Huffman coding to further compress the already compressed stream.

Because MPEG uses motion estimation and compensation techniques, three different frame types, depending on the temporal references at encoding time, are obtained:

- I (Intra coded) frames are coded without any reference to other images. In that sense, I frames are similar to JPEG frames. I frames have the lowest compression rate and can be used as points for random access in the MPEG stream.

- P (Predictive coded) frames require previous I or P frames for coding and decoding. This achieves a higher compression ratio than I frames.

- B (Bidirectionally interpolated) frames make reference to a forward and a backward I or/and P frames for encoding and decoding. B frames achieve the highest compression ratio of all three types of frames.

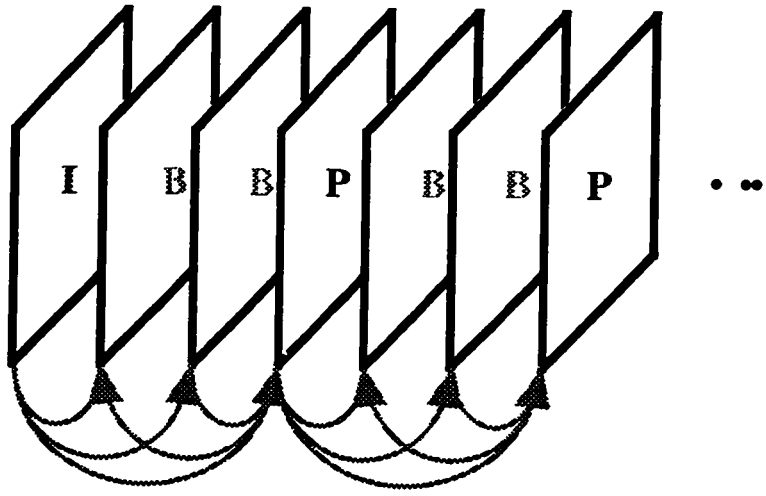

Figure 3.3.5 MPEG Video Bit Stream

Figure 3.3.5 shows an example of a MPEG video bit stream with the temporal coding reference for each frame. The number of B and P frames used in the sequence defines how fast a stream can be randomly accessed as well as the compression ratio achieved. A practical combination which achieves good compression and fast indexing is **IBBPBBPBB-IBBPBBPBB....** In addition to having three different types of frames, the MPEG video stream differs from its MJPEG counterpart in three other ways:

- I, P and B frames have different frame sizes (on average) with a typical bandwidth ratio of 5:3:1. This adds a little complexity to the buffer management process.

- The transmitted stream and the decoding order are not the same as the display order of the frames. For instance, the stream in figure 3.3.5 will be transmitted as follows: **I1 P4 B2 B3 P7 B5 B6 I10 B8 B9** ....This requires additional buffering at the client.

- The MPEG video stream has a layered syntax, unlike the MJPEG stream which consists of adjacent frames. The main layers of the MPEG video bit stream syntax of concern to us are: the sequence layer, which contains general information about the stream, consists of several groups of pictures; the group of picture layer consists of a number of frames and usually allows random access to the first image of the group; finally the picture layer contains a whole picture. Additional layers are, in hierarchical order, the slice, the macro-block and the block layers. Each of these layers contains coding and decoding information for the data contained within.

All these distinctions with the MJPEG stream commands that extra care must be taken when decisions to drop frames are considered by the MPEG PSC. The MPEG PSC is similar in concept to the MJPEG PSC since it is still based on buffer occupancy. However, the MPEG PSC has to handle the extra complexities introduced by the MPEG video bit stream. We devise a selective frame drop/repeat scheme. In the case of overflow threats only B frames are discarded, the number of which depends

on the buffer occupancy and the capacity of the decoder (software case) to match the target frame rate. I or P frames are not dropped because they are needed to decode all subsequent B or P frames until an I frame is reached. In the case of underflow threats, only B or I frames are repeated. P frames are not repeated because the decoder needs two different P or an I and P frame to decode subsequent B frames. The MPEG PSC is given below in pseudo-algorithmic notation:

```
Get compressed frame
If (MSC signal drop late frame)
  if (frame type is B)
     drop frame
  else
     drop next B frame
Get buffer occupancy (BO)
if (BO > BOH)
  drop next B frame in buffer
if (BO < BOL)
  if (frame type P)
     repeat previous B frame
  else
     send frame twice to MSC
if (BO normal)
  if (dropped frames not nil)
     if (current frame I or B)
        repeat frame
     else
        decode frame
  if (repeated frames not nil)
     if (current frame B)
       drop frame
     else
```

```
      decode frame
if (frames dropped/repeated nil)
   decode frame
```

Chapter Four

Predecoder Synchronization Controller: Implementation

The functionalities of the PSC as described in the previous chapter were incorporated in the synchronization module of the MCRLab News-On-Demand prototype for both MJPEG and MPEG video bit streams. The MCRLab NoD prototype was developed by Brimont in [BRI95] with the emphasis on multimedia synchronization at the client. In this chapter, we first describe the hardware setup and the software tools and components used to develop the PSC modules. We then compare the original prototype to the new modified or augmented prototype (with PSC capability) in order to: first relate our work to that in [BRI95] and second to illustrate the changes/additions carried out. The third section of this chapter describes a stream structure developed to overcome the server limitations of the original prototype. Based on this structure, computations for buffer dimensioning to avoid starvation and overflow at the client

are derived. The final two sections of this chapter consider the implementation of the MJPEG and MPEG PSCs in detail. It is noted that only details pertinent to the implementation of the PSC are presented. For a general description and detailed software component interactions of the whole prototype, the reader is referred to [BRI95].

4.1 Hardware and Software Setup

The development of the prototype along with the PSC modifications was carried out on two IBM RISC/6000 workstations (model IBM power station 360). One workstation served as a server and the other as a client. The workstations were equipped with 64 Mbytes RAM, 4 and 1 Gbyte SCSI hard disks (server/client) and 20" IBM SVGA monitors. Additionally, the client workstation was equipped with an IBM Ultimedia audio capture & playback adapter and an Ultimotion JPEG graphics adapter capable of rendering MJPEG video to an NTSC interface at 15 frames/s (320x240 pixels). Both workstations were connected using 100 Mbits/s ATM adapters via a Newbridge Vivid local ATM switch to the OCRInet metropolitan ATM network. They were also connected to MCRLab's Ethernet local area network.

On the software side, the two workstations were running IBM AIX 3.2.5 operating system (IBM's version of UNIX). The majority of the development was done using traditional UNIX system programming and interprocess communications functionalities. The software was compiled using IBM xlC C/C++ compiler from the IBM CSET++ for AIX version 2 package. The MJPEG display process was developed using IBM Ultimedia Services 2.1 for AIX, an Application Programming Interface (API) to support

video and audio services on IBM AIX operating system [ULT94]. More precisely, this was done using the Movie Capture and Playback Objects API. The MPEG decoder and Display process was based on a public domain MPEG-2 decoder/encoder implementation by the MPEG Software Simulation Group, a affiliate of the MPEG standard setting group ([MPEG95] Document 5). The MPEG-2 decoder was modified in order to be integrated with the prototype and to accommodate the PSC functionality.

4.2 The Original Versus The Modified Prototype

As discussed and illustrated in section 3.2 (figures 3.2.1 and 3.2.2), the MCRLab prototype consists of a server side and a client side. The server side consists of a database server and media servers. The client side provides delivery scheduling and synchronization recovery. In this section, we contrast the original implementation as described in [BRI95] to the new prototype after the integration of the PSC functionality. The emphasis will on the modifications made and the reasons these modifications were done. In the process, the limitations of the original prototype are discussed.

We begin the discussion with the server side. Some minor changes were made to the Database Server, particularly to the presentation scenario structure and manipulation part, in order to accommodate a larger number of media segments and media relations. The reason being that the MJPEG test clip, a 4 minutes 15 frames/s CBC news clip, was segmented to 4 second interval LDUs for the purpose of inter-stream synchronization at the client as opposed to 10 second intervals used in the original prototype for MPEG video. The original database structures and file formats only

accommodated 99 segments. This was increased to 999 segments allowing for a more flexible segmentation of a video or audio clip. Moreover, the Server Media Synchronization Controllers (SMSCs) were found to transmit their segments (or data objects) as soon as they can without any rate control whatsoever. This lead to high buffering requirements at the client (more than 10 second for each media). In addition , this lead to a traffic pattern from server to client that peaked each 10 seconds with the rest of the interval idle. In order to overcome this limitation, a stream structure was developed to allow some rate control during the transmission at the server. This stream structure is outlined in the next section. Moreover, the original SMSCs made no distinction between Ethernet and ATM networks in terms of TCP packet size. The packet size used was 1024 Bytes each. While this was convenient for Ethernet due to the standard 1500 byte limitation on the MAC packet of the transmission medium, the ATM performance suffered due to the windowing flow control mechanism of TCP requiring frequent acknowledgments by the client's transport system. To overcome this limitation, the TCP packet size for ATM transmission was increased to 8192 Bytes.

At the client side, The original prototype (figure 4.2.1) only supported MPEG video streams. It was then modified to include the rendering of MJPEG video streams (figure 4.2.2) using the Ultimotion JPEG graphics adapter. The MJPEG PSC and MJPEG MSC were combined into a single process because the decoding entity is in hardware. More on this will be outlined in section 4.4. Also, a MJPEG display process was written and integrated in the prototype to send coded JPEG frames to the hardware decoding board. The MPEG video stream handling in the original prototype had a major design flow both in terms of inter- and intra- stream

synchronization . In fact, the MPEG decoder and display processes were kept as a single entity, which was completely independent from the MPEG MSC (Video Fifo_CMSC in figure 4.2.1). The only link between both processes is a FIFO channel which feeds the decoder/player pair with an MPEG video stream at playback, in addition to termination/pause signals. The MPEG MSC, in this case, has no control on the time the frames are sent to the screen, relying on a decoding time estimate which is highly variable in the case of software decoding.

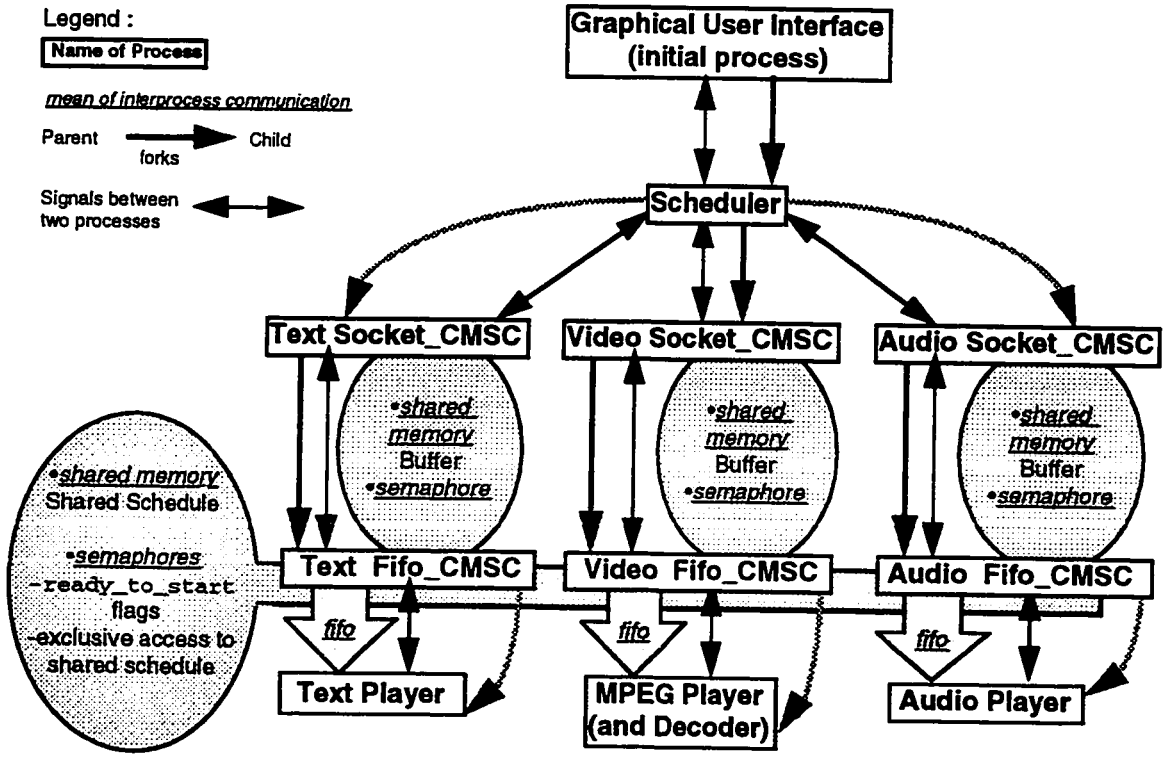

Figure 4.2.1 Client Architecture For Original Prototype

For this reason and in order to incorporate the MPEG PSC into the prototype a major redesign was done. The PSC was combined with the decoder entity, which now deposits decoded frames into a second shared memory space (figure 4.2.2). The MPEG MSC

was combined with the display process and now handles decoded frames instead of a coded stream. This architecture enables full control over the timing of frame display since the only delay incurred by the display process is a fixed dithering delay. More details about the MPEG PSC and MSC implementation will be given in section 4.5.

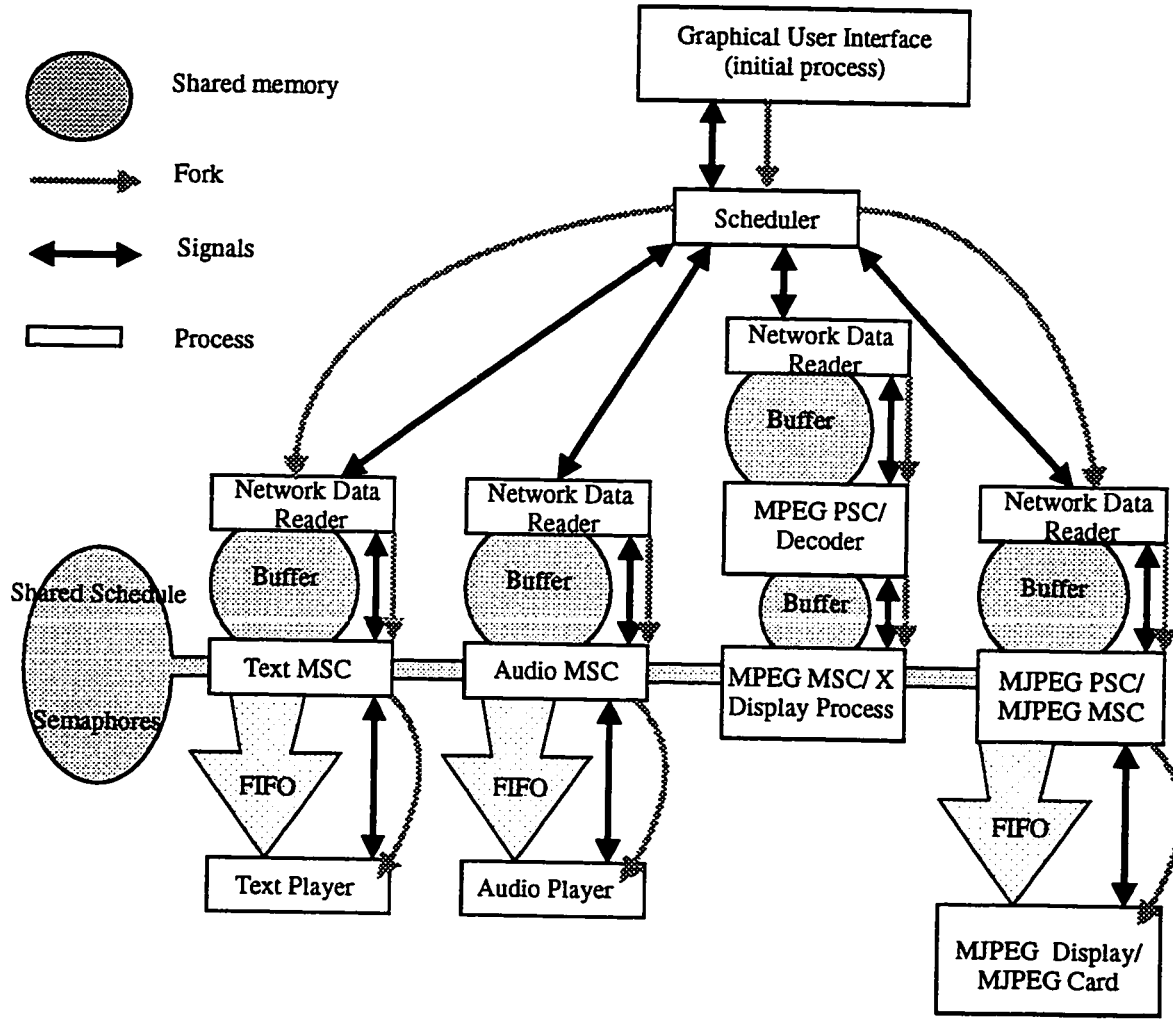

Figure 4.2.2 Client Architecture For Modified Prototype

4.3 Stream Structure

Figure 4.3.1 describes the stream structure transmitted by the server to the client. A logical data unit (LDU) is the smallest inter-media synchronization granule. It is used for synchronizing video with audio at playback. Each LDU is composed of *M* logical transmission units (LTU). Each LTU is composed of *N* data units. The transmission period between successive LTUs, say number *i* and *i+1*, spans the time required to render the data units of LTU *i* at the client. For a MJPEG stream the data units are actual frames and the period between transmission of successive LTUs is *N/R* where *R* is the frame rate of the stream.

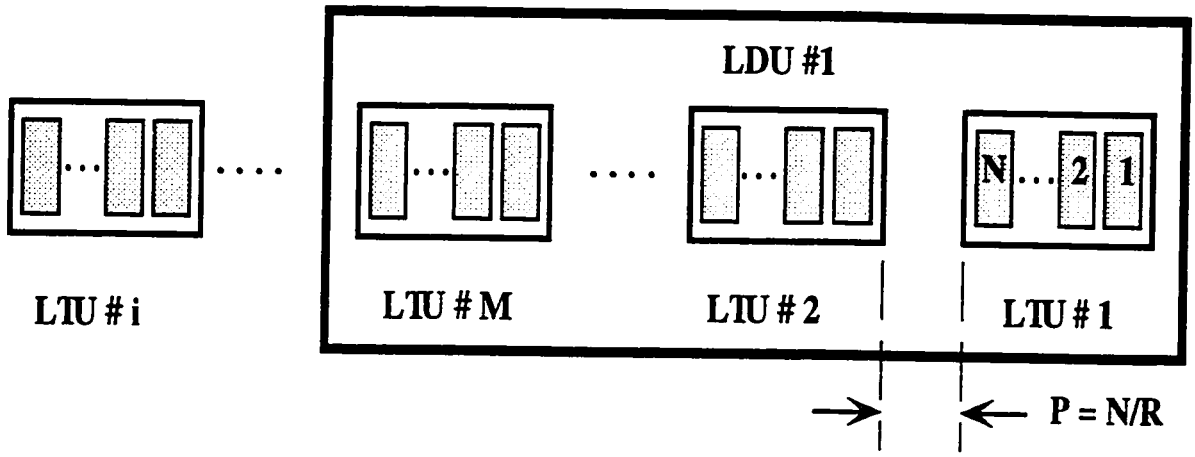

Figure 4.3.1 Stream Structure

For MPEG streams, due to its layered syntax, the data unit's size is chosen as the internal buffer size of the decoder, typically 2048 bytes. The period between the transmission of successive LTUs is a lower bound estimate on the number of frames contained in the LTU divided by the frame rate. The stream structure described here simulates a constant bit rate transmission of the video stream in the sense that it ensures

that the data transmission spans the presentation period evenly, therefore reducing the buffering requirement at the client. The choice of M and N has a direct impact on inter- and intra- media synchronization. A large M implies a longer data segment and might perpetuate inter-media asynchronies. A small M implies more frequent inter-media synchronization at the client. A large N implies higher buffering needs at the client while a small N might affect intra-media synchrony.

Based on the above stream structure we propose to compute the minimum buffering delay to avoid starvation threats and the maximum buffer size to avoid overflow threats. It is important to make the distinction between the buffering delay which is a fixed startup delay to smooth out network jitter and the buffer size itself which can be several orders of magnitude higher than the buffering delay. The computations are based on the bounded delay model which assumes that the network delay is guaranteed not to exceed a maximum of Δ_{max} and is subject to a minimum (due to propagation and buffering at network switching nodes) of Δ_{min}. This assumption is quite valid for ATM networks due to characteristics like resource reservation, QoS negotiation and reliable transmission. This, however, does not mean that the PSC or even the SSP functionalities become obsolete. In fact these entities are by definition exception handling entities which are always needed due to the statistical nature of the underlying network parameters. This being said, The maximum jitter that a frame might encounter during transmission is defined to be (here we adopt the definition of Rangan et al. [VEN 95]):

$$J_{max} = \Delta_{max} - \Delta_{min}$$

The next two subsections detail the derivation which is similar in nature to that carried in [HUA96], but with a different stream structure and a different synchronization mechanism (in our case a non blocking mechanism with PSC functionality).

4.3.1 Buffering Delay For Starvation Threats

Starvation implies that the client misses the presentation of a media frame due to an empty buffer at the time instant the frame is required. The worst case scenario that might lead to a starvation threat at the client is when the first LTU (as defined in the stream structure part) experiences a Δ_{min} delay, while the remaining LTUs will experience a Δ_{max} delay. Figure 4.3.2 shows this exact situation, Where N is the number of frames in the LTU and p is the period of presentation for one frame.

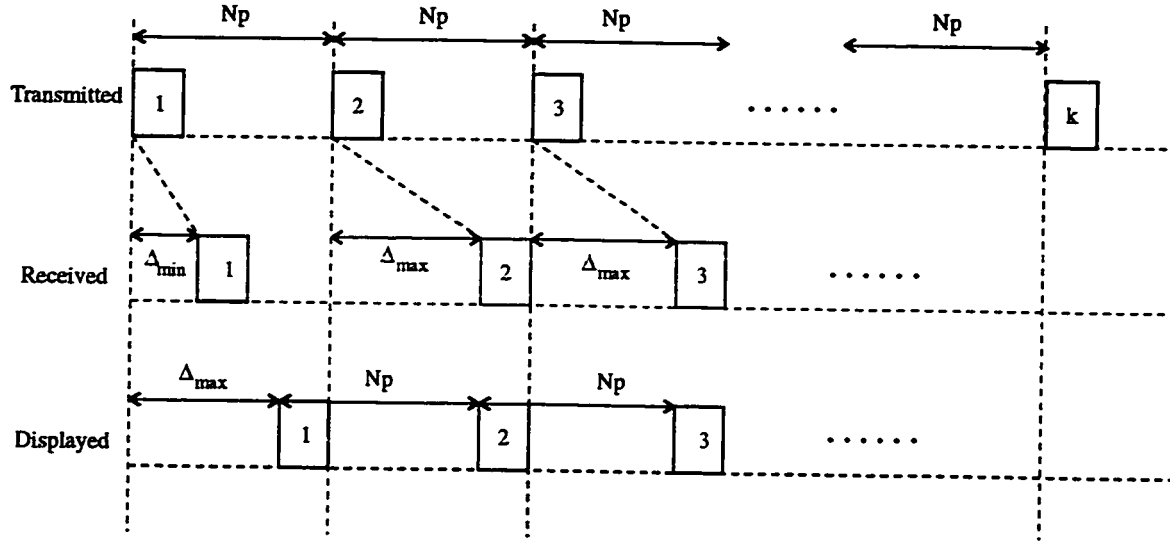

Figure 4.3.2 Worst Case Scenario For Starvation at Client Buffer

It must also be noted that the operating system processing of the data and the decoding time add up to the minimum buffering delay at the client. According to figure 4.3.2, the buffering delay, therefore, should be:

$$\Delta_{max} - \Delta_{min} + t_{decode} + t_{processing}$$

In the case of MPEG video bit streams, the decoding overhead of the first I and P frames must be taken in consideration. This is because if the first I frame is immediately displayed (i.e. after $\Delta_{max} - \Delta_{min}$) the subsequent P and B frames must be decoded in a period p (in order to maintain stream continuity and display frames in order and on time). This might be costly in the case of a software decoder.

4.3.2 Buffer Dimensioning For Overflow Threats

Overflow threats are the result of bursty arrivals (assuming no clock drift involved in this case). These can only occur when the maximum jitter that can be experienced by an LTU is greater than the display period of the LTU. A possible scenario is : an LTU i experiences maximum jitter while all other successive LTUs will experience minimum delay. Hence a burst of LTUs separated by Np will occur at the client while still in display of LTU i, causing a build up of buffer occupancy over time. Eventually, an overflow threat occurs triggering appropriate action by the PSC.

In order to approximate the buffer size to avoid (or limit) overflow threats, we shall distinguish between three time lines as illustrated in figure 4.3.3. The transmission time line corresponds to the time the server starts transmitting data. The reception time line corresponds to the instant the client starts receiving data. The display time line describes the instant the rendering of the multimedia presentation starts following a buffering delay to absorb network and operating system jitter.

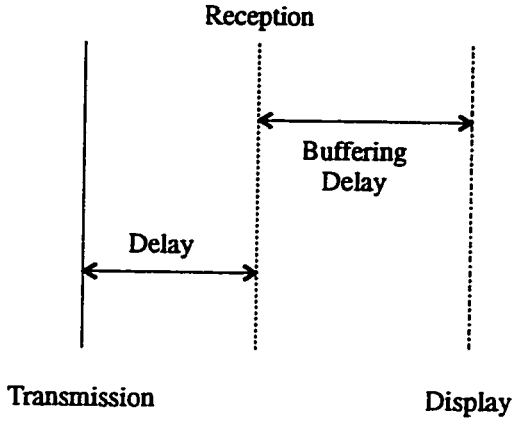

Figure 4.3.3 Important Time Lines For Client Buffer

The scenario described earlier might well become the worst case scenario if we determine the maximum possible buffered units at the reception of LTU *k*. To do that we consider the latest the start of the display time line can be. Clearly, this happens when the first LTU experiences maximum delay Δ_{max}. Therefore, at transmission time of LTU *k*, $(k - 1)Np$, there will be a (Δ_{max} + buffering delay) of presentation to go in unit $k - 1$. Assuming a buffering delay as computed in the previous subsection this results in $2\Delta_{max} - \Delta_{min}$ of buffering. Therefore, at the reception of LTU *k* (which according to our scenario experienced a delay Δ_{max}), the maximum buffer occupancy will be $\Delta_{max} - \Delta_{min}$ of data units. The scenario discussed above is outlined in figure 4.3.4. Let *y* be the number of subsequent LTUs that will arrive during the period of LTU *k*, causing a build up of buffer occupancy. The following holds true:

$$(k + y - 1)Np + \Delta_{min} = (k - 1)Np + \Delta_{max} + Np$$

which yields

$$y = 1 + \left\lceil \frac{(\Delta_{max} - \Delta_{min})}{Np} \right\rceil$$

To which is added a $\left\lceil \frac{(\Delta_{max} - \Delta_{min})}{Np} \right\rceil$ of data already in buffer at the reception of LTU *k*.

Therefore the buffer size to avoid overflow should be:

$$1 + 2\left\lceil \frac{(\Delta_{max} - \Delta_{min})}{Np} \right\rceil \text{ LTUs}$$

or

$$N + 2\left\lceil \frac{(\Delta_{max} - \Delta_{min})}{p} \right\rceil \text{ frames}$$

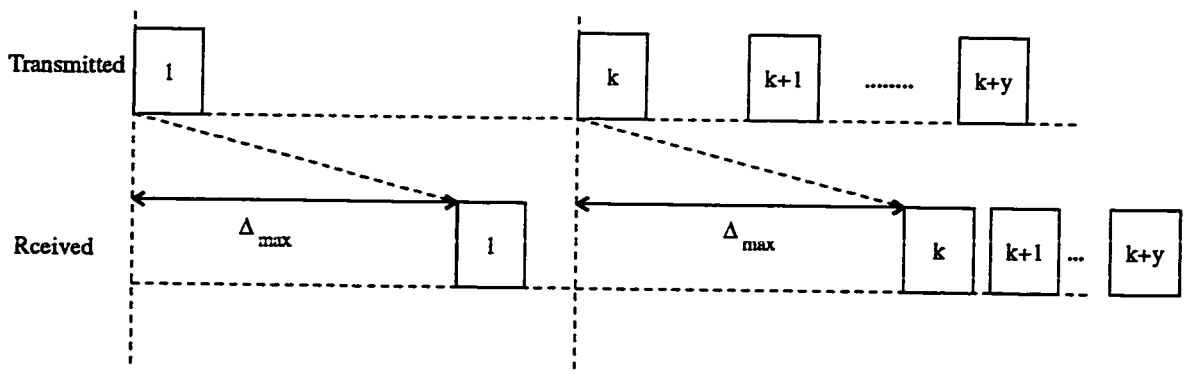

Figure 4.3.4 Worst Case Scenario For Overflow Threats At Client Buffer

A more stringent condition will be to size the buffer to avoid overflow threats. So assuming BOL to be the lower buffer threshold, the buffer size should be:

$$N + 2\left\lceil \frac{\Delta_{max} - \Delta_{min}}{p} \right\rceil (1 + BOL) \text{ frames}$$

4.4 MJPEG PSC Architecture

The nature of the decoding entity has a direct impact on the placement of the MSC and PSC synchronization entities at the client. MJPEG video streams are decoded in hardware with little or imperceptible delay. The PSC and MSC had to be positioned before the decoder in this case (figure 4.4.1). Furthermore, the PSC and the MSC functionalities were grouped into a single UNIX process. The PSC controls the shared memory ring structured buffer space accessed by the network data reader. The MSC controls the sending of the frames to the display process through a FIFO communication link. The display process sends the frames according to their time stamps to the hardware decoder which is connected to a TV screen. The MSC also takes care of inter-stream synchronization by communicating with the other MSCs.

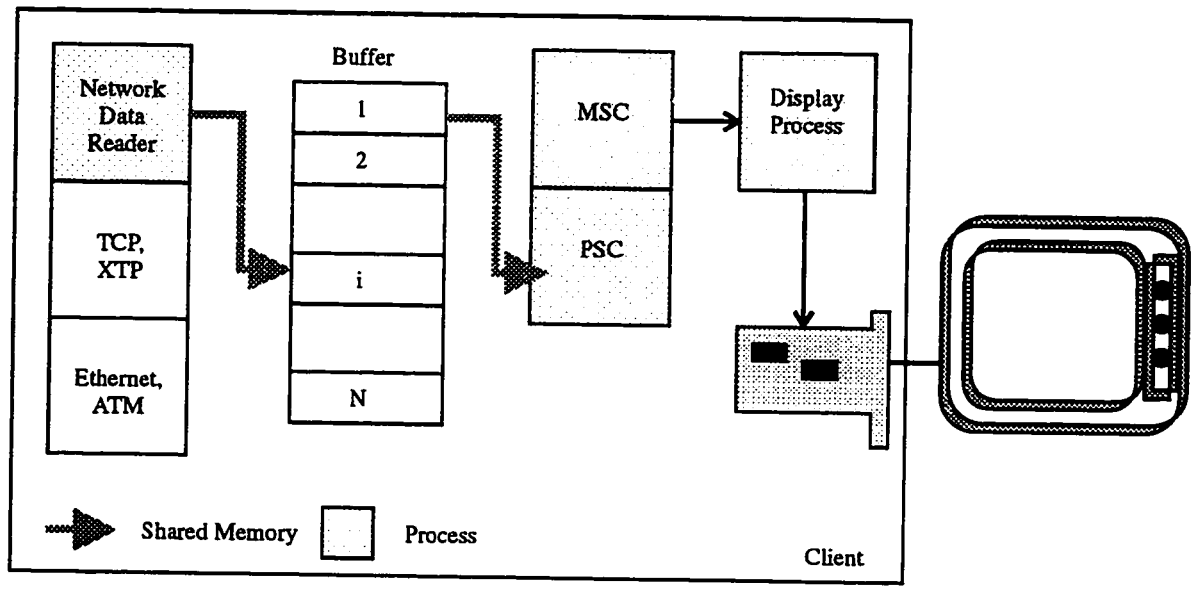

Figure 4.4.1 MJPEG PSC Architecture

The most important implementation details in the case of the MJPEG PSC are: first, the ring structured shared memory client buffer; and second the mechanism used to check buffer occupancy at the client. We describe each of these two aspects in the following paragraphs.

A MJPEG video stream consists of a number of consecutive variable size frames with no syntactical structure linking them. The client does not know the frame sizes in advance and the Scheduler process, therefore, cannot allocate a static shared memory buffer before the article is selected. When the Scheduler retrieves the information about the client from the meta-database server, it also retrieves the number of frames included in the MJPEG stream as well as their respective sizes. The Scheduler then determines the maximum frame size to be allocated for each MJPEG buffering unit. The buffer allocation itself is left to the Network Data Reader process, which is responsible for spawning the PSC/MSC process. Because of this dynamic buffer allocation at runtime, the MJPEG shared memory ring buffered structure was modeled as shown in figure 4.4.2. The MJPEG buffer structure consists initially of a number of MJPEG buffering units. Each MJPEG buffering unit consists of three fields:

- A *written* boolean flag which indicates whether the buffering unit contains a complete frame (1) or does not contain a complete frame (0).
- A *size* integer field containing the size of the written frame
- And a *pointer* to character field which points to the allocated maximum frame size space appended dynamically to the end of the MJPEG buffer structure.

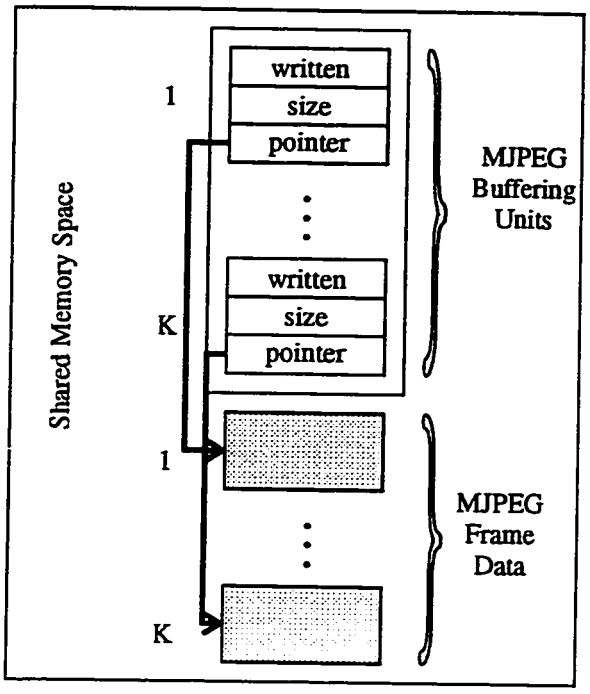

Figure 4.4.2 MJPEG Shared Memory Buffer Structure

When the shared memory structure is initialized the Network Data Reader allocates for each pointer a maximum frame size memory space which is appended to the end of the structure. The space appended at the end of the buffer structure is accessed using the index of its corresponding MJPEG buffering unit and the size of the structure as a whole.

In order to check for buffer occupancy the PSC makes use of a multi-valued Semaphore Inter Process Communication (IPC) primitive called *buffer_occup*. The *buffer_occup* semaphore is shared by the PSC and the Network Data Reader. Whenever, the Network Data Reader finishes writing a frame in the buffer, it sets *buffer_occup* to the value of the index of the buffering unit just written. Before retrieving a frame from the buffer, the PSC calculates the difference between the index of the buffering unit being currently read and the value of *buffer_occup* (modulo the

total number of buffering units). In this way, the PSC keeps track of the buffer occupancy in terms of the number of frames in the buffer relative to the total number of frames the buffer can hold. Ofcourse, whenever a frame is read by the PSC or written by the Network Data Reader, its *written* flag is reset or set accordingly The PSC also makes use of two local flags: *overflow_threat* and *starvation_threat* and an integer value *frames_dropped* to implement the PSC algorithm as described in section 3.3.5.

4.5 MPEG PSC Architecture

MPEG video streams are decoded using am MPEG-2 software decoder from the MPEG Software Simulation Group which was modified to fit into the prototype. The†PSC now lies before the decoder while the MSC lies after the decoder (figure 6). Two shared memory ring structured buffer spaces were needed: the main buffer lying between the network data reader and the PSC and an auxiliary buffer to contain decoded frames before their actual display by the MSC. The PSC was blended with the MPEG decoder in the sense that frames are dropped before decoding and they are repeated after decompression, saving a significant amount of processing by the decoder. In fact, the separation between the two entities in figure 4.5.1 is only a logical one. The MSC was combined with the X Display process into a single entity (again they are separated in the figure for logical purposes). The MSC controls the time the decoded frames are sent to the X display server. Moreover, the MSC, upon its auxiliary buffer starvation, will instruct the PSC to accelerate the stream by dropping late frames, thus the feedback arrow from the MSC to the PSC. Finally, the MSC takes care of updating the shared

schedule and maintaining inter-stream synchronization with the other MSCs in the system.

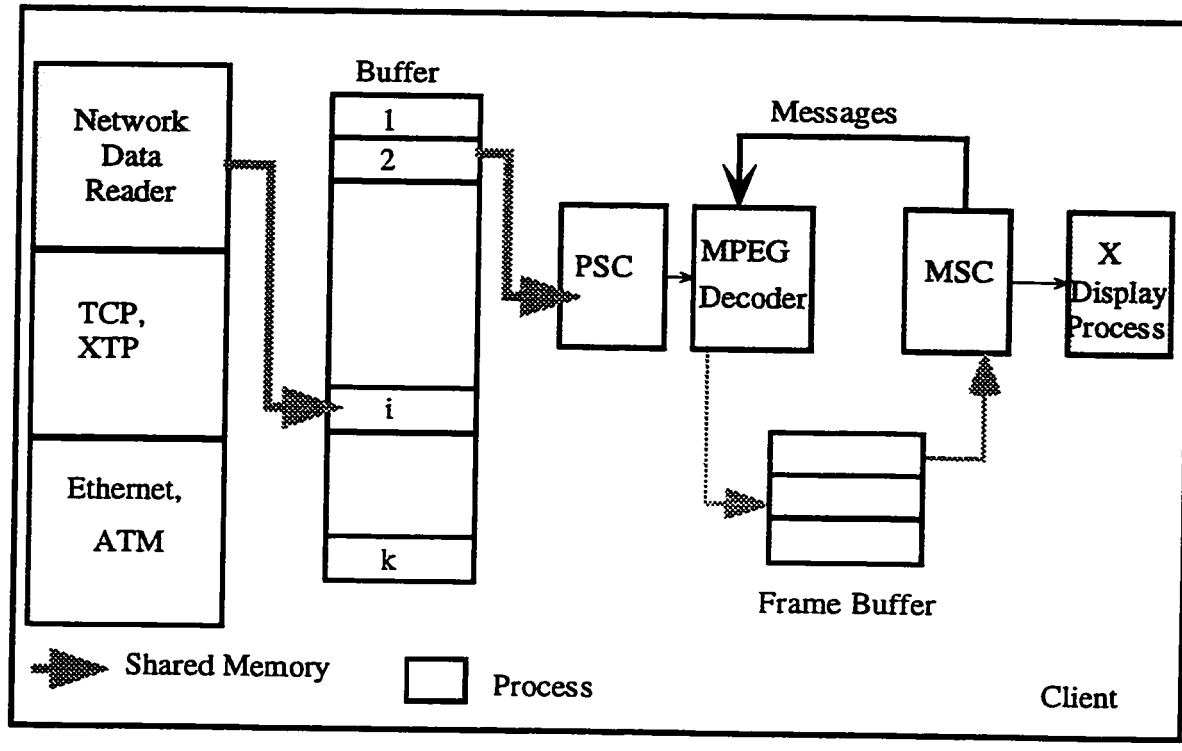

Figure 4.5.1 MPEG PSC Architecture

Having described the general aspects of the MPEG PSC architecture, we discuss the operation of the PSC/Decoder pair in more details. Then we discuss the format of the frame (or auxiliary) buffer shared by the MPEG PSC and MSC. Finally, we describe the communication between the PSC and the MSC. It is to be noted that the client buffer structure used in the MPEG PSC is the same as that in the original prototype, except for a minor modification of the buffering unit size from 1024 bytes to 2048 bytes to match the decoder's internal buffer size. Moreover, the buffer occupancy checking and the frame dropping control variables are similar to those presented for the MJPEG PSC and are not elaborated upon in this section.

When the PSC reads the first data unit of the received MPEG stream, it uses the information within to initialize the decoder internal parameters. These parameters include the frame width and the frame height of the video clip. These two are then used to compute the size of the luminance (Y) and the chrominance (U,V) components of the decoded frame according to the sampling information (4:4:4, 4:2:2 etc...) of the coded MPEG video stream. The PSC, then initializes and allocates the shared memory auxiliary buffer space lying between the PSC/Decoder and the MSC/Display processes. Having done all the initialization and before starting the decoding actions, the PSC forks the MSC/Display process. The PSC then keeps filling the decoder's internal buffer with the data stream along with the monitoring of the buffer occupancy. To skip frames, in overflow threat situations, the PSC uses the Next_Start_Code() function of the decoder to parse through the data stream (in the internal decoder buffer) repeatedly, until a PICTURE_START_CODE (0x100) is encountered. The PSC then checks the type of the frame and makes its decision based on the algorithm in section 3.3.6. To repeat frames, in starvation threat situations, the PSC/Decoder process writes the current decoded frame (depending on the frame type as in the algorithm in section 3.3.6) twice to the auxiliary frame buffer. In this way, the PSC only drops coded frames and only repeats decoded frames. This turns out to be very important due to the savings in processing time made. The PSC also keeps track of dropped/repeated frames in order to be able to recover the normal decoding sequence, when the buffer occupancy returns to normal.

The auxiliary frame buffer size (as opposed to the number of frames it holds which is a modifiable static constant) is not known in advance since different clips will have

different frame sizes and aspect ratios. As described in the above paragraph, the PSC takes care of the dynamic allocation of this buffer space. Figure 4.5.2 shows the structure of the auxiliary buffer used which is similar in concept to that of the MJPEG PSC buffer . The exception is that instead of the space appended at the end of the buffer structure being single fielded, it is now triple fielded with three pointers *y_pointer*, *u_pointer* and *v_pointer*.

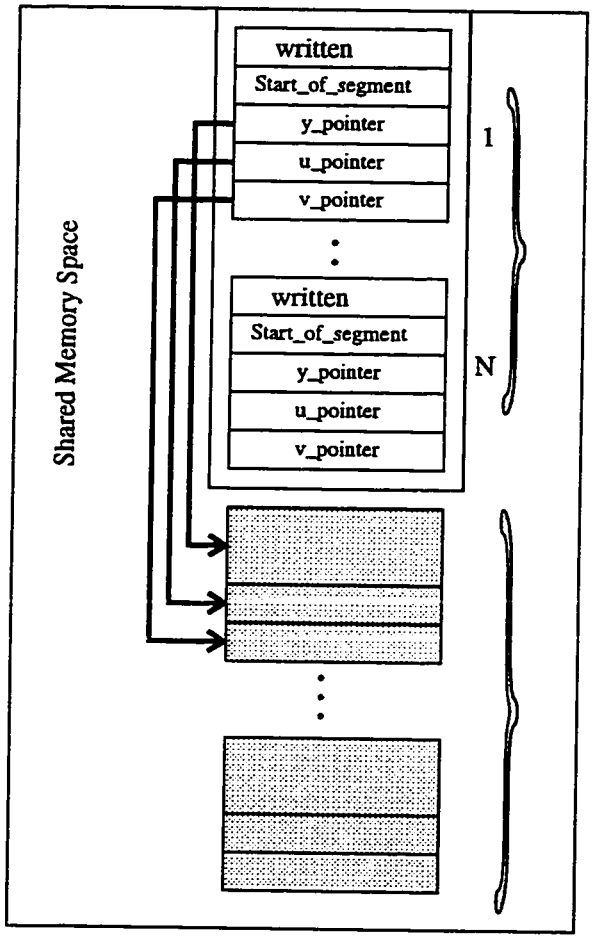

Figure 4.5.2 Auxiliary Frame Buffer Structure

Moreover the auxiliary buffer structure contains an important parameter needed by the MSC, namely, the *start_of_segment* flag. This boolean variable is set

to 1 by the PSC whenever the decoded frame being written into the slot is the first frame of a given segment (or logical data unit). This information is used by the MSC to initialize/reschedule the shared schedule among the different MSCs, thereby, maintaining inter-stream synchronization. The MSC resets the flag after its use.

Finally, the last point of concern in the implementation of the PSC is the communication between the MSC and the PSC. It is to be noted that the auxiliary frame buffer is never under the threat of an overflow, due to the controlled action of the PSC and the Decoder. In fact if the buffer frame is full, the decoding will stop until one slot is freed. However, the frame buffer can face starvation, in case the decoding rate cannot match the playback rate. To rectify such situations, the MSC and PSC share a common semaphore called *frame_starvation*. If the MSC is running short of frames, it will set the multi-valued semaphore to the number of frames it needs in order to catch up with its schedule (this can occur in an incremental way). The PSC, along with its buffer monitoring and frame drop/repeat considerations, checks the value of *frame_starvation* and adjusts its actions to balance out the situation as described in section 3.3.6. The PSC then resets the semaphore accordingly.

As a final remark to the implementation of the MPEG PSC, it is to be noted the significant complexity of such scheme with respect to the MJPEG PSC. This is attributed to the major differences between both streams in terms of frame types and syntax.

The testing of the PSC for MJPEG streams shows good rendering continuity of a 15 frames/second news clip with less than 2/3 of a second buffering space

at the client. The MPEG PSC seems to suffer from the dithering overhead incurred by the frames using the X display server. This might be alleviated by using shared memory to communicate with the X server.

Chapter Five

Conclusion

The first contribution of this thesis is a comprehensive review of multimedia synchronization, especially in distributed multimedia system. A thorough discussion of multimedia synchronization errors, their causes and the possible strategies to prevent/recover from such anomalies was presented. The discussion also considered in detail the specification of multimedia synchronization and the QoS issues related to multimedia synchronization. This allowed a complete comprehension of the multimedia synchronization problem from all view points of the multimedia system. One aspect of this synchronization view, which did not encounter much attention in the literature, is that of coded streams. The following paragraphs summarize the work presented in this thesis with respect to such area.

Coded video streams are expected to become the predominant type of multimedia traffic transmitted and are one of the most demanding in terms of bandwidth, delay and processing requirements [TS96]. Time-dependent coded video streams are very

sensitive to delay and delay variation. Moreover, they are more vulnerable to data loss than non coded streams. In addition, they may be of a variable bit rate (VBR) nature making it difficult to accurately estimate their buffering requirements. furthermore, they may have a different presentation, transmission and decoding orders which is the case with MPEG video. Also, coded video streams have a high throughput, which is even orders of magnitude higher after decompression, necessitating fast real time processing at the level of the coded stream in order to avoid huge buffering needs at the client. Finally, coded video streams require a decoding entity (hardware/software) at the client. These distinctions dictate special provisions in terms of synchronization control of these streams at the client.

We presented a client based application level buffer control scheme to maintain stream continuity, avoid data overflow and underflow conditions and compensate for possible clock drifts at the client. The control policy, referred to as the PSC, uses buffer occupancy as its main control parameter. Based on high (BOH) and low (BOL) buffer occupancy thresholds, overflow threats and starvation threats are detected. Time contraction (selective frame drop) and expansion concepts (selective frame repeat) are used to prevent these data overflow and starvation threats to become real conditions, respectively. In addition, the PSC mechanism compensates for dropped/repeated frames after the buffer occupancy returns to normal in order to recover inter-stream synchronization. In essence, the PSC mechanism blends the decoding process with the intra-media synchronization of continuous coded video streams.

We specified PSC mechanisms for both MJPEG and MPEG video bit streams. In the case of MJPEG streams, examples were given illustrating the effect of the PSC under both underflow and overflow conditions. The PSC was found to smoothen the stream, creating localized variations in the rate of displayed stream. This was found to avoid long frame freeze and abrupt scene change situations. In the case of MPEG streams, a selective frame drop/repeat policy is devised due to the existence of three types of MPEG frames (I, P and B) and the syntactical nature of the stream. In the case of overflow threats, only B frames are discarded. I or P frames are not dropped because they are needed to decode all subsequent B or P frames until an I frame is reached. In the case of underflow threats, only B or I frames are repeated. P frames are not repeated because the decoder needs two different P or an I and P frame to decode subsequent B frames.

A prototype implementation was carried out for both MJPEG and MPEG video bit streams. The MJPEG video stream was decoded in hardware using the IBM Ultimotion JPEG graphics adapter. The MPEG video bit streams were decoded using a MPEG-2 public domain decoder from the MPEG Software Simulation Group. For the purpose of the implementation, a stream structure was developed that allowed uniform transmission of media units from the server over the course of their presentation period. This allowed for reduced buffer requirements at the client site (less than 2/3 of a second). Based on this stream structure and based on a bounded delay network model, buffering delay (for starvation prevention) and buffer size (for overflow prevention) values were derived. In addition to the stream structure, the prototype implementation showed the impact the nature of the decoder has on the architecture of the

synchronization scheme. In the case of hardware decoder, the PSC had to be combined with the MSC just before the decoder. In the case of a software decoder, the PSC lies before the decoder. The MSC had to be positioned after the decoder.

Future work should consider three aspects of the PSC mechanism:

- Performance evaluation of the new augmented synchronization prototype should be carried out using real test video streams over a metropolitan area ATM network such as OCRINet. This will illustrate in a concrete way the impact of the PSC on synchronization as well as the reduced buffering requirements at the client side.

- Enhancements to the PSC functionalities should be considered by allowing selective information dropping. This implies lowering the quality of the displayed frames instead of dropping them completely while still carrying the main goals of the PSC. This will involve a tradeoff between the involvement of the PSC in regulating the stream as opposed to the quality of the displayed picture.

- Finally, a feedback version of the PSC functionality could be envisioned. This will allow the server to drop/repeat frames. Again a tradeoff between PSC functionality , complexity of implementation and server performance imposes itself at this point.

References

[AND91] D. P. Anderson and G. Homsy, "Synchronization Policies and Mechanisms in a Continuous Media I/O Server," International Conference on Multimedia Information Systems, Singapore, January 1991.

[APT95] R. T. Aptekar, J. A. Fisher, V. Kisimov and H. Nieshlos, "Video Acceptability and Frame Rate," IEEE Multimedia, pp. 32-40, Sept. 1995.

[BAQ96] S. Baqai, M. Woo and A. Ghafoor, "Network Resource Management for Expertise Wide Multimedia Services," IEEE Communications, Vol. 34, pp. 78-83, Jan. 1996.

[BLA96] G. Blakowski and R. Steinmetz, " A Media Synchronization Survey: Reference Model, Specification and Case Studies," IEEE JSAC: Synchronization Issues in Multimedia Communications, Vol. 14, No. 1, January 1996.

[BRI95] R. Brimont, *Implementation and Performance Analysis of a Multimedia Synchronizer*, Dept. of Electrical Engineering, University of Ottawa, 1995.

[BUR94] T. M. Burkow, "Operating system Support for Distributed Multimedia Applications: A Survey of Current Research," Pegasus Paper 94-8, University of Cambridge, Computer Laboratory and University of Twente, June 1994.

[CAR94] S. Carrier and N. D. Georganas, "Multimedia Electronic Mail: A Pragmatic Approach," Proc. IEEE COMSOC Multimedia '94, Kyoto, Japan, May 1994.

[CEN95] S. Cen et al., " A Distributed Real-Time MPEG Video Audio Player," Proceedings of the 5th Int. Workshop on Network and Operating System Support for Digital Audio and Video, April 1995, Durham, NH.

[CHE96] H. Y. Chen and J. L. Wu, "MultiSync: A Synchronization Model for Multimedia Systems," IEEE JSAC: Synchronization Issues in Multimedia Communications, Vol. 14, No. 1, January 1996.

[COR95] M. Coreira and P. Pinto, "Low Level Multimedia Synchronization Algorithms on Broadband Networks," ACM Multimedia '95, San Francisco, CA., 1995.

[DEL94] L. Delgrossi, R. G. Herrwich an F. O. Hoffmann, "An Implementation of ST-II for the Heidelberg Transport System," Internetworking Res. Experience, Vol. 5, 1994.

[DUP92] S. Dupuy, W. Tawbi and E. Horlait, "Protocols for High Speed Multimedia Communications Networks," Computer Communications, Vol. 15, No.6, July/August 1992.

[FER92] D. Ferrari, "Delay Jitter Control Scheme for Packet Switching Internetworks," Computer Communications, Vol. 15. No. 6, July/August 1992.

[FOR93] M. Fortier, "A Store-and Forward Architecture for Video-on-Demand Service," Proc. ICCC Multimedia Communications '93 Conference, Banff, Apr. 1993.

[FUR94] B. Furht, "Multimedia Systems: An Overview ," IEEE Multimedia, Vol. 1, No. 1, Spring 1994.

[GEO94] N. D. Georganas, *Topics in Communications II: Multimedia Communications*, ELG 7177 Lecture Notes, Dept. of Electrical Engineering, University of Ottawa, 1994.

[GEO96] N. D. Georganas, " Synchronization Issues in Multimedia Presentational and Conversational Applications," Proceedings of the Pacific Workshop on Distributed multimedia Systems, June 1996, Hong Kong.

[HAM72] C. Hamblin, "Instants and Intervals," in Proc. of the First Conference of the International Society for the Study of Time, pp. 324-331, 1972.

[HUA96] C. M. Huang and R. Y. Lee, "Achieving Multimedia Synchronization Between Live Video and Live Audio Streams Using QoS Controls," Computer Communications, Vol. 19, pp. 456-467, 1996.

[HUI90] J. Y. Hui, *Switching and Traffic Theory for Integrated Broadband Networks*, Kluwer Academic Publishers, 1990.

[HUI96] J. Y. Hui et al., " Client-Server Synchronization and Buffering for Variable Rate Multimedia Retrievals," IEEE JSAC: Synchronization Issues in Multimedia Communications, Vol. 14, No. 1, January 96.

[ISO93] ISO Multimedia and Hypermedia Information Coding Experts Group, ISO/IEC JTC 1/SC 29/WG 12, "Information Technology- Coded Representation of Multimedia and Hypermedia Information (MHEG), Part 1: Base Notation (ASN.1)," Committee Draft ISO/IEC CD 13522-1, June 1993.

[JPE93] JPEG (1993), ISO/IEC/JTC1 Information Technology- Digital Compression and Coding of Continuous-Tone Still Images, ISO/IEC IS 10918.

[KAW91] M. Kawaraski and B. Jabbari, "B-ISDN Architecture and Protocol," IEEE JSAC, Vol. 9, No 9, December 1991.

[KUR94] T. Kurita, S. Lai and N. Kitawaki, "Effects of Transmission Delay in Audiovisual Communication," Electronics and Communications in Japan, Part 1, Vol. 77, No. 3, 1994.

[LAM94.1] L. Lamont and N. D. Georganas, "Synchronization Architecture and Protocols for a Multimedia News Service Application," Proc. IEEE International Multimedia Computing and Systems Conference, Boston, May 1994.

[LAM94.2] L. Lamont, L. Li and N. D. Georganas, "Centralized and Distributed Architectures for Multimedia Presentational Applications," Proc. of 3rd International Conference on Broadband Islands, Hamburg, Germany, June 1994.

[LAM96] L. Lamont, L. Li, R. Brimont and N. D. Georganas, " Synchronization of Multimedia Data for a Multimedia News on Demand Application," IEEE JSAC: Synchronization Issues in Multimedia Communications, Vol. 14, No. 1, January 1996.

[LAZ90] A. A. Lazar, G. Pacifici and J. S. White, "Real-Time Traffic Measurements on MAGNET II." IEEE JSAC, Vol.8., No. 3, pp. 467-483, Apr. 1990.

[LEB92] J. Y. Le Boudec, "The Asynchronous Transfer Mode: A Tutorial," Computer Networks and ISDN Systems, Vol. 24, pp. 279-309, 1992.

[LEG91] D. LeGall, "MPEG: A Video Compression Standard for Multimedia Applications," Communications of the ACM, Vol. 34, No. 4, pp. 45-68, Apr. 1991.

[LIK94.1] L. Li, A. Karmouch and N. D. Georganas, " Multimedia Teleorchestra with Independent Sources: Part 1- Temporal Modeling of Collaborative Multimedia Scenarios," ACM/Springer Journal of Multimedia Systems, Vol. 1, No 4, February 1994.

[LIK94.2] L. Li, A. Karmouch and N. D. Georganas, " Multimedia Teleorchestra with Independent Sources: Part 2- Synchronization Algorithms,"

ACM/Springer Journal of Multimedia Systems, Vol. 1, No 4, February 1994.

[LIT90.1] T. D. C. Little and A. Ghafoor, " Synchronization and Storage Models for Multimedia Objects," IEEE JSAC, Vol. 8, pp. 413-426, April 1990.

[LIT90.2] T. D. C. Little and A. Ghafoor, "Network Considerations for Distributed Multimedia Object Composition and Communication," IEEE Network Magazine, Vol. 4, pp. 32-49, Nov. 1990.

[LIT91.1] T. D. C. Little and A. Ghafoor, " Spatio-Temporal Composition of Distributed Multimedia Objects for Value Added Networks," IEEE Computer, Vol. 24, pp. 42-50, Oct. 1991.

[LIT91.2] T. D. C. Little and A. Ghafoor, "Multimedia Synchronization Protocols for Broadband Integrated Services," IEEE JSAC, Vol. 9, pp. 1368-1382, Dec. 1991.

[LYL92] J. Lyles and D. Swinehart, " The Emerging Gigabyte Environment and the Role of Local ATM," IEEE Communications Magazine, Vol. 30., pp. 52-58, 1992.

[MEY93] T. Meyer, W. Effelsberg and R. Steinmetz, "A Taxonomy on Multimedia Synchronization," Proc. 4th Int. Workshop on Future Trends in Distributed Computing Systems, 1993.

[MIL92] D. Mills, "Network Time Protocol (Version 3): Specification, Implementation and Analysis," Internet RFC 1305, March 1992.

[MMC95] Multimedia Communications Quality of Service, Final MMCF Document, Multimedia Communications Forum, 1995.

[MPE93] MPEG-1 (1993), ISO/IEC/JTC1 Information Technology- Coding of Moving Pictures and Associated Audio for Digital Storage Media Up to About 1.5 Mbits/s, ISO/IEC IS 11172.

[MPE95] MPEG-2, ISO/IEC/JTC1 Information Technology- Generic Coding of Moving Pictures and Associated Audio, ISO/IEC DIS 13818, March 1995.

[NAH95] K. Nahrstedt and R. Steinmetz, "Resource Management in Networked Multimedia Systems," IEEE Computer Magazine, Vol. 28, No. 5, May 1995.

[PAT93] K. Patel, B. C. Smith and L. A. Rowe, "Performance of a Software MPEG Video Decoder", ACM Multimedia '93, Anaheim, CA., August 1993.

[PAT94] S. Patel, G. Henderson and N. D. Georganas," Multimedia Fax/Mime Interworking," Proc. 1st IEEE International Conference on Multimedia computing Systems, Boston, May 1994.

[PEN93] W. B. Pennenbaker and J. L. Mitchel, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, New York, 1993.

[RAV93] K. Ravindran and V. Bansal, "Delay Compensation Protocols for Synchronization of Multimedia Data Streams," IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 4. August 1993.

[ROS89] F. Ross, "An Overview of FDDI: The Fiber Distributed Data Interface," IEEE JSAC, Vol.7, No 7, September 1989.

[ROT95] K. Rothermel and T. Helbig, " An Adaptive Stream Synchronization Protocol," Proceedings of the 5th Int. Workshop on Network and Operating System Support for Digital Audio and Video, April 1995, Durham, NH.

[ROW92] L. A. Rowe and B. C. Smith, "A continuous Media Player," Proc. 3rd Int. Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, CA., November 1992.

[ROW94] L. A. Rowe et al., " MPEG Video in Software: Representation, Transmission and Playback," High Speed Networking and Multimedia Computing, IS&T/SPIE Symp. On Elec. Imaging Sci. & Tech, San Jose, CA, February 94.

[SCH90] G. Schurmann and U. Holzmann-Kaiser, "Distributed Multimedia Information Handling and Processing," IEE Network Magazine, Vol. 4, No. 11, pp-23-31, Nov. 1990.

[SMI95] J. M. Smith and K. Nahrstedt, "The QoS Broker," IEEE Multimedia, Vol. 2, No. 1, pp. 53-67, Spring 1995.

[STE90] R. Steinmetz, "Synchronization Properties in Multimedia Systems," IEEE JSAC, Vol. 8, pp. 401-412, Apr. 1990.

[STE94] R. Steinmetz, "Data Compression in Multimedia Computing- Principles and Techniques," Multimedia Systems, Vol. 1, pp. 166-172, 1996.

[STE95.1] R. Steinmetz, "Analyzing the Multimedia Operating System," IEEE Multimedia, Vol. 2, No. 1, Spring 1995.

[STE95.2] R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications & Applications*, Prentice Hall, Upper Saddle River, New Jersey, 1995.

[TSA96] R. P. Tsang, D. H. C. Du and A. Pavan, "Experiments with Video Transmission Over an Asynchronous Transfer Mode (ATM) Network," Multimedia Systems, Vol. 4, pp. 157-171, 1996.

[ULT94] Ultimedia Services 2.1 for AIX: Guide and Reference, International Business Machine Corporation, July 1994.

[VEN95] P. Venkat et al., " Feedback Techniques for Continuity and Synchronization in Multimedia Information Retrieval," ACM Trans. Inform. Syst., Apr. 1995.

[VOG95] A. Vogel, B. Kerherve, G. Von Bochmann and J. Gecsei, "Distributed Multimedia and QoS: A Survey," IEEE Multimedia, Vol. 3, pp. 10-19, Summer 1995.

[WAL91] G. Wallace, "The JPEG Still Picture Compression Standard," Communications of the ACM, Vol. 34, No. 4, Apr. 1991.

[WIJ96] D. Wijesekera and J. Srivastava, "Quality of Service (QoS) Metrics for Continuous Media," Multimedia Tools and Applications, pp. 127-166, March 1996.

[WOE86] D. Woelk, W. Kim and W. Luther, "An Object-Oriented Approach to Multimedia Databases," Proc. of ACM SIGMOD '86, pp. 311-325, Washington, D.C., May 1986.

[ZHA93] L. Zhang et al., "RSVP: A New Resource Reservation Protocol," IEEE Network, Vol. 7, No. 5, pp. 8-18, Sept. 1993.

[ZNA95] T. Znati, R. Simon and B. Field, " A network-Based Scheme for Synchronization of Multimedia Streams," Multimedia Synchronization Workshop,held in conjunction with IEEE International Conference on Multimedia Computing and Systems, Whashington D.C., 1995

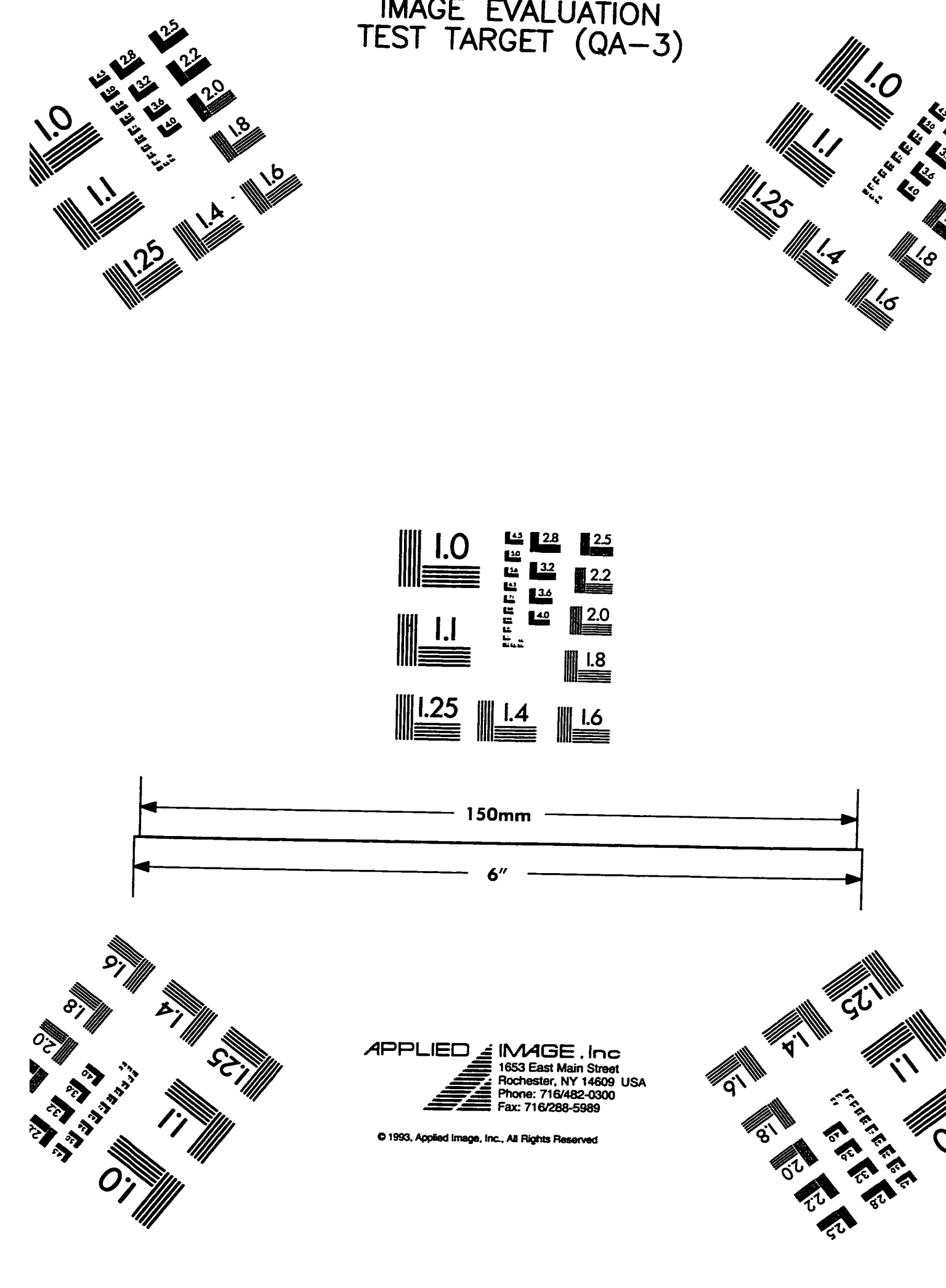
IMAGE EVALUATION
TEST TARGET (QA-3)
1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
150mm
6"
APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989
© 1993, Applied Image, Inc., All Rights Reserved