National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Canada

The Design and Implementation of a Real-time Multimedia Synchronization Control System over High-speed Communications Networks

Lian Li

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, 1994

ISBN 0-612-00540-2

Canada

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

I hereby declare that I am the sole author of this thesis.

I authorize the University of Ottawa to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Lian Li

I further authorize the University of Ottawa to reproduce the thesis by photocopying or by other means, in total or in part, at the request of the other institutions or individuals for the purpose of scholarly research.

Lian Li

Abstract

Synchronization is considered as a key issue in distributed multimedia systems. In a real-time multimedia presentation, data objects of different media types or coding formats are delivered from distributed media-storing servers to the remote client simultaneously over high-speed networks. The multiple streams need to be synchronized so that the multimedia document can be presented in the way specified by its creator. The synchronization research involves issues such as temporal relationship modeling, extending network protocols and supporting the implementation of applications where the synchronization control mechanisms integrate with other system functionality, such as the ATM network transmissions, the video coding/decoding and the distributed database management.

In this thesis, we investigate a software synchronization control system for a target presentational application, i.e., a *Multimedia News-on-demand* service. Relying on the Quality of Services (QoS) supported by the ATM-based virtual connections, the system prevents major multi-stream mismatches through a delivery scheduling operation. Moreover, the synchronization errors brought by the inevitable network delay variations are recovered through a Stream Synchronization Protocol (SSP) in order to preserve the presentation quality. We apply the Time Flow Graph (TFG) to model the temporal relationships among the media components so that the scheduling and recovering operations can be efficient. Synchronization QoS parameters are employed in the SSP control. In addition, the differences between the characteristics of coded and uncoded data streams are taken into account. We present a priority-based synchronization control for coded data, e.g., the MPEG-2 video stream.

For the implementation of such a control system, we elaborate a set of data structure specifications and algorithms. As well, we develop the software modules to implement the synchronization control prototype.

Acknowledgments

First, I would like to thank my supervisor Dr. N.D. Georganas for his constant support and encouragement during my stay at the University of Ottawa.

I would also like to thank the members of the Multimedia Synchronization research group for always being available for discussions and providing valuable comments. My special thanks go to Li Li, Louise Lamont and Grant Henderson, for their encouragement and help in my thesis work. I would also like to thank Dr. A. Karmouch for providing me the opportunity to work with his research team for a few months, which gave me valuable experience in implementation. My thanks are also due to the members of the Multimedia Communications Research Laboratory, who have always been helpful and fun people to be with.

I would give my thanks to the support staff members of Electrical Engineering for their help, to Amanda Lauzon, Lucette Lepage and Michele Roy.

Finally, I would like to thank my family for their consistent support through all the years of my education.

Contents

List of Figures

List of Tables

List of Symbols

MSC	Media Synchronization Controller process
OCPN	Object Composition Petri Nets model
PSC	Pre-decoder Synchronization Controller process
SSP	Stream Synchronization Protocol
TFG	Time Flow Graph model
QoS	Quality of Service

Chapter 1

Introduction

1.1 Multimedia Applications and Synchronization

Advances in computer technologies have led to the development of powerful workstations with audio/video capabilities, and server machines with high capacity storage devices. Similarly, advances in networking technologies promise the availability of high-speed ATM-based networks that are characterized by bandwidth-on-demand. These developments have spurred interests in the development of *distributed multimedia applications*. The potential new applications include [Fur94]: multimedia information systems, on-demand multimedia services, distance learning, collaboration and conferencing, etc.

A *Distributed Multimedia System*, where a number of workstations and server machines are interconnected by one or more networks, offers greater promise than the *Stand-alone* multimedia workstation. On the other hand, managing multimedia in a distributed system brings in a number of research problems. *Multimedia Integration* becomes a great challenge in the development of distributed multimedia applications.

.*Multimedia Integration*

Multimedia systems are defined as systems capable of creating, storing, retrieving and transporting a variety of media types, such as text, voice, graphics, animation, images, audio and full motion video. First, the *variety* of media types is an important feature of modern information systems. Second, in order to deal with this variety, *integration* is a critical concern. In essence, multimedia systems are attempting to solve the problems of information management by integrating the various forms of media into the computer/communications infrastructure. There are two benefits of achieving this level of *integration*[Bla91]:

1) The computer can help in the task of managing and processing the information.

2) Information users only have to deal with one integrated environment rather than a number of separate information subsystems.

The above is the main motivation behind the current research on the multimedia integration subject.

.*Multimedia Synchronization*

In the context of this thesis, the term *multimedia synchronization* refers to the temporal integration of the distributed multimedia information. A distributed multimedia system (refer to Fig.1.1.a) includes multiple sources of various media either spatially or temporally[Lit90] to create composite multimedia documents. Spatial composition links various multimedia objects into a single entity dealing with object size, rotation, and placement within the entity. Temporal composition creates a multimedia presentation by arranging the objects corresponding to the temporal relationships. The *synchronization* control is to maintain the temporal relationships among the multimedia objects.

There are basically two types of synchronization, i.e., *continuous synchronization* and *synthetic synchronization*. The *continuous synchronization* requires constant synchronization of lengthy events. An example of continuous synchronization is the *lip-sync* in video telephony, where the audio signal captured at the recorder and the video signal captured at a camera are transmitted to a remote receiver site and then are continuously played back with the audio matching

the speaker's lip movement. The *synthetic synchronization* exists in a multimedia document, where the temporal ordering required in playing back the media components is explicitly specified. An example of synthetic synchronization is a slide show with blocks of voice allotted to each slide (refer to Fig. 1.1.b). These two types of synchronization may coexist in the same application.

Since the network introduces variable delays to the delivery of the objects, synchronization control is required so that the multimedia data objects can be played back according to the temporal relationships among the objects.

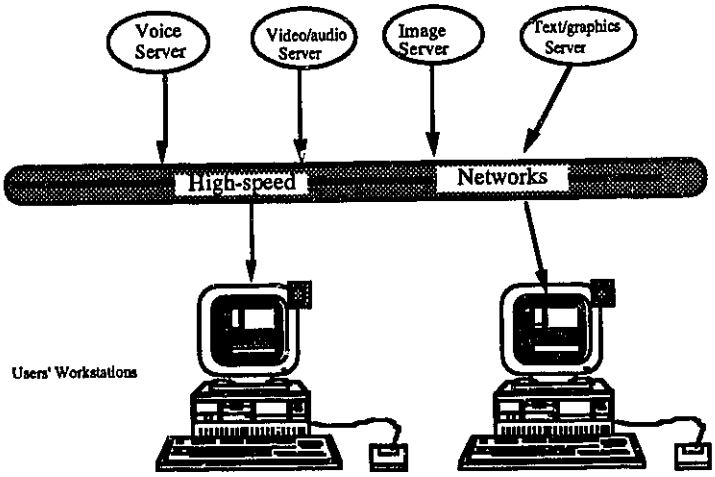

Figure 1.1.a An example of a distributed multimedia system

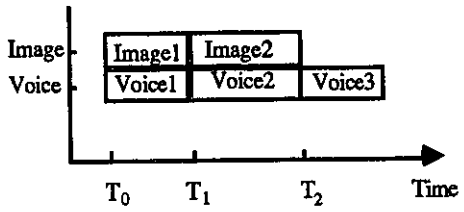

Figure 1.1.b The temporal composition of a multimedia document

Research work is required to fully understand the temporal integration requirements of the distributed multimedia applications and the corresponding implications for the synchronization control system design. The thesis focuses on the synchronization problem of the *Distributed*

Multimedia On-demand system, particularly the design and implementation issues of a synchronization software control system for a distributed *Multimedia News-on-demand* application. The thesis work relates to the CITR major project entitled "Broadband Services".

1.2 Introduction to the CITR "Broadband Services" Project[Dra94]

An important class of distributed multimedia applications is the presentational one where multimedia documents featuring continuous (voice and video) and/or discrete (text/and image) data are accessed interactively by remote users. Example applications include: electronic newspaper, digital library, home shopping, and distance education. The success of this type of interactive services is heavily dependent on the ability to deliver the service to a large community of users in an effective manner.

The Canadian Institute of Telecommunications Research (CITR) is carrying on a *Broadband Services* project which is focused on the enabling technologies required for the development of presentational applications involving one or several multimedia databases accessed over distance through a broadband network. Of particular interest are the service architecture and the software infrastructure that support the implementation of applications that are efficient and flexible. Video objects are generally large. The storage and communication requirements of these objects create challenges in the design of systems that support large user communities, especially in the areas of video file servers and the management of user expectation with respect to quality of service guarantees. These areas will be investigated in this major project. The elaborate data management requirements of multimedia applications also create challenges in database design. Other issues to be addressed include video encoding to impact on the design of video file servers, multimedia databases, and telecommunications software. They also affect the design of application programming interfaces for interactive multimedia applications.

This major project has an experimental component where a target application will be developed and demonstrated on an ATM test bed. Currently, *Multimedia News-on-demand* is a strong candidate for the target application. The six constituent projects are designed such that they are complementary and that the results can be integrated into a working prototype. The titles of these projects are listed below:

. Multimedia Data Management

. Scalable Video Encoding for Database storage

. Quality of Service Negotiation and Adaptation

. A Distributed Continuous-Media File System

. Synchronization of Multimedia Data for Presentational Applications

. Project Integration

Our research group at the Multimedia Communications Research Lab. (MCRLab), University of Ottawa, is engaged in the project entitled "Synchronization of Multimedia Data for Presentational Applications". The general objective of this project is to develop and evaluate a complete software control system for synchronizing multiple data streams generated from distributed media storage database servers. The control system developed will be used to support the *Multimedia News-on-demand*, the target presentational application of the major project.

1.3 Thesis Outline

Recent research in our group and elsewhere has focused on the architecture design and implementation of the real-time multimedia synchronization control system.

A review of the multimedia synchronization research is given in Chapter 2. Both the application requirements and synchronization technologies are addressed. On the broadband network (e.g., ATM network), data streams of different media may be transferred on different virtual connections, according to their traffic characteristics and QoS requirements. Furthermore

the Synchronization QoS parameters[Ste92] are applied to describe the multi-stream mismatching that can be tolerated by applications of different types. In stored information retrieval, a delivery schedule can be made before the real-time transmission starts based on *a priori* presentational requirements represented in a temporal model. Moreover, synchronization error recovery applies to the data streams upon their arrivals. The synchronization control schemes can be applied to a variety of synchronization control "granularity" units. Research on layered synchronization architectures is also introduced.

Chapter 3 describes the conceptual design of a complete software synchronization control system for a target real-time presentational application, the *Multimedia News-on-demand* service. Temporal models are discussed in the aspect of their capabilities to support synchronization scheduling. The functional entities required to perform the synchronization consist of a data delivery *Scheduler* and pairs of *Media Synchronization Controllers* (MSCs) at both ends of each stream connection, i.e., the server MSCs at the Media Server's site and the client MSCs at the client's site. The problem of synchronizing distributed servers in an integrated document delivery is partly solved by connection QoS guarantees and delivery scheduling. The other concerns lie in the synchronization recovery performance, such as, fast accommodation to the random network disturbance, efficient buffer management, the impact of data loss which may occur when the buffer overflows, etc. A *Stream Synchronization Protocol* (SSP) is introduced to the client MSCs to deal with the synchronization errors among the data objects when late packets are received at the client's workstation via separate stream connections. A priority-based synchronization control scheme for a coded stream, the *Pre-decoder Synchronization Controller* (PSC) and MSC control over a MPEG-2 video stream in particular, is also investigated. In the network transmission architecture, the SSP which is an application protocol, resides above the QoS supporting network protocol, such as ST-2, which in turn resides above the ATM transmission subsystem.

Chapter 4 presents the implementation issues related to the above design. The data structures of the scenario specification, the temporal model of Time Flow Graph, the schedules, and a set of scheduling algorithms are provided. In the software implementation, the UNIX

supported interprocess communication mechanisms are fully utilized for concurrent process control, such as shared memory, semaphores, sockets, and signals. The receiving buffer (e.g., the buffer controlled by the client synchronization controllers) structure is implemented by a linked list. User interactions are also considered. The experimental prototype is going to demonstrate the synchronization control system over an ATM-based network.

Chapter 5 concludes the thesis by summarizing the contributions and suggesting future work.

1.4 Publications Arising from this Research

1. L. Lamont, L. Li, N. D. Georganas, " Centralized and Distributed Synchronization Architectures for Multimedia Presentational Applications", Proc. the 3rd International Conference on Broadband Islands, Hamburg, Germany, June 1994.

2. L. Li and N. D. Georganas, " MPEG-2 Coded- and Uncoded- Stream Synchronization Control for Real-time Multimedia Transmission and Presentation over B-ISDN", Proc. ACM Multimedia'94 Conf., San Francisco, U.S., Oct., 1994.

1.5 Technical Terminology and Definitions

Media Object

An information item in mono-media. One page of text file, a segment of audio, and a video frame can all be media objects.

Presentational Scenario

A document structure specifying the temporal relationships among the multimedia objects, which characterizes the time ordering of the media objects in a multimedia presentation. For instance, a multimedia article which consists of video and text objects is requested, by its document presentational scenario, to present the video object immediately after the presentation of the text object. The p*resentational scenario* is created by the document designer and is stored in the multimedia database.

Temporal model

A mathematical specification which represents the temporal relationships among the media data objects. The temporal relationships can be either implicit for the live data which are generated simultaneously, or explicit as specified by a document *presentational scenario.*

Presentation Schedule

A temporal specification of the presentational scenario in terms of the presentation start time for each object in a multimedia document.

Delivery Schedule

A temporal specification in terms of the start time for the delivery of each object in a multimedia document from the media-storing database server machines.

Transport Data Unit

The application specified data unit of a media object, which is transferred between the application layer and the underlined transport network.

Presentation Interval

The time period between the presentations of two subsequent transport data units. For a continuous media type object such as video, the interval can be the average holding time of each video frame on the screen; for a discrete media type object such as text, the interval can be the holding time of displaying a whole page of text on the screen.

Jitter

The delay variation in a single data stream.

Skew

The time lag between related data streams.

Chapter 2

Multimedia Synchronization Review

2.1 Multimedia applications and their requirements

Powerful and cost-effective personal computers and workstations, high-capacity storage devices, high-speed integrated services digital networks [Her92] have made multimedia system development technically feasible. In this section, the general requirements of the multimedia applications will be addressed. The impacts on the high-speed networks and data compression techniques will be discussed afterwards.

2.1.1 Application Requirements

The requirements of multimedia applications include the following aspects, which are demanding technology development in the areas of storage, processing and network transmission.

.*Real-time Performance*

Multimedia information requires real-time performance in handling a number of media types. This is particularly true for the continuous media types such as audio and video. The real-time distributed multimedia applications rely on bounded end-to-end delays. For instance, a two second delay in an interactive *video-on-demand* service is acceptable to the clients, while the same amount of delay is unacceptable in teleconferencing. The research on improving real-time performance involves fast processors, real-time operating systems, high-speed networks, light-weight protocols, and video and audio compression algorithms.

.*Dynamic and Flexible Management*

In order to help meet a variety of performance requirements, a multimedia system needs flexibility in the way it utilizes the available resources. The system management should support allocating requested resources according to a number of quality of service criteria. This task may provide a reduced service to the applications which are willing to accept a level of degradation.

.*Integration*

Multimedia encompasses a wide range of data types. In addition, within each type there exists a variety of qualities of service. As discussed previously, one of the key challenges of a multimedia system is to integrate the various types and qualities of service within one system framework. In addition, multimedia communication implies that several data streams may have to be handled in parallel. Since these data streams belong to a single application and are therefore not independent, ways for synchronizing the streams must be found[Shep90].

2.1.2 Related Technical Advances

Before focusing on the area of the synchronization control in the following section, we will first discuss some important issues which are mostly relevant to multimedia synchronization.

1) Multimedia Network Communications

Many applications, such as video-on-demand, video conferencing and collaborative work systems, require networked multimedia. In these applications, the multimedia objects are generated from the server and played back at the clients' sites. Existing telecommunications facilities do not significantly support multimedia communications. They do not offer a common access to different networks or encourage collaboration. From the network's points of view, the most important requirements of multimedia communications are:

. High speed and changing bit rates

. Synchronization of different information types

. Suitable standardized services supporting multimedia applications

. Various service qualities

Multimedia applications place new communication requirements on the existing communications networks and protocols[Fur94]. These issues are discussed in more detail below.

.*Bandwidth*

The presentation rate of a sequence of objects, such as video frames, is nominally equal to the rate at which they are recorded. In a real-time network presentational application where the objects (e.g. the video clips) are continuously retrieved, transferred and played back at the destined end machine, network bandwidth requirements can be very high. Multimedia networks require a very high transfer rate or bandwidth, even when the data are compressed. For example, an MPEG-1 session requires a bandwidth of about 1.5 Mbps. MPEG-2 through JPEG will take 4 to 10 Mbps, while the projected required bandwidth of HDTV is 5 to 20 Mbps[Arm92]. Besides being high, the transfer rate must also be predictable. Networks must provide a variety of constant, variable, and burst-oriented bit rates of up to about 100 Mb/s per connection, as required by different and changing needs. Typical bit rates to be expected for the various information types have been compiled in [Arm92].

.*Reliability*

Another important communication requirement is the "reliability" of communication services, which can be viewed at multiple levels: per bit, frame, packet, channel, or connection. The reliability, expressed in terms of Bit Error Rate (BER) and Packet Error Rate (PER), represents the number of errors per time unit for bits and packets, respectively. The detected error rate of a communication channel is dependent on factors including the transmission medium, check-sum algorithm, and expected rate of packet loss from buffer overflows. The effect of packet and bit errors can have very different consequences, depending on the data transmitted. For example, an error in digitized voice can result in an audible click in a phone connection. For inter-frame coded video, lost bits or packets can interrupt image display for several seconds[Lit90].

The level of error provision also impacts time performance. To provide an error-free service, error control protocols are required. These protocols must provide error detection and retransmission and/or correction, which reduce ability to meet real-time performance specifications. Traditional networks are used to provide error-free transmission, since the traditional data file transfer can tolerate delays but can hardly tolerate any error. However, most multimedia applications tolerate errors in transmission without retransmission or correction. Typical limits of acceptable bit error rates are 10^{-10} to 10^{-12} for still image, 10^{-8} to 10^{-9} for moving pictures using an intra-frame compression technique, and 10^{-10} to 10^{-11} for moving pictures using an inter-frame compression technique[Arm92]. Meanwhile the fast developing optical network link is far more reliable than the traditional copper link. The ATM network which is built on optical transmission media can provide services with BER in the range of 10^{-8} to 10^{-12}. In future multimedia communications protocol design, the importance of the real-time performance may outweigh the reliability requirement. Light-weight network protocols are desired which do not provide error-correction to the network users.

.*Network Delays and Delay Variations*

The continuous media traffic such as audio and video often requires more network throughput and more stringent delay bounds than the conventional discrete media traffic does. A late audio packet may be of no use in a conversation and can be discarded. Delay jitter is the variation of the time which packets spend on a network connection[Fer92]. For good quality of reception, continuous media streams require that the jitter be kept below a sufficiently small upper bound.

.*Network Quality of Service Supports*

CCITT (now ITU-TS) defined the Quality of Service (QoS) as the "collective effect of service performance which determines the degree of satisfaction of the user of the services"[CCI]. The network quality of service parameters specified by the application include: throughput, delay, delay jitter, BER, etc. The multimedia network must avoid or compensate the undesired impacts in order to guarantee the required service quality. In the case of multimedia applications, this can lead to separate connections with different performance parameters for data on the one hand and for video and audio on the other[Arm92].

The network must also be flexible in a quantitative way, providing suitable service quality on demand. Asynchronous Transfer Mode (ATM) is suitable for multimedia traffic; it provides great flexibility in bandwidth allocation by assigning fixed length packets, called cells, to virtual connections. ATM can also increase the bandwidth efficiency by buffering and statistically multiplexing bursty traffic at the expense of cell delay and loss. ATM thus provides an integrated communications service in Broadband Integrated Services Digital Networks (B-ISDN). The ATM Adaptation Layer (AAL) produces AAL-PDUs from the application transport units. Four types of services have been defined as the ATM AAL services, i.e. connectionless delay insensitive, connection-oriented delay insensitive, connection-oriented delay sensitive & CBR traffic, and connection-oriented delay sensitive & VBR traffic transmission services. The ATM AAL also

allows network users to specify a variety of QoS for the requested virtual connections. The ATM-based network and its QoS supports are essential for the multimedia application system design.

.*Multi-channel Dependency*

In a multimedia environment, the various types of information which travel on different communications connections may have mutual dependencies. For instance, the video and the corresponding audio in video teleconferencing, or the separately stored image and voice annotation components of a multimedia document which are retrieved to the client in real-time. System and network support for multi-stream synchronization becomes a necessity for most multimedia applications.

.*Standardization*

The OSI/RM model is the standard model for computer networks and has gained very broad acceptance. This model reflects the conventional communications infrastructure which is suitable for applications such as remote terminal access and file transfer. With the emergence of new high-speed network standards such as B-ISDN, one of the challenges that OSI must meet is to show that it can be extended or adapted to support the wide range of traffic across the networks[Shep90]. The multi-connection synchronization and multi-casting features are of great concern.

2) Multimedia Compression

Present multimedia systems require data compression for three reasons[Fur94]: the large storage requirements of multimedia data, the relatively slow storage devices that cannot play multimedia data (specially video) in real time, and the network bandwidth that does not allow real-time video data transmission. For example, a single frame of a color video , with 620*560 -pixel frames at 24 bits per pixel, would take up about 1 Mbyte. At a real-time rate of 30 frames per second, that equals 30 Mbytes for one second of video. Modern image and video compression

techniques reduce these tremendous bandwidth requirements. Advanced techniques can compress a typical image at a ratio ranging from 10:1 to 50:1, achieving video compression up to 2,000:1[Fur94].

The MPEG standard is intended for compressing full-motion video. It uses inter-frame compression, achieving compression ratios of up to 200:1 by using motion compensation between successive frames. MPEG specifications also include an algorithm for compressing audio data at ratios from 5:1 to 10:1. The present standard, called MPEG-1, compresses 320*240 full- motion video which requires a minimum data rate of 1.5 Mbps. MPEG-2 will compress 720*480 full-motion video in broadcast television and video-on-demand applications. It will require a data rate in the range of 4 to 10 Mbps and will provide VCR-quality video[Fur94].

The compression algorithms reformat the data stream structures in various ways. Moreover, the inter-frame compression algorithms introduce correlation between the frames in a video sequence. Consequently, the synchronization among the compressed streams becomes more complicated than the synchronization of the natural form data streams, such as the uncoded video frame sequences.

Intensive activity is being carried on by the Moving Pictures Experts Group(MPEG) for designing a generic multi-stream synchronization standard. In the MPEG-1 standard[MPEG-1], a video stream and its associated audio stream can be multiplexed together according to their temporal relationships. The MPEG decoder, by using its internal buffers, will provide a complete solution for synchronization among its elementary streams, e.g., video and audio. However, network transmission introduces random delays and jitters to the MPEG stream, causes overflow or starving in the decoder internal buffer, and thus degrades the presentation quality. MPEG has recently produced its Committee Drafts of the MPEG-2 Standard[MPEG-2]. The MPEG-2 takes aim at supporting a wide range of broadcast, telecommunications, computing, and storage applications. A Transport Packet Stream, which is composed of fixed length Transport Packets, is introduced for the transportation environment where errors are likely. However, ATM AAL connection-

oriented stream transmission has to adapt to MPEG and ATM Forum is studying these issues[IBM93][Wel93].

2.2 Multimedia Synchronization Technology Review

The reported multimedia temporal synchronization studies focus on three major aspects: temporal modeling of synchronization constraints in the multimedia presentation, synchronization algorithm design based on the temporal models, and extending network protocols for multimedia data transfer with synchronization requirements.

2.2.1 Temporal Modeling of the Multimedia Data Integration

The temporal relationships between the media may be implied as in the simultaneous acquisition of voice and video, or may be explicitly formulated as in the case of a multimedia document which possesses voice annotated text. In order to properly schedule the synchronization of media with vastly different presentation requirements, the relationship between media must be established. In information retrieval, a "scenario" specifies the desired time ordering of the document media components in the presentation[LiLi94.2-1].

The temporal model which mathematically specifies the temporal relationships can be classified into two catalogs[LiLi94.2-1]. One is based on time intervals and the other is based on time points. A temporal interval is characterized as a nonzero duration of time in any set of units, for example, "a week" or "100ms" . This is contrasted with a time instant which is a point structure with a zero length duration, e.g., "12:00 am".

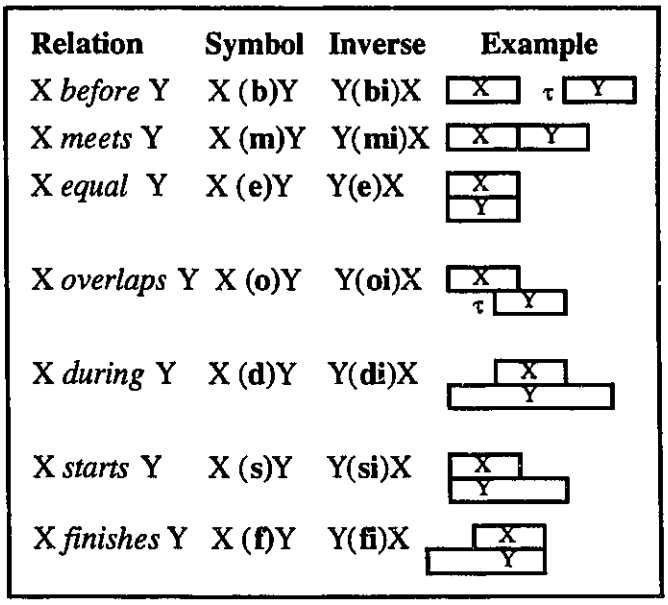

Figure 2.2.1 The seven basic relations between temporal intervals

Hamblin[Ham] presents a logic of intervals which is very useful in the development of a synchronization scheme. Given any two intervals, there are thirteen distinct ways in which they can be related. These relations indicate how two intervals relate in time; whether they overlap, or precede, etc. In Fig.2.2.1 we show seven of the thirteen relations since the remainder are inverse relations. For example, *after* is the inverse relation of *before*, or equivalently, $before^{-1}$ is the inverse relation of *before*. Note that the *equality* relation has no inverse.

In the following, two interval-based temporal models are described, i.e., the OCPN and the TFG models.

1) OCPN

The Object Composition Petri Nets (OCPN) model was proposed by Little and Ghafoor[Lit90]. The OCPN is an extension of a marked PetriNet , which, in turn, is derived from a simple PetriNet. PetriNet is chosen as the modeling tool for its ability to specify real-time process interaction based on hard, interprocess timing relationships as required for multimedia

presentation. Processes are represented by places and are assigned duration and resources. The duration determines the length of time for which a place locks its token and the resource. Additionally, a PetriNet is amenable to analysis including Markov process modeling[Lit90]. The OCPN [Lit90] is defined as a directed graph (N=(T,P,A,D,R,M)) where:

. T (set of transitions) = {t1, t2, ..., t_n};

. P (set of places) = {p1, p2,..., p_m};

. A (arc) : {P x T}U{T x P} -> I, I={1,2, ...,}

. M (mark): P -> I, I={0,1,2, ...}

. D(duration) : P-> *R*

. R (resources) : P -> {r_1, r_2, ..., r_k}

M is a mapping from the set of places to the integers, which represents the tokens in the places in the net. D and R are mappings from the set of places to the set of real numbers representing the duration values, and from the set of places to a set of resources, respectively.

Transitions are represented by vertical lines. Places are shown by circles. Arcs are directed lines between places and transitions. A place is marked if it has a token, indicated by a dark dot. A transition fires when each of its input places has a token. A transition occurs instantaneously. Upon firing, the transition removes a token from each of its input places and adds a token to each of its output places. After receiving a token, a place remains in the active state for the interval specified duration. During this interval, the token is locked. When the place becomes inactive, or upon expiration of the duration, the token becomes unlocked.

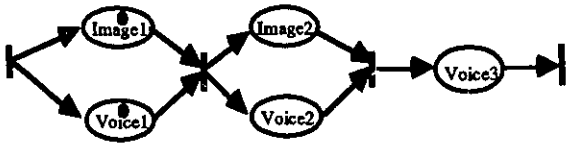

Figure 2.2.2 The OCPN of the multimedia slide presentation

For example, a multimedia slide presentation (Fig.1.1.b) consists of a sequence of synchronized voice and visual elements of varying duration. The presentation of the two streams of information occurs concurrently. Fig. 2.2.2 shows the corresponding OCPN representation.

The corresponding OCPN representations for the seven basic temporal relationships are shown in Fig.2.2.3. It is possible to represent arbitrarily complex synchronization requirements with this technique. Various degrees of granularity are possible to be represented in the OCPN. In addition, the OCPN model can well support the packet scheduling at the transport level[Lit91]. Research has also been done to extend the OCPN to incorporate user's interactions in interactive multimedia applications.

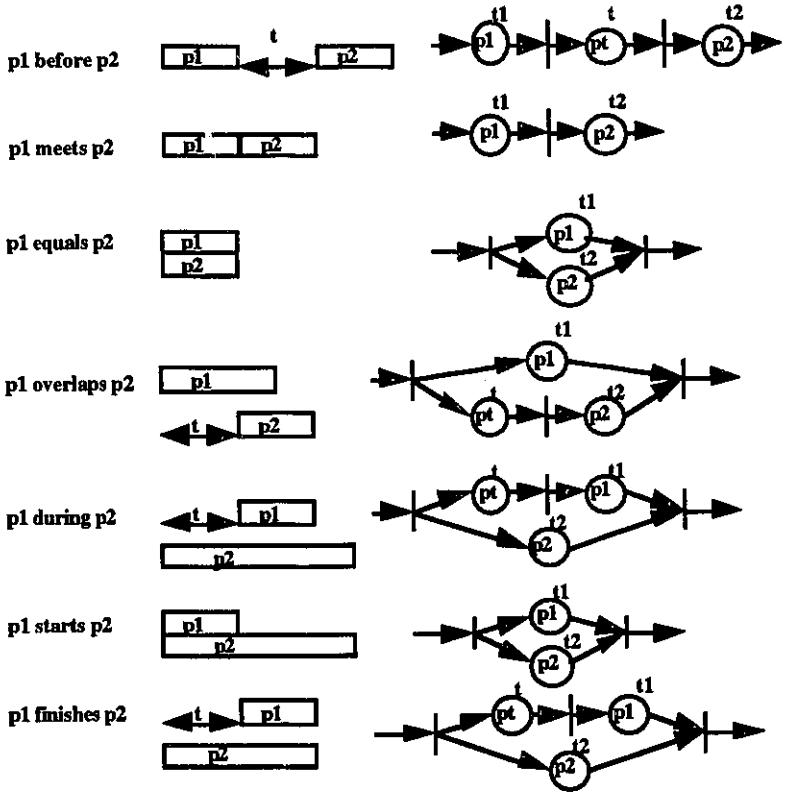

Figure 2.2.3 The seven basic temporal relations and the OCPN

2) TFG

In [LiLi94.2-1], the model of interval vectors and the involved temporal information are maintained in a Time Flow Graph (TFG). The TFG provides distinct graphic structures for the basic interval relationships, as shown in Fig.2.2.4. In contrast, the OCPN model relies on the duration information in the representation. The basic OCPN models for the relations of "during", "finishes" and "overlapping" can only be distinguished by the different duration values of the involved intervals. Because of the strength of representing the relative temporal relationships with the unknown interval values, the TFG model can handle the "fuzzy" scenarios that contains imprecise synchronization constraints, such as unknown object presentation durations and relative event occurring times[LiLi94.2-2].

A TFG is defined [LiLi94.2-1] as a directed graph $\{\Delta_N, \Delta_t, E\}$, for a scenario which has an object interval set N. Δ_N is a set of nodes for the object interval vectors. Δ_t is a set of transit nodes and E is a set of directed edges.

For an object interval vector n_X, its model Δ_X is composed of an interval node N_X, representing the interval of the object, and δ node(s), representing the flexible time intermission between this object interval and the other intervals. Δ_X can be one of the following forms:

$(\delta_X, N_X, \delta_X)$, (δ_X, N_X), (N_X, δ_X), (N_X)

If the start time of n_X has been specified in the scenario, for example $n_X(s)n_Y$ or $n_Y(m)n_X$, Δ_X will not have a δ_X ahead of N_X; if the end time of n_X has been specified in the scenario, for example $n_X(f)n_Y$ or $n_X(m)n_Y$, Δ_X will not have a δ_X behind the N_X; if neither the start time nor the end time of n_X has been specified in the scenario, Δ_X will have δ_X nodes at both ends of N_X; At last if both the start time and the end time of n_X have been specified, no δ_X node is in the Δ_X. In the model, the δ_X nodes do not exist independently and only belong to the node N_X.

The fixed intermissions between object intervals in the scenario are represented by a special type of nodes N_τ. N_τ has a duration of τ. N_τ is included in Δ_X in $n_Y(o)n_X$ (or in Δ_Y in $n_X(o)n_Y$).

The transit nodes have zero durations. There are two special types of transit nodes in the Δ_t, N_S and N_e. The TFG can be decomposed into a sequence of activity vectors, the *AC*s, which

are aggregated sub-graphs of the nodes. Each activity starts with one and only one N_S, and ends with one and only one N_e. The start time of N_S is earlier than any other node in that activity. The N_e node is a common end node in a TFG representing an activity. The start time of N_e is later than any other node in that activity.

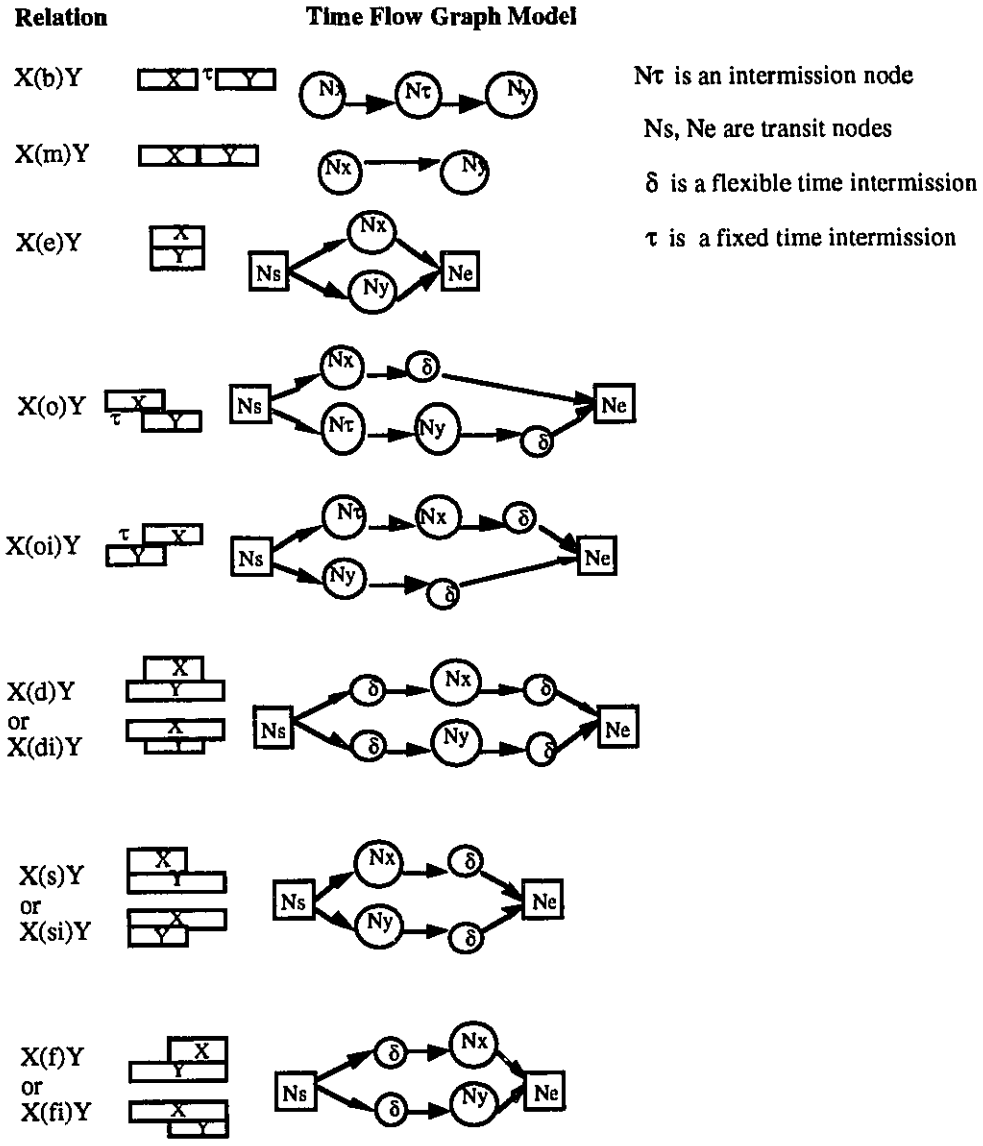

Figure 2.2.4 The basic temporal relationships and the TFG representation

Considering that in the distributed multimedia system, the precise time values such as the presentation durations may not be globally known at the early stage of the presentation scheduling, the temporal model should allow scheduling functions to perform with an uncertainty of information. Additionally, the network fluctuation will bring in delay variations to the remote presentation. The presentation start time and the duration should be adjustable, in order to accommodate the synchronization errors and preserve the high quality of presentation. This requires the temporal model to preserve its consistency and correctness in face of modifications on the involved interval values. The TFG, on the one hand can represent all types of interval relationships without the precise knowledge of the interval durations; On the other hand, this interval-based model supports temporal scheduling with the temporal knowledge it has acquired. Therefore, the TFG model becomes a good candidate in distributed synchronization control. A fully distributed delivering schedule algorithm was proposed in [LiLi94.2-2] based on the TFG model.

2.2.2 Real-time Synchronization Control Schemes

There are two basic approaches to multimedia communications synchronization. One is a real-time synchronization control scheme, the other is a store-and-forward scheme. In the latter approach[For93][Mil93], the multimedia components of a document are transmitted to the destination and stored until all components arrive. Then the multimedia document is played back according to the temporal relationships defined in the document scenario. Multimedia data delivery is performed as file transfer. It requires simple synchronization control, but needs huge buffer storage (esp. for video components) and causes large response delay for user interactions.

The real-time application requires information to be transmitted in continuous mode. Real-time synchronization control is adopted to guarantee a continuous delivery and the required temporal relationships among the multiple data streams. Several synchronization control methods have been introduced for solving the problem of stream synchronization.

1) Synchronization Control Units

When the data objects are delivered onto the communication network as data streams, the temporal relationships have to be maintained during the transmission in order to realize a real-time data presentation at the receiver site. Therefore, synchronization is required not only at the object composition level, but also at the multimedia stream transmission level.

There are continuous media (CM) streams, such as voice data stream or video data stream, as well as discrete media (DM) streams, such as computer I/O data [Ste92]. The concept of Logical Data Unit (LDU) was introduced to the CM streams in [Ste92] as the "granularity" of a media stream in the synchronization control. For instance, the LDU of a video stream can either be a single frame or a well-defined sequence of frames (Fig.2.2.5). Each kind of medium has an associated LDU. The CM streams thus are viewed as sequence of LDUs. The temporal relationships among the LDUs of different media can be described.

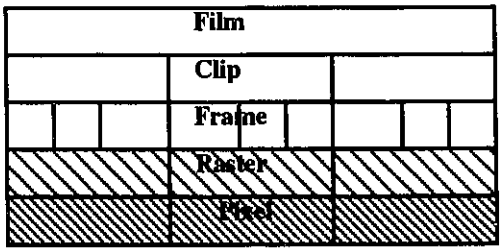

Figure 2.2.5 The granularity of LDUs

2) Intra-stream & Inter-stream Synchronization

Because of the random network delay, the continuity of the data stream will be interrupted. As shown in Fig. 2.2.6 [LiLi92.6], the random network delay introduces jitters (delay variations) to the data stream during transmission. These jitters destroy the presentation quality at the receiver. This continuity problem, called *"intra-stream synchronization"*, exists in all kinds of packet switching networks. To solve this problem, network supported delay variation guarantee and a

jitter smoothing buffer at the receiver are essential. In [Fer93], Ferrari proposes a delay jitter control scheme that ensures media synchronization as long as a bound on delay can be guaranteed and the source and destination clocks are kept in synchrony. Every node on the transmission path in the network is involved in the synchronization control by regulating the random delay which has been introduced to an arriving packet by the previous hop. However, it might be difficult to implement this scheme on an ATM network because it introduces too much workload on the intermediate nodes that carry thousands of connections.

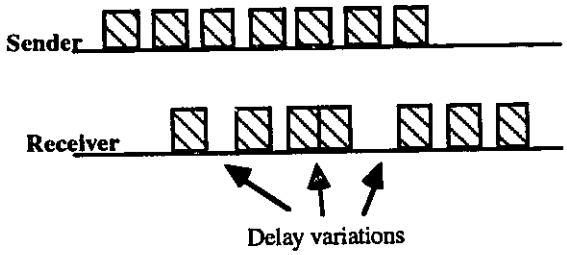

Figure 2.2.6 Delay variations introduced by the random network delays

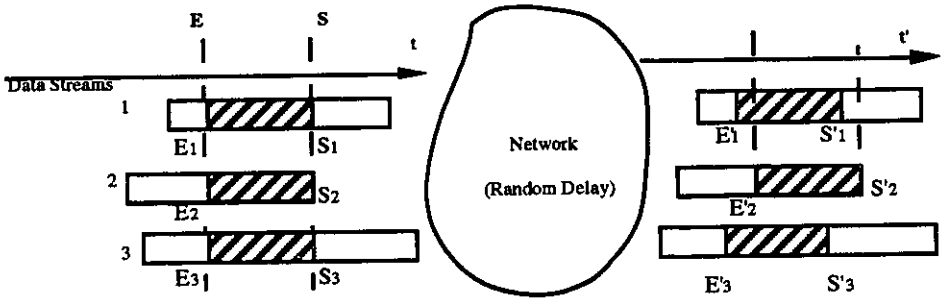

Figure 2.2.7 Mismatches in the simultaneous multi-stream delivery

The variable network delays also give rise to mismatches among the multiple data streams which are delivered simultaneously (Fig. 2.2.7) [LiLi92.6]. The synchronization constraints among the related streams are required in various multimedia applications, therefore *"inter-stream synchronization"* should be provided. "Tight coupling" is required in some applications. A

typical example of strict time matching is the "lip-sync" problem in video conferencing. The movements of the speakers' lips have to match exactly their voice. On the other hand, sometimes, tight coupling is not strictly required. In the case of a music service application, for instance, the Hi-Fi music and a background video are sent to a remote user for the purpose of entertainment. The display of the video showing, e.g. views from nature, does not require any strict matching with the music. In this case, a "loose coupling" is sufficient. In video conferencing, for example, voice data may couple the video with a tight "lip-sync" while short text annotations loosely couple the voice by displaying the summary of the speech [LiLi94.5].

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. background 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

Table 2.2.1 Quality of service for synchronization purposes

The "tight" and "loose" coupling should be described quantitatively for synchronization control. This is a very important issue in multimedia synchronization studies. From the

experiments conducted recently by Steinmetz in IBM Heidelberg[Ste93], the synchronization errors that can be tolerated by human perception vary in different application scenarios. For instance, 80 msec mismatching in lip-sync of voice and video can be perceived as disturbing by the application users. On the other hand, if the text annotation is played out 200 msec earlier than the voice, the presentation can still be well accepted. We call these application defined parameters the *"skew tolerance parameters"*. Skew tolerance parameters determine the inter-stream synchronization goal. The issue of specifying the different coupling types and the corresponding skew tolerance parameters, and also the derivation of the parameters between the indirectly related data streams have been thoroughly studied by Steinmetz[Ste93]. According to his study, Steinmetz has proposed that the skew tolerance parameters be taken as the Synchronization QoS (S-QoS) parameters and included in the QoS requirements for the multimedia transmission network (see Table 2.2.1).

3) A Single Virtual Connection or Multiple Virtual Connections

A straight-forward approach to solving the inter-stream mismatching problem is to multiplex the data (LDU) from different streams onto a single virtual connection (Fig. 2.2.8)[Ste92]. These combined LDUs are time-stamped at the source and demultiplexed at the destination according to the time stamps. Since the combined LDUs are delivered in order, this scheme greatly simplifies the synchronization control. However, this solution would demand from the network connection the most stringent QoS parameters of all involved media. A more efficient solution is to carry each medium stream on a specific highly matched connection.

Different data streams have very different characteristics. For instance, voice data are very sensitive to the response time but can tolerate certain data transmission errors. On the other hand, video data are very sensitive to transmission errors and also require a very high throughput. Data of different media by nature cannot possibly be described by one set of traffic parameters and located on a single connection. In many distributed multimedia applications, data streams can originate from sources at different sites. For example, they may be retrieved from distributed server machines. Then, it will not be efficient to put all the data streams on one connection. On

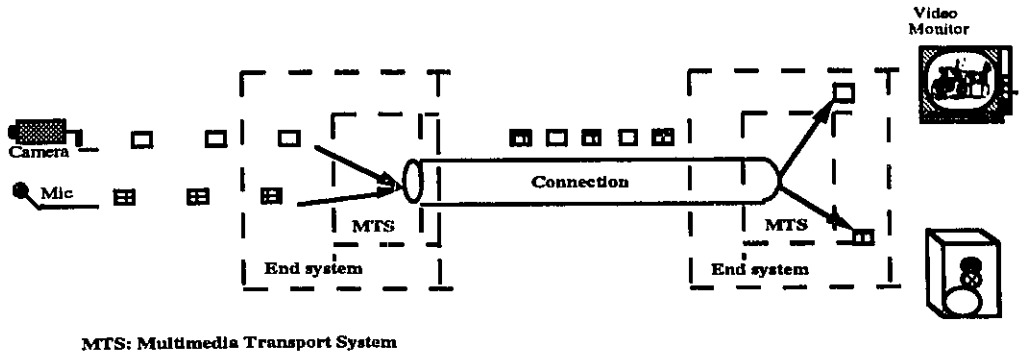

Figure 2.2.8 Multiplexing related streams onto a single network connection

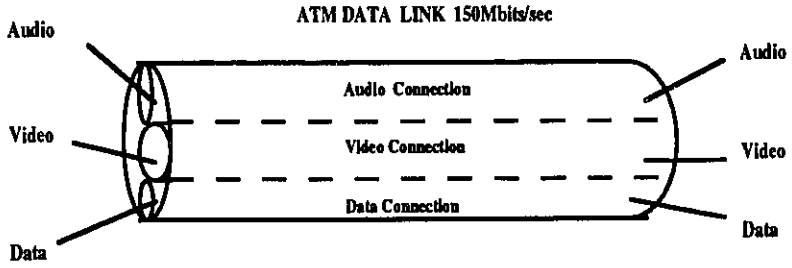

Figure 2.2.9 Multiple virtual connections over an ATM network

broadband ISDN, such as ATM (Asynchronous Transfer Mode) networks, a connection is assigned to a certain class of traffic, for example, a voice connection or a video connection etc., as shown in Fig.2.2.9 [Ste92] . For efficient utilization of the bandwidth, as well as for flow and traffic control on the ATM network, connections are managed according to the traffic characteristics, such as the average peak rate, the burst length of the source, etc. Meanwhile, the QoS (Quality of Service) parameters required by every connection, such as the tolerable cell transfer delay, cell loss and the smoothing requirement have to be supported by the underlying ATM network.

4) Synchronization Control Markers or a Synchronization Control Channel

When multiple data streams are transferred on separate virtual connections, the matching among the data streams becomes a problem. Extensive work has been done on this subject[Ste92][Shep92][Lit91.12][LilLi94.5].

The main idea is to synchronize multiple streams according to control units, i.e., all the data belonging to the same control unit should be displayed at the same time. Synchronization actions are carried out on the multiple streams at the beginning of each control unit. There are two types of synchronization actions, either to delay the presentation of the data units on the fast stream, or to skip a few data units on the slow stream. Because of the random network delays, the mismatching among the streams probably accumulates inside a control unit. The smaller the synchronization control units are, the tighter will the multiple stream be synchronized, though smaller control units also cause more transmission overhead and require more synchronization actions. Performance evaluation for different segmentation schemes to produce synchronization control units can be found in [LiLi94.5].

The synchronization control unit can be identified by the "synchronization markers" which are inserted by the source into the data streams. These markers modify the user data and the modification may not be easy to carry out on the live data in a real-time system. The synchronization channel scheme described in [Shep90] uses an additional channel to carry synchronization information. As a result, the user data need not be modified. However, it introduces overhead in the network since an additional real-time control channel is required.

5) Synchronization Scheduling

It is worth noting that the data storage sources give us more freedom in controlling the packet production times than that of the live data sources. With the retrieval schedule, the data from separate sources will arrive at the destination just before their playback deadline. Therefore, there are less mismatches among the arriving streams and smaller receiving buffers are required.

In [Lit90], Little et al. proposed a computing schedule before transmitting the data packets. Based on the *a priori* knowledge of the temporal specification using OCPN model, a playout deadline for each media components can be generated. A thorough analysis was also given on the end-to-end delay over a packet switching virtual connection of either a packet or an object (which contains a large number of packets). A retrieval schedule was in turn generated which would compensate major delay mismatches among the separate data streams (Fig. 2.2.10).

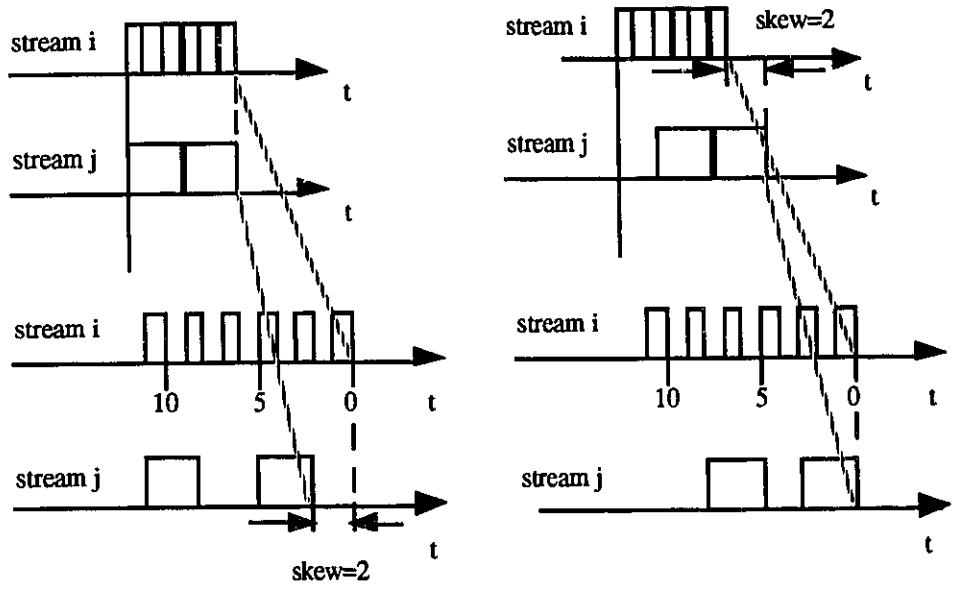

Figure 2.2.10 Compensating the mismatches by the delivery schedule

6) Synchronization Error Recovery

Although schedules and estimates can be made for the multimedia data to be transferred, random network disturbances and synchronization errors are inevitable. Scheduling the traffic alone is not sufficient to maintain a simultaneous multi-stream delivery. Therefore, at the receiver, certain compensation is still necessary, when the synchronization error occurs.

The synchronization error recovery performed on the synchronization control units relies on the deadline meeting mechanism. A Logic Time System (LTS) was introduced[And90] to the continuous media synchronization control, in which the data units with the same time stamp are

displayed or collected at about the same time. The rate of running LTS is either device driven or connection driven. A logic device adjusts its rate by skipping and pausing. Skipping speeds up a device while pausing slows it down. For example, a video output device can pause while re-displaying the last frame, or skip a frame and start to display the successive frame in order to speed up. The pausing and skipping are primitive synchronization operations at a receiver. Skipping data due to the violation of deadline introduces data loss to a stream. Since the compressed data streams are very sensitive to the data loss, the synchronization control over the coded streams need further studies.

Instead of the local synchronization recovery, Rangan et al. present a feedback technique for multimedia on-demand services[Ran91]. The centralized multimedia server uses feedback units transmitted by the receivers to detect the asynchrony among them. Correction of the asynchrony is made at the source (the multimedia server) by speeding up or slowing down the traffic on the various media streams. When using this scheme, it might be difficult to detect and correct the asynchrony before the user can notice them when the network is heavily loaded. A real-time synchronization method is needed to display the document in real-time and to preserve the synchronization among the different streams.

7) Layered System Model for Synchronization Control

Different levels of synchronization control can be employed at different layers in a multimedia architecture. In [Lit91], Little et al. proposed a two-level synchronization control architecture. The lower level, called the Network Synchronization Protocol (NSP), provides functionality to establish and maintain individual connections with some specified synchronization characteristics. The upper level, the Application Synchronization Protocol (ASP), supports an integrated synchronization service for multimedia applications. The ASP uses temporal relationships of an application's data objects and manages the synchronization of arriving data for playout. The proposed NSP and ASP are mapped to the session and application layers of the OSI reference model, respectively.

Nicolaou proposed a three layer multimedia architecture in [Nic90]. The multimedia mechanism layer provides the multimedia communications supports. The distributed computing systems layer controls the applications. The interface between the distributed computing systems layer and the multimedia mechanism layer is referred to as the Orchestration layer which implements the heterogeneity management mechanism.

Considering the various tightness of synchronization requirements, the synchronization control can be performed at different control unit granularities. For example, the application wishes to control a video stream by the video frame level, while the video stream is implemented using a protocol which transmits four samples per frame. These samples are referred to as Physical Synchronization Frame (PSF)[Nic90], which are intended as the unit of synchronization within the communication sub-system, i.e., in the multimedia mechanism layer. The video frames are served as Logical Synchronization Frames (LSF), which are buffered, displayed or re-displayed under the application synchronization control. The LSF is the control unit of the Orchestration layer.

Research work has been carried on in the areas of realizing multi-channel synchronization at the application or session layer, designing the light-weighted transport protocols, integrating the multimedia application with the ATM transmission services, and the QoS negotiation and enforcement at the various network levels.

Further research is required to design suitable synchronization control sub-systems to meet the various requirements of the multimedia services, and to incorporate with the other components in the multimedia system. The issues which relate to the ATM communications services, the QoS negotiation, and the data coding/decoding schemes need to be fully investigated. After all, prototype experiments of synchronization schemes on ATM-backbone networks are highly desired for system development and implementation.

Chapter 3

Design of the Real-time Synchronization Control System

3.1 The Multimedia News-on-demand System

The CITR *Multimedia News-on-demand* is being considered as a target multimedia presentational application. In the following part of the thesis, the real-time synchronization aspects related to the *Multimedia News-on-demand* system are investigated.

Multimedia news may include components such as video, audio, image, graphics, text, etc. The multimedia news is delivered to its users through a broadband network. This provides a much more expressive, animated and timely service than a traditional newspaper does. Moreover, the users have an interactive environment over which the flow of information can be controlled. They can survey the news database taking advantage of world-wide news sources, choose the topics they are most interested in, "read" the news content in the same way as if they were viewing tapes from a VCR. This user-interaction feature distinguishes multimedia news-on-demand from TV news broadcasting. An integrated multimedia news system over a broadband network is introduced as follows (Fig.3.1):

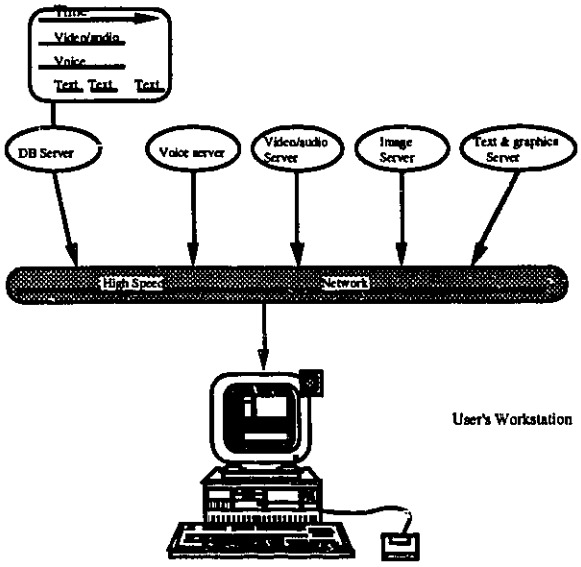

Figure 3.1 A *Multimedia News-on-demand* presentational application

In a Multimedia News system, multimedia news is created, edited and stored in a distributed news database. The database consists of a special database server and multiple media servers, e.g., an MPEG video & audio server, a voice server, a text and graphics server, an image server, etc. These media servers perform storage and retrieval operations, which are adapted to specific media types or data formats. By introducing the media servers, the performance bottleneck in centralized multimedia storage and retrieval could be removed. The database server stores all the document structures (esp. the temporal structure called document *scenario*). It also handles database management for news generating and updating.

The user's workstation is capable of a multimedia presentation. It has a screen, a loudspeaker, hardware decoders, as well as decoding and displaying software. The presentation is characterized by the spatio-temporal integration of the multimedia objects[Lit90.9]. The temporal integration produces the required presentation sequence of the objects. The document scenario represents the temporal relationships[Lit90.4] among the data objects involved in the document. Through a user interface, the user is able to issue commands to browse the index and select a topic, and to a play-back a news clip in chosen speeds forward or backward.

The remote users' workstations are connected to the Multimedia News database through a Broadband-ISDN. Considering that the user may not have gigabytes of memory or disk space dedicated to multimedia news retrieval, the multimedia news article is progressively transmitted over the network and is presented to the user simultaneously. Data in different media are transferred over different ATM virtual connections or streams according to their traffic characteristics and transmission requirements. For instance, voice is transferred on a 64 kbits/sec connection while high quality uncompressed digitized video requires a connection of more than 30 Mbits/sec. In order to support this real-time application, a real-time Stream Synchronization Protocol (SSP) is required to preserve the temporal presentation of the data objects at the receiver[Lam94.5]. A priority-based synchronization control scheme is provided for coded streams. The stream synchronization control integrates into the multimedia communication architecture.

3.2 The Time Flow Graph for Scheduling

Before we discuss the architecture of the real-time synchronization control system, we have to first choose an appropriate temporal model in order to capture the various synchronization requirements and to support the real-time synchronization operations. To compare the temporal models, we have considered the following important characteristics that are relevant to our work.

1) Interval-based vs. Point-based

In [Ste92], the "relative synchronization operation" was defined, referring to the imprecise and relative temporal requirements involved in a presentation scenario. Contrasted with the "absolute synchronization operation"[Ste92], which is required directly through the time parameters, the "relative synchronization operation" is described only by certain references without explicit time parameters in terms of the start and end times, or durations. For example, "present object A at the

time T_A" is an absolute synchronization operation. On the other hand, "present object A simultaneously with object B" is a relative synchronization operation.

In order to provide a presentation time schedule, the system must internally translate the requirements in the relative synchronization operations to the absolute operations. It is necessary for a temporal model to represent the relative and fuzzy temporal requirements, and to support the scheduling operation.

The "point based" models represent relatively rich temporal relationships that can be involved in a multimedia presentation[Heo92][Buc92]. But they do not take into accounts the interval durations. Therefore this type of models cannot readily generate a time schedule for the absolute synchronization operations. In contrast, the "interval based" temporal models, which provide support to the interval scheduling, were thus chosen for the real time synchronization system design.

2) Fixed Intervals vs. Flexible Intervals

We observed that in many applications the time relationships among the objects are imprecise and flexible. For example, a scenario may require that subtitles be presented during the presentation period of a video show (Fig. 3.2.a). The time difference, τ, between the start time of the text subtitle and the corresponding video clip is not specified. This scenario implies that the start time of the subtitle display can be delayed by a certain amount as long as the subtitle stays on the screen for a sufficient period and disappears before the video clip ends. Such temporal flexibility of the scenario allows the application to tolerate certain amount of the network delay variation which is inevitable in the remote presentational multimedia applications. The temporal model we choose should be able to represent this type of relaxed synchronization constraints among the media objects. The synchronization recovery operations based on such a temporal model can thus become more efficient, by taking advantage of the application tolerance of certain time shifting.

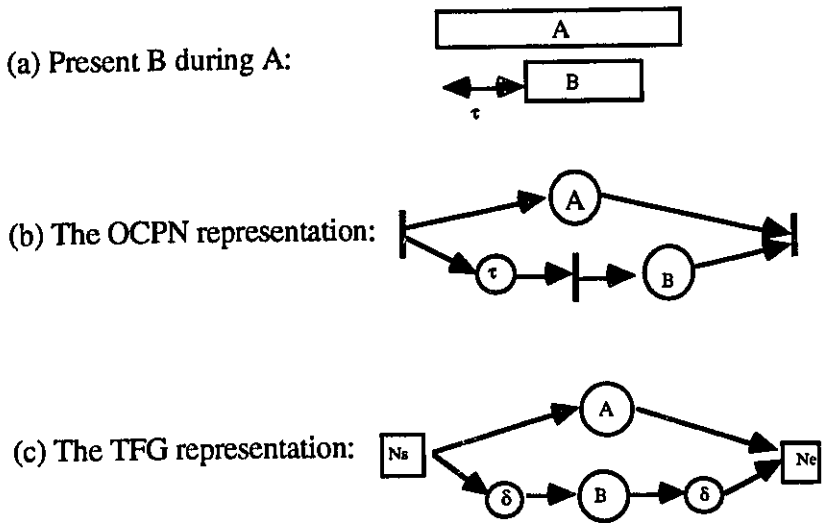

Figure 3.2 The temporal relationship and the corresponding models

In the OCPN model, the time interval with a fixed duration value is taken as the primitive. The flexible intermissions among the objects are converted to the fixed ones as the OCPN model is built up. For example, in the scenario mentioned above, if the duration of the object A and the object B are D_A and D_B respectively, the OCPN model (Fig. 3.2.b) will specify an intermission node between the node A and the node B with a duration value of τ, $\tau < D_A - D_B$. Consequently the presentation schedule specifies the start time of A is T_A and the start time of B is $T_B = T_A + \tau$. With the tight dependence on the interval duration values, the OCPN model loses much of the tolerance which is possessed in the original presentational scenario. The OCPN model for the relationships of "during", "finishes" and "overlaps" can only be distinguished by the different duration values of the intermission intervals. For example, in the Fig.3.2.b, if the value of τ equals $D_A - D_B$, the relationship between A and B becomes "A finishes together with B"; If τ is larger than $D_A - D_B$, the relation becomes "B overlaps A". Therefore, in order to maintain the consistence of the model, change of the duration value of the intermission τ in the OCPN is not allowed. In the example of

"B is presented during A", if the data of A arrives late for the scheduled start time T_A by a value of ξ, the start time of B has to be postponed by ξ since the intermission between T_A and T_B has been fixed to a specific value of τ. As a result, the start time of B is postponed and additional buffer space is required to store the data from B's network connection.

The TFG models, on the other hand, support the flexible temporal relationship specification. Therefore, if one object is late for its scheduled start time, the synchronization error is accommodated by the flexible intermissions, so that the unnecessary synchronization error recovery operations are avoided. The consequent buffer requirement of the postponed schedule is also reduced. The TFG model introduces a δ node to represent the flexible intermission between the object nodes. The model distinguishes the different relationships by different graph structures. The relationships of "during", "finishes" and "overlaps" are specified in different graphic structures instead of the duration values of intermissions. Fig.3.2.c shows the TFG representation of the example scenario "B is presented during A". If the data of A arrives late for the scheduled start time T_A by a value of ξ, the start time of B remains the former value T_B as long as the value of ξ is smaller than that of D_A-D_B. The display of the object B is not postponed and no additional B buffer is necessary.

Based on the above considerations, we chose TFG to be the temporal model in the synchronization system design and implementation. From the multimedia document scenario, a Time Flow Graph is generated on which the scheduling algorithm can be executed.

3.3 Synchronization Control System Design Overview

In the *Multimedia News-on-demand* service, the real-time performance and the buffer space demand on a remote user's workstation are both constraints on the system design. Instead of synchronizing the multimedia components in presentation in a store-and-forward manner, we designed a real-time synchronization control system to guarantee a continuous delivery and

presentation and, in the meanwhile, to maintain the required temporal relationship during presentation. The functional entities required to perform the synchronization consist of a data delivery Scheduler and pairs of Media Synchronization Controllers (MSCs) at both ends of each connection, i.e., the server MSCs at the Media Server's site and the client MSCs at the user's site (Fig. 3.3).

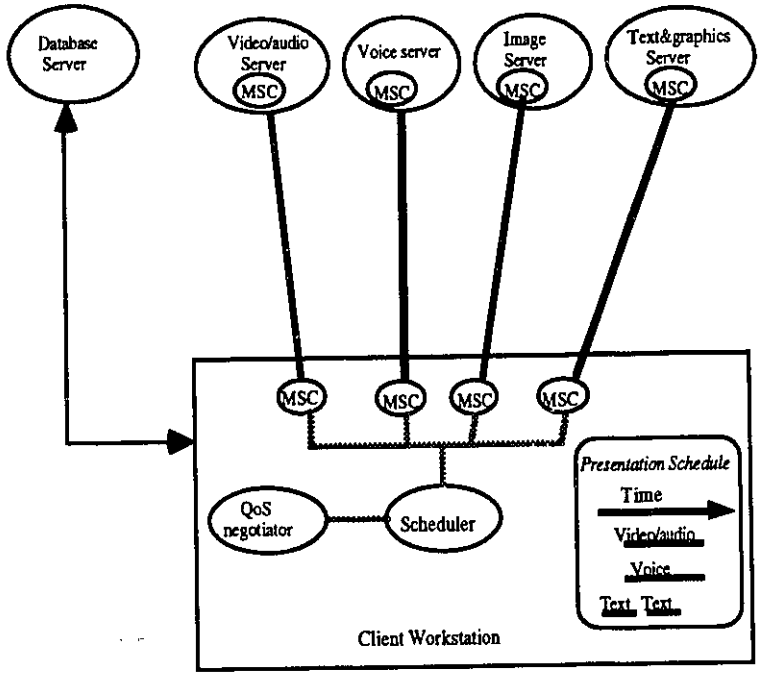

Figure 3.3 The functional entities of the multi-stream synchronization

Real-time stream synchronization control is built on the concepts of prevention and recovery. The prevention schemes are employed before the start of the document delivery. When the *database server* accepts a document presentation request from one of its remote *clients*, a *scheduler* at the client's site is invoked and starts to develop a delivery schedule. Appropriate connections from the distributed media servers to the client's workstation are set up to reserve the network resources for the performance guaranteed transmission. A pair of MSC processes at both ends of each media stream connection are invoked, which are responsible for the data delivery and receiving respectively. Each object component involved in the requested multimedia document will

be sent at a scheduled delivery time, so that it will reach its destination around the time when the component is required for the document presentation. As the delivery schedule takes into account the network connection characteristics and provides compensation for variable delays over separate streams, major synchronization errors among media components in a target document presentation are thus prevented. In addition, fine-tuning operations are taken at the receiver, the client's site, to deal with problems caused by the network delay fluctuation, clock mismatches, and any inaccurate estimations in the delivery scheduling. Given that some late arrivals may occur in one of the streams, the client MSC processes have to adjust the target presentation schedule for the subsequent media objects so that the synchronization constraints among them will still be held true at the time when the subsequent objects are played back. Consequently, recovery buffers are necessary. An efficient buffer management scheme can be developed based on the knowledge of the presentation schedules.

3.4 The Prevention Operations

3.4.1 QoS Negotiation for the Virtual Connections

ATM network connections support guaranteed Quality-of-Service(QoS) to network users. This feature greatly assists the connection-oriented multimedia transmission. The multimedia data streams have far different characteristics. For instance an uncoded voice stream requires small delay jitter, but it can tolerate packet loss. On the contrary, a coded video stream demands high bandwidth and is very sensitive to packet loss, but it can tolerate delay jitter around 100 ms with the aid of recovery schemes such as replaying the previous picture. To achieve efficient network utilization, different media streams are transmitted though separate network connections which guarantee certain quality of service. The description of each data object specifies the desired QoS

for the presentation and transmission. For example, a piece of coded video object may have an object Id, the transmission requirement class of delay-sensitive & loss-sensitive traffic, the object size of certain number of frames, the data-unit-size which is the decoded frame size(in bytes), the playback rate of 30 frames/sec., the playback jitter of 3 frames/sec. The desired network transmission QoS parameters can be derived as follows:

1. The class of traffic, is the same as the class of the transmission requirements in the object description;

2. The transmission rate requirement of an object C_i can be calculated as follows: If the data object is of the continuous media type, such as video and voice,

C_i = playback rate (units/sec) * (unit-size + overhead) (bytes);

If the data object is of the discrete media type, such as text and graphics,

C_i = 1 (units/sec) * (unit-size + overhead) (bytes).

If there are more than one object to be retrieved from the same media server in the same presentation request, the transmission rate of the connection is chosen as the highest transmission rate requirement of the involved objects,

$C = \max(C_i)$;

3. The delay variation σ = playback jitter (bytes).

A QoS negotiator deduces the network transmission QoS from the application specified presentation QoS. It is also responsible for the three-party quality of service negotiation among the client, the servers and the network management module [Vog93]. It is possible that several tuples of QoS parameters are stored for a certain media object, which correspond to different levels of presentation quality. The network may provide the transmission service as request or a degraded service according to the network traffic load. The client can also be involved in the QoS negotiation by specifying the user's preference on the services and the cost. The QoS negotiator determines the tuple of presentation QoS which is acceptable for the client, the servers and the

transport system. It specifies the transmission characteristics for each medium stream, the stream connection may be opened up in the QoS negotiation phase. Guaranteed connection average delays and delay variances are fundamental for the data delivery scheduling.

3.4.2 Delivery Scheduling

The data delivery schedule is used to compensate the different delays experienced by the data objects over the separate streams. For a data unit, D_{total} is the total delay from the server delivery to the presentation at the remote client workstation. To meet a target presentation schedule at the client workstation, the data unit should be delivered to the connection D_{total} time units earlier.

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

D_{total} is composed of three types of delays:

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

where $D_{e\text{-}e}$ is the end-to-end network delay on the connection; $D_{buffering}$ is the synchronization buffering delay to compensate the predictable network delay variations; and $D_{decoding}$ is the average decoding delay for the coded data stream. $D_{e\text{-}e}$ is approximated by a Gaussian distributed random variable, which can be measured by the mean μ and variance σ^2. In order to prevent the data unit from missing the presentation schedule due to the end-to-end delay variation, the data unit should be put into the connection $D_{buffering}$ time units in advance. $D_{buffering}$ is determined by the probability P(fail) which is the acceptable late arrival probability specified by the multimedia application,

$$P[(D_{e\text{-}e} - \mu - D_{buffering}) > 0] < P(Fail);$$

For example,

$$P[(D_{e\text{-}e} - \mu) > \sigma] < 16\%,$$

so that setting the $D_{buffering}$ to σ will guarantee the late arrival error to be less than 16%. The buffering delay is spent at the recovering buffer of the synchronization controllers at the client's workstation. Given the μ, $D_{decoding}$ and $D_{buffering}$ for each connection, the D_{total} can be found by

$$D_{total} = \mu + D_{decoding} + D_{buffering}.$$

The delivery schedule can be derived from D_{total} and the presentation schedules. By opening up appropriate connections and using media object delivery scheduling, major synchronization errors among multimedia components are prevented.

3.4.3 Interactions during the Delivery Scheduling

A scheduler on the client's site schedules the media object delivery so that the data will experience very little buffering at the destination before being played back. When the multimedia database receives a request for viewing a specific news document, it sends the document presentation scenario to the scheduler on the client workstation. The scheduler determines which media types are involved in the scenario and creates the corresponding client MSCs. A client MSC is responsible for receiving and transferring to the user interface the object(s) of a particular media type which arrived from a single stream connection. The scheduler opens up a control channel to each media-server involved in the scenario. Several server MSC processes are activated on each media server's site, corresponding to the client MSC processes at the client's site. The scheduler gets the QoS parameters for each media stream from the QoS negotiator. It notifies the server MSCs to open a unidirectional communication stream to a peer client MSC. The MSCs at the different servers' sites open up a connection to the client site with the proper transmission characteristics for each media stream. After the connections are opened, the server MSCs return the QoS that will be guaranteed by the network. The returned end-to-end average network delays(μ) and delay variations(σ) are the parameters that the scheduler will use to compute the start time of each object. The average decoding delay is also an important parameter when calculating

the total delay on a coded stream. Then the scheduler calculates the total delay on each stream connection based on the network delay values and the decoding delay value. The scheduler extracts the temporal relationship information from the scenario and generates a Time Flow Graph (TFG), on which the scheduling functions can be performed. The scheduler produces a presentation schedule which allows the client MSCs to present objects at presentation start times so that the synchronization requirement in the given scenario is satisfied. Then a delivery schedule is produced which allows the server MSCs to deliver the media objects at the specified delivery times so that the objects will arrive at the client MSC at the presentation times after experiencing the different total delay over the stream connections. The scheduler then sends the delivery schedule to all the server MSCs involved in the document delivery.

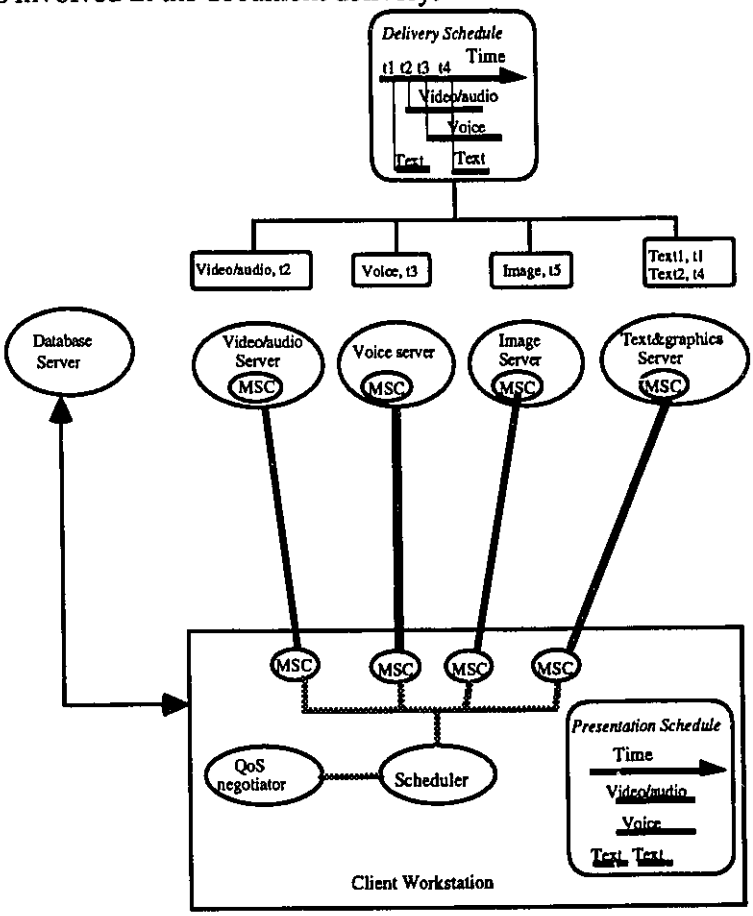

Figure 3.4 The delivery scheduling scheme

For example, in the Fig. 3.4, if the object Text1 is the first object to be delivered, the delivery time offset of Text1 is t_1=0, the Video/audio object has t_2, the Voice object Voice has t_3, and so on. The server MSCs are informed of these delivery time offsets. When the user requests to start the presentation, the scheduler informs the involved server MSCs and thus triggers the document delivery. Each server MSC then delivers the data object(s) according to the scheduled time offset specified in the delivery schedule.

During the above procedures, if any of the involved server MSC is not able to open up a connection because of unacceptable transmission characteristics, it will notify the scheduler. The scheduler in turn notifies the client that the session is aborted.

3.5 The Stream Synchronization Protocol for Client Synchronization Error Recovery

3.5.1 The Stream Synchronization Protocol

Scheduling and predicting the traffic are not sufficient to maintain a simultaneous multi-stream delivery since the networks introduce random network disturbances. Random network delay destroys the continuity of the data stream by introducing gaps and jitters during the data transmission. Therefore certain compensations at the receiver are necessary when synchronization errors occur. A Stream Synchronization Protocol(SSP)[Lam94.5] describes the protocol recovery operations of the client MSCs. The concept of an "intentional delay" is used by the various streams in order to adjust their presentation time to recover from network fluctuations. The document scenario specifies the temporal relationships among the media components in a particular document presentation. These relationships are modeled by a TFG. Based on the TFG model, a presentation schedule for every involved media data object (media component) is derived. The common presentation schedule is shared by the concurrent client MSCs. The client MSC processes are required to start and end their operations according to the shared schedule, so that the multimedia

news article will be played back in synchrony. SSP control is implemented by mutual process state checking and presentation re-scheduling.

Communication between the client MSCs is required when the client MSC receives the first transport unit of a media object, and when synchronization error recovery is needed because of late data unit arrivals (units not being received in the allowable time frame).

The synchronization errors that can be tolerated by human perception vary in different application scenarios (referring to section 2.2.2 of this thesis). Different skew parameters are defined and stored in the presentational scenario. The client MSCs use the skew parameters to determine the mismatching tolerance time admitted by the application before applying synchronization error recovery.

When arriving data units exceed the skew tolerance parameters, a less rigid policy may be applied. The application is aborted only when the skew between related streams has been exceeded a number of times indicating that there is a problem on one or many of the streams. These control policies are application dependent.

3.5.2 The Shared Presentation Schedule and the Intentional delays

To prevent the tolerated synchronization errors from being rampant to the following stream presentation, the client MSC which detects the late arrivals should make necessary changes in the shared presentation schedule, according to the synchronization requirements extracted from the TFG model. Certain amount of "intentional delay" is placed on the presentation of the related media objects. While the intentional delay is in effect, each affected client MSC may have to handle the extra buffering demand or it may have to extend the object display by an additional period of time.

By performing such a presentation re-scheduling operation, the multimedia presentation will be adapted to the random network disturbances. The presentation schedule is stored in a table in the shared space, and it is accessible to one client MSC process at a time. Therefore the

consistency of the shared schedule is guaranteed through the re-scheduling operations of the concurrent client MSCs.

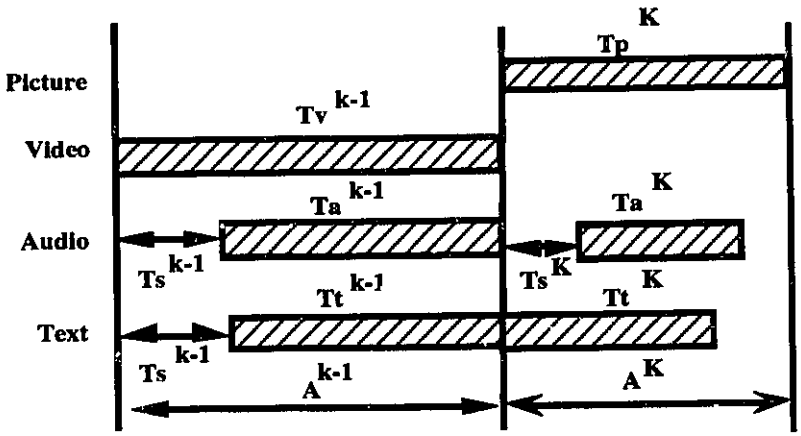

Figure 3.5.1a Parameters in the presentation requirement

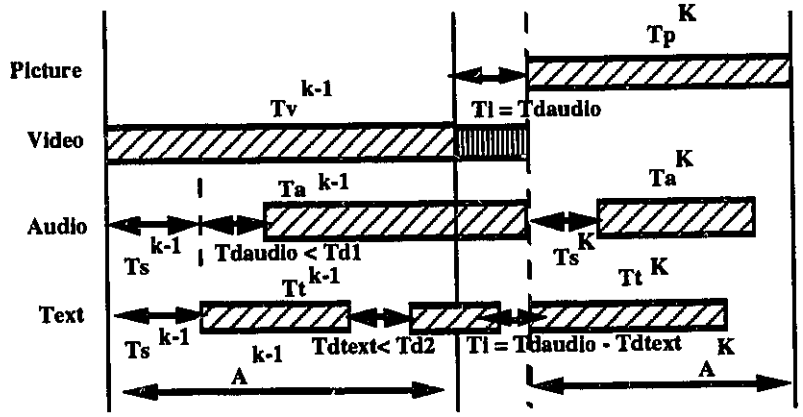

Figure 3.5.1b Mismatch and recovery

For instance[Lam94.5] in Fig.3.5.1a, the MSCs involved in the first activity of the scenario, get the video duration time length T_v, the starting time difference for the text and voice activity T_s and the text and audio duration time T_a & T_t from the presentational scenario. The skew tolerance time between the audio and the text objects T_{d1}, and between the video and the audio/text objects T_{d2} are also retrieved. The client MSCs accept late arrivals as long as the mismatching does not exceed the tolerance T_{d1} and T_{d2}. Late arrivals in at least one of the objects cause gaps in the

display. In order to filter the gaps and thus decrease the mismatching at the end of an activity, an "intentional delay" T_i is placed on the objects involved in the following activity. The intentional delay T_i for the objects that are delayed by T_d in the previous activity will correspond to $T_{dmax} - T_d$, where T_{dmax} corresponds to the longest delay encountered in the previous activity. While the intentional delay T_i is in effect, each MSC involved should help decrease the interruption caused by the filling gaps. The MSC associated with the video, fills in the gaps by repeating the last frame received in the previous activity while the other MSCs simply pause their display to the screen. In Fig.3.5.1b, the audio and text are delayed by T_{daudio} and T_{dtext} in the first activity, where $T_{daudio} > T_{dtext}$. In the second activity, intentional delays of T_{daudio} and $T_{daudio} - T_{dtext}$ are placed on the picture and text objects in order to preserve the temporal relationship between these objects.

In certain situations, one can also filter the gaps by adding an intentional delay within an activity. For instance, if the video object in Fig.3.5.1a arrives late, the audio and text MSCs can adjust their presentation within this activity by adding an intentional delay equal to the video object's delay.

The first transport unit that arrives on a stream triggers the MSCs to start executing the predefined presentation schedule. When no late arrivals occur, the signaling traffic between the MSCs is reduced to a minimum. Alternatively, when network delay fluctuations occur, the MSCs will communicate to one another and a new presentation schedule will be produced. The idea of adding intentional delays to the subsequent objects allows for the synchronization to be recovered without having to discard or skip late packets. This method, which can also be referred as a "time expanding policy"[LiLi92.6], is very well suited for applications such as *Multimedia News-on-demand* services, since it ensures a high quality multimedia presentation at the receiver.

Figure 3.5.2 illustrates the operations taken by one of the concurrent client MSC processes.

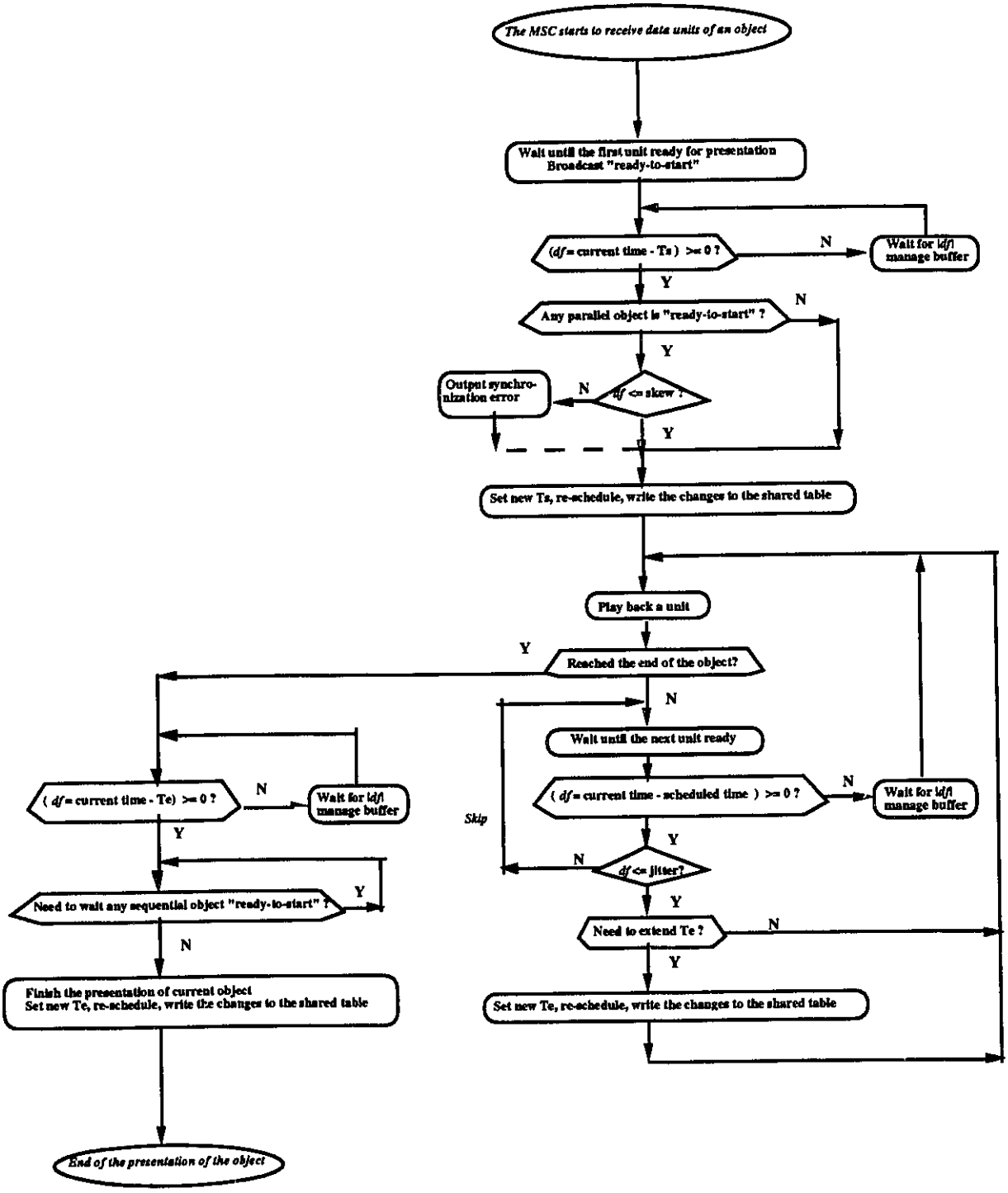

Figure 3.5.2 The flow chart of one of the concurrent client MSC processes

Ts: the scheduled start time of the object presentation

Te: the scheduled end time of the object presentation

Concerning the relative long propagation delay in high-speed networks, no re-schedule message is fed back to the scheduler or the media-servers at the sending end. As the recovery scheme relies on additional local buffering, efficient buffer management is required. With knowledge of the scheduled or re-scheduled start times and end times, the client processes can predict the future buffer requirements and request for more buffer allocation in advance. Also the processes can periodically skip a few low priority data to avoid buffer overflows during the waiting time of the stream. Otherwise, without re-scheduled times, the processes at the client will have to wait blindly for certain signals and cannot predict buffer requirements in advance.

3.5.3 The TFG model's Impact on the Efficiency of the Recovery Operations

Because the TFG model introduces the flexible time intermission to preserve the flexibility of the synchronization requirements, late arrivals occurring on one object can be accommodated by the tolerance of the application, so that the number of objects whose schedule are to be affected by the recovery operation and the amount of the intentional delay to be added are reduced to the minimum.

For example, the TFG model in Fig. 3.5.3a shows the object B can start any time after the object A finishes, while the object C has to start immediately after the object A finishes. There is no constraint in the time difference between the presentation start time of B and that of C. In the initial presentation schedule (Fig. 3.5.3b), the presentation of object B is at 5, which equals to the

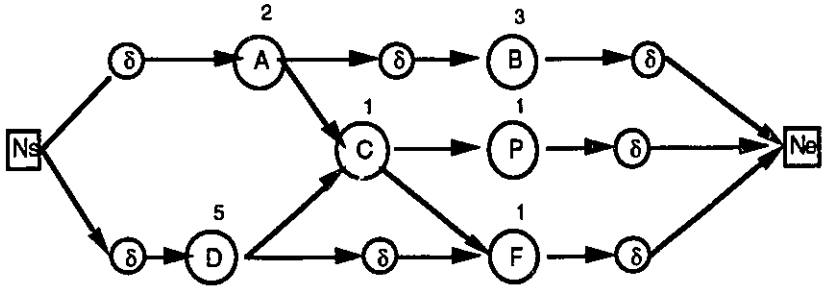

Figure 3.5.3a The TFG of a presentational scenario

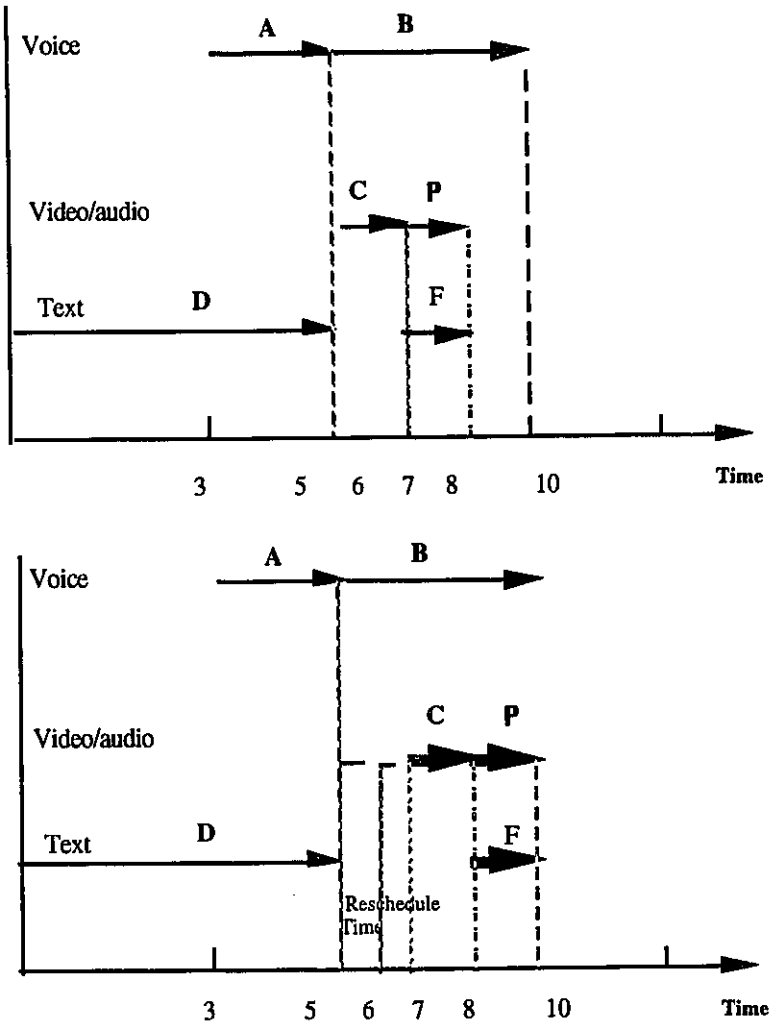

Figure 3.5.3b The re-scheduling scheme if the object C ends 1 time unit late

Plain line_ former values as in the initial presentation schedule

Bold line_ changed values after re-scheduling

start time of C. If the client MSC of video/audio detects late arrival in the object C and decides to extend the scheduled presentation by 1 time unit, consequently the start time of P and F is postponed by 1 time unit too. But the presentation of the object B is not affected. Moreover, the activity can still finish at the scheduled time 8. No intentional delay need to be added to the following activities.

3.6 Synchronization Control over Coded Data Streams

In multimedia communication, transmission bandwidth constraints and storage capacity limitations impose a reduction on the number of bits representing a data stream. Advances in coding techniques are the key to reduced bandwidth requirements. For instance, a raw full motion video requires up to 100 Mbps bandwidth. The MPEG video coding algorithm is capable of providing compression ratios of 100:1 with a picture quality that resembles that delivered by standard VCR tapes[For93]. As the transmission of coded data becomes a practical solution for multimedia real-time applications, we look into the coded stream characteristics, to develop synchronization control schemes which are suitable for their new requirements. Our synchronization control system handles both coded and uncoded streams. A continuous coded stream should be decoded and presented continuously and correctly. Inter-stream synchronization, as discussed in the last section, also applies to the coded streams. Therefore, the decoder should support bounded decoding delay and jitter. Some coding schemes also provide other synchronization support. For example, MPEG-1 coding can guarantee synchronization between a video stream and its associated audio stream[MPEG-1]. However, the MSC pairs are still employed since they can adapt to each type of coded stream and maintain the synchronization among all involved streams.

The important parameters induced by a particular coding scheme are: the throughput, the nature of the bit stream (variable or constant), the compression/decompression delay and jitter bounds, and the structure of the stream[Taw93]. The network transmission bit error rate and delay jitter also have an impact on the coded stream and may lead to quality degradation of the decoded data. We must consider the decoder's capability to adapt to the degradation of its input stream and apply the necessary corrective measures.

3.6.1 The Pre-decoder Synchronization Controller

The client synchronization control system for a coded continuous stream consists of a client MSC and a Pre-decoder Synchronization Controller (PSC). The system diagram is shown in Fig.3.6.1. The client MSC maintains the multimedia synchronization requirements through the Stream Synchronization Protocol (SSP), and the PSC smoothes out the network jitter and provides support to the MSC.

The PSC gets its input from the network and passes the coded data to the decoder according to their target decoding time-stamps. With continuous media coding schemes, the target decoding time-stamps are generated by the encoder and are stored in the coded data unit's structure. When the first coded transport unit of a stream arrives at the client PSC, it is buffered for a predetermined time (the initial PSC delay) and then it is delivered to the decoder for decoding. The PSC checks the difference between the time-stamp of the subsequent unit and that of the previous one to determine the subsequent delivery time. The PSC reconstructs an acceptable timing stream, in order to prevent the decoder's internal buffer from overflows. With the PSC, the decoder is isolated from network delay fluctuations. Therefore, one can choose and combine various types of decoders and network transport facilities.

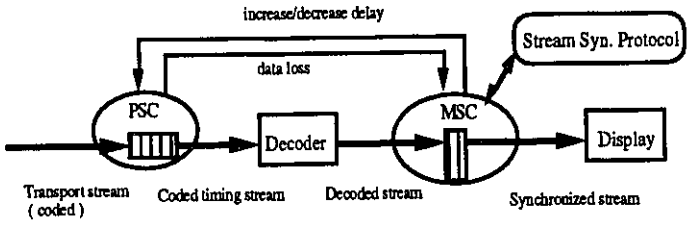

Figure 3.6.1 The client synchronization control scheme for a coded stream

The PSC employs buffering and occasionally skipping to reconstruct a timing stream according to the given time-stamps. When a transport unit arrives early, it is buffered until its target scheduled decoding time is reached. When the PSC detects that a unit is too late for its target

scheduled decoding time, it may skip the unit if the subsequent unit is already stored in the buffer. Considering the limited size of the PSC buffer, possible large network delay fluctuations and the tight timing requirements for the decoder input stream, data may have to be occasionally discarded due to PSC buffer overflows.

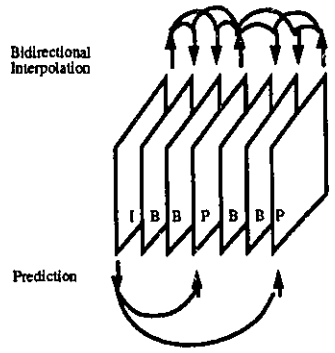

Figure 3.6.2 Temporal picture structure of MPEG video

The data units in a coded stream have different significance for data decoding quality. Priority control should be considered in the PSC, so that the occasional data skipping or discarding does not dramatically degrade the presentation quality. For example, the MPEG-2 [MPEG-2] standard has a presentation unit of pictures, which are coded into three types of coded pictures of different compression ratios. An I-picture (Intra-frame picture) is independently coded and provides reference for P and B pictures. A P-picture is predictive coded, which requires reference of past I-picture for its decoding. A B-picture is a bidirectionally-predicted picture, which requires reference of past and future I-picture and/or P-picture for its decoding(Fig. 3.6.2). Compared to that of P and B-pictures, the data units of the I-picture are of foremost importance for video decoding and displaying quality. If some transmission units of an I-picture are lost, the presentation quality of a group of pictures that require reference to the lost I-picture will be degraded. If some transmission units of a B-picture are lost, only the quality of the present picture will be degraded. If all transmission units have the same priorities, the units corresponding to I-pictures will face higher probabilities of loss than those of B-pictures that are composed of a smaller number of data units. Referring to section 3.6.2, the PSC can extract the

priority information from the particular data structure of the coded stream. In the case of buffer overflow, only the low priority units are discarded.

3.6.2 Priority-extracting Algorithm for MPEG-2 Video Streams

The transport units for an MPEG-2 stream are "Transport Packets". The target decoding time-stamps are derived from the program-clock-reference fields in the Transport Packet header. If the priority control at the granule of a picture is desired, the PSC will apply the priority extracting algorithm to detect the picture coding type. The PSC will assign higher priority to the Transport Packet which carries video data of the I frame, and lower priority to the packet which carries the data of the B or P frames.

The MPEG-2 standard[MPEG-2] has defined *start codes* in its video stream. These *start codes* are reserved bit patterns that do not otherwise occur in the video stream. The *start code* for a coded picture (either of the I, P or B picture format) is 00000100 in hexadecimal. After this *picture_start_code* and a 10-bit code for *temporal_reference*, there follows a 3 bit *picture_coding_type* code which identifies whether a picture is an intra-frame coded picture(I), predictive-coded picture(P), bidirectionally predictive-coded picture(B), or of other coding types.

001 intra-coded(I)

010 predictive-coded(P)

011 bidirectionally- predictive-coded(B)

The coded video stream is carried in packetized elementary stream (PES) packets. As shown in Fig. 3.6.3, the PES packet consists of a PES packet header followed by packet data. PES packets are carried by Transport Packets. The Transport Packet is of a fixed length of 188 bytes. PES packets may be much larger than the size of a transport packet. The first byte of each PES packet header is located at the first available payload location of a Transport Packet.

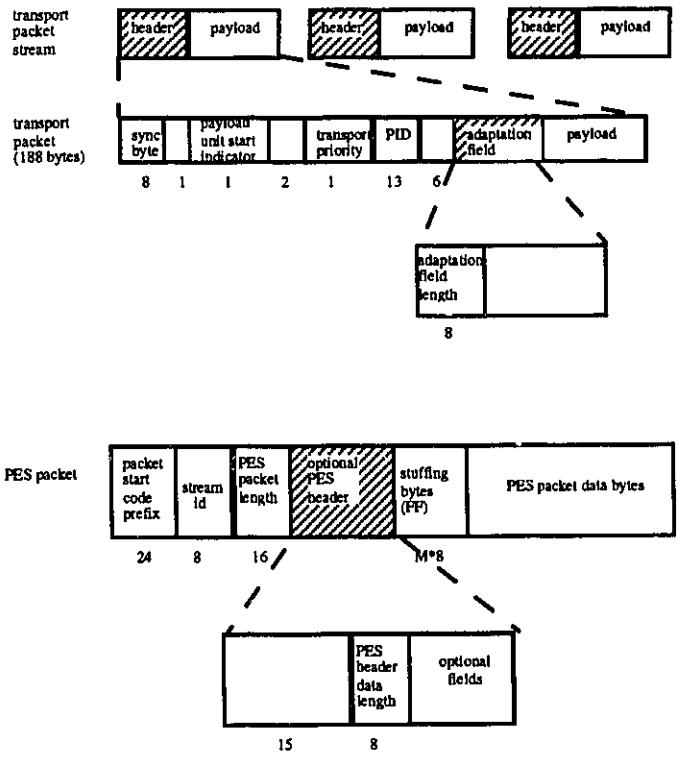

Figure 3.6.3 MPEG-2 transport stream syntax

A PES packet contains coded bytes from one and only one elementary stream. On decoding, de-multiplexing is required to reconstitute elementary streams from the multiplexed Transport Stream with the aid of Packet ID (*PID*) codes in the Transport Stream.

When the payload of the Transport Packet contains PES data, the *payload_unit_start_indicator* is set to '1', if the Transport packet commences with a PES packet header, or '0' if no PES header in this Transport Packet payload.

The *transport_priority* field in the Transport Packet header is a one bit indicator. When set to '1', it indicates that the associated packet is of greater priority than other packets having the same PID which do not have the bit set to '1'. In the MPEG-2 specification[MPEG-2], this field may be changed by data link specific encoders or decoders. In our priority-based synchronization control at the PSC, the *transport_priority* field of the Transport Packet which carried the data of the coded video stream, will be set to '1' or '0' through our priority extracting algorithm.

Referring to Fig.3.6.4, the PSC priority extracting algorithm is done by checking the Transport Packet header and the PES packet header to locate the data of the video stream, and scanning them for the *picture_start_code* so that the *picture_coding_type* of video picture can be found. Then the *transport_priority* field in the Transport Packet header is set to '1' if the *picture_coding_type* shows the video picture is an I-picture, or '0' whether it is a P-picture or a B-picture. If the *picture_start _code* is not found, the *transport_priority* will keep the same value as that of the last found priority which is stored as the *old_priority*.

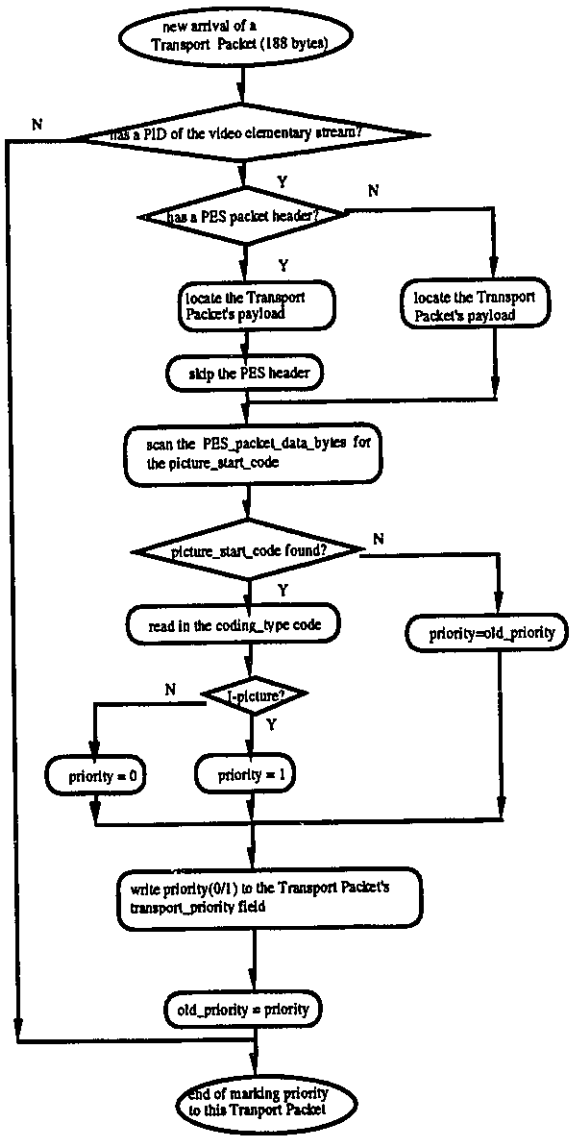

Figure 3.6.4 The priority extracting algorithm for the transport packets in a MPEG-2 video stream

The *PID* code in the Transport Packet header identifies the video stream. If the *PID* doesn't match the specified video stream *PID*, no priority extracting operation will be applied to the Transport Packet. The *payload_unit_start_indicator* shows if the PES header appears in the Transport Packet's payload before the video stream data. If there is no PES header in the payload, the whole payload is used by video stream data; otherwise, the video stream data appear in the *PES_packet_data_bytes* field after the PES header in the Transport Packet's payload. With the information in the PES header, such as the *PES_header_data_length*, it is easy to skip the PES header and locate the *PES_packet_data_bytes* field.

3.6.3 PSC Priority-based Stream Timing Control

Priorities obtained from the transport unit structures make the PSC synchronization control more efficient and provide better presentation satisfaction. The MPEG-2 PSC, for example, will manage its buffer overflow and starving based on the Transport Packet priorities.

Work of the MPEG video committee suggests[MPEG-1] that allotting P-pictures about 5 times as many bits as B-pictures and allotting I-pictures up to 3 times as many bits as P-pictures gives good results for typical natural scenes. For simplicity, here we assume the number of the Transport Packets carrying data of a B-picture is N, the number of the Transport Packets carrying data of a P-picture is $5N$, and the number of the Transport Packets carrying data of an I-picture is $15N$. There are 2 B-pictures between each pair of I or P pictures.

When the PSC buffer is full, the PSC scans its buffer from the end of the queue. A Transport Packet of the lowest priority is removed from the queue to prepare an empty space for the next Transport Packet coming from the network. Alternatively, the to-be-removed packet could be kept until the new packet which is of a higher priority is ready to come in.

Table 3.6.1 PSC buffer overflow control

The transport stream ordering: (In pictures)	**1I, 4P, 2B, 3B, 7I, 5B, 6B, 10P, 8B, 9B, ..**
The target presentation stream without loss: (In pictures)	**1I, 2B, 3B, 4P, 5B, 6B, 7I, 8B, 9B, 10P, ..**
Transport stream: (in Transport Packet)	1I1 1I2 ..1I15N, 4P1 ..4P5N, 2B1 ..2BN, 3B1 .. 3BN, 7I1 ~~7I2~~ ..7I15N, 5B1.. buffer full discarding
The presentation stream with loss:	**1I, 2B, 3B, 4P,** ~~4P~~, ~~4P~~, ~~4P~~, ~~4P~~, ~~4P~~, ~~4P~~, ..
The transport stream: (in Transport Packet)	1I1 1I2 ..1I15N, 4P1 ..4P5N, 2B1 ..2BN, 3B1 .. ~~3BN~~, 7I1 7I2 .. 7I15N, 5B1..
Control priorities:	(1) (1) ..(1), (0) ..(0), (0) .. (0), (0) .. (0), (1) (1) .. (1), (0).. discarding, buffer full
The presentation stream with priority-based loss control:	**1I, 2B,** ~~2B~~, **4P, 5B, 6B, 7I, 8B, 9B, 10P, ..**

In the example shown in Table 3.6.1, the PSC buffer is full when the second Transport Packet of an I-picture arrives. From the end of the queue, the PSC finds a priority 1 packet of the I-picture 7I which follows a priority 0 packet of the B-picture 3B. The priority 0 packet of 3B is removed from the buffer. A new arrival Transport Packet of priority 1 is then added in the buffer. Only one picture (3B) is damaged in the resulting display stream. The previous picture 2B is re-displayed to smooth the video presentation. If the PSC applies synchronization control without the knowledge of priorities, the in-coming Transport Packet of the I-picture 7I will be discarded due to buffer overflow. By losing the data of an I-picture(7I), the B-pictures (5B and 6B) and the P-picture (10P) which have references to 7I are also damaged. The damage will cause a remarkable pause in the resulting video display.

When the PSC buffer is empty, the PSC checks the arriving data unit to decide if it is beyond the delay tolerance. If it violates the tolerance, the unit is skipped; otherwise it is sent directly to the decoder. By skipping a late transport unit, the transport unit which follows the skipped one will be decoded right away so that the stream speed could catch up. Here are some

concerns of the tolerance values. If the delay tolerance is zero, every late unit will be dropped, but before the stream speed catches up there could be a significant pause in the video presentation. If the delay tolerance is set too large, too many late units may be accepted and occupy the decoder, thus the catching up speed will be slowed down. The worse situation is that, if all units are of the same control priority, some important data units are skipped along with less important units, thus when the stream speed finally catches up to the target presentation schedule it may not be properly decoded since data which provides reference did not enter the decoder. Therefore, we define a relatively larger delay tolerance for higher priority data units. That implies the speed catching-up is mainly achieved by skipping the less significant data for presentation. If a higher priority data unit is skipped, all its immediately following units of lower priorities will also be skipped without delay tolerance checking.

Table 3.6.2 PSC buffer starving control

The transport stream ordering: (In pictures)	**1I, 4P, 2B, 3B, 7I, 5B, 6B, 10P, 8B, 9B, ..**
The target presentation stream without loss: (In pictures)	**1I, 2B, 3B, 4P, 5B, 6B, 7I, 8B, 9B, 10P, ..**
Transport stream: (in Transport Packet)	1I1..1I15N, 4P1 ..4P5N, 2B1 ..2BN, 3B1 .. ~~3BN~~, ~~7I1~~ 7I2 ..7I15N, 5B1 5B2.. late, late discarding
The presentation stream with loss:	**1I, 2B,** 2B, **4P,** 4P, 4P, 4P, 4P, 4P, 4P, ..
The transport stream: (in Transport Packet)	1I1..1I15N, 4P1 ..4P5N, 2B1 ..2BN, 3B1 .. ~~3BN~~, 7I1 7I2 ..7I15N, ~~5B1~~ 5B2
Control priorities:	(1) .. (1), (0) .. (0), (0) .. (0), (0) .. (0), (1) (1) .. (1), (0) (0) late, late late .. late, late discarding discarding
The presentation stream with priority-based loss control:	**1I, 2B,** 2B, 4P, 4P, **6B, 7I, 8B, 9B, 10P, ..**

In the example shown in Table 3.6.2, the PSC notices that its buffer is starved when it waits for the *Nth* Transport Packet of a B-picture (3B). Then a packet of priority 0 and a packet of priority 1 arrive one after another, both late according to their time stamps. The PSC skips the priority 0 packet because the packet delay tolerance is violated, while accepts the priority 1 packet

of the I-picture (7I) because the packet delay tolerance is not yet violated. The following priority 1 packets meet the same situation. Then a packet of priority 0 arrives, being late and also violating its delay tolerance. The PSC again skips the packet. Finally, the presentation stream catches up its speed at the 2nd packet of the picture 5B. During the process of speeding-up, there is a 2 picture pause in video displaying. On the contrary, without this priority-based control, the Transport Packet of the I-picture may be skipped together with the Transport Packets of the B-picture. When the stream speed catches up at the 2nd packet of the picture 7I, the following P and B pictures could not be decoded without correct reference to the previous I-picture. The resulting video presentation will be badly damaged.

3.6.4 Communication between the Client PSC and MSC on a Coded Stream

The client MSC maintains the multimedia synchronization according to the presentation schedule through the SSP. The functions of the client MSCs are generic for all types of streams. The presentation schedule is characterized in terms of uncoded data units, such as the number of pictures, the duration of audio playing, etc., which is only meaningful to the MSC.

The client MSC may inform the PSC to adjust the initial PSC delay. Therefore, the PSC timing control can be set up with respect to the presentation schedule. According to the inter-stream synchronization requirements, a client MSC may have to delay the presentation of its next media object since an intentional delay has been added to scheduled presentation start time of that object. Due to the extremely high throughput of the decoded continuous data stream, the client MSC buffer may not be sufficient for buffering the delayed data. For instance, one second delay of a TV standard video sequence will accumulate 150 Mbits. Therefore the client MSC has to inform the PSC to buffer the delayed data in their compressed formats before decoding, by adding the intentional delay to the previous PSC delay. On the other hand, when the PSC skips or discards transport unit, it should notify the client MSC of the data loss.

3.7 Network Architecture for Remote Presentational Applications

3.7.1 Integrating to ATM-based Networks

ATM is a high-bandwidth, low-delay cell-transport technique[Bou92]. It is proposed to provide an integrated communication service in B-ISDN. Given the importance of quality and service guarantees in multimedia communication, ATM seems to be an adequate technology for future applications. For the last few years, ATM has been adopted as the basis for most standardization efforts of networking technology in the local, metropolitan and wide area environments. Through ATM user-network interfaces, ATM connections could be set up to support specified QoS requirements of the network users. The multimedia communication system for news document progressive presentation is based on an ATM network. The scheduling and stream control functions rely on the service provided by the underlined ATM virtual connection.

As we are considering a multimedia multi-stream transmission, the applications would require a large variety of QoS which is beyond the scope of the ATM QoS specification. A QoS selector may be introduced in order to map network users' QoS requirement to the ATM network services[Dub94].

3.7.2 Internetworking user premise network with B-ISDN

There are many proposals for high speed LAN/B-ISDN internetworking[Pit93][Sut93]. An ATM-LAN is considered as a good candidate for future high speed LAN design, as it can integrate today's LAN as well as ATM networking. In our remote multimedia presentational application, the most important issue relating to internetworking is how could an internetworking protocol support

end-to-end bounded QoS (such as delay and loss). The ST-2 standard[CIP92] is designed as an internetwork protocol to replace IP for high-speed networks. The ST-2 protocol supports end-to-end connection-oriented transmissions, based on the concept of *stream*. Users can specify FlowSpec to describe the required characteristics of a stream, including bandwidth, delay, and reliability parameters, with both desired values and minimal allowable values. We see these QoS features of ST-2 necessary (if not yet sufficient) for our application. The MSC and PSC are needed in case of ST-2 or ATM not being sufficient to bound their statistic QoS in given tolerances.

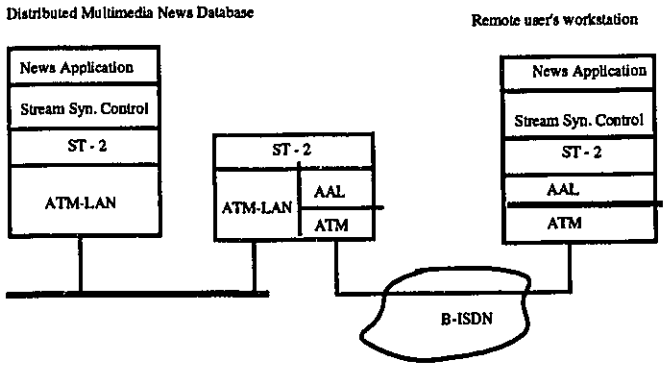

Figure 3.7 The protocol stack for stream synchronization control

Figure 3.7 shows the communication protocol stack in which our stream synchronization control system is sitting above the ST-2 protocol to provide multi-stream synchronization support to the multimedia presentational application. With respect to the focused multimedia applications, a multimedia transport service is assumed that could exist between the ST-2 and the stream synchronization control system. The QoS negotiator will be in effect before the connections are set up.

Chapter 4

Implementation

4.1 System Hardware and Software Components

In Fig. 4.1, the synchronization control system is shown in conjunction with the other software components in the *Multimedia News-on-demand* application.

The synchronization control software system consists of the processes of the scheduler, the client Media Synchronization Controllers (client MSCs), the Pre-decoder Synchronization Controller(PSC), and the server Media Synchronization Controllers (server MSCs). These processes are linked with control channels. Interfaces are required between the synchronization control system and the other sub-systems, such as the QoS negotiator (QoS Negot), the decoder, the database server, the media-storing servers (e.g. the Continuous Media file servers, the Non Continuous Media file servers, or the Archive storage servers). The Application Programming Interface (API) functions are required for system integration. Additionally, some control channels are needed among the geographically distributed software modules. The messages exchanged on the control channels should be specified.

The implementation will be done based on the recently developed hardware and software techniques. The system platform is built on IBM RS/6000 PowerPC workstations. Ultimedia

servers will provide powerful multimedia handling facilities. MPEG coding and decoding are also supported. Token ring LAN and an ATM backbone WAN environment are considered for the multimedia data transportation. ST-2 and the multimedia transport protocols will provide QoS negotiation and enforcement. The C++ programming language has been selected for system integration. The work described in this chapter is part of the prototype development for the synchronization control sub-system. The implementation has been done in C in a UNIX system.

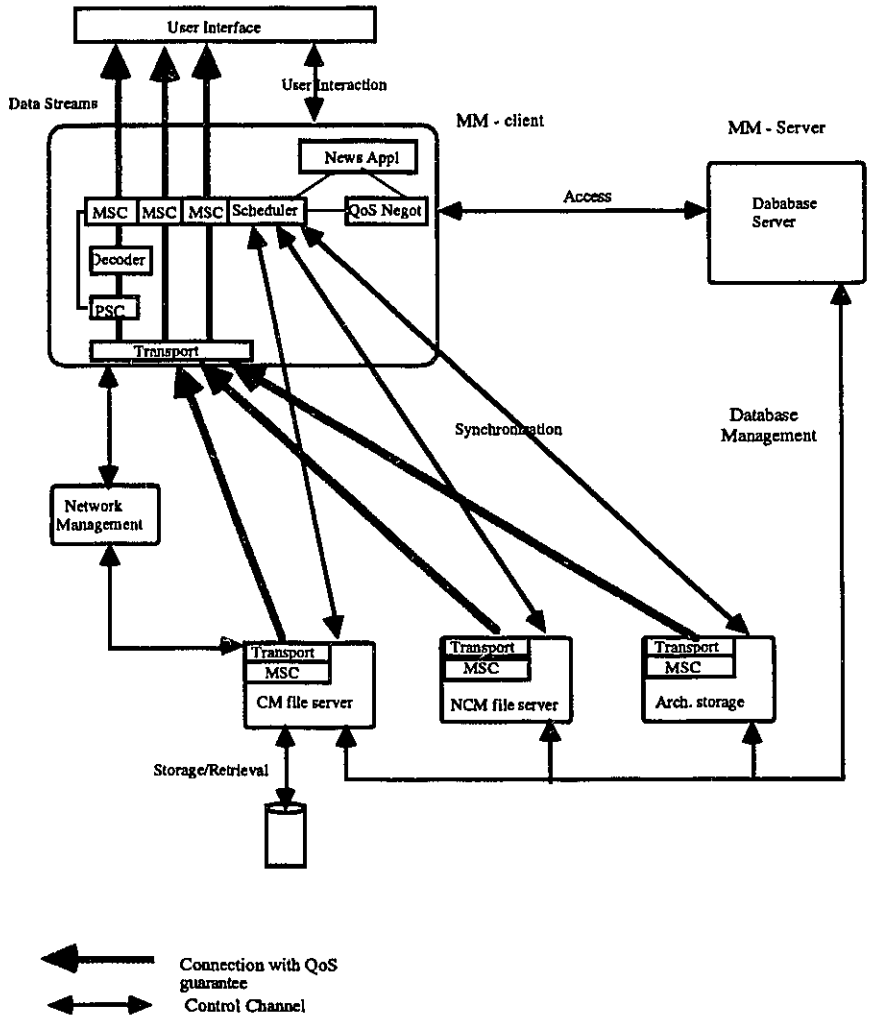

Figure 4.1 Software components in the *Multimedia News-on-demand* service

4.2 The data structures and algorithms

Overall, the basic data structures consist of the specifications for the scenario, the TFG, the shared presentation schedule table, and the delivery schedule. The basic algorithms include TFG generating, presentation scheduling, presentation re-scheduling and delivery scheduling. We will discuss these data structures and algorithms in the following sections. (Refer to Fig. 4.2.1)

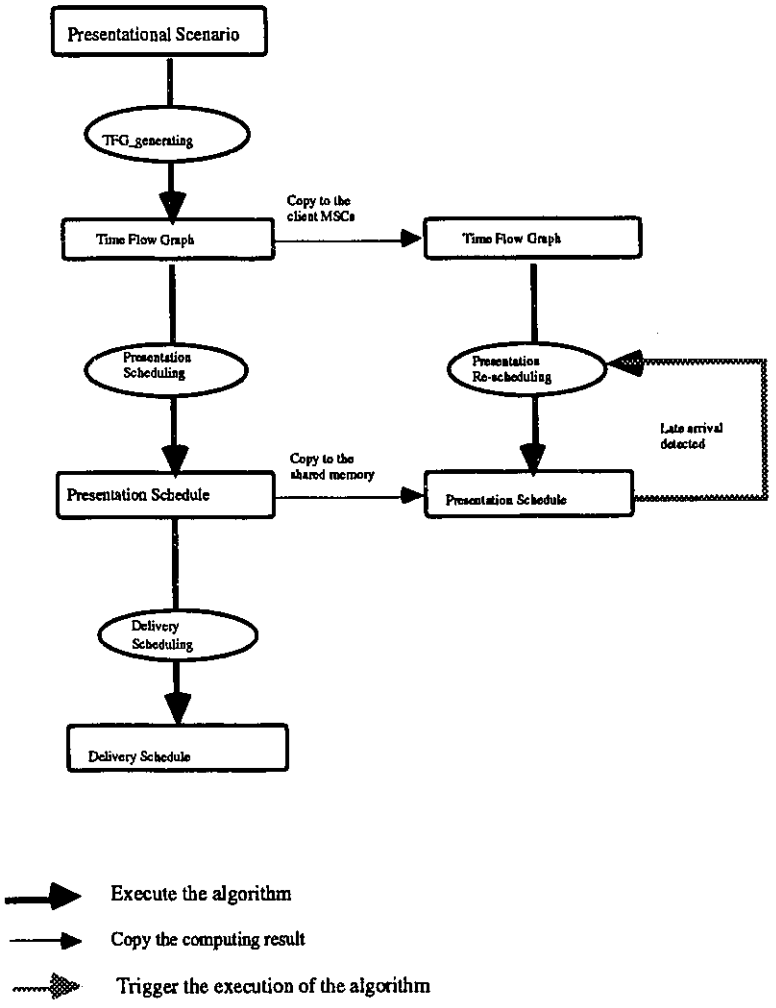

Figure 4.2.1 The relation of the data structures and the algorithms

4.2.1 Scenario Specifications

Requirements for a scenario specification

1) It describes rich temporal relationships among the multimedia objects, including the flexible intermissions, skew tolerances of the relationships, etc. If a scenario is satisfied, all the synchronization requirements of the application are met.

2) A scenario can be satisfied by at least one presentation time schedule.

3) The scenario is well structured and efficient in order to facilitate message exchanging in distributed database environment.

Scenario specification

The scenario specification proposed in [LiLi94.2-1] has the strengths of flexibility and efficiency. However, the skew tolerances are not included in the specification. Additionally, we simplified the scenario by taking advantage of the *a priori* knowledge of the duration values and the pre-defined activities.

Refer to the appendix for the scenario specifications in *BNF style* description. The scenario is a single thread of activities. Each activity is braced with "{ }". The activity is composed of a number of *objects* and the various *relations* among them.

The *object* is represented as a tuple of *(media_Id, object_Id, duration).* The *media_Id* identifies the media-server where the component is stored. The *object_Id* is unique for each component on the same media-server. The *duration* is the presentation duration. If the component is of a continuous media type, its *duration* is the playback duration. If the component is of a discrete media type, its *duration* is the playback duration plus the holding time.

The *relation* is represented by the *relation_Id*, the involved two *objects*, a possibly pre-defined *fixed intermission* and/or a *skew* tolerance. The *relation_Id* can be either "b", "m", "e", "o", "d", "s", or "f", which stands for the seven basic mutual relationships, "before", "meets", "equal", "overlaps", "during", "starts together", and "finishes together" respectively.

A Scenario Example

Scenario-1:

<start> { [(Audio, A, 2), (Audio, B, 3), (V/A, C, 1), (Text, D, 5), (Text, F, 1), (V/A, P, 1)],

[A(b)B(,), A(m)C(,), A(f)D(,κ2), D(b)F(,), C(m)P(,), P(s)F(,κ5)] } <end>

where:

1). There is only one activity, including the objects A, B, C, D, F, and P.

2). The audio object A has a duration of 2 time units, the audio object B has a duration of 3 time units, the video/audio object C has a duration of 1 time units, the text object D has 5, and so on.

3). The mutual relationships include: the object A plays back before B, with a flexible intermission between A and B; A finishes together with D, with a skew tolerance of κ2; A meets with C; D plays back before F, with a flexible intermission between D and F; C meets with P; and P starts together with F, with a skew κ5.

4.2.2 Time Flow Graph

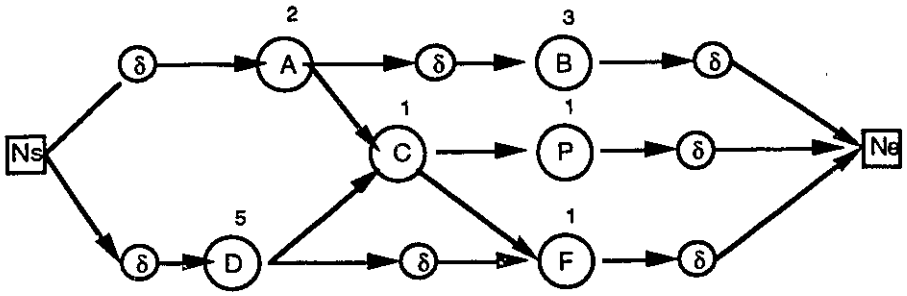

Figure 4.2.2 The TFG of Scenario-1

The mathematical model of interval vectors and the involved temporal information are maintained in a Time Flow Graph (TFG)[LiLi94.2-1]. In the synchronization control system under development a major simplification was made to the TFG. In the simplified TFG, the activities are always in sequential order without time overlapping. The rich temporal relationships among the

media components can still be represented by the relationships among the objects (nodes) within the activity. We assume that the aggregation of objects into an activity has been done off-line, i.e., at the time when a scenario is created. The real-time scheduling operation at the activity level was thus greatly reduced. The simplified TFG of a scenario becomes a directed link of activity sub-graphs. The end time of an activity, Te, always equals the start time of its following activity, Ts. There is no node shared by more than one activity.

A TFG Example:

Scenario-1: Refer to Fig. 4.2.2.

TFG Definitions:

```
TFG {
        number_of_activities,
        activity_pointer -> { activities}
                                /* the pointers to the activities which are list in a sequential order */
        };

Activity {
        number_of_objects,
        StartTime_activity,             /* scheduled start time for presentation,                  */
                                        /* time offset to the start time of the first activity     */
        components -> { nodes}          /* the pointers to a set of nodes                          */
                /* the first node of the components is always the common start node Ns,            */
                /* while the last one is the common end node Ne                                    */
        };

Node {
        object_Id,      /* a unique node ID in an activity */
        media_Id,       /* source MSC ID */
        duration,       /* presentation duration */
        jitter,         /* presentation jitter tolerance */
```

StartTime_node,

/* scheduled start time for the object presentation */

/* It is the time offset to the start time of the activity */

Ready-to-start, /* The address of the shared variable which is set to TRUE

after the enough data units arrive at the receiver */

Number-of-ancestor,

Ancestor(Number-of-ancestors) -> {pointer_to_node, δ},

/* δ is TRUE if there is a flexible intermission between the object

and its ancestor*/

Number-of- children,

Children (Number-of-children) -> {pointer_to_node,δ},

/* δ is TRUE if there is a flexible intermission between the

object and its children*/

Skew /* which is min[$\kappa(r_i)$], where r_i is any parallel

relationship the object involved in */

};

Property of simplified TFG:

Any node *N* in a TFG satisfy all the following rules:

Let N.Ts = N.StartTime_object, N.Te stands for the end time of the presentation.

Rule-1: N.Te = N.Ts + N.duration.

Rule-2: if N.ancestor(j).δ = FALSE,

N.Ts = N.ancestor(j).Ts + N.ancestor(j).duration.

Rule-3: if N.ancestor(j).δ = TRUE,

N.Ts >= N.ancestor(j).Ts + N.ancestor(j).duration.

From rule-2 and rule-3 we have:

Rule-4: N.Ts > = max[N.ancestor(j).Ts + N.ancestor(j).duration]

4.2.3 TFG Generating Algorithm

This algorithm generates a TFG specification from a given scenario. Refer to the appendix for the TFG_generating algorithm.

4.2.4 Presentation Scheduling Algorithm

Although various temporal relationships can be involved in a scenario, the major issue of scheduling is to determine the start time for each object which is related of other objects. Thus the start time which involves multiple objects should be captured from the TFG model. The presentation scheduling is to define the absolute time operation from the relative temporal relationships specified in a scenario.

A presentation schedule is composed of the start times of all nodes in the TFG of an scenario, where all the rules of TFG apply. A presentation schedule is in the tuples of *(media_Id, object_Id, StartTime_activity, StartTime_node).*

The fuzzy temporal requirements specified in a particular scenario or the corresponding TFG can be translated into more than one absolute presentation schedules. A presentation scheduling algorithm will produce one and only one presentation time schedule from a TFG model conditioned on the given duration parameters. The components in the distributed system will thus posses a common target presentation schedule which is locally derived from the scenario information through the TFG presentation schedule algorithm. Besides, the resulting presentation schedule is always the one which has the shortest total presentation duration among all other satisfactory schedules. This restriction is based on the observation that an ideal presentation does not have display gaps or frozen screens, and that the synchronization controller will recover the mismatch among the multiple streams by expanding the initial presentation schedule.

The presentation algorithm applies to two levels of TFG, i.e., the activity level and the object level. Since the simplification has been made that the activities are played back in sequential order, the main issue of scheduling become that of deriving a sub-schedule for the object nodes within each activity. In the following, we shall focus on the algorithm which applies to the sub-graph of nodes in an activity.

Algorithm Design

A schedule of start times satisfies the three rules in the TFG property at every node through every edge. It can be exhaustively verified through all existing paths from N_S to N_e. We combine the schedule generation and verification together in an exhaustive path search operation. While searching for a new path, all flexible intermission values are initialized to zeros, and the start time of each encountered node is set. A path search operation stops when it reaches a node which it has reached in previous paths searches. If the new value of the start time is not larger than its former value, then the old value is verified as true; Otherwise, update the start time to be the larger value, according to Rule-3, and propagate the changes to a minimum number of nodes which have to change their start time so that the rules of a TFG will still hold true. The change propagating operation is done by a recursively re-scheduling algorithm (i.e., the *recur_resch* algorithm, referring to the appendix). Through re-scheduling, some flexible time intermission values are updated to non-zero ones. As a result, the schedule will be computed and verified.

4.2.5 The Presentation Re-scheduling Algorithm

The presentation re-scheduling algorithm is the same as the presentation scheduling algorithm, except that a short-cut calculating is employed which takes advantage of the previously existing schedule. For example, if the changes of the schedule are isolated within an activity, in other words, if the end time of the activity keeps its former value, no re-scheduling will be performed on the following activities. In the activity to which the re-scheduling algorithm applies, changes are only made for a minimum number of objects; the other objects will stick to their former scheduled times (by taking advantage of the flexible time intermissions in the TFG). Therefore, the re-scheduling calculation is fast enough for real-time synchronization control.

4.3.6 Delivery Scheduling Algorithm

A Delivery schedule is the tuple of (*media_Id, object_Id, T_d*), where the *media_Id* and the *object_Id* are the same as those in the scenario. The T_d is the delivery start time of a media object, i.e., the time offset to the beginning of the document delivery.

The scheduler receives the document scenario from the database-server. It produces the presentation schedule in terms of *(media_Id, object_Id, StartTime_activity, StartTime_node)* using the Presentation Scheduling Algorithm. In the initial presentation schedule, the start time of the document presentation is set to zero. The T_{pi} is the time offset of the presentation of the *i*th document component, T_{pi} = StartTime_activity + StartTime_node. In order to satisfy the timing order specified in the presentation schedule, the scheduler derives a delivery schedule for the server MSCs to start their object component deliveries. Such a delivery schedule compensates the variable delays over separate stream connections.

Letting the earliest delivery time to be zero and the delivery time offset of the *i*th component to be T_{di}, then the target presentation time of the *i*th component is defined in equation [1]:

$$T_{ti} = T_{di} + D_{total\text{-}i}, \qquad [1]$$

where $D_{total\text{-}i}$ is the total delay of the *i*th media component starting from the delivery onto its stream connection until its presentation time. The equation [2] represents that the timing relationship among the target presentation times is the same as that in the presentation schedule:

$$(T_{ti} - T_{tk}) = (T_{pi} - T_{pk}) \qquad [2]$$

Refer to the appendix for the Delivery Scheduling Algorithm. The total delay is obtained through steps 1 and 2 in the Delivery-Scheduling Algorithm for each stream. Considering both the total delays and the presentational deadlines, the media object which is to be delivered at the earliest time can be found by step 3 in the algorithm. Step 4 records the earliest delivery time to be zero, and derives the corresponding target presentation time. Applying the equation [2] to step 5, all other streams' target presentation times are obtained. Consequently, all streams' delivery times are

derived through the equation [1] in step 6. For the stream which has more than one sequential object components, we can derive the delivery schedule for the subsequent object components by step 7.

4.3 The SSP implemented among the client MSCs

To implement the Stream Synchronization Protocol (SSP) among the concurrent client MSC processes, we have to solve the following problems:

1) Transfer data from the network stream connection to the client MSC's synchronization control buffer upon data arrival, and transfer the data from the buffer to the user interface upon scheduled presentational deadline;
2) Support dynamic buffer management;
3) Guarantee mutual exclusion on access to shared presentation schedule;
4) Allow the state variable of a client MSC process to be accessible to other client MSC processes.

4.3.1 Timing Control

The client MSC process receives data from the connection and periodically plays-back the data at the user interface. To solve the synchrony problem between the receiving and the playing back operations, a playback alarm clock process is introduced which sends alarming signals back to its parent process periodically according to the playing back time interval. At the start time of the continuous presentation of a media object, the client MSC forks a child process, the playback alarm process with a specified time interval value. Refer to Fig. 4.3.1. Upon catching an alarm signal (implemented by SIGUSR1 in the UNIX system) sent back from the playback alarm process, the client MSC interrupts its receiving operation and transfers a data unit from the synchronization

control buffer to the user interface for presentation. Then the client MSC continues its receiving operation until the next alarm signal arrives. The client MSC keeps track of the number of the data units being played back and the number of the data units being received. When the end of the object presentation is reached, the client MSC process kills the playback alarm process. If there is a subsequent object to be presented by the same client MSC, another playback alarm process will be created while the corresponding time interval value is being specified.

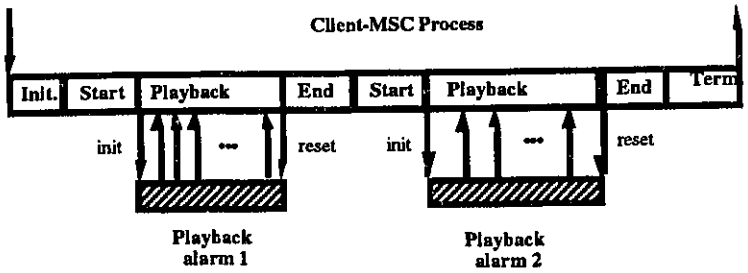

Figure 4.3.1 The interaction of a client MSC process and the playback alarm processes

An alternative process structure consists of a pair of producer and consumer processes for each client MSC. The producer process keeps on receiving data from the connection, while the consumer process periodically transfers the data from the buffer to the user interface. Because the client MSC's synchronization control buffer should be accessible to these two processes, a shared memory control mechanism is required. Considering the dynamic buffer management required by the SSP implementation, we were determined to avoid the unnecessary complexity of the buffer management introduced by using a shared memory control mechanism on the client MSCs' synchronization control buffers. Therefore, the widely used producer/consumer process structure was rejected in our system implementation. Refer to Fig. 4.3.2 for the process structure implemented.

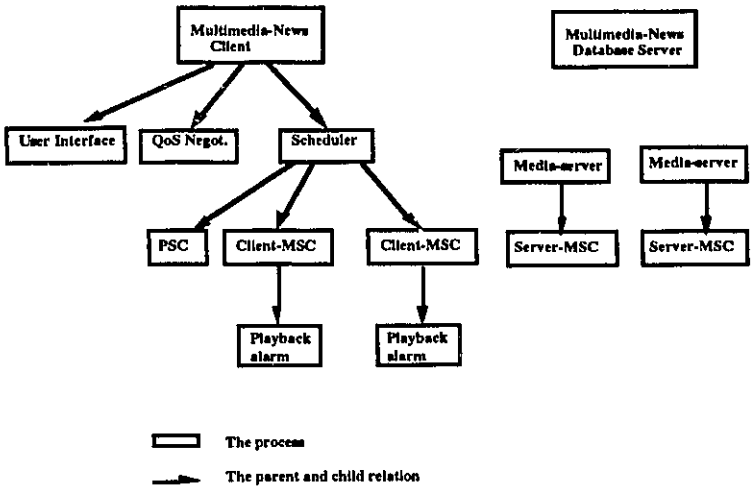

Figure 4.3.2 The process structure

4.3.2 Buffer Structure

The data arriving before the scheduled presentation deadline are stored in the client MSC's synchronization control buffer. Buffer space must be provided to absorb a certain amount of network delay variation. Based on the statistic delay variation value which is guaranteed by the underlined network stream connection, the scheduler calculates the buffering delay, i.e. $D_{buffering}$, which is brought by the data buffering at the client MSC in order to smooth out the variable network delay. The initial buffer size is determined by the data unit size and the buffering delay.

The dynamic buffer management issue is raised by the re-scheduling operation. If a presentation schedule is extended to a later time point, additional volume of data may accumulate at the buffer during the time difference between the former presentation schedule and the updated one. Additional buffer space should be allocated to prevent buffer overflows. In the meanwhile, the operations of reading from and writing to the buffer should remain generic in spite of the changes in buffer allocations. We chose the buffer structure to be a linked list, which may consist of a variable number of buffer units. Each buffer unit is connected to a subsequent buffer unit by a pointer. The last buffer unit links back to the first unit, and a circled linked list of buffer units is

thus formed. The buffer units do not have to be allocated with a continuous physical memory address. They can be dynamically added to or removed from the linked list.

Each buffer unit has the same size as that of a data unit which is specified by the presentation requirement for the object, such as a frame of a decoded video or a page of text. A client MSC initializes a linked list structured buffer for each of its sequential object. The MSC can play back the data of one object while receiving data of another object. A write-pointer records the address of the data unit in the linked list in which new arrived data will be put by the receiving function, while a read-pointer records the unit address which contains the data being lastly displayed. The two pointers proceed in the same direction around the circled linked list. The two pointers are not allowed to overrun each other. When the write-pointer catches up to the read-pointer, the buffer overflows. In case that buffer overflows, either more buffer allocation is required or some data are discarded. When the read-pointer is going to catch up to the write-pointer, the buffer starves. Then the read-pointer stops at the previous address without proceeding to the next buffer unit, and the client MSC is allowed to playback the last data unit.

4.3.3 Exclusive Access to the Shared Presentation Schedule

The presentation schedule is initially produced by the scheduler and is stored in the shared memory. The shared memory is accessible to concurrent client MSCs (Fig. 4.3.3). The shared presentation schedule regulates the playback procedures of the different client MSC processes, so that the presentation could remain in synchrony. If a re-scheduling operation is completed by one of the client MSC, an updated presentation schedule will overwrite the previous one in the shared memory. The shared memory mechanism eliminates the need for complex message exchanges among the client MSCs in supporting the SSP control.

In UNIX, a shared memory space can be allocated and the exclusive memory access is achieved by a binary semaphore. Only when the semaphore value equals to 1, can a process access the shared memory. The semaphore value decreases by 1 and will remain 0 until the

process releases the shared memory. In our software implementation, the scheduler allocates a shared memory space from the system and sets up a semaphore for exclusive memory access control. When it is created, each client MSC process inherits a copy of ID of the shared memory and the ID of the related semaphore. Whenever a client MSC receives the first data unit of its first media object, it accesses the shared memory after obtaining the semaphore value. If the presentation schedule is still composed of the relative time offsets initialized by the scheduler, the client MSC updates the schedule by converting the time offset 0 to the current value of the system time clock, and converts other time offsets accordingly. Then the client MSC releases the shared memory and increases the semaphore value by 1.

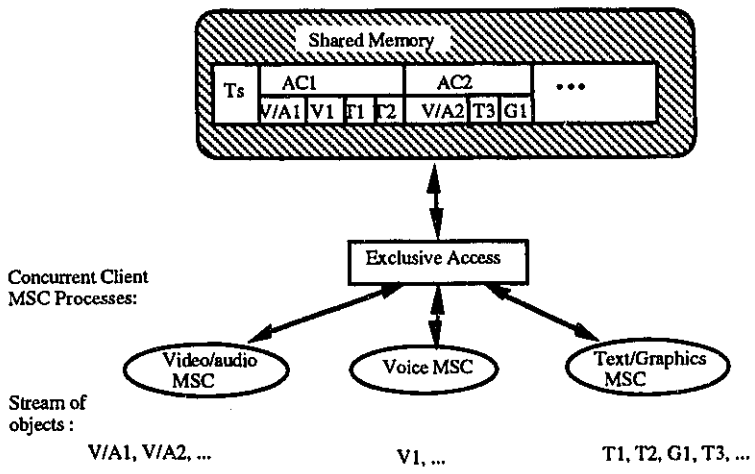

Figure 4.3.3 Client MSCs' exclusive assess to the share presentation schedule

The concurrent client MSCs also access the shared memory to check the current system clock with the presentation schedule. After performing a re-scheduling algorithm, a client MSC produces a new presentation schedule and writes it to the shared memory so that other client MSCs are able to synchronize their presentation operation accordingly.

After the whole document presentation is completed, the scheduler removes the shared memory and the related semaphore before the scheduler exits.

4.3.4 State Variables

Each object has a binary state variable, "ready-to-start". The "ready-to-start" variable of each object is reset to zero before any data unit is received. When the first data unit of an object is received and is ready to be sent to the presentation device, the client MSC sets the state variable of this media object to 1. After the object has been played back, the value is reset to zero again.

The start time of a late object is not predictable. Thus if an object X is supposed to meet with the subsequent object Y, object X may have to extend its presentation until the object Y is "ready-to-start".

If the skew tolerance between the parallel objects is specified, each of the involved objects should check the "ready-to-start" states of the other. If the object X is supposed to start together with the object Y and the first data unit of X arrives later than that of Y, the client MSC in charge of the object Y sets its state "ready-to-start", and it puts the current clock value to the shared schedule. It then proceeds to start presentation. Then the client MSC of X will find the state of Y is "ready-to-start" and it will have to make sure the time difference between the current system clock and the start time read from the shared schedule doesn't exceed the skew tolerance.

The "ready-to-start" state variable of one client MSC can be read by other client MSCs. This shared variable is implemented as a semaphore. The address of the "ready-to-start" semaphore of an object is specified in the corresponding object node in the TFG. The scheduler initializes the semaphore array for the "ready-to-start" variables and store their addresses in the TFG. The client MSC gets a copy of the TFG when the client MSC process is created. Through the exclusive access control of the semaphore, the client MSC is able to update its own "ready-to-start" variable and to check other client MSC's current state without conflicting.

After the presentation is completed, the scheduler removes the "ready-to-start" semaphore array before the exiting by itself.

4.4 User Interactions[Lam94.5.2]

In a *Multimedia News-on-demand* service, the user may choose a multimedia document from the database for viewing, then start and close the presentation. The user may also request to pause the presentation to view a certain portion, to skip a portion of the document, or to scan the presentation backward and forward. The scheduler and the client and server MSCs are involved in dealing with these user interactions, which are related to the document presentation. The scheduler performs the delivery scheduling only once for each document, at the time when the user selects the document for viewing. The scheduler serves as a master who dispatches relevant commands to the MSCs.

4.4.1 Selecting a Document for Viewing

This is the phase in which the scheduler invokes the client and server MSCs, and initiates the multimedia presentation. The synchronization error prevention described in section 3.4 is activated. Data streams are set up with negotiated QoS parameters. A delivery schedule is produced and sent to the server MSCs. Messages are exchanged between the scheduler and the database server, the scheduler and each involved media-server, and the scheduler and each server MSC. API calls are needed to retrieve the document scenario from the database server, to get QoS parameters for the negotiation, to create a MSC process on the client and server machines, to open up virtual connections, and to send and receive messages form the connections. The *query_decoding_time* API provides the scheduler with the average decoding delay to schedule delivery.

4.4.2 Starting the Presentation

When the user wishes to start viewing the article, the scheduler multicasts a *start_request* message to the server MSCs. The server MSCs return the *start_confirm* massage and start executing the delivery schedule. The *start_delivery* API function is called by the server MSC every

time an object has to be retrieved by a media-server. Each server MSC sends its data object to a peer client MSC according to the delivery time schedule. The client MSCs execute the SSP to control the presentation and use the API calls to display the data objects on the user interface.

4.4.3 Pausing the Presentation

When a user wishes to pause the presentation of the article, the scheduler multicasts a *pause_request* message to both the client and server MSCs. When the client MSC processes receive the message, they start buffering the data received on their streams. Upon reception of a *pause_request* message, the server MSCs return a *pause_confirm* massage to the scheduler and stop executing the delivery schedule. The server MSCs call the *stop_delivery* API function which will have the media-servers stop retrieving their respective data objects.

4.4.4 Scanning the Presentation Forward or Backward

When a user selects to scan the presentation backward or forward, the speed of the motion must be selected. The scheduler process multicasts a *scan_request* message and the server MSCs return a *scan_confirm* message to the scheduler. The video server MSC calls the *scan* API function. The video media-server uses the speed to determine how many frames to skip before transmitting a frame. The other server MSCs simply call the *stop_delivery* API function which will make the media-servers stop transmitting the data until the server MSCs are notified of the next start point in the delivery time schedule. Meanwhile when the client MSCs receive the *scan_request* message, they stop applying the SSP until they are advised to proceed to the display from a new start point in the presentation time schedule. Only the client MSC associated with the video stream displays the frames that it receives while other client MSCs discard the received data.

4.4.5 Starting the Presentation after Pausing

When a user wishes to start the presentation of an article after pausing, the scheduler multicasts a *start_request* message to both the client and server MSCs. The message contains the

status of the presentation (e.g., pausing). When the client MSC processes receive the message, they start displaying the buffered data and accepting data received on their stream connection. Upon reception of the *start_request* message, the server MSCs return the *start_confirm* message to the scheduler and call the *start_delivery* API function which will have the media-servers resume the retrieval.

4.4.6 Starting the Presentation after Scanning Backward or Forward

When a user wishes to start the presentation of an article after scanning backward or forward, the scheduler multicasts a *start_request* message to both the client and server MSCs. The message contains the status of the presentation (e.g., scanning). Upon reception of the *start_request* message, the server MSCs return the *start_confirm* message to the scheduler and call the *seek* API function which will inform the media-servers of the new start point in the delivery schedule. The media-servers will derive the number of units that should be scanned. When the client MSC processes receive the *start_request* message, they resume executing the SSP

4.4.7 Closing the Document Presentation

When a user wishes to close the presentation of the document, the scheduler multicasts a *close_request* message to the server MSCs. Upon reception of the message, the server MSCs disconnect their respective stream to their peer client MSC. Once a stream has been disconnected, the media-server removes its server MSC process by calling the *kill_server_msc* API and returns a *close_confirm* message to the scheduler. When the scheduler receives the *close_confirm* message it removes the client MSCs.

4.5 Prototype Implementations

For the first prototype of the real-time synchronization control system for the Multimedia News application, we make several simplifications to the general problem of the network document presentation.

1) Documents are delivered from a single server workstation to a single client workstation. The database server and the media servers all reside in the same server workstation. They are implemented as concurrent independent processes which are running all the time in the server workstation. Multiple virtual connections are set up between the servers and the client workstation over a single physical network link. In the future system, the database server and the media servers will be geographically distributed in the computer network.

2) The database server and media servers are simulated by the servers which retrieve the requested objects from the corresponding files on the magnetic disk and deliver the file context over virtual connections.

3) The network virtual connections are over a dedicated token ring link. The virtual connections over an ATM backbone wide area network are going to be implemented as the project proceeds and as the relevant hardware and software components becomes available.

4) The client workstation is dedicated to the task of the Multimedia News presentation. In a real system environment, a workstation must perform many tasks simultaneously.

Note that these simplifications are made to eliminate much of the resource contention for the network, operating system and some workstation resources, such as buffer space. We expected the implementation of the multimedia synchronization schemes to be difficult even under these conditions. We determined to make our first step manageable. With the technical progress made together by the research groups of the CITR Multimedia News application, we are going to integrate the database server, the continuous media servers, the multimedia transport system over ATM networks and our real-time multi-stream synchronization system into a sophisticated

Multimedia News network presentation system. By that time, the conditions on our first prototype system will be removed.

In addition, because we were determined to demonstrate the outcome of the synchronization control system, we are in the process of developing a user interface which is capable of presenting MPEG-1 video/audio and text captions simultaneously.

The specific cases of the various multimedia document presentation scenarios that are investigated for the purpose of system testing and demonstration are:

1) MPEG video/audio show along with text captions in another language.

The lower part of the video window is overlapped by a text window. The captions context is refreshed and updated from time to time corresponding to the video/audio context. The scenario is shown in Fig. 4.5.a.

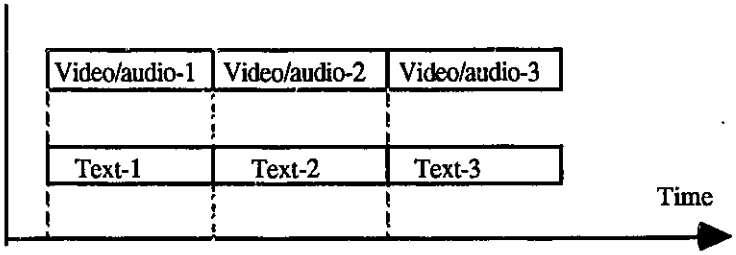

Figure 4.5.a The parallel presentational scenario for demonstration

2) An MPEG-1 video/audio clip such as a part of the latest video record of a football game followed by a page of text such as the current season scoreboard. The video window is immediately replaced by the text window. The scenario is shown in Fig. 4.5.b.

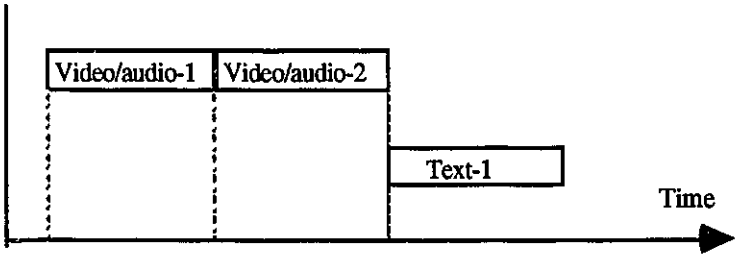

Figure 4.5.b The sequential presentational scenario for demonstration

These two scenarios demonstrate the basic multi-stream synchronization requirements in the multimedia document presentation, i.e., the parallel inter-stream synchronization and the sequential inter-stream synchronization.

We are now in the early stages of the first experimental prototype implementation. Module coding and testing have been carried out for the supporting functions which were discussed in the previous sections of this chapter. Application Programming Interface (API) calls are provided for the algorithm executions, the SSP controls, and the network transportation. We are building up the synchronization control system by using these supporting functions. An experimental system will integrate the synchronization control sub-system with the user interface, the simulated database server and media servers.

Chapter 5

Conclusions

5.1 Summary of the Thesis

The thesis presents a complete software control system for synchronizing multiple data streams generated from distributed media-storing database servers. The different media are transmitted through an ATM network for presentation on the client workstation. A scheduler at the client workstation derives a stream delivery schedule to compensate variable network delays. Besides the traffic prediction and scheduling of the object delivery, synchronization recovery is performed at the receiver, the client's workstation, before the playback of the multiple data streams. A stream synchronization protocol (SSP) prevents the synchronization errors form being rampant in the display, even during network delay fluctuations.

The SSP uses skew tolerance parameters to guarantee the control for the different types of parallel data streams. In addition, the differences between the characteristic of coded and uncoded streams are taken into account. We extend the synchronization mechanism to support the control over the coded data streams.

The contributions of this thesis work are as follows.

1) Applying the Time Flow Graph Model to the Synchronization Control System

The temporal models are compared in terms of their capabilities in supporting synchronization scheduling. The TFG model is able to represent the temporal relationships among the media objects without precise interval values. The TFG can also represent the flexible intermission intervals which is a favorable feature for improving the efficiency of the synchronization recovery. On the other hand, it can accommodate modification of the interval values while keeping the relative temporal relationships in consistency. Therefore, the TFG is found very useful for synchronization recovery control of the SSP. Additionally, in future research work, the TFG model will be required for a fully distributed scheduling algorithm to perform[LiLi94.2-2]. Thus the TFG is chosen for the current system development. It becomes a fundamental mathematical model which is applied to the complete software synchronization control system.

2) Developing a Complete Real-time Synchronization Control System

In our real-time synchronization control system, the synchronization requirements specification (i.e., the scenario) is extended to include the Synchronization QoS parameters [Ste93]. The TFG was chosen as a generic temporal model. The two aspects of a real-time synchronization control, i.e., the traffic prediction and delivery scheduling, and the synchronization error recovery are both investigated. The QoS is assumed to be provided by the ATM-based virtual connections. No feedback message is needed for the real-time recovery. Presentation re-scheduling to accommodate the mismatches among the multiple data streams at the client's workstation is conducted through the Stream Synchronization Protocol (SSP). Moreover, we also consider the integration problem if the synchronization stream control system joins with the database servers, coding/decoding algorithms, and user interactions. The relevant API and message exchanges are also considered in the implementation.

3) Synchronization Control over Coded Streams

In the system under consideration, the data objects are of different types of media and in different coding formats. Only video and its associated audio are in MPEG-2 format controlled by an MPEG video/audio. Synchronization between an MPEG stream and other streams is investigated in this thesis. Some overlapping between the functions of our MPEG stream controller and the proposed MPEG-2 decoder is suggested. Our MPEG stream control, as one part of the overall synchronization control facilities, supports external stream synchronization of the MPEG stream with other streams. Moreover, by introducing a Pre-decoder Synchronization Controller between the network interface and the MPEG-2 decoder, the decoding quality is enhanced by avoiding decoder internal buffer's starving and overflow which are caused by the network delay fluctuation conditions. A priority-based synchronization control mechanism for the coded data streams is provided. The investigation of the MPEG-2 video stream provides insights on coded stream behavior. The control scheme elaborated in this thesis can be adjusted to apply to other formats of coded streams. Our work contributes to the generic multimedia synchronization control for both coded and uncoded data streams in a real-time distributed environment.

4) Implementation

As part of the prototype development, implementation has been done in the following three aspects.

. the data structure specification and scheduling algorithm design

. the SSP control mechanism implementation and testing

. the programming flow chart is also provided for the scheduler and the client MSCs

5.2 Suggestions for Future Work

The synchronization system described in this thesis is based on several assumptions:

1) QoS supported by ATM-based virtual connections. In the real world, the ATM network is currently being constructed and tested. QoS negotiation support is not fully available.

2) We rely on the ST-2 networking protocol for our internetworking connections passing an ATM backbone network. Yet the ST-2, as an experimental protocol, is far from being mature, it needs to support a variety of existing LAN environments, and its enforcement of QoS guarantees are not yet fully implemented.

3) A multimedia real-time transport protocol is supposed to be available, which provides the MSC with transport units and reports the possible data loss or error to the upper layer, such as the MSCs or the PSC.

All the above network protocol support problems need in-depth investigation. A QoS negotiator is also supposed in the system design which will deal with the network QoS negotiation.

Moreover, the synchronization prototype implementation requires the system integrating with the decoder, the user interface and the simulated database server machines. The experiment over an ATM-based network should be carried out. The impacts of the real-world network fluctuation on the compressed data streams is an important subject. The experiment will provide valuable experience in choosing synchronization QoS parameters, understanding the impacts of applying rigid or loose control rules in synchronization recovery.

Bibliography

[And91] D. Anderson and G.Homsy, "A Continuous Media I/O Server and its Synchronization Mechanism", IEEE Computer, Oct., 1991.

[Arm92] H. Armbruster and K. Wimmer, "Broadband Multimedia Applications Using ATM Networks: High-Performance Computing, High-capacity Storage, and High-Speed Communication", IEEE Journal on Selected Areas in Communications, Vol. 10, No. 9, Dec., 1992.

[Bla91] G. Blair, D. Hutchinson and D. Shepherd, "Multi-Media Systems", Tutorial 4, 3rd IFIP Conf. on High Speed Networking, Berlin, Mar., 1991.

[Bou92] J. Le Boudec, "The Asynchronous Transfer Mode: a Tutorial", Computer Networks and ISDN Systems 24 (1992), pp.279-309.

[CIT88] CCITT Recommendation I.350, "General Aspects of Quality of Service and Network Performance in Digital Networks, including ISDN", Blue Book, 1988.

[CIP92] CIP working group, "Experimental Internet Stream Protocol", Version 2 (ST-II), Oct. 1992.

[Dra94] Draft Research Program 1994-95, Broadband Services, Feb. 11, 1994.

[Dub94] D. Dubois, N. D. Georganas and E. Horlait, "A QOS Selector for Multimedia Applications on ATM Networks", Proc. IEEE ICC'94, New Orleans, May 1994.

[Dup92] S. Dupuy, W. Tawbi and E. Horlait, "Protocols for High-speed Multimedia Communications Networks", Computer Communications, Vol.15, No.6, July/Aug., 1992.

[Esc93] J.Escobar, D. Deutsch, and C. Partridge, "Flow Synchronization Protocol", Proc. IEEE ICC'92, Geneva, Switzerland, May 1993.

[Fer92] D. Ferrari, A. Gupta, M. Moran and B. Wolfinger, "A Continuous Media Communication Service and its Implementation", Proc. IEEE GlobeCom'92.

[Fer92.7] D. Ferrari, "Delay Jitter Control Scheme for Packet-switching Internetworks", Computer Communications, Vol. 15, No. 6, July / Aug. 1992, pp.367-373.

[For93] M. Fortier, "A Store-and-forward Architecture for Video-on-Demand Service", Proc. ICCC Multimedia Communications'93 Conf., Banff, April 1993.

[Fun94] C. Fung and M. Pong, "MOCS: An Object-Oriented Programming Model for Multimedia Object Communication and Synchronization", Proc. IEEE Inter.Conf. on Multimedia Computing and Systems, Boston, 1994.

[Fur94] B. Furht, "Multimedia Systems: An Overview", IEEE Multimedia mag., Vol. 1, No. 1. 1994.

[Ham] C. L. Hamblin, "Instants and Intervals," in Proc. 1st Conf. Int. Soc. for the Study of Time, J. T. Fraser et al., Eds. New York: Springer-Verlag.

[Her92] R. G. Herrtwich, "The HeiProjects: Support for Distributed Multimedia Applications", IBM European Networking Center Technical Report No. 43.9206, 1992.

[Hoe92] P. Hoepner, "Synchronizing the Presentation of Multimedia Objects", Computer Communications, Vol.15, No.9, 1992 .

[IBM93] IBM Corporation, "Video Clock Recovering and ATM Environment", ATM_Forum/93-977, Nov., 1, 1993.

[JPEG] JPEG, "Information technology - Coding of Joint Pictures and Associated Audio for Digital Storage Media up to about 1,5 Mbit/s", Draft International Standard ISO/IEC DIS 11172.

[Kar90] A.Karmouch, L.O.Barbosa, N.D. Georganas, and M.Goldberg, "A Multimedia Medical Communications System", IEEE Journal of Selected Areas on Communications, Vol. 8, No. 3, Apr. 1990, pp.325-339.

[Lam94.5] L. Lamont and N. D. Georganas, "Synchronization Architecture and Protocols for a Multimedia News Service Application", Proc. IEEE Inter. Conf. on Multimedia Computing and Systems, Boston, U.S., May 1994.

[Lam94.5] L. Lamont, "Component Interactions and Messaging System for Multimedia Synchronization", Report, MCRLab, University of Ottawa, May 2, 1994.

[LiLi92.4] Li Li, A.Karmouch, and N.D.Georganas, "Real-time Synchronization Control in Multimedia Distributed Systems", Proc. IEEE Multimedia'92, Monterey, U.S.A. Apr., 1992 .

[LiLi92.6] Li Li, A.Karmouch and N.D. Georganas, "Synchronization in Real Time Multimedia Data Delivery", Proc. IEEE ICC'92, Chicago, U.S.A, June 1992. pp.322-1.

[LiLi92.7] Li Li, A. Karmouch and N. D. Georganas, "Real-Time Synchronization Control in Multimedia Distributed Systems", ACM Computer Communication Review, July,1992.

[LiLi93.4] Li Li, A. Karmouch and N. D. Georganas, "Performance Modelling of Distributed Data Integration in Real-time Multimedia Systems", Proc. ICCC Multimedia Communications'93, Banff, Canada, April, 1993.

[LiLi93.6] Li Li, L. Lamont, A. Karmouch and N. D. Georganas, "A Distributed Synchronization Control Scheme in a Group-oriented Multimedia Conferencing System", Proc. 2nd International Conference on Broadband Islands, Athens, Greece, June, 1993.

[LiLi93] Li Li, A. Karmouch and N.D. Georganas, "Real-time Synchronization Control in Multimedia Distributed Systems", ACM Computer Communication Review, Vol. 22, No. 3, 1993, pp. 79-86.

[LiLi94.2-1] Li Li, A. Karmouch, and N. D. Georganas, "Multimedia Teleorchestra with Independent Sources: Part 1 - Temporal Modeling of Collaborative Multimedia Scenarios", ACM Journal of Multimedia Systems, Vol. 1, No. 4, Feb. 1994.

[LiLi94.2-2] Li Li, A. Karmouch, and N. D. Georganas, "Multimedia Teleorchestra with Independent Sources: Part 2 -synchronization algorithms", ACM Journal of Multimedia Systems, Vol. 1, No. 4, Feb. 1994.

[LiLi94.5] Li Li, A. Karmouch, and N. D. Georganas, "Multimedia Segment Delivery Scheme and its Performance for Real-time Synchronization Control", Proc. IEEE ICC'94, New Orleans, May, 1994.

[Lit90.4] T. D. C. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects", IEEE Journal on Selected Areas in Communications, Vol. 8, No. 3, Apr. 1990.

[Lit90.5] P. B. Berrs, C. Y. R. Chen, A. Ghafoor, C. C. Lin, T. D. C. Little and D. Shin, "Architecture for Distributed Multimedia Database Systems", Computer Communications, Vol. 13, No. 4, May. 1990.

[Lit90.11] T. D. C. Little and A. Ghafoor, "Network Considerations for Distributed Multimedia Object Composition and Communication", IEEE Network Mag., Nov.1990, pp.32-49.

[Lit91.10] T. D. C. Little and A. Ghafoor, "Spatio-Temporal Composition of Distributed Multimedia Objects for Value-Added Networks", IEEE Computer Mag., Oct. 1991, pp.42-50.

[Lit91.12] T. D. C. Little and A. Ghafoor, "Multimedia Synchronization Protocols for Broadband Integrated Services", IEEE Journal on Selected Areas in Communications, Vol. 9, No. 9, Dec. 1991, pp.1368-1382.

[Loe92] S. Loeb, "Delivering Interactive Multimedia Documents over Networks", IEEE Communications Mag., May 1992.

[Mil93] G. Miller, G. Baber and M. Gilliland, "News-on-Demand for Multimedia Networks", Proc. ACM Multimedia'93 Conf., San Diego, Aug., 1993.

[MPEG-1] MPEG, " Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5 Mbit/s", DRAFT International Standard ISO/IEC DIS 11172, 1992.

[MPEG-2] ISO/IEC JTC1/SC29/WG11, N0601, "Coding of Moving Pictures and Associated Audio", MPEG93, Nov., 1993.

[Nic90] C. Nicolaou, "An Architecture for Real-Time Multimedia Communication Systems", IEEE Journal on Selected Areas in Communications, Vol. 8, No. 3, Apr. 1990, pp.391-400.

[Oro92] L.Orozco-Barbosa, A.Karmouch, N.D.Georganas and M.Goldberg, "A Multimedia Inter-hospital Communications System for Medical Consultations", IEEE Journal on Selected Areas in Communications, Vol. 10, No. 7, Sept. 1992, pp.1145-1157.

[Peh91] B. Pehrson and S. Pink, "Multimedia and High Speed Networking in MultiG", Computer Networks and ISDN Systems 21(1991) 315-319.

[Pit93] D. A Pitt and J. A Ruela, "The Broadband LAN as a Metropolitan Area Network", Computer Communications, Feb., 1993

[Ram93] S. Ramanathan and P. V. Ragan, "Adaptive Feedback Techniques for Synchronization Multimedia Retrieval over Integrated Networks", IEEE/ACM Trans. on Networking, Vol.1, No.2, Apr. 1993, pp.246-260.

[Ran91] P. V. Rangan and D. C. Swinehart, "Software Architecture for Integration of Video Services in the Etherphone System", IEEE Journal on Selected Areas in Communications, Vol. 9, No. 9, Dec. 1991.

[Ran92] P. V. Rangan, H. M. Vin, and S. Ramanathan, "Designing An On-Demand Multimedia Service", IEEE Communications Magazine, July 1992, pp.56-64.

[Rav93] K. Ravindran and V. Bansal, "Delay Compensation Protocols for Synchronization of Multimedia Data Streams", IEEE Trans. on Knowledge and Data Engineering, Vol. 5, No. 4., Aug. 1993.

[Rob91] John Robinson, E. Rubinov, and et al., "A Multimedia Interactive Conferencing Application for Personal Workstations", IEEE Trans. on Communications, Vol. 39, No. 11, Nov. 1991.

[Sad90] H. Sadamoto and S. Matsushita, "Synchronization Control between Text and Voice for Multimedia Message Communication", Proc. ICCC 90 International Conf. on Computer Communication, 1990.

[Shep90] D. Shepherd and M. Salmony, " Extending OSI to Support Synchronization Required by Multimedia Applications", Computer Communications, Vol. 13, No. 7, 1990, pp. 399-406.

[Shep92] D. Shepherd, D. Hutchinson, F. Garcia and G. Coulson, "Protocol Support for Distributed Multimedia Applications", Computer Communications, Vol. 15, No. 6, July/ Aug. 1992, pp.359-366 .

[Ste90.4] R. Steinmetz, "Synchronization Properties In Multimedia System", IEEE Journal on Selected Areas in Communications, Vol. 8, No. 3, Apr. 1990, pp.401-412.

[Ste90.9] R. Steinmetz, R. Heite, J. Ruckert and B. Schoner, "Compound Multimedia Objects-Integration into Network and Operating System", Proc. 1st Intl. Workshop on Network and OS Support for Digital Audio and Video, Berkeley, CA, USA, Nov. 1990.

[Ste92] R. Steinmetz, "Multimedia Synchronization Techniques: Experiences Based on Different System Structures", Proc. IEEE Multimedia'92, Monterey, U.S.A., Apr., 1992.

[Ste93] R. Steinmetz and C. Engler, "Human Perception of Media Synchronization", Technical Report 43.9310, IBM European Networking Center, Heidelberg, 1993.

[Sut93] S. L. Sutherland and J. Burgin, "B-ISDN Internetworking", IEEE Communications Mag., Aug., 1993.

[Sye93] A. Syed and C. L. Williamson, "An Experimental Evaluation of Video Coding Algorithms for ATM-based Networks", Proc. ICCC Multimedia Communications'93 Conf., Banff, Canada, Apr. 1993.

[Taw93] W. Tawbi, F. Horn, E. Horlait and J.-B. Stefani, "Video Compression Standards and Quality of Service", The Computer Journal, Vol. 36, No. 1, 1993.

[Taw93] W. Tawbi, L. Fedaoui and E. Horlait, " Dynamic QoS issues in Distributed Multimedia Systems", Proc. 2nd Intl. Conf. on Broadband Islands, Athens, Greece, 1993, pp.69-75.

[Vog93] A. Vogel, (Ed.) " Project Overview, QoS Architecture and QoS Negotiation Protocol", Report, Oct. 27, 1993.

[Wel93] G. Wells, U.S. West Advanced Technclogies, "SRTS and Variable Bit Rate Video (MPEG II) ", ATM_Forum/93-931, Nov., 16-19, 1993.

[Wit93] V. Witana and A. Seneviratne, "Operating System Support for Multimedia Applications", Proc. ICCC Multimedia Communications'93 Conf., Banff, Canada, Apr. 1993.

[Zie90] C. Ziegler and G. Weiss, "Multimedia Conferencing on Local Area Networks", IEEE Computer, Sept. 1990.

Appendix

A.1 The Scenario Specifications in the *BNF Style* Description:

<scenario> ::= <start> <activity> {,<activity>} <end>
<activity> ::= <object_in_activity> <relation_in_activity>

<object_in_activity> ::= <object> {, <object>}
<object> ::= <media_Id> <object_Id> <duration>

<relation_in_activity> ::= <relation> {, <relation>}
<relation>::=<object_Id><relation_Id><object_Id>[<fixed_intermission>][<skew>]
<relation_Id> ::= b|m|e|o|d|s|f

B.1 TFG_generating Algorithm:

1. Initialize a TFG.

    1.1 Generate a TFG, set *number_of_activities.*

    1.2 Generate activities, set *number_of_objects.*

    1.3 Generate nodes.

    Set: *media_Id, object_Id, duration, jitter and skew* as specified in the given scenario.

    Reset: *StartTime_activity, StartTime_node =0;*

    Ready-to-start = False;

    Number-of ancestors = *Number-of-Children* = 0;

2. Link the pairs of nodes according to the mutual relationships

    Assume X,Y is a pair of nodes with a relationship r, X(r)Y

2.1 Link the pairs of nodes which have sequential relationships.

Case(r=b): /* call function before(X,Y, τ); */

if (the fixed intermission τ =0)

then link X to Y as Y's ancestor with δ = TRUE;

else {

generate a transit node N_t with the duration equal to τ;

call function meet(X, N_t);

call function meet(N_t, Y); }

Case(r=m): /* call function meet(X,Y); */

link X to Y as Y's ancestor with δ = FALSE;

2.2 Link the pairs of nodes which have parallel relationships.

Case(r=s): /* call function start(X,Y) */

if (an ancestor of X(or Y) with δ = FALSE is found)

then call function meet(X's ancestor,Y)

or meet(Y's ancestor, X)

else {

generate a transit node N_t;

call function meet(N_t,X);

call function meet(N_t,Y); }

Case(r=f): /* call function finish(X,Y); */

if (a child of X(or Y) with δ = FALSE is found)

then call function meet(Y, X's child)

or meet(X, Y's child);

else {

generate a transit node N_t;

call function meet(X, N_t);

call function meet(Y, N_t); }

Case(r=e): {

call function start(X,Y);

call function finish(X,Y); }

Case(r=o): if (the fixed intermission $\tau = 0$) and (X's duration < Y's duration)

then set τ = (Y's duration - X's duration);

if (an ancestor of Y with δ = FALSE is found)

then call function before(Y's ancestor, X, τ);

else {

generate a transit node N_t;

call function before(N_t, X, τ);

call function meet(N_t, Y); }

if (a child of X with δ = FALSE is found)

then call function before(Y, X's child, 0);

else {

generate a transit node N_t;

call function meet(X, N_t);

call function before(Y, N_t, 0); }

Case(r=d): if (an ancestor of Y with δ = FALSE is found)

then call function before(Y's ancestor, X, τ);

else {

generate a transit node N_t;

call function before(N_t, X, τ);

call function meet(N_t, Y); }

if (a child of Y with δ = FALSE is found)

then call function before(X, Y's child, 0);

else {

generate a transit node N_t;

call function before(X, N_t, 0);

call function meet(Y, N_t); }

3. Repeat step 2 for all mutual relationships in an activity.

4. Form boundary of the activity, assuming the current activity is ACi

4.1. Generate ACi.N_S, insert it to the beginning of the ACi.components

if X.number-of-ancestors = 0, any node X in ACi{components}

then call function before(N_S, X, 0);

4.2. Generate ACi.N_e, append it to the ACi.components

if X.number-of-children = 0, any node X in ACi{components}

then call function before(X, N_e, 0);

5. Repeat steps 2 to 4 for all activities.

B.2. Presentation Scheduling Algorithm

B.2.1 Presentation Scheduling Algorithm:

In each activity, the initial start time of all nodes are zeros.

1). Start an OPEN list with only one node N_S. Start an empty CLOSED list and an empty UPDATED list.

2). If the OPEN list is empty, succeed in presentation scheduling of the activity.

3). Denote the first node in the OPEN list to be *N*. Remove *N* from the OPEN list. The new start time of *N* is stored in *ts_new*. The head node of *N* is stored in *hd*.

4). If *N* has already been in the CLOSED list, then go to 7)

5). Append *N* to the CLOSED list.

6). Expand N to all its children nodes, insert them to the beginning of the OPEN list. In the OPEN list record the new start time of them as *N.StartTime_node* + *N.duration*, and record the node N as the head of the newly expanded nodes. Return to 2).

7). If *ts_new* is not larger than *N*.Ts, then return to 2);

8). Store the increment parameter, x= *(ts_new - N.StartTime_node)*. Empty the UPDATED list.

9). Apply a **recur_resch** algorithm in order to recursively propagate to the nodes which are related to N and add x to their former start times. At the same time move N and other updated nodes from the CLOSED list to the UPDATED list.

10). Append the nodes in the UPDATED list to the CLOSED list. Return to 2)

B.2.2 Recur_resch Algorithm:

Assume N_i is the node being calculated.

1) If the CLOSED list is empty, return TRUE.

2) If the node N_i is not in the CLOSED list, return.

3) Move N_i from the CLOSED list to the UPDATED list.

4) Update the node N_i by adding x to its former start time;

5) For each of N_i's ancestor with which d is equal to FALSE, except the node which is *hd*, call **recur_resch** with increment parameter of x and *hd*. Whenever all the **recur_resch** being called return TRUE, return TRUE.

6) For each of Ni's children, set *x_new* = N_i*.StartTime_node* + N_i*.duration - the child's former start time*. If *x_new* is larger than zero, call **recur_resch** with increment parameter of *x_new*. Whenever the **recur_resch** being called returns TRUE, return TRUE.

7) Return.

B.3 Re-scheduling Algorithm

B.3.1 Re-scheduling Algorithm for the Activities:

Assume the object *i* encounters an arrival which is *x* time unit late. The client MSC which is in charge of this object will locate its corresponding node *N* in the TFG.

1). Store the start time of the N_e node in the current activity to be *ts_old.*

2). Call the **re-scheduling algorithm for the nodes** in the current activity.

3). . If the start time of the N_e node is larger than *ts_old*, then extend the start times of the following activities by *x*.

B.3.2 Re-scheduling Algorithm for the Nodes(in an activity):

1). Start a CLOSED list with all the nodes in the activity where the node N is in. Start an empty UPDATED list.

2). Extend the duration of the object node *N* corresponding to the object *i*, by *x*

3). Set the head node as the node *N*. For each child of the node *N*, apply **recur_resch** algorithm.

4). Succeed.

B.4 Delivery Scheduling Algorithm:

To be performed by the scheduler, assuming there are *n* media types (streams) in the requested document scenario:

1) Find the buffering delay of each stream:

$$P[\ (D_{e\text{-}e} - \mu) > D_{buffering}\ ] < P(Fail);$$

2) Find the total delay of each stream:

$$D_{total} = \mu + D_{decoding} + D_{buffering};$$

3) Among the media object components which is the first one to be delivered on its stream connection, find the component which has the earliest delivery time:

$$\text{Get } k, \quad \text{if} \quad (D_{total\text{-}k} - T_{pk}) = \max (D_{total\text{-}i} - T_{pi});$$

4) Set $T_{dk} = 0$, get the target presentation time for the *k*th component:

$$T_{tk} = D_{total\text{-}k};$$

5) Get the target presentation time for the *i*th component, $0 <= i < n$

$$T_{ti} = T_{tk} + T_{pi} - T_{pk};$$

6) Get the target delivery schedule for the *i*th component., $0 <= i < n$

$$T_{di} = T_{ti} - D_{total\text{-}i}.$$

7) For the stream which has *m* sequential object components following the first component, derive the delivery schedule for the *l*th object components, $0 < l <= m$

$$T_{dil} = T_{di} + T_{pil} - T_{pi};$$