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

DESIGN OF A STORAGE AND RETRIEVAL MODEL FOR MULTIMEDIA DATA

Ruihong Wang

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

in

Electrical Engineering

Ottawa-Carleton Institute of Electrical Engineering

Department of Electrical Engineering

Faculty of Engineering

University of Ottawa

OTTAWA, ONTARIO, K1N 6N5

ISBN 0-612-00595-X

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.

Ruihong Wang.

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

Ruihong Wang.

Acknowledgments

First of all, I wish to express my profound gratitude to Dr.A.Karmouch, my supervisor, for the constant help and support he gave me during my stay at University of Ottawa.

I would like to thank Dr. T. Yeap for his technical suggestions. I also want to thank all the past and current students in Multimedia Communication Research Lab. My special thanks to Lian Li, Jie Li, Nail Hirzalla, James Emery, Dr. Bala Madaparthi, and Dr.Sudhakar Ganti with whom I had good technical discussions.

I want to thank all the support staff members of Electrical Engineering for their help, to Amanda Lauzon, Lucette Lepage, Michèle Roy and Suzanne St-Michel.

I also acknowledge the financial support which I received from the NSERC and the OCRI(Ottawa Carleton Research Institute), for which I am sincerely grateful.

Finally, I am truly grateful to Jun, my husband, and to my family for their consistent support, without which this work would not have been possible.

Abstract

One of the most important and challenging aspects in multimedia technology is the design of a storage and retrieval system that will support the recording and the playback of multimedia data from a secondary storage device. This multimedia storage and retrieval system must be capable of handling discrete media (i.e. text, images and graphics) and continuous media(i.e. audio, video and animation). It should also provide facilities to handle the synchronization information between two or more data objects in a multimedia application.

In this thesis, multimedia technology and its applications are first introduced followed by a review of multimedia requirements. An overview of the current storage and retrieval techniques for multimedia information is then given. Following, our storage model for multimedia data is presented. This model can support the storage of both discrete media and continuous media on a hard disk, it also offers methods to store the synchronization information of data objects on the disk.

Unlike conventional data placement techniques, a special storage pattern is used in the model for the storage of continuous media streams. By using the storage pattern, real-time constraint can be guaranteed during the retrieval of a continuous media stream. A control scheme for simultaneous retrieval of multiple streams is then discussed in a single disk head system. The control scheme can guarantee the continuous retrieval of multiple streams simultaneously. By combining this control scheme with buffering techniques, a dynamic access control algorithm is developed to control the acceptance of the new retrieval request.

Computer simulation is used to help the analysis of the storage model and the simultaneous retrieval control scheme. The results show that the storage and retrieval model can provide a good storage efficiency, and can guarantee the continuous retrieval of delay sensitive media streams.

Contents

List of Figures

Chapter 1

Introduction

In this chapter, we first describe the general background of multimedia systems and applications. Then, we state our research motivation and objective. Finally, we give the thesis outline.

1.1 Multimedia Systems and Applications

A multimedia system combines a variety of information sources, such as voice, graphics, animation, images, audio and full-motion video, into a wide range of applications. It may consist of stereo speakers, high-resolution color displays, megabytes of RAM, fast processors for video and audio, fiber-optic network connections, hundreds of megabytes of disk capacity, pointing devices, text-entry units, and associate multimedia software development tools. The multimedia represents the merging of three industries: computing, communication, and broadcasting.

With the help of multimedia equipment and software tools, a computer is becoming more powerful, more useful, more fun, and less abstract in the way it represent things. Through the multimedia system, users, instead of using conventional photographic, audio, newspaper, and television services, will process, exchange, and transform information in

new ways.

Multimedia Research and Technology

The development and progress of high technologies in computer science and electrical engineering are helping to make multimedia systems technically and economically feasible. Researchers are working within established computer areas to transform existing technologies, or to develop new technologies, for multimedia application. Currently, an important technical transformation is taking place in the key domains of information processing and computer communications [28]:

- The emergence of high speed networks with powerful workstations and new storage devices, imposes new ways of processing information and provides valuable opportunities for the design of multimedia applications which were impossible with existing technologies.

- Distributed system configurations where several powerful workstations share resources located at remote sites by communicating through LANs. Other configurations may cover wider areas such as LAN interconnections through MANs (Metropolitan Area Network) and even WAN (Wide Area Network) allowing communications between users geographically dispersed.

- Multiple information vehicles such as video, audio and text are all integrated as a single entity (the so called multimedia document) with their temporal relationships. Multimedia documents require high performance processing, large storage systems with which to be manipulated, and faster networks to be transmitted.

- Optical fiber will overcome the limitation of the current networking technology in providing high-speed networks, thus permitting the transfer of large amounts of multimedia information over a single channel in an integrated and synchronized manner.

All of these evolving technologies aim at providing the ability for people to communicate and exchange information in a "natural" way by integrating processing and communication of multimedia information. Thus, the new information age will be dominated by multimedia technology which will support all possible types of information and will provide new facilities for their acquisition, transfer, manipulation, storage and retrieval.

Multimedia Application

The development of high technologies also helps the emergence of multimedia applications. Applications in education, entertainment, travel, real-estate, medicine, administration and publishing are emerging at a fast pace. Multimedia electronic mail, video conferencing systems and other kinds of more advanced multimedia applications, such as cooperative design, joint editing, etc., are becoming part of our daily life.

Multimedia applications fall into two classes: stand-alone and distributed. A stand-alone application is a local application which requires that all the resources and presentation materials are locally available. Currently, stand-alone applications, such as workstation with DVI(Digital Video Interactive)[5], [7], [8], the CD-ROM database [40], video playback with editing capabilities [2], [4] are already on the market with affordable prices. The more attractive multimedia applications are remote applications, being carried out in a distributed system over the networks. Real-time video conferencing [67], for instance, allows a group of users to interact through their workstations or terminals over a network. Users may have speakers, monitors and cameras associated with their workstations. At any time, users can display the pictures of all the participants on their monitors. In the mean while, the participants can exchange files, graphics, images and other types of data, while talking on the phone. The electronic work-place, an organization-wide system that integrates various information processing and communication activities, gives another interesting example of the distributed multimedia applications. The study of such systems is growing into a new multidisciplinary field: Computer-Supported Cooperative Work

(CSCW) [50]. As an example, joint editing is one of the popular CSCW applications that are being developed [67][50][51]. In joint editing, a group of users will cooperate to produce a multimedia document. One of the users, for instance, may paste a picture while another is editing a text portion. Other types of data such as video, audio (Hi-Fi music), voice etc. may also be involved in the editing. Another example of a typical distributed application is when users at different physical locations browsing simultaneous multimedia news retrieved from a central multimedia database.

Distributed multimedia applications have advantage over the stand-alone applications in the following aspects:

- Resources located on other systems may be used.
- Presentation materials located on other systems may be used.
- The user may use multiple multimedia applications to accomplish his or her tasks.
- There may be multiple users using multiple application and sharing data dynamically.

Resources in a multimedia system can be expensive peripheral devices, such as a professional VCR, a hardware MPEG [27] to JPEG [26] movie converter, and remote sensing equipment. In the distributed computing environment, the user can leverage these resources from other system. By sharing equipment among multiple users, the cost is shared, lowering the cost to multimedia users, lowering the cost of multimedia desktops for users. Of course, the user can still use local resources. The user does not need to copy presentation materials to their local machine in order to use them. They can use presentation materials located on a central resource. This simplifies the management of data by limiting the number of redundant copies. Users can also share the data with one another to leverage their work through the shared use of material they generate.

Distributed multimedia applications can be categorized into two classes: real-time and store-and-forward applications. From the communications point of view, real-time applications are those that require information to be transmitted in continuous mode from

one location to another (real-time delivery). An example of such application is the video conferencing system mentioned above [51]. In this application, the information exchanged among the users or between the multimedia database and the users is transmitted on the network and displayed on the workstations at almost the same time. On the other hand, store-and-forward applications, by nature, are not real-time applications. An example of such application is the multimedia mailing system where the documents are transferred from one location to another in store-and-forward mode. At the receiver side, the message is stored in the mail-box until the user is ready to read it.

The trend in developing multimedia systems is to ensure that they can support real-time distributed applications. Real-time distributed multimedia applications require a high performance distribution environment that will meet the overall new multimedia requirements in terms of processing, integration, storage and communications.

In the design of the distributed multimedia system, the special technical requirements of multimedia applications are considered to be the most important. In chapter 2, we will analyze the requirements of multimedia applications.

1.2 Motivation and Objectives of Our Research

Multimedia communications has been around for more than 10 years, ever since the first videodisc was announced in 1978 for home entertainment [35]. Now, although more applications for learning and entertainment are being produced on CD-ROM and PC desktop, there are still some years to go for fully distributed and complex multimedia applications. Researchers in universities and research centers have been working hard to overcome the limitation of the current computer system and to help the emergence of a common standard for multimedia.

In the multimedia information research laboratory, University of Ottawa, we are currently exploring a concept of multimedia research that combines many aspects of communications and information processing. This vision of a multimedia information system

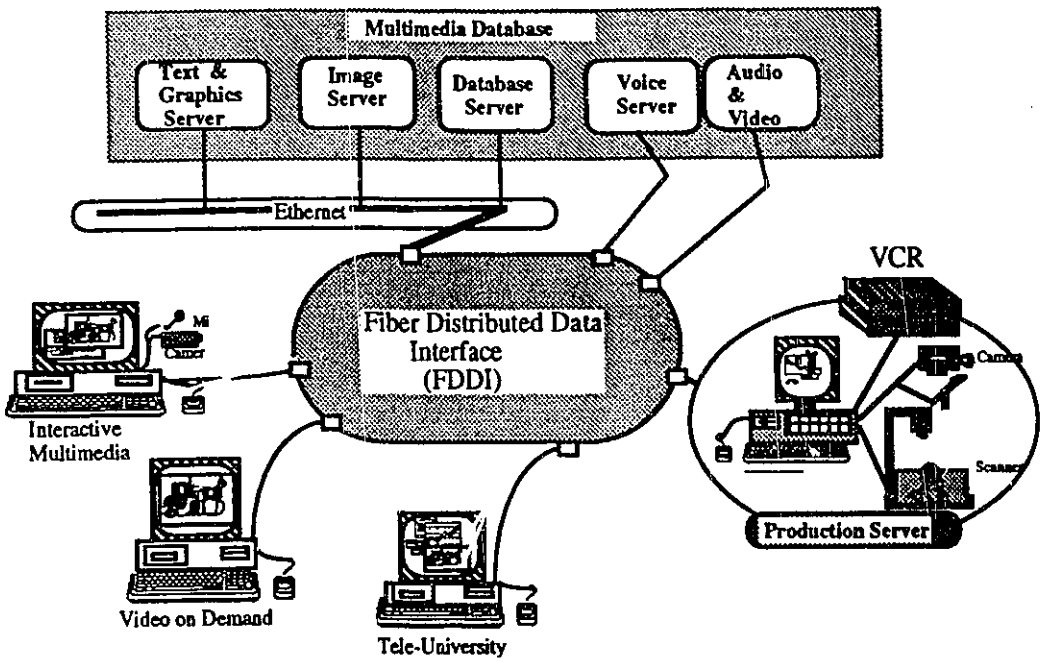


Figure 1.1: Generic Architecture of MEDIABASE

has resulted in the creation of the MEDIABASE project. The purpose of MEDIABASE is to develop an advanced high performance real-time multimedia distributed information system. Figure 1.1 shows the generic architecture of MEDIABASE and illustrates that several multimedia workstations are connected through communication networks(i.e. FDDI). The multimedia production server and the multimedia database are also connected through networks (i.e. FDDI or Ethernet). Thus, users can work at workstations at different locations and share the same production server and database by using high speed networks. The design of a distributed information system such as MEDIABASE, faces many technical challenges. These challenges range from the system architecture down to the physical data storage. In the MEDIABASE project, our particular focus is on document architecture, database model, high-level communication and synchronization protocols, and real-time physical storage of multimedia data. Our MEDIABASE system is aimed at serving interactive multimedia applications such as distance learning, multimedia newspapers, and video-on-demand.

One of the requirements of any system supporting a distributed multimedia application is the need of providing the storage and retrieval method for multimedia data. As a matter of fact, in the effort of developing multimedia distributed application systems, the storage and retrieval of multimedia data is one of the most important elements of the whole system. The work reported in this thesis, as a sub-project of MEDIABASE project, is the first step toward the design and development of a multimedia file system which will be used to manage the multimedia files. This project is named as MEDIAFILE.

The biggest challenge of designing a multimedia file system is the integration of continuous media such as audio and video. The integration of digital video and audio media in database requires an appropriate storage structure and organization of the media on a dedicated secondary storage device. The storage technique employed must meet the real-time deadlines required for the rendition of the media. In the case of video, for instance, real-time deadlines must be met for the display of successive frames in order for motion to flow smoothly without any gaps or interruptions. The retrieval of continuous media from a secondary storage device must be performed according to the appropriate data rate. The data retrieval rate from the disk depends on the characteristics of the storage device, the data placement on the disk, the compression rate achieved, the number of frames (e.g. video) required and the chosen granularity for the basic object (e.g. frame) manipulated. MEDIAFILE aims to provide an integrated storage and playback system that physically stores and retrieves multimedia information from a dedicated secondary storage device. The system is developed for single and multi-users environments.

The objective of this thesis is to describe a multimedia storage model to integrate all types of media on the storage device, and to propose a retrieval control scheme to guarantee simultaneous real-time retrieval of multiple continuous media streams.

1.3 Organization of The Thesis

The thesis is organized as follows:

In chapter 2, we describe the requirements of multimedia applications, particularly focusing on the requirements for the design of a multimedia file system. Current research works in the related area are also briefly introduced in this chapter.

In chapter 3, we present our design of a MEDIAFILE storage model and a multimedia data structure. The simulation results of our algorithms are also given. In the last part of this chapter, we summarize our work and state our contributions.

In chapter 4, a detailed analysis of the simultaneous retrieval of multiple continuous media is presented. First, we analyze the retrieval process of a single continuous stream. Then, the retrieval control scheme for the simultaneous retrieval of multiple streams is described. The dynamic access control algorithm is introduced to control the acceptance of retrieval request. At the end, to verify our analysis, simulation results are shown.

In chapter 5, we summarize our current research work and compare it with other related works. The suggestions for further research are then given in this chapter.

Chapter 2

Multimedia Requirements and an Overview of the Development of a Multimedia File System

In this chapter, we present the requirements of multimedia application, and review the research trend in development of multimedia file system. First, the term "multimedia" which is used in the following chapters is clearly defined. In section 2.2, the general requirements of multimedia applications are introduced. In section 2.3, we analyze the requirements of a multimedia file system which should be respected during the system design. The current development of multimedia file systems and the trend of the storage and retrieval technologies are reviewed in section 2.4. Section 2.5 gives the summary of this chapter.

2.1 The Definition of Multimedia

There is not a standard definition for multimedia, but if we look up the root of the word, we can understand it very simply. A medium denotes a type of information vehicle such as text, graphics, drawings, animation, voice, audio and video. Human beings use different information vehicles to communicate each other. Each of these vehicles is

adapted to one of our senses, and may be used simultaneously and synchronously with others. A film, for example, presents a sequence of images and a sound track. The soundtrack is itself a compound of two different types of information vehicles: one is the voice and the other is music. Similarly, a written report uses typed words, drawings and graphs to express ideas. These are all examples of a medium — a vehicle of information for humans.

In computers, since all types of data are represented in digital format, a medium is not only a certain type of data, but also an output device and a method by which the computer "displays" (i.e. sends to a user) data in a form humans can understand. In addition, the computer must acquire the data representing the information and provide methods for storage and manipulation of the data.

The distinguishing factor between media in a computer is the "display" method. According to their "display" method, we categorize media into two groups: discrete media, and continuous media. Discrete media includes text, graphics, images etc. The display of these media has no strict time constraints. They are time independent media. Most discrete media are already very well handled by the current computing systems.

In the case of continuous media, such as voice, audio and video, the information itself may be expressed as a function of time, and require to be displayed on the screen in a limited time constraint. Usually continuous media requires large storage space, large I/O bandwidth, and large network bandwidth. Therefore continuous media brings major challenges to the current computing systems.

The term "multimedia" takes a different meaning for different people. In general, an application that deals with more than one media can be called a multimedia application. Displaying a movie with video and audio, or an E-mail with text and graphics are two examples of multimedia applications. In our MEDIABASE project, we define the term "multimedia" as the collection of discrete media - text, graphics, and still images, and continuous media - voice, audio, animation and video.

2.2 General Requirements of Multimedia Applications

The complexity of multimedia applications stresses all the components of a computing system. The architecture of a computing system must provide high bus bandwidth and efficient I/O for multimedia applications. A multimedia operating system is needed to support new data types, real-time scheduling, and fast-interrupt processing. Multimedia applications require very high storage and memory capacity, fast access time, and high transfer rate. New networks and protocols are necessary to provide the high bandwidth, low latency and low jitter required for mutimedia applications. New object-oriented, user-friendly software development tools, as well as tools for data retrieval and management are required for networked and distributed multimedia systems.

In the following, we will discuss some important requirements of multimedia application in respect to hardware, compression/decompression, synchronization and networks. The requirements of storage and retrieval of multimedia data will be discussed in details in section 2.3.

2.2.1 Hardware Requirements

In the development of computing system, the application software is usually pressed to catch up with the advances in the hardware. However, as far as multimedia is concerned, it is the hardware that must keep pace with what the software can do.

Multimedia with full-motion video and audio requires processing rates, bus bandwidths, main memory and video/graphics memory, mass storage, I/O bandwidth and transmission bandwidth at least two orders of magnitude beyond those required for text and data. Therefore, the traditional desktops, workstations and microprocessors are not capable of handling multimedia data. The root of the matter is that none of the current computers is very friendly to full-motion video. With current desktop and consumer hard-

ware platforms, most multimedia presentations to date have been a medley of cartoon-like animation, music, voice, and photograph-quality still images digitized by means of a scanner. Full-motion video on a computer requires to add a device for video acquisition, to increase the capacity of mass storage and memory, to speed up the processing rate, and to expand the bandwidth of I/O and the network. However, all of these will increase the cost of the system dramatically. For example, a video capture board may cost as much as its host system. So, developing powerful multimedia hardware at relatively low cost is essential for a multimedia system.

Currently, new multimedia hardware products are being developed in research and industry laboratories, and some of them can already be seen on the market. For example, highly integrated video-frame-grabbing and digitizing circuits from companies such as Chips and Technologies, Cirrus Logic, and Philips Semiconductor have already spurred the proliferation of video capture and preprocessing boards selling for less than 300 dollars a piece.

The bottleneck in developing multimedia computing systems is caused by the bandwidth problem. Running a HDTV program at 30 frames a second would requires a bandwidth of almost 120 MB/s, which is already beyond the reach of current desktops and workstations. The industry is addressing the bandwidth bottleneck, first, by compressing the video and audio data into small file sizes and, second, by expanding the bandwidth of disk I/O, the buses and network links on which those files travel. Neither alone is sufficient. So, in the following section, we will describe the requirements of compression and decompression.

2.2.2 Compression/Decompression Requirements

Compression/decompression techniques clearly play a crucial role in digital multimedia applications. Data compression/decompression is needed for three reasons:

- the huge storage requirements of multimedia data.

- relatively slow storage devices that can not retrieve multimedia data, especially full-motion video, in real-time.
- network bandwidth that does not allow real-time transmission of video or audio data.

For example, a single frame of a colour video, with 620x560 - 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. If a multimedia application requires more than 30 minutes of video, 2,000 images, and 40 minutes of stereo sound on each side of a laser disk, this application would require about 50 Gbytes of storage for video, 15 Gbytes for images, and 0.4 Gbytes for audio. That requires a total of 65.4 Gbytes of storage on the whole disk. With the multimedia file likes this, even if we have enough storage available, we may not be able to playback the video and audio in real-time due to the insufficient disk transfer rate of the storage device, and insufficient bandwidth of the network.

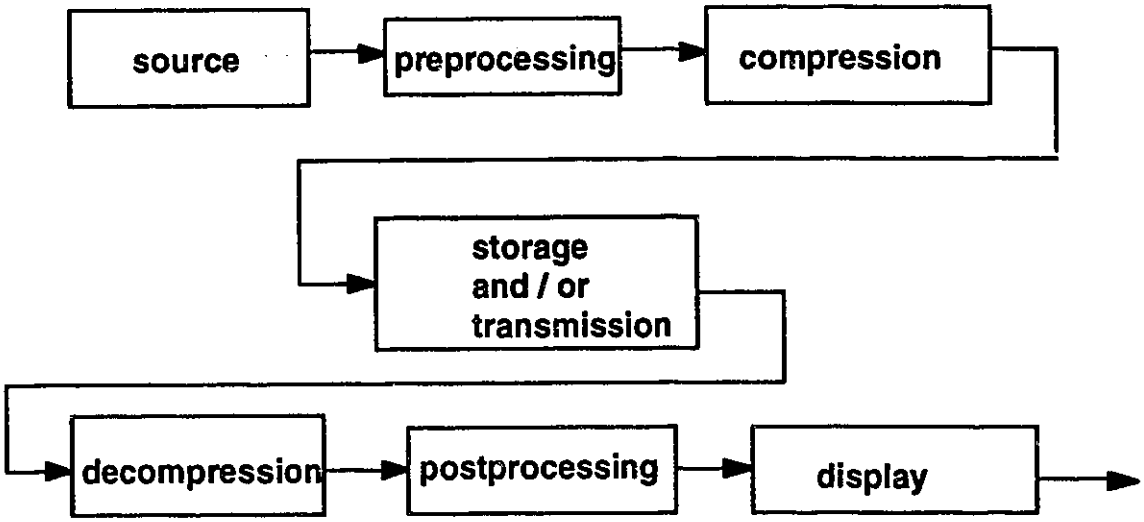


Figure 2.1: Compression and Decompression in Multimedia System

In order to store and transfer more digital information, we need to use compression technologies during storage process and use real-time decompression technologies during its retrieval, as shown in figure 2.1. Different applications have different requirements for the compression and decompression cycle. Some require real-time decompression and can afford to wait for a long compression process, others may require the reverse. The reason is that the importance of display quality varies across applications. For example, some applications require high quality images and long video segments, so a small delay will not affect the quality of display; others may require very rapid changes among short segments, in which delays of several seconds may not be acceptable. Therefore, different compression/decompression schemes must be applied according to different requirements.

A number of design choices are made during the compression stage, including image size(full-screen, cropped, or reduced), resolution, pixel depth, and compression factors. All these factors together will determine the amount of disk space used, image quality, and decompression time etc. JPEG [26] and MPEG [27] are currently two industry-standard algorithms for compression of photo-quality still images and full-motion images respectively. The compression ratios of using these algorithms range from 20:1 to 200:1. However, there are still many difficult technical problems such as dealing with real-time decompression problem, reducing the cost of compression, reducing the amount of hardware and special software, improving the quality of images, and ensuring the smooth motion sequences need to be solved.

2.2.3 Synchronization Requirements

Synchronization is considered as one of the most important issues in a distributed multimedia system. Synchronization problem exist not only in multimedia presentation, but also in the storage/ retrieval, and the transmission of multimedia data.

In a multimedia system, an information item in mono-media is referred to as a data object [70]. For example, one page of text file or a segment of audio can both be data objects. In multimedia data integration, several data objects of different media are

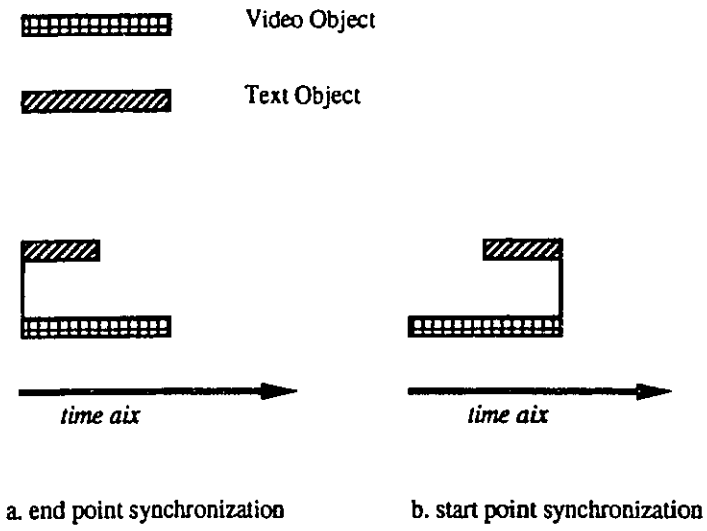


Figure 2.2: Synchronization Scenario

combined to form a new multimedia object, which is a composite data object. The data object is taken as the basic data unit in the multimedia system.

Typical synchronization requirements involved in a multimedia application can be classified into two kinds: intra-media synchronization and inter-media synchronization. In the playback of a continuous media stream, such as audio or video, the playback rate of a media stream must meet exactly its recording rate. For instance, a NTSC video is recorded in 30 frames per second, then, to guarantee the display quality of this video, data has to be retrieved from the disk continuously and be displayed on the screen at the rate of 30 frames per second. This intra-media synchronization is important in applications such as telephony, and video-on-demand. Intra-media synchronization only exists in the playback of a single continuous media. On the other hand, inter-media synchronization is required in multimedia data integration, where the temporal relationships among the data objects have to be handled. Different temporal relationships implemented among the same set of data objects can result different different display scenarios. Two data objects(e.g. the video object and the text object in figure 2.2), have different display

durations, if they are synchronized at the start point of their display (start together), we will have the display scenario as shown in figure 2.2b. However, if they are synchronized at the end point of their display(end together), see figure 2.2a, the display scenario will be different. Thus, the temporal relationship between media objects have to be stored along with the objects on the disk, so that they can be used during their retrieval and transmission to guarantee a synchronized presentation.

To meet the synchronization requirements of multimedia data, a multimedia system must have the following capabilities:

- store the temporal relationships between data objects.
- schedule the retrieval of related data objects according to their temporal relationships.
- allow a fast access to the stored data objects, so that their retrieval deadlines can be met.
- transmit related objects over the network in a synchronized way.
- predict the network delays of different data objects and resynchronize the related data objects before their playbacks.

2.2.4 Network Requirements

The integration of all multimedia services within a single broadband network poses new communication requirements of switching, multiplexing, transmission, control and processing of voice, audio and video signals. Table 2.1 contrasts traditional data transfer and multimedia data transfer.

Characteristics	Data Transfer	Multimedia Transfer
Data Rate	Low	High
Traffic Pattern	Bursty	Stream-oriented highly bursty
Reliability Requirements	No loss	Some loss
Latency Requirements	None	Low, for example, 20ms
Temporal Relationship	None	Synchronized Transmission

Table 2.1. Network Communications: Traditional Vs. Multimedia

Data Rate

Because of the large data size of a multimedia file, multimedia networks require a very high transfer rate or bandwidth, even after the data is compressed. For example, full-motion video service using high definition television (HDTV) requires a transfer rate of 200 - 300 Mbit per second, even after compression. Consider the synchronization requirements, besides being high, the transfer rate has to be predictable.

Reliability

Traditional networks are used to provide error free transmission. However, most multimedia applications can tolerate errors in transmission due to corruption or packet loss without retransmission or correction. In some cases, in order to meet real-time requirements or achieve synchronization, some packets are even discarded.

Latency

Multimedia networks must provide the low latency required for interactive operation. Since multimedia data must be synchronized when it arrives at the destination site, networks should provide synchronized transmission with low jitters [46], [47], [48].

Multipoint

In multimedia networks, most communications are multipoint as opposed to traditional point-to-point communication. For example, when multimedia conferences involve more than two participants, the network needs to distribute information in different media to each participant [68].

Therefore, traditional networks with low transfer rate and unpredictable latency and

jitter, such as Ethernet and Token-ring, can not carry multimedia data. New high-speed networks such as FDDI, DQDB, and B-ISDN which have data rates of 100-200 Mb/s, provide more power to transfer multimedia objects. Among these new networks, optical network technology can support the Broadband Integrated Services Digital Network(B-ISDN) standard, expected to become the key network for multimedia applications.

Since different type of data objects may have different requirements for their transmission, multimedia applications require network systems to have flexibilities in several respects [56]. As far as the public and private telecommunication networks and in particular the switching systems within the network are concerned, the asynchronous transfer mode (ATM) will play a significant role in realizing the flexibility and economy necessary for multimedia communications. ATM provides great flexibility in bandwidth allocation by assigning fixed length packets, called cells, to virtual connection. ATM can also increase the bandwidth efficiency by buffering and statistically multiplexing bursty traffic at the expense of cell delay and loss.

In the above discussion, we have introduced the general requirements of multimedia applications. These requirements must be met before the multimedia applications can be realized.

2.3 MEDIAFILE Requirements

As an important part of a computer system, the file system also faces a lot of new challenges from the requirements of multimedia applications. The objective of our MEDIAFILE is to handle real-time storage and retrieval of multimedia data. The defining characteristic of multimedia is the involvement of continuous media such as voice, audio and video. The design of MEDIAFILE not only has to meet the traditional requirements of a file system (e.g. easy manipulation, random accessibility, reliability etc.), but also has to meet the special requirements of multimedia.

There are three important features that differentiate a digital multimedia file system from a traditional text file system.

Large File Size

Multimedia files have very large storage requirements. Figure 2.3 illustrates the storage requirements for a multimedia application consisting of various media types. Compressing the images by a ratio of 15:1 and the video by factors of 30:1 and 200:1, the total storage requirements for this application will still be over 2 Gbytes. Large storage requirements ask for a more efficient storage model. However, if the file system is to act as a basis for supporting media services such as document editing, mail, distribution of news and entertainment, etc., it must provide mechanisms for manipulation (e.g. insertion or deletion) and sharing stored data. For these mechanisms to be efficient on large size of multimedia data, the file system must minimize copying of data on the disk.

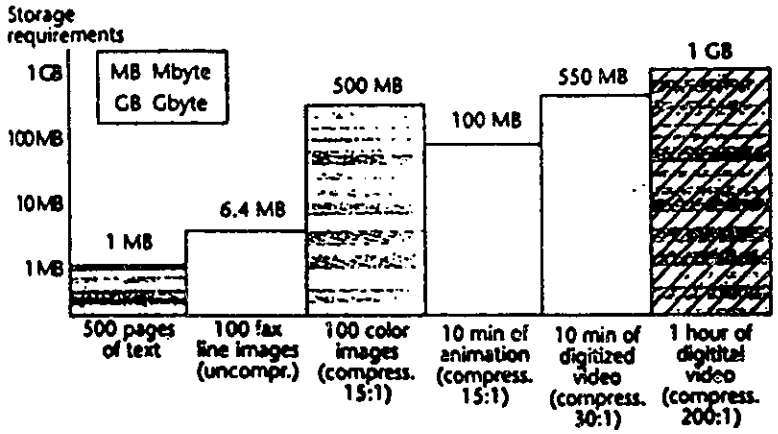


Figure 2.3: Example of Storage Requirements

In the design of our storage model, we try to find a trade off between the efficient usage of the disk space and the efficient manipulation and sharing of data.

Continuous Recording and Retrieval of Media Streams

Recording and retrieval of audio and video are continuous operations. The file

system must organize multimedia data on the disk so as to guarantee that their storage and retrieval is performed at their respective real-time rates.

To ensure continuous retrieval of media streams, the storage model has to employ constrained placement of media blocks. The size of each media block and the separation between successive media blocks of a media stream are determined such that the time to access each media block is bounded by the playback duration of the block. The details of retrieval constraints will be addressed in chapter 3.

While the control of the data placement of media blocks only suffices for ensuring the continuity of one stream retrieval, satisfying the retrieval of multiple streams simultaneously, without violating any of their continuity requirements, forces multimedia file system to have retrieval and access control mechanisms. Our scheme to support simultaneous real-time retrieval streams is presented in chapter 4.

Synchronization of Multiple Data Streams

Suppose that a multimedia file consists of three components: audio, video, and text. Generally, these three components of a multimedia file are separated at the input and they arrive at the file system as three different streams. Similarly, during retrieval, these streams are routed to different output devices. Storing these media together may entail additional processing for combining them during storage, and for separating them during retrieval. On the other hand, if the three media are stored separately, the file system must explicitly maintain temporal relationships among the media so as to ensure synchronization between them during retrieval and transmission. As part of our storage model, the storage of synchronization information is introduced in chapter 3.

We have discussed the requirements of a multimedia file system. The design of a file system that addresses these requirements is the main subject matter of our MEDIAFILE project. As the first step of this project, this thesis mainly focuses on the design of a storage and retrieval model which can handle continuous media such as audio and video.

2.4 Overview: The Development of Multimedia File Systems

From our discussion in section 2.3, we know that the biggest challenge in designing a multimedia file system is the storage and retrieval of continuous media such as audio and full-motion video. For several years, researchers in industries, universities and research centers have been working toward the design and development of new storage models, new methods of data management, new control mechanisms etc.. The aim of these efforts is to make computers more friendly to the storage and retrieval of multimedia data, especially to audio and full-motion video. In the following, we will first review the development of storage technology by comparing the difference between optical storage devices and magnetic storage devices. An overview of the strategies of multimedia data placement is given in section 2.4.2. Finally, various approaches which can guarantee simultaneous retrieval of multiple continuous streams are introduced.

2.4.1 Storage Technology for Multimedia Data

Recalled from last section, the volume of data managed by a storage system is rising dramatically. The ever-increasing storage demands of individual applications, the migration of large database applications, and the introduction of the new groupware, imaging, and multimedia applications contribute to the need for expanding the capacity of storage and increasing the data transfer rate of storage drivers. To meet the storage and retrieval requirements of multimedia files, we first have to choose an appropriate storage device.

Our computing environment contains many types of storage, from the registers that feed the processor pipelines to the jukeboxes that store archival data. The different types of storage in a computer system form a pyramid like hierarchy, as shown in figure 2.4, with fast, expensive devices at the top and slow, inexpensive ones at the bottom [34].

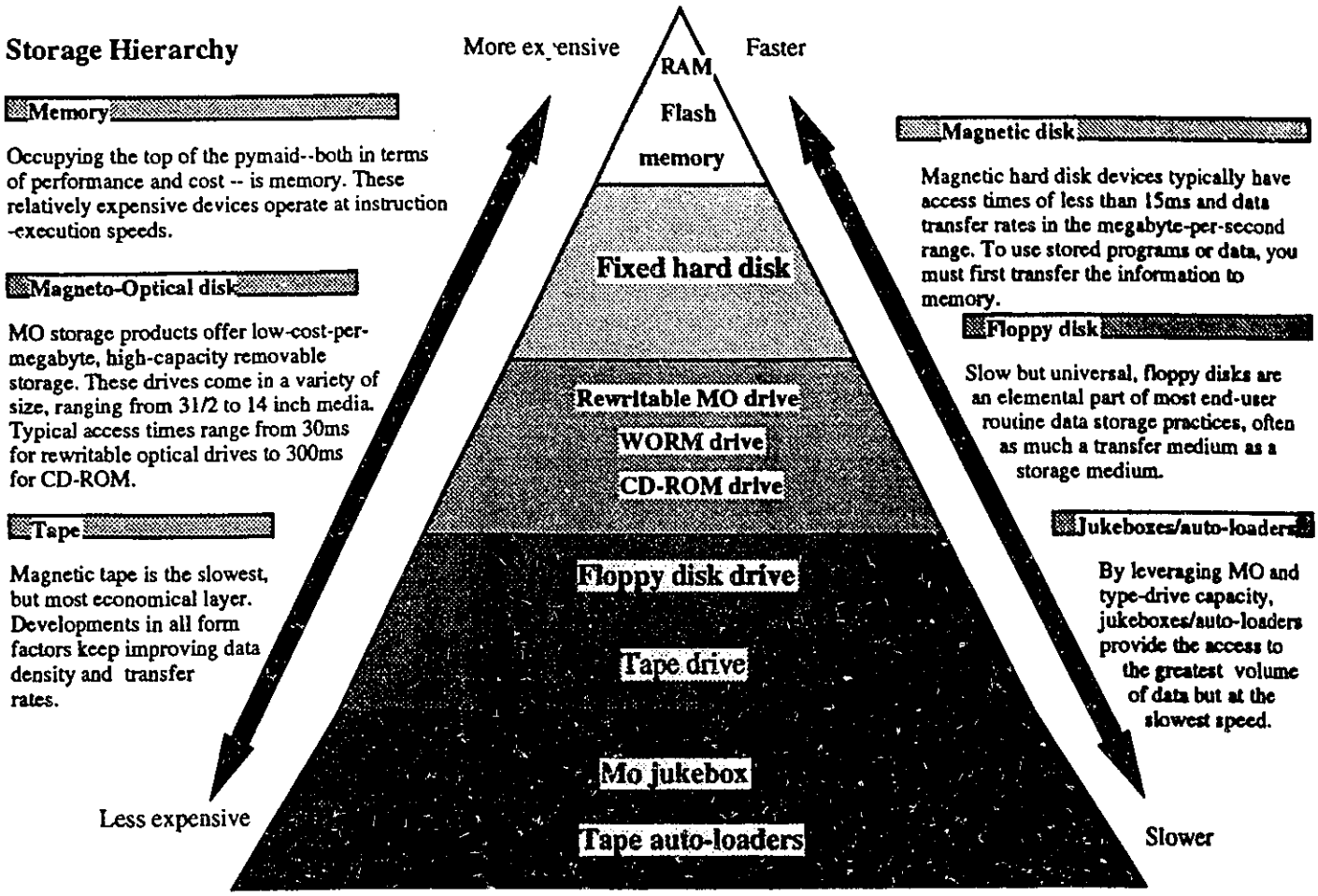


Figure 2.4: Storage Pyramid

The top of the storage pyramid consists mostly of DRAM and associated caches that are used for temporary storage of programs and data. As applications such as multimedia databases, and operating systems become more memory hungry, we will see the average memory of computer systems rise to accommodate such requirements. In the bottom part of the storage pyramid are traditional storage devices – floppy disk drives, tape drives, mo jukeboxes, and tape auto-loaders. These devices provide the access to the greatest volume of data, but at a very low speed.

As we analyzed in section 2.3, multimedia files require a very large storage capacity, as well as a fast data transfer rate to guarantee the continuous retrieval of media such as audio and video. Therefore, the storage solution for multimedia data is to choose hard disks or optical disks, which reside the middle part of storage pyramid, as the main storage device. In the following, we will overview and compare these two storage devices in terms of their storage performance(e.g. capacity, data transferate, access time etc.)

Hard Disk

Hard disk technology is the most rapidly evolving storage technology. In the past few years, great gains have been made in the key performance of hard disk such as areal density (which dictates drive capacity), access time, and data transfer rate. Phil Devin, vice president of storage technologies at Dataquest in San Jose, California, foresees disk density improvements of 60 percent to 65 percent annually, prices falling by 12 percent per quarter, and product life cycles of less than a year [34]. The improvement of data density of the disk is not the only good news. Drive latency, which is a function of rotation speed, is also improved. Spin speed has already been increased from 5400 rpm to 7200 rpm. Spindles spinning 33 percent faster means data is available sooner, combined with increased areal density, the effect is a high data transfer rate and low seek time.

For example, IBM's DFMS and DFHS families of high performance 3.5-inch drives rotate at 7200 rpm. These 1-, 2.1-, and 4.3-GB drives feature an industry-leading areal density of 564Mb per square inch, a seek time of 8.6ms, and the industry's highest data

transfer rate of 12.2 MB/s [34]. The Barraenda, another 7200-rpm drive from Seagate Technology, offers 8ms seek times and a capacity of 4.1 GB. With lower seek time, improvements in caching, and higher spin speeds, access time is getting better as well.

For databases which are continuously on-line and constantly being updated, a hard disk with high data transfer rate and low seek time can be the suitable storage device. However, a hard disk has a high cost per Mbyte compared with an optical disk. For databases which require a very large storage capacity and on which changes are rarely made, for example archival database, it is better to use an optical disk as the main storage device.

Optical Disk

The principal advantages optical disks have over hard disks are greater bit density, removability, and low cost. These combine to make an optical disk ideal for library and archive applications.

Optical disks come in various sizes ranging from about 7cm to about 30cm. There are various categories of optical disks, such as read-only disks, write-once-read-many-times disks(WORMs), and erasable disks such as magneto-optical disks. Data on read-only disks or WORMs can niether be erased nor rewritten. Compact disks (i.e. CDs) are one of the formats of read-only optical disks. CDs also have many variations, including CD-DA (compact disk digital audio), CD-I (compact disk interactive), CD-V (compact disk video), and CD-ROMXA (compact disk read only memory extended architecture)., each holds digital data and uses the same physical encoding scheme. The differences between these formats lie in the amount of data integrity that is achieved by the encoding scheme, the data coding schemes that are used or allowable, and the allowed disk data topography.

The capacity of compact disks has already improved from 650 MB to 2 GB. Rewritable optical disks are also under development, and are becoming more capacious and attractive. We anticipate that the capacity of some 3.5-inch rewritable disks will exceed 500 MB [11].

Although optical devices offer large storage capacity and lower cost, their average seek times are about three times longer than the hard disk and their data transfer rates are much slower compared to that of the hard disk. The data transfer rate of current optical storage device is ranged from 200 KB/s to 2 MB/s based on [11]. The other strong drawback of optical storage devices is their lack of reversibility.

Currently, because of its large capacity, low cost and removability, optical storage devices (e.g. CD-ROM, CD-I) are widely used in the desktop computer for multimedia applications such as entertainment and training. CD-ROM , one of the popular optical devices, is also used by large engineering companies as a way of distributing large software applications. With the low data transfer rate and high access time, it is very hard for an optical disk to handle well the real-time storage and retrieval of continuous media such as audio and video. The lack of reversibility prevents the usage of an optical storage device in some multimedia applications such as an on-line multimedia database. However, with the development of new optical technologies, we expect to see reversible CDs with a higher data transfer rate and a faster access time in the next few years.

Above, we have reviewed the technologies of both the hard disk and the optical disk. Since our MEDIABASE project focuses on the interactive multimedia database, in which high data transfer rate and fast access to the stored data are essential, we, currently, have chosen to use the hard disk as our main storage device, and we have concentrated on the study of multimedia data placement strategies and retrieval control schemes for continuous media streams. However, our storage and retrieval schemes are not limited to the hard disk, the same methods may be extended to high performance erasable optical storage devices in the future.

2.4.2 Data Placement Methods for Multimedia Data

With huge data size and heterogeneous data types, multimedia applications made data placement one of the most important issues in the design of multimedia file systems.

In the traditional file system, data is stored at the random locations on the disk, and related data objects are linked by pointers according to their sequence. However, this data placement method is not suitable to multimedia data. One of the reasons is that the retrieval of media such as voice, animation, audio, and video has real-time requirements. If the data objects are randomly located, the seek time overhead and rotational latency between the two consequence data objects may be long enough to violate the real-time constraints of continuous retrieval. Thus, new data placement strategies have to be developed for the storage of the delay-sensitive multimedia data to guarantee the continuous retrieval.

Besides, the synchronization requirements of data objects also made traditional file systems no longer suitable. As pointed out section 2.3, different media streams can be arranged for simultaneous retrieval, requiring the multimedia file system has to have the abilities to store and to modify the synchronization information. This allows the multimedia file systems to be used during the retrieval, transfer and presentation processes.

In short, a new storage model needs to be designed in order to store and to retrieve both discrete and continuous media, and in order to meet the synchronization requirements between different data objects.

Several multimedia file systems have been proposed in the past few years. However, most of these systems have focused on the storage of still images with text [39], [40], [41]. The design of the storage system for delay-sensitive media has only started. Yu et al [6] introduce techniques for merging the storage of real-time audio files on read-only optical disks. Anderson et al. [42], [43] give a high-level description of a distributed system that includes delay-sensitive data without describing in detail how the real-time requirements of data retrieval are met. Abbot [44] has examined a storage placement strategy that yields a high probability of improved system throughput, but does not absolutely guarantee meeting the real-time constraints. Park and English [45] give a strategy for using lower-quality, lower-bandwidth audio data when contention for bandwidth occurs. However, their method does not provide a guaranteed minimum display quality.

Our insterest concentrates on the work reported by Genmmell and Christodoulakis

in [24], and the storage model developed by Rangan and Vin [12]. The work reported in this thesis is motivated mainly by their experience.

In the Gemmell and Christodoulakis paper, a theoretical frame work is developed for real-time requirements of digital audio playback. The real-time requirements are described in terms of the consumption rate of the audio data and the nature of the data retrieval rate from the secondary storage device. Making use of their framework, bounds are derived for buffer space requirements for three common retrieval scenarios (details can be found in section 2.4.3). They also describe and compare the different data placement strategies for multimedia data streams in general.

According to their paper, the placement strategies of multimedia data can be categorized as:

- Scattered noninterleaved data placement.
- Contiguous noninterleaved data placement.
- Contiguous interleaved data placement.
- Scattered interleaved data placement.

Placement of data may be either interleaved or noninterleaved, contiguous or scattered. An interleaved placement groups together data that are read at the same time. This implies that the synchronization information is completed before placement and that changing synchronization involves moving the data on the storage device. In a contiguous strategy, audio samples or video frames, in the order they will be read, are stored together in a contiguous fashion, one after another, on the storage device. In a scattered strategy, the data may be split up into blocks and placed in various locations on the storage device.

There are various advantages and disadvantages to each strategy. Interleaving is used in an attempt to reduce seeking. By placing blocks that must be read at the same time, side by side, minimum seeking between them is guaranteed. However, synchronization information must exist in order to do interleaving, and it cannot change without rewriting the data. Adjustment of synchronization is very flexible under a noninterleaved

scheme. Another benefit of using noninterleaved placement is that, when a certain part of a data stream is used several times, it need only be recorded once on the storage device. With interleaved placement, it would have to be copied once each time it is to be used.

Scattered placement allows more flexibility than contiguous placement, so that external fragmentation may be avoided. External fragmentation occurs when small gaps must be left on the storage device because no file is small enough to fill them. However, there is more overhead involved in keeping track of the locations of scattered blocks. In addition, contiguous interleaved placements require much less seeking during playback than scattered interleaved placements.

Currently, the placement strategy used by most of the multimedia applications is contiguous interleaved placement. Compact disk, the best-known digital audio storage media, CD-I(i.e. compact disk interactive), and DVI(digital video interactive) are examples of using contiguous placement strategy for data storage. For systems using magnetic storage devices as the storage media, little information regarding placement is available. However, Rangan et al attempt to use scattered placement strategy for the storage of full-motion video. Anderson et al. developed their continuous media file system by assuming contiguous placement of multimedia data on a magnetic disk [32]. The review of Anderson's media file system will be given in the next section.

In the storage model developed by Rangan et al, a storage pattern can be obtained for each video stream. The storage pattern, in general, is that a sequence of video frames is organized on the storage device in terms of media blocks separated by gaps of free space. Their model relates device characteristics to the media recording rate, and derives the block size and gap sizes that result in continuous retrieval. In addition, the model provides a merging and layout mechanisms for multiple streams so as to utilize disk space efficiently. In the merging of video streams, gaps between media blocks of one stream are filled with media blocks of other streams. As a result, the storage pattern and the continuity of the second stream may not be maintained strictly for each media block. To guarantee the continuous retrieval of the second stream, they introduce read-ahead and buffering of finite number of blocks during retrieval in order to preserve the storage

pattern and continuity properties on average over a finite number of blocks.

Rangan et al's storage model can guarantee the real-time requirements of video retrieval, and can reduce the data copying amount required during data editing. Using the merging, their disk usage can approach 90 percent. However, read-ahead and buffering during the retrieval increased buffer requirements and complicates the computational performance.

We have reviewed various data placement approaches for multimedia data. In chapter 3, the storage model of multimedia developed in our lab will be introduced.

2.4.3 Approaches for the Simultaneous Retrieval of Multiple Continuous Streams

The low I/O bandwidth of the current disk technology, and the large seek time overhead as well as the slow rotation speed of the current disk drive present a real challenge to some multimedia applications (e.g. video-on demand). In these multimedia applications, different continuous streams may be required to be displayed simultaneously. Sometimes the display of one video stream may require synchronization with the other video or audio stream that is already running. Thus, a real-time access to large amounts of storage and to the continuous retrieval of these media streams is required.

Currently, there are several techniques to support a real-time retrieval and display of multiple media streams. These techniques include scarifying the quality of the data [7], [9]; using RAID as a high bandwidth secondary storage device [30]; disk multitasking and data replications[31]; also various retrieval control polices [29], [30], [32], [13].

The central idea behind RAID(Redundant Arrays of Inexpensive Disk) is to construct an I/O subsystem which consists of a disk controller and an array of disk drives. The controller distributes a file or a multimedia object across all the drives in order to provide a high data rate upon its retrieval. In order to minimize the probability of the data becoming unavailable in the presence of disk failures, RAID has introduced a taxonomy

of five different organization of disk arrays, beginning with mirrored disks and progressing through a variety of alternatives with different performance and reliability characteristics. For multimedia systems, a limitation of RAID is its lack of control on the placement of the data. Consequently, one cannot control the rate at which the I/O subsystem produces data. For example, consider a 2-h video object with a 45 Mb/s bandwidth requirement (this object is 40 gigabytes in size). Ideally, the I/O subsystems should produce the object at a continuous bandwidth of 45 Mb/s for a period of two hours. However, if a RAID I/O subsystem consists of 100 disks, each disk with a bandwidth of 10 Mb/s, then the data will be produced at a rate of 1000 Mb/s, overflowing the memory buffers of a display station. It is also unclear whether RAID can support a real-time display of several objects simultaneously.

The work reported by Ghandeharizadeh and Ramos [31], can be extended to RAID to enable it to support a multimedia information system. Continuous retrieval of multiple media streams is guaranteed by using multitasking or replication techniques. In their paper, several different approaches for disk multitasking or replication are described in a shared-nothing architecture (i.e. processors do not share disk drives or random access memory and can only communicate with one another by sending messages using an interconnection network).

Multitasking is when a disk drive is required to retrieve a portion of each requested fragment in one scan of the disk drive (instead of the entire fragment as with a batch-order). Multitasking enables the system to support the simultaneous display of multiple objects, but it wastes the bandwidth of each disk drive by causing it to perform expensive seek operations. By replicating the fragments of an object across several processors, the system can use a fragment of an object that resides on an idle processor to service a request. The replication approach does not incur the seek time overhead, so it can support a higher number of simultaneous displays. However, it requires the system to provide a high amount of disk storage in order to store multiple copies of the data.

No matter which approach is used to support the simultaneous display of multiple objects, a retrieval control scheme is always needed to avoid contention of the storage

device. Various retrieval control schemes are proposed to guarantee the real-time retrieval rate of simultaneously displayed objects. Among them, the most noteworthy schemes are proposed in CLSA system [29], CFMS [32], Genmmell and Christodoulakis's paper[24], and Rangan's paper [13].

CLSA (Constrained-Latency Storage Access) system has been developed for applications that have strict deadlines for the completion of some secondary storage accesses. These applications must look for prefetching to satisfy constraints on transaction times. To satisfy constraints, worst-case latency must be avoided. In CLSA system, data is prefetched from the slow lower level to the faster upper level of the storage hierarchies. The system has to analyze the retrieval requirements, and generate a prefetch schedule according to the different applications, and according to the latency of the storage device.

In the CLSA system, applications are categorized as periodic, scripted and probabilistic. Periodic applications are the easiest for developing a prefetch algorithm. Scripted applications (when an explicit storage request is known in advance), with synchronization requirements between different objects, are not supported in existing computer systems. Probabilistic applications, for which the target of required events is unknown until the runtime, can not be anticipated in present computer systems. In the paper [29], a prefetch algorithm is developed for periodic applications. The existing challenges and possible supporting techniques for scripted applications are also discussed.

Anderson et al describe the design of a continuous media file system (CMFS) for digital audio and video that supports multiple concurrent real-time sessions. Their proposal includes a model for bounding access times and allocating the resources for the throughput requirements of each session. A request for a session to read (or write) a real-time file is given an acceptance test to determine if its maximum throughput and minimum buffer requirements can be met concurrently with the previously guaranteed sessions. Each real-time file in the CFMS is created with a maximum rate parameter that constrains the size and frequency of seek times that may be incurred in sequentially accessing the file. The disk-scheduling algorithm reads (or writes) enough data for each session during a cycle to make sure that none of the readers (or writers) run out of data

between cycles. The CFMS algorithm is developed by assuming that the data on the disk is stored contiguously.

A general theoretical framework for studying the performance of an audio playback system with a dedicated storage device is given in [24]. By comparing the data consumption rate with the data production rate of a dedicated storage device, the buffer requirements of single channel playback is analyzed. the data production rate depends on the device characteristics, the data placement on the device, the compression rate achieved at any point in the playback stream (if employed), in addition to the number of samples recorded per second, and the granularity of the sample. To realize a multi-channel playback, a free time model and a channel model is proposed. In the free time model, free time of the storage device in each reading period is used to allow different channels to share the storage device. In the channel model, channels (i.e. part of the storage device bandwidth) are reserved for the playback using.

Based on their own data storage model [33], Rangan et al proposed their algorithm for supporting the multiple HDTV (high density TV) subscribers. In their paper[13], they study various policies(e.g. round robin and quality proportional) for servicing multiple subscribers simultaneously, and propose algorithms by which a HDTV server can enforce these policies without violating the real-time retrieval rates of any of the subscribers. The quality proportional algorithm retrieves video frames at a rate proportional on an average to the HDTV playback rates of requests, but uses a staggered toggling technique by which successive numbers of retrieval frames are fine tuned individually to achieve the servicing of an optimal number of subscribes simultaneously.

In the above, we have briefly reviewed different approaches for the retrieval of multiple continuous streams. Based on our storage model introduced in chapter 3, a retrieval control scheme is studied in chapter 4.

2.5 Summary

In this chapter, we defined the term "multimedia". Then, we showed the general requirements a multimedia system must respect. We also analyzed the requirements of a multimedia file system, and gave the limitation of our thesis research. Finally, we have reviewed existing storage and retrieval technologies for multimedia. In the following two chapters, we present our storage and retrieval model for multimedia information.

Chapter 3

A Storage Model for Continuous and Discrete Media

3.1 Introduction

We recall from chapter 2 that a multimedia system requires to have a new physical data structure for their efficient storage and retrieval. This data structure must meet the real-time deadlines required for the rendition of the media (e.g. audio and video). It should also provide facilities for the handling of synchronization between two or more media.

A few research centers and universities have been working towards the area of design and implementation of multimedia file systems. The central issue investigated in these laboratories was the design of a proper storage strategy for the delay-sensitive media (e.g. digital audio, animation, and video). Of particular interest is the storage model developed by Rangan and Vin [12], [13], [33]. In this model, recall section 2.4.2, a storage pattern can be obtained for each media stream. In addition, the model provides a merging and layout mechanism for multiple streams. However, the merging method used is independent of the storage pattern causing violation in the real-time requirement for the storage and retrieval of the streams from a disk. In order to guarantee such a requirement, read-

ahead and buffering techniques, for a finite number of blocks, have been introduced. This requires not only a large buffer, but also demands complex computations to be performed during a retrieval process. The work reported by Genmmell and Christodoulakis in [24] describes and compares different data placement strategies on the disk for audio data streams. Continuous retrieval from the disk is achieved by means of buffering techniques. The relationship between the display start-time of the media and the required buffer size is also analyzed. However, the paper does not give details about the data structure employed to store audio and video streams. Also, it is not clear from the discussion how the model can handle the continuity requirement for the storage.

In this chapter, we propose an approach which is oriented toward the definition of a complete multimedia data structure that can satisfy the following main requirements.

1. Multimedia data structure: the proposed data structure should permit storage, manipulation and retrieval of both discrete and continuous media.

2. Media synchronization: Media synchronization refers to two situations. First, a single time-based media is processed with respect to time (i.e. continuous media) and second, two or more media are processed in parallel with respect to either time or one of the media types [46], [47], [48]. The storage model should handle these two situations for both files and records.

In the following section, we present our storage model and associated data structure, and in section 3.3, we show how the real-time requirements of continuous media can affect the storage patterns. A novel merging and layout algorithm to manipulate media blocks and gaps in a disk is proposed in section 3.4 and discussions on how new streams are inserted in the available gaps are presented in the section 3.5. Our indexing method and the file structure with the synchronization strategies is defined in section 3.6 while the analysis on the storage efficiency of the proposed algorithm using computer is presented in section 3.7. Finally, we summary our work in section 3.8.

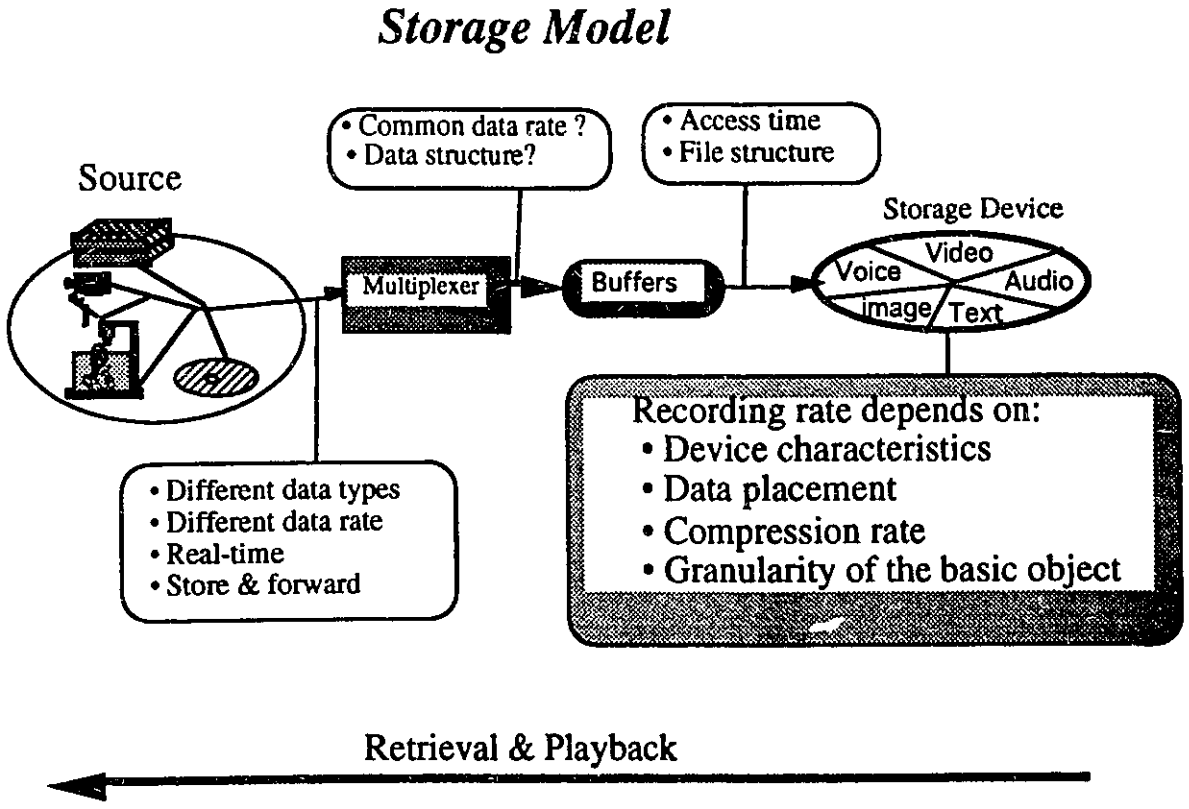


Figure 3.1: Storage and Retrieval Model for Multimedia Information

3.2 MEDIAFILE Structure

We are interested in designing a complete file structure for the management of multimedia information. The objective of the storage model described in this chapter is, first to optimize the disk utilization, second to minimize the amount of data reallocated during data manipulation, third to reduce the buffer requirements during retrieval and fourth to support user defined synchronization among media.

Figure 3.1 shows our storage model. Different types of data arriving at the multiplexor at different data rates are combined into one data stream with a common data rate to be stored in a disk. Different data streams can be stored separately from each other. For example, video data may be stored separately from audio data. Similarly text and image may be interleaved with either audio or video, or stored on a separate location on the disk. Synchronization information which ties different kinds of data together is stored in the file and record attributes. A special storage pattern which consists of media blocks with gaps in between them is designed for the storage of audio and video media. Merging and layout algorithms are also defined to provide an efficient use of the storage space. That is, when merging different streams, media blocks of a new stream are uniformly filled (interleaved) into the existing gaps on the disk. This is achieved by applying merging conditions and the layout algorithms to the stream being stored. The storage pattern with the merging and layout algorithms can provide an efficient use of the disk space and guarantee continuous storage and retrieval of each media blocks of the streams.

3.3 A Storage Pattern for Continuous Media Streams

3.3.1 Parameters, Notations and Assumptions.

In the following we define the parameters, notations and assumptions that are used in this chapter:

- Media stream: a sequence of audio samples or video frames that is captured or displayed in real-time.
- physical sector: one physical sector can occupy from 512 bytes up to 4096 bytes.
- Media block size(M): number of physical sectors occupied by a media block. Logically, M is the basic object of a continuous media stream.

- Gap size(G): number of nonfilled physical sectors between two successive media blocks on the disk.
- G_{max}^{new}: the maximum number of physical sectors between two successive media blocks of a new media stream.
- Storage pattern: (M, Gmm). On the disk, a media stream is stored as a sequence of media blocks; every two successive media blocks are separated by a number of physical sectors(G_{mm}) which can be filled by media blocks of other stream or left nonfilled.
- R_{dt}: data transfer rate of the storage device.
- R_c: data consumption rate, i.e. the rate of data being emptied from buffer.
- R_i: data acquisition rate, i.e. the speed at which data is being captured.

We assume the followings:

- Data transfer rate is much faster than data consumption rate.
- Media block size and gap size are integers.
- The processing (reading, writing, displaying, and recording) is carried out in a pipelined fashion. Figure 3.2 shows an example.

3.3.2 Real-time Conditions for Media Streams.

To record a video or audio stream, the real-time condition requires that all the incoming data must be stored. This means that the data transfer rate between the buffer and the disk is fast enough (with limited buffering) to guarantee a loss-less data storage. To retrieve a video or audio stream, the real-time condition requires the data block to be available at the display buffer before or at the time of its playback. The details for the

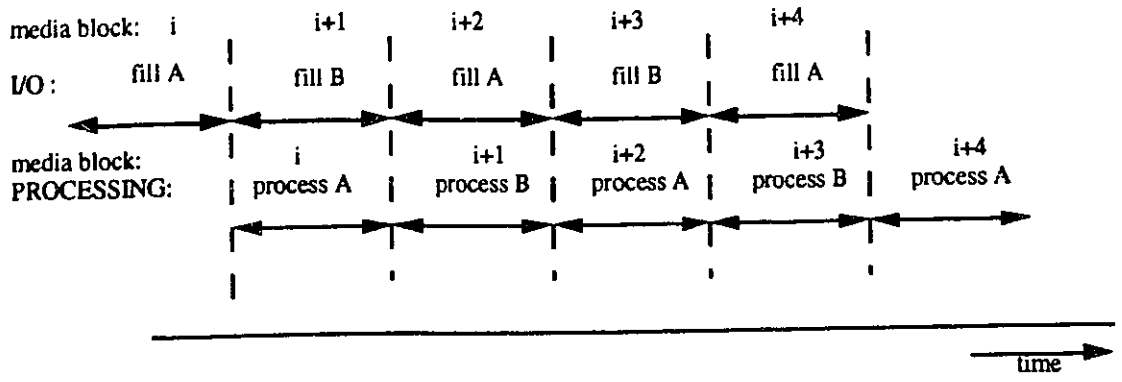


Figure 3.2: Pipelined Processing

real-time requirements are given below.

Since we are assured that the data transfer rate is faster than data consumption rate, to avoid information loss, the recording or retrieval can be carried out in a pipelined fashion.

The recording process is shown in figure 2. While data is being transferred from buffer A to the disk, buffer B is filled by data acquired from the device. If the buffer size is equal to the media block size, the real-time condition can be expressed as follows:

$$\frac{G+M}{R_{dt}} \leq \frac{M}{R_i} \tag{3.1}$$

where $\frac{G+M}{R_{dt}}$ is the time required to locate the address and transfer the data from buffer to disk. $\frac{M}{R_i}$ is the time to fill a buffer of media block size.

Note here that M in this equation refers to any media blocks of a data stream except the first block. When the first media block is stored or retrieved, it may take longer to locate the first block at the beginning of a stream because of the seek time and rotational

latency incurred when a request to write or read on a disk is performed. Once the first block is located, the time to locate the following media blocks depends only on the rotation time.

To retrieve a stream, the real-time condition is almost the same as the recording. It can be expressed as:

$$\frac{M+G}{R_{dt}} \leq \frac{M}{R_c} \tag{3.2}$$

where $\frac{M}{R_c}$ is the time to consume all the data in a buffer.

The above real-time conditions are one of the critical factors in designing the storage patterns. In the following section, we will discuss the storage pattern of the media streams.

3.3.3 Media Block Size and Gap Size

For a given system, data transfer rate (R_{dt}) is known, but data acquisition rate(R_i) or consumption rate (R_c) may vary from stream to stream depending on the quality of the media. We also need to determine the Media block size M and the gap size G_{mm}. For a given stream(R_c and R_i are given), if we assign a value M to media block size, the maximum value of gap G_{max}^{new} can be derived from equation 3.1 or equation 3.2.

Media block size

Usually, media block size(M) is chosen according to the application needs. For video, it can be chosen equal to the size of a video frame, or equal to the size of a given packet which is a group of video frames or a group of video pixels. Once the media block size is known, it becomes the basic object of this media. Here, we do use the same media block size for all the data streams of a media type, even if the data acquisition rate or

the data consumption rate of these streams are different. In terms of storage pattern, the only difference among these streams is the gap size(G_{mm}) which is affected by different data rate. This makes it possible to guarantee the real-time requirements of each media stream at the recording level.

Multimedia applications frequently require that separate data streams such as audio and video be synchronized during their transmission or display. This means that information about the relationships among the media must be stored on the disk. At the storage level, if the media block size of the video streams is chosen first, then depending on its value, we can choose the media block size of audio streams, such that the time necessary to display an audio block is equal to the time necessary to display a video block. Thus, the synchronization control between audio and video can be simplified during storage, retrieval, transmission and display.

Gap Size

The tradeoff between effective storage and effective data manipulation is reflected in the selection of the gap size of a stream. A gap G_{mm} of a data stream can be zero and up to G_{max}^{new}. Zero means that the media blocks of a data stream are stored contiguously without gaps. If a gap size is larger than zero, then the media blocks of a data stream are scattered on the disk. The continuous placement requires much less seek time during retrieval than scattered placement. However, contiguous placement does not support efficient data manipulation. For example, in cartoon movie creation, new pictures need to be added into the display sequence once a while. In case of contiguous placement, adding a new media block into a data stream may require a complete reorganization of the data. This is because pointers to link media block inserted together cannot be used as in the case of traditional text data. The real-time requirements of each data stream have to be considered during the insertion process. Therefore, if we leave space between two successive media blocks, we may optimize the amount of data copied during insertion.

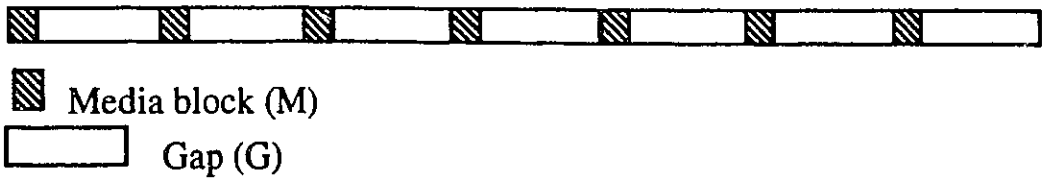


Figure 3.3: Example of Storage Pattern

The maximum gap size obtained from real-time conditions indicates that the real-time requirements of data streams can be guaranteed if the gap between two successive media blocks of a stream is less than the maximum gap size. Figure 3.3 shows an example of storage pattern.

As shown in Figure 3.3 , if the disk space is unlimited, each data stream can be stored using scattered placement and each two successive media blocks of a data stream can be separated by G_{max}^{new}. However, in practice, too many existing nonfilled gaps between media blocks is not suitable for an efficient use of the disk space. To reduce the nonfilled gaps, media blocks with the same data type are interleaved with each other on the disk. Therefore, the size of the gap(G_{mm}) between two successive media blocks for a given stream depends on the quality of media (i.e. the real-time requirement) and the layout algorithm which is used to determine the interleave method. This will be discussed in the next section.

3.4 Merging and Layout Algorithm.

To improve disk utilization, media streams of the same type (e.g. video) are merged during their storage. The merging process is controlled by a layout algorithm, so that, each time after merging, media blocks and nonfilled gaps are uniformly distributed on the

disk and the real-time requirements of all streams are satisfied at the recording level.

3.4.1 Merging Conditions

For a given M, the maximum gap size of a new stream is determined by the real-time condition that can be obtained from equation 3.1:

$$G_{max}^{new} = (\frac{R_{dt}}{R_i} - 1) \times M \tag{3.3}$$

Because disk space is limited, we can not guarantee that a data stream be stored using storage pattern, (M, G_{max}^{new}). It may have to be merged into the nonfilled gaps of existing streams. Therefore, the gap(G_{mm}) between two successive media blocks of a data stream depends not only on its real-time conditions, but also on the merging method. The size of non-filled gap(G) of two successive media blocks after merging is also determined by the merging method.

Obviously, a data block of a new stream S_{new} (M, G_{max}^{new}) can be merged into a gap of stream S(M, G) which already exist on the disk, if the following basic condition is true:

$G \geq M$

The above inequality means that the existing gap size must be larger than or equal to the media block size. But this condition is not enough to determine if all the media blocks of the new stream S_{new} can be merged into the existing gaps of S with respect to the real-time requirements. The reason is discussed below.

To layout media blocks of a new stream, we can simply fill out the first available gap with data blocks from the new stream, then fill out the second gap, the third gap etc.. Thus, after merging, there is no nonfilled gap between media blocks. Although the real-time requirements can be satisfied, extensive data reorganization is needed whenever

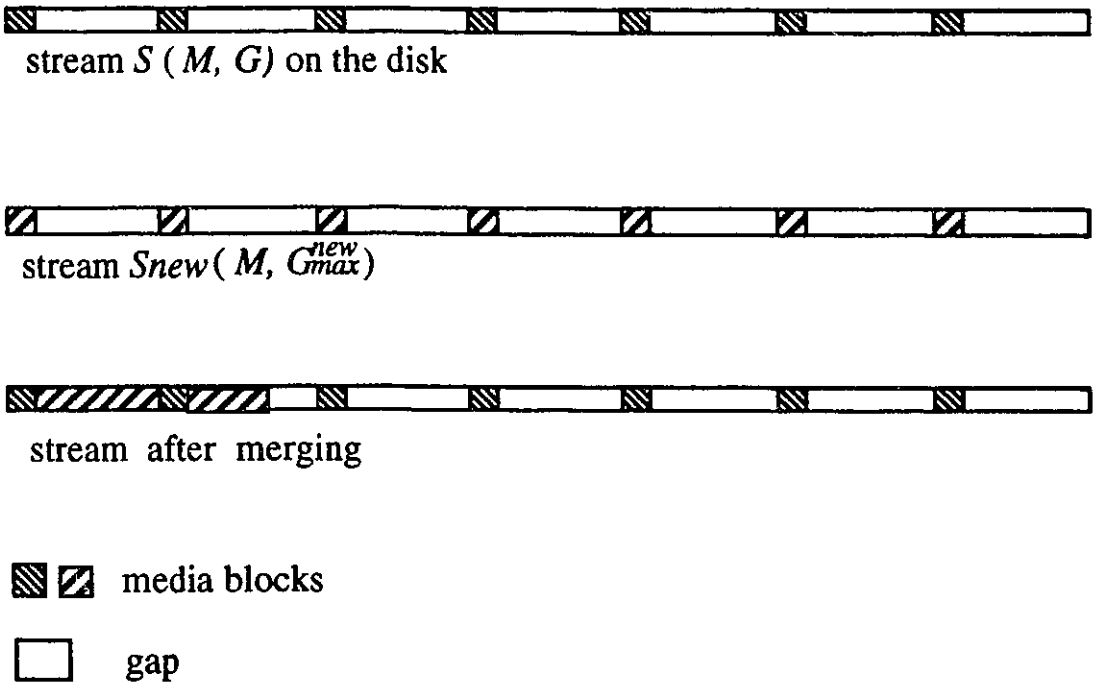


Figure 3.4: Simple Merging and Layout Method

new data blocks are inserted. This scenario is shown in Figure 3.4.

To provide a flexible data management, we choose to have the largest nonfilled gap between the merged and the existing media blocks. After each merging, the resulting non-filled gaps can be used during the next merging process. Thus, the two successive media blocks of a data stream are separated by nonfilled gaps and media blocks from other data streams. In this method, we may only need to reallocate a few merged streams when new media blocks are inserted. Thus, the layout algorithms are used to organize data streams on the disk. In the following section, we describe the layout algorithms in greater detail.

3.4.2 Layout Algorithm

The main objective of the layout algorithm is to provide a uniform distribution of the media blocks on the disk after each merging. It also provides a set of nonfilled gap allowing new streams to be stored according to the real-time constraint. The uniform distribution is achieved by storing the media blocks of a new stream at the center of the existing nonfilled gap, so that the gap is split into approximately two equal gaps. This is called the central merging method.

To store all the media blocks of a new data stream on the disk, we need to know the maximum gap size of the new stream(G_{max}^{new}) and the available nonfilled gap size of the disk(G). From equation 3.3, we see that G_{max}^{new} depends on R_i or R_c. In the following, we will use video stream retrieval as an example to discuss the layout algorithm. The same algorithm can be used for audio or voice.

In our example, we assume that stream S is stored as media blocks and gaps interleaved according to storage pattern (M, G) of the stream, as shown in Figure 3.5. G is the nonfilled gap size of the stream and M is the same for all video streams. A new stream S_{new} with the same media block size M and maximum gap size G_{max}^{new} is going to be merged with stream S. We assume that the merging condition is satisfied under all circumstances. In order to manage the nonfilled gap on the disk, we keep information of nonfilled gaps in a special file called gap list.

We will discuss the algorithm under three scenarios:

1) G_{max}^{new} of the new stream is equal to G of the disk: $G = G_{max}^{new}$

2) G_{max}^{new} of the new stream is larger than G of the disk: $G < G_{max}^{new}$

3) G_{max}^{new} of the new stream is less than G of the disk: $G > G_{max}^{new}$

Let us now discuss each scenario in detail.

When searching for nonfilled gaps on a disk for a new data stream, G_{max}^{new} of the new

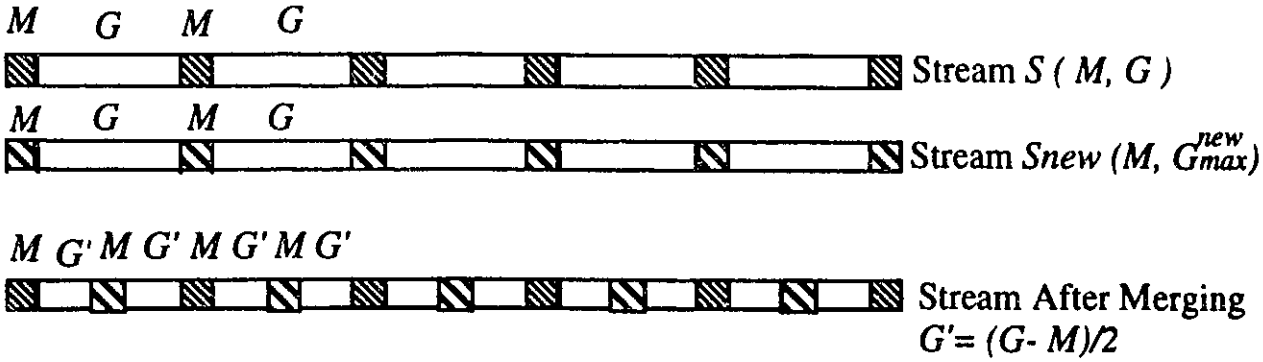


Figure 3.5: Layout Algorithm: the first scenario

stream is compared with G of an existing stream which is the size of the non-filled gap. If $G = G_{max}^{new}$, media blocks of S_{new} are placed into the center of nonfilled gaps one by one and the nonfilled gap size becomes G' after merging:

$$G' = \frac{G - M}{2} \tag{3.4}$$

Figure 3.5 depicts the first scenario. Then, we let $G = G'$ and update the nonfilled gap list.

If $G_{max}^{new} > G$, since G_{max}^{new} is the upper bound of the gap of this new stream, we can always choose a value between zero and the upper bound to make merging possible and easier. The simplest way to achieve the merging is to let $G_{mm}^{new} = G$, making it similar to the first scenario. An example of the second scenario is given in Figure 3.6.

To control the storage process under the above two scenarios, we initialize the storage pointer before merging with the address of the first available nonfilled gap in the disk. The first media block of the new data stream S_{new} is stored from the location given by:

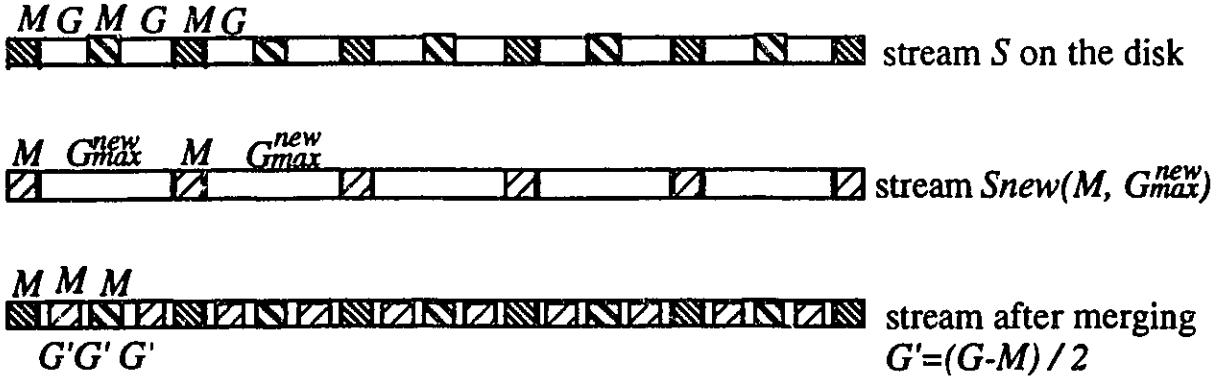


Figure 3.6: Layout Algorithm: the second scenario

$$\text{storage starting position} = \text{the pointer address} + \frac{G - M}{2} \tag{3.5}$$

Then the pointer is moved for G sectors to the second storage starting position and the second media block of S_{new} is stored. This process is repeated until all the media blocks of S_{new} are stored on the disk(see figure 3.7).

Under the third scenario, when the gap size G of stream S is larger than the maximum gap size of a new data stream G_{max}^{new}, more than one media blocks of S_{new} must be placed into one G. This will guarantee that the real-time requirement of S_{new} is satisfied. To obtain the maximum nonfilled gap size after merging, we have to consider the worst case after the media block is stored. The worst case occurs when two media blocks of the new data stream are separated by another media block. So, to avoid collision between the old and the new blocks, the size of maximum gap of S_{new} must be larger than the size of the media block.

One possible size of nonfilled gap(G'') after merging is

$$G'' = \frac{G_{max}^{new} - M}{2}$$

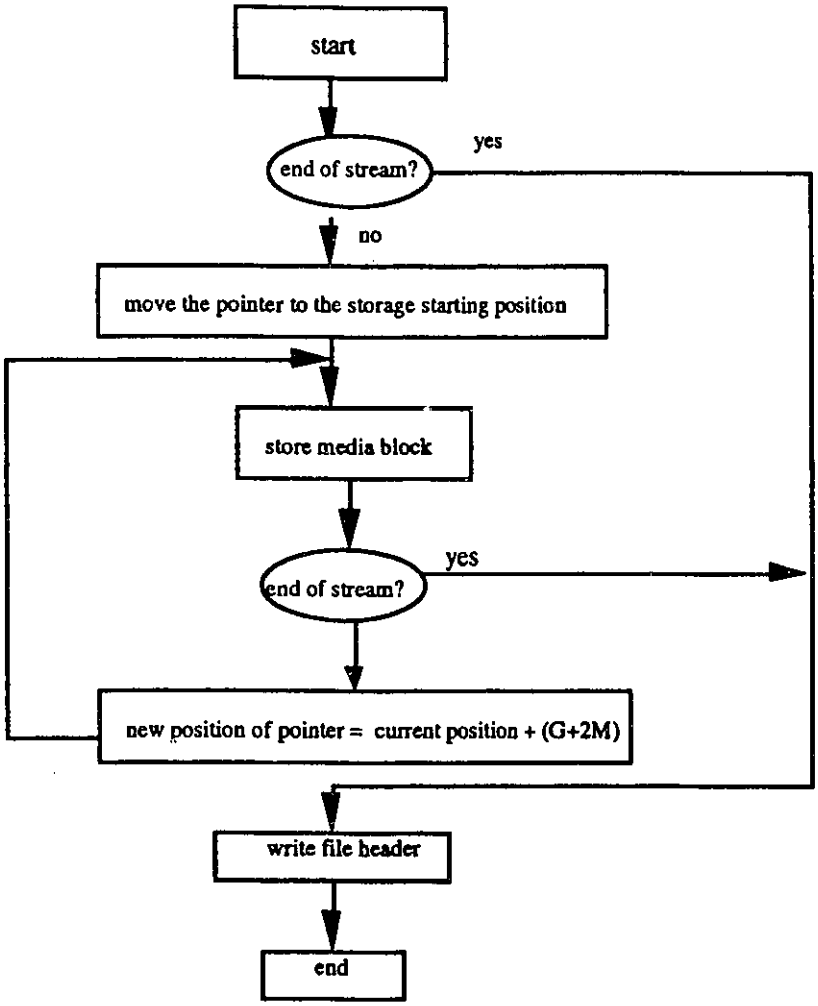


Figure 3.7: A Flow Chart of Layout Algorithm under either first or second scenario

If $G" \leq M$, which means that the nonfilled gaps after merging is not large enough to be used by another new stream, then S_{new} can be stored into the nonfilled gaps of S by using contiguous placement method.

If $G" > M$, we would like to control the allocation of media blocks of S_{new}, so that the nonfilled gaps are uniformly distributed after merging. Now let us first get the number(N) of media blocks that can be stored in one G. Considering the worst case, we have:

$$2 * (\frac{G_{max}^{new} - M}{2}) + (N-1) * (\frac{G_{max}^{new} - M}{2}) + N * M = G$$

Where $2 * (\frac{G_{max}^{new} - M}{2})$ indicates the gaps at the ends, $(N-1) * (\frac{G_{max}^{new} - M}{2})$ is the gaps in the middle, and $N * M$ is the total merged media blocks. From the above equation, we obtain:

$$N = \frac{G - \frac{G_{max}^{new} - M}{2}}{M + \frac{G_{max}^{new} - M}{2}} \tag{3.6}$$

If N is not an integer, it need to be rounded up.

Now that we know the number of media blocks that can be filled into one gap, we can get the gap size G' of the disk between two merged media blocks of the new stream:

$$G' = \frac{G - N * M}{N + 1} \tag{3.7}$$

If G' is not an integer, we keep only the integer part. When we allocate media blocks of S_{new}, G' may be rounded down or rounded up by one sector. We apply this method to all the media blocks of data stream S_{new}. Figure 3.8 shows an example of this scenario. In the example, the size of media block is $3sectors$, the gap size of the disk is $25sectors$. The new stream with maximum gap size $13sectors$ requests to be stored. Using our layout algorithm two, $G"$ equals to $6sectors$ which is larger than M. Thus, we

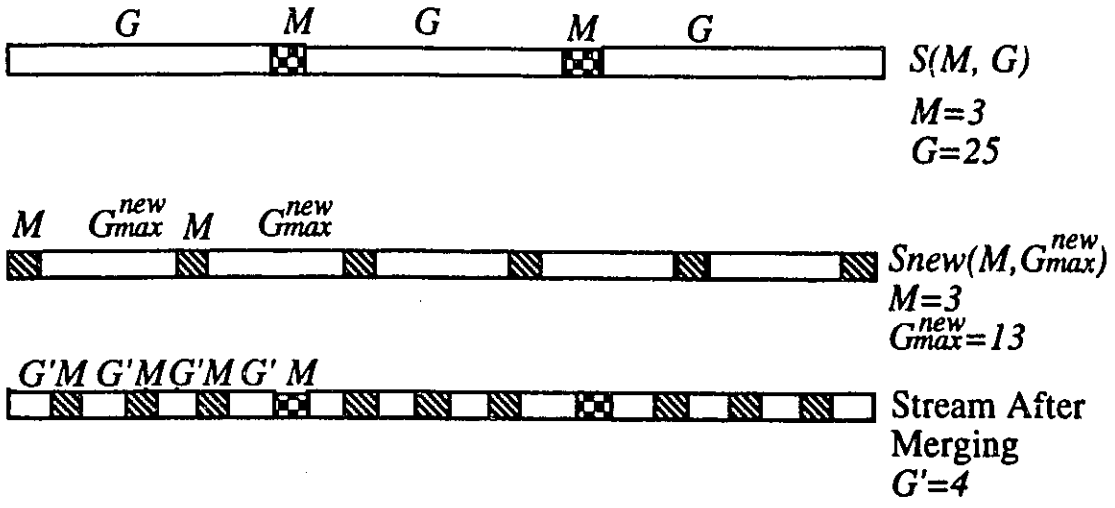


Figure 3.8: Layout Algorithm: the third scenario

get N equals to $3sectors$ by round up the result of equation 3.6. From equation 3.7, we get G' equals to $4sectors$. Therefore, the new G is $4sectors$ after merging, and gaps are still uniformly distributed on the disk.

Using the above layout algorithms, we can successfully merge any data stream on the disk. The storage efficiency of the proposed algorithm will be analyzed using computer simulations in section 3.6.

3.5 Discussions

3.5.1 Effect of Compression

Audio and video data require large disk space for their storage. As a result, techniques to minimize storage space on the disk (e.g. compression/decompression) are considered important and the compression ratios of different data streams have very significant effects

on the storage requirement. For example, one second of NTSC video requires several megabytes of storage space, but after compression, it may only need several hundreds kilobytes of storage space. There are several compression schemes, that give different compression ratios, and different video or audio quality. Standards such as JPEG [26] and MPEG [27] can be used to compress still images and motion pictures with sound information, respectively.

In our storage model, a compression device may be connected between the sensor and the multiplexor in figure 1 and the decompression device may be connected between the storage device and the display workstations. The MEDIAFILE requires an efficient compression scheme to reduce the storage space. However MEDIAFIALE does not need to know the details about the compression scheme used. To store a compressed media stream, we need only to know the data rate obtained after compression.

The process of a data stream going through the MEDIAFILE system can be seen in a manner similar to the data encapsulation and data decapsulation process. After compression, the "code.header" is added to the data object. When this data object is stored on a disk, the "disk.header" is added to it. In the retrieval process, these headers are decapsulated and only original data is sent for display. Figure 3.9 gives a simple example.

3.5.2 Adding and Removing Media Blocks

Adding and removing operations can be applied to the media blocks or at a higher level to a group of media blocks. To remove a media block from a data stream, the following media blocks in the stream are shifted by one position for each deleted block on the disk.

To add a media block to a data stream, media blocks on the disk must be moved ahead in order to meet the real-time deadlines of all the streams. Media blocks of a given data stream may be stored contiguously. Different media blocks pertaining to different

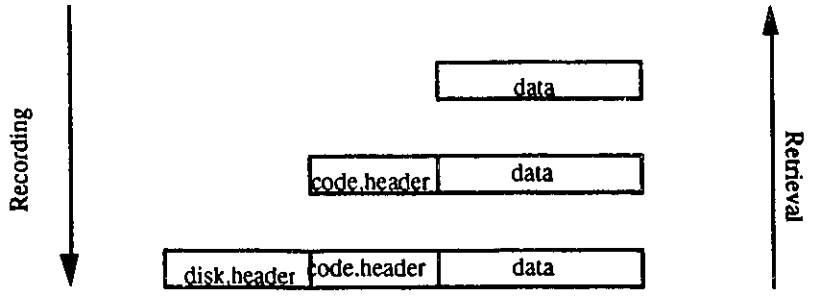


Figure 3.9: Data Encapsulation and Decapsulation

streams may also be stored contiguously. In the former case, to insert media blocks into one stream, almost all the data streams have to be rearranged on the disk. However, in the latter case, when we want to insert media blocks in a data stream, we only need to move one or two merged streams instead of reorganizing all the data streams on the disk. Therefore, by using the merging method and layout algorithms, the amount of data moved during an insertion is reduced.

3.6 File Structure and Index File

In this section, a file structure is defined to store and synchronize various media on the disk. An index file structure is also introduced to permit the random access of data.

One of the primary use of MEDIAFILE in our laboratory is to store and retrieve multimedia documents. The architecture of such documents is described in [25]. Five main objects may be defined within a multimedia document. They are independent object blocks, sequential object blocks, concurrent object blocks, media and scene layer objects. From the application point of view, a user can retrieve a multimedia document or an independent media file directly. The user can also retrieve parts of existing docu-

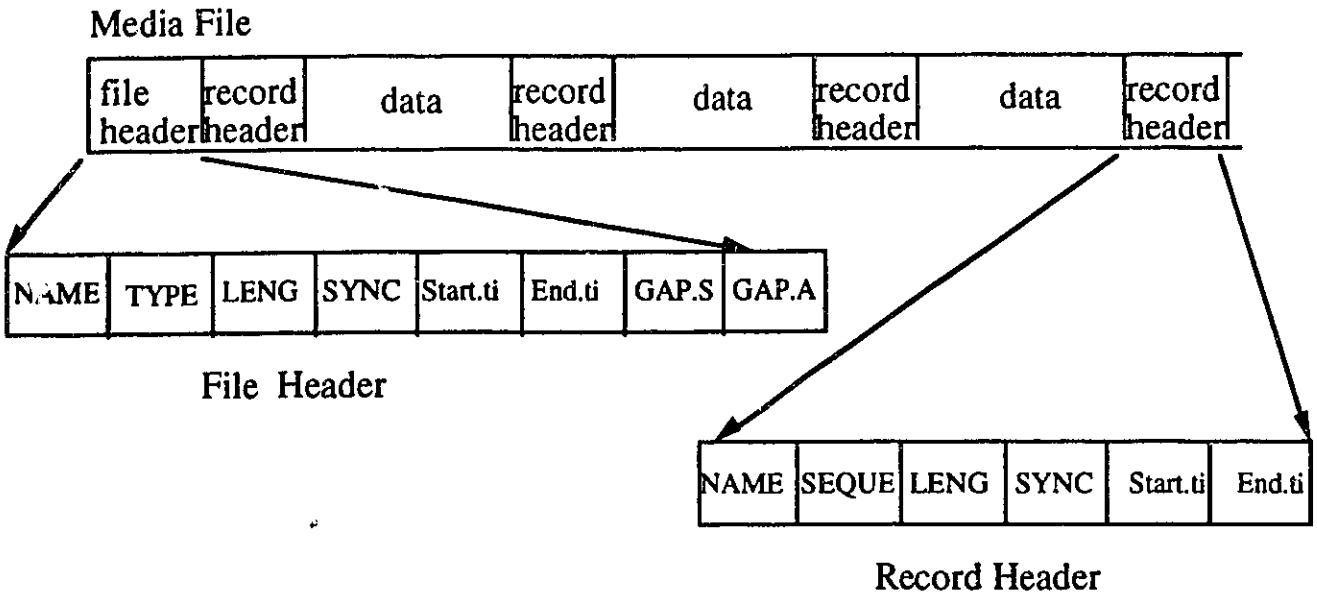


Figure 3.10: File Structure

ments or files to create new multimedia documents. In a newly created document, the user specifies the interrelationship among media files. The MEDIAFILE should keep all the synchronization information, and be able to arrange data on the disk to meet the user requirements.

Figure 3.10 describes the structure of a MEDIAFILE. Each file contains a file header and a sequence of records. Each record contains its own header. The meaning of each item is explained in the following.

In a file header:

- NAME: the name or ID of the file. It is used to index each file.
- TYPE: the media type of a file. This gives the information if the file has a real-time requirement.
- LENG: the total length of the file.
- SYNC: names or ID of all the related files.

- Start.ti: the display starting time of the file. It can be used to synchronize media files at the start point.
- End.ti: the display ending time of the file. It can be used to synchronize media files at the end point.
- GAP.S: the nonfilled gap size of the file.
- GAP.A: the starting address of the first nonfilled gap of the file.

In a record header:

- NAME: the name or ID of the record.
- SEQUE: the sequence number of this record in the file.
- LENG: the total length of the record.
- SYNC: sequence numbers of the related records of other file.
- Start.ti: the display starting time of the record. It can be used to synchronize media records at the start point.
- End.ti: the display ending time of the record. It can be used to synchronize media records at the end point.

Using this structure, we can define a file structure for both continuous and discrete media. When the user creates a multimedia document composed of discrete media, we need only to write or modify the header of the file or the data unit. However, for audio and video, we may also need to rearrange part of the media streams or data units to satisfy real-time conditions.

In a given file structure, two levels of synchronization are defined: at the file level and the record level. At the file level, information such as SYNC., Start.ti and End.ti(see figure 3.10) is stored in the file header and used to synchronize media files. At the record

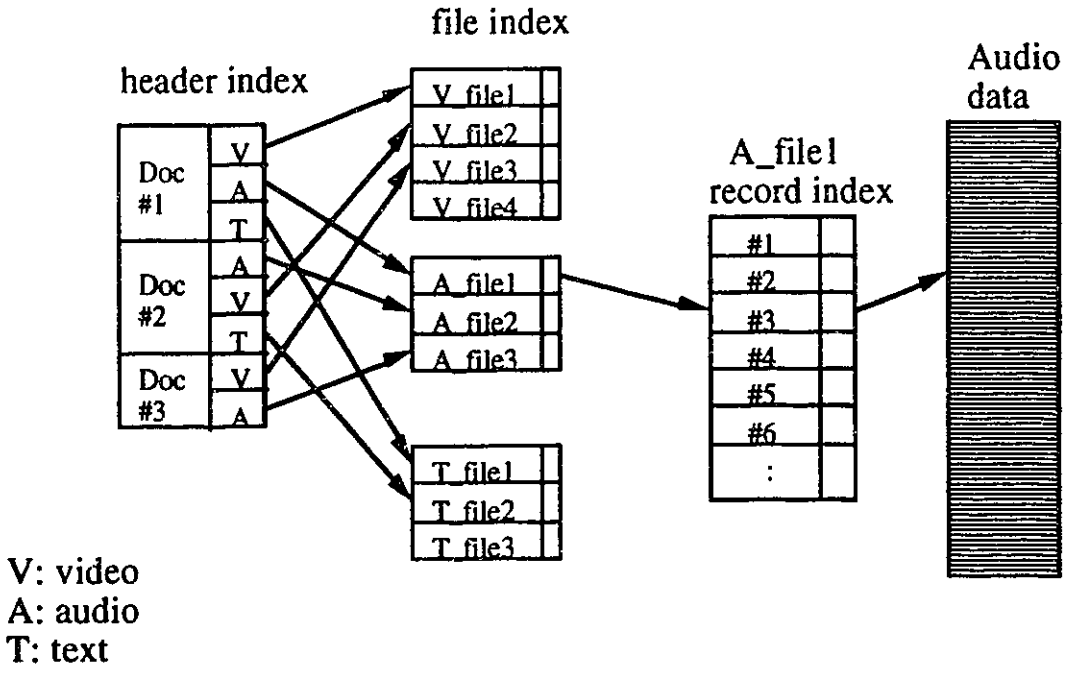


Figure 3.11: Index Structure

level, synchronization information located in the record header is used to synchronize media records.

A multilevel index structure is defined to organize all the information on the disk. An example is given in Figure 3.11. A header index includes the names of multimedia documents and the pointers to the related files. At the file index level, each media type has its own index file. Each index file contains headers for the files of the same type. The pointers to the address of the records are also stored in the index file.

3.7 Computer Simulations

The simulations were performed on the computer. The purpose of the simulations is to determine the storage efficiency of the proposed layout algorithm and also to de-

termine whether all the merged data streams meet their respective real-time requirements.

Simulation results show that the algorithm has the best case storage efficiency of 85 % and the worst case of 60 % when the value of G_{max}^{new} of is between M and $5M$. When the value of G_{max}^{new} is greater than $6M$, the storage efficiency is always less than 50 %. The results also show that the merging and layout algorithm is able to maintain the continuity requirement of each data stream when all data streams are merged in a disk.

The details of the computer simulation, the model and results, can be found in our technical report[49]. The simulation is done by Dr. Yeap.

3.8 Summary

In this chapter, we have discussed a storage model for the management of both discrete and continuous media. Merging and layout algorithm for storing continuous data streams has been presented. This algorithm is designed to meet the multimedia requirements of the data streams and to use the disk space efficiently. Computer simulations were performed to evaluate the storage efficiency of the proposed merging and layout algorithm. In addition, methods for integrating different media files on the disk and supporting user defined intermedia synchronization at the storage level were discussed.

Our storage model is designed with respect to the continuous retrieval of a single media stream by using buffers of media block size. However, in a multi-user environment, multiple media streams may need to be retrieved simultaneously. Hence it is desirable to have a retrieval control scheme which can guarantee the continuous retrieval of the multiple media streams. In the next chapter, we present our scheme for supporting the real-time retrieval of multiple streams.

Chapter 4

A Control Scheme for Supporting The Simultaneous Retrieval of Multiple Streams

4.1 Introduction

We recall from chapter 2, multimedia application involves large amount of data that needs to be stored, retrieved, manipulated, processed and exchanged between a group of users. A possible configuration for such applications may be composed of a set of workstations connected to a multimedia server through a high speed network(e.g. FDDI, B-ISDN). User's requests may arrive at the server at the same time, so the file server must have the capability to serve all the requests within a specific deadline (i.e. in real-time). Furthermore, in contrast with traditional applications, multimedia applications usually involve real-time media (e.g. audio and video), and therefore, the file server has to guarantee not only a fast response time, but also the continuous retrieval conditions of the streams. Although some current file servers can provide a fast response and methods for stream retrieval, they can not guarantee continuous retrieval of several media streams simultaneously.

The scenarios of simultaneous retrieval can be different. In the simplest scenario, all users request the retrieval of the same media stream. For example, they subscribe to the same piece of music at the same time. In this case, the server needs only to retrieve the required audio stream from the disk once and then multicast it to all the users. However, in practice, different users may request the retrieval of different streams. Even when the same media stream is being requested by different users, there may be phase differences among their requests (such as each user retrieving a different portion of a stream at the same time). Furthermore, each user may request the retrieval of a multimedia file which contains more than one media stream. The complexity of different retrieval scenarios demand the development of new retrieval control schemes to guarantee various retrieval requirements.

As noted in chapter 2, the bottleneck in the retrieval of multimedia file is the disk I/O bandwidth. The current research trend is to use multiple read/write heads and multiple disk drivers to increase the total bandwidth. There are several papers (see section 2.4.3) in which the problems of simultaneous real-time retrievals are addressed. Most of their studies focus on developing control methods for a storage system that has multiple read/write heads or has multiple disk drivers. However, with the development of hardware technologies, disk I/O bandwidth will improve considerably. This will allow a single disk head storage system to support the retrieval of multiple real-time streams simultaneously. In a multiple disk head system, one can think of a simple scheme that simultaneous retrieving streams are handled by different disk heads (i.e. one stream per disk head). However, this solution will limit the number of streams that can be retrieved simultaneously to the number of disk heads. Another drawback of this method is that it could not use the bandwidth of the disk efficiently. This is because that different streams may have different retrieval rates(i.e. the different bandwidth requirements). It is very important to improve the usage of I/O bandwidth. There are several ways to improve the usage of the disk I/O bandwidth. Our work described in this chapter will focus on the analysis of retrieval process and the control schemes for simultaneous retrieval in a single

disk head system. The storage model which is introduced in the previous chapter is used for our analysis. Although this model is designed for database applications that require editing and updating constantly, it still needs to support the simultaneous retrieval of multiple streams, especially when a multimedia file contains more than one continuous media streams.

The contribution of this chapter lies in the fact that it thoroughly addresses a control scheme using a single disk head storage device to support the real-time retrieval of several streams simultaneously. A complete analysis of the relationship between the number of simultaneous retrieving streams and their affecting factors (e.g. disk characteristic, data placement on the disk, and the buffer requirements) is presented.

In the following section, we analyze the retrieval process of a single media stream, and we present our control scheme for simultaneous retrieval of N streams. In section 4.3, we show how the characteristics of both storage devices and applications can affect the number of simultaneous streams. A novel dynamic access control algorithm is proposed in section 4.4. Discussions of the retrieval start time of a new stream and the synchronization problems during the retrieval are presented in section 4.5. The analysis on the simultaneous real-time retrieval control scheme using computer simulation is presented in section 4.6. Finally, in section 4.7, we draw our conclusions and discuss plans for future work.

4.2 The Retrieval of a Real-Time Stream

4.2.1 Retrieval of a Single Media Stream

As discussed in chapter 3, according to the real-time condition, the continuous retrieval of a single stream can be done by using at least two equal sized buffers between

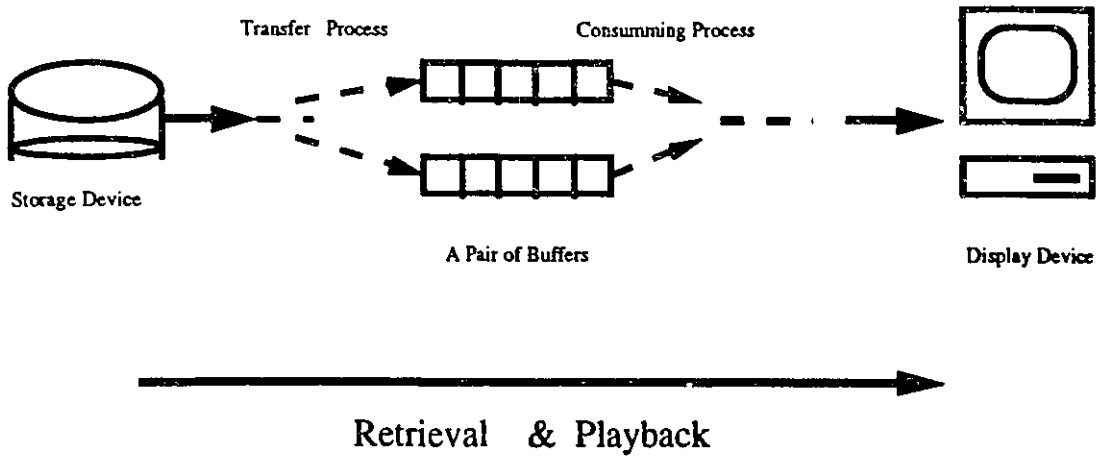


Figure 4.1: Retrieval of One Media Stream

the storage device and the display device. To retrieve a media stream from the disk, a transfer process and a consuming process are working together in a pipelined fashion. The function of the transfer process is to locate a media block on the disk and transfer it to an empty buffer, while the role of the consuming process is to retrieve data from the buffer and send it over the network or for display at a local station. The scenario of the retrieval of a single stream is shown in figure 4.1.

The data transfer rate of the transfer process depends on the characteristics of the storage device. Different devices may have a different data transfer rate. A storage device with a higher data transfer rate will be able to support more real-time retrieval users simultaneously. In our study, we assume that the data transfer rate is fast enough to support the real-time retrieval of more than one continuous streams simultaneously. In the following description, R_{dt} is used to indicate the data transfer rate of a single disk-head storage device.

The buffer consumption rate (denoted by R_c) of a consuming process depends on both the playback rate of the stream (i.e. the display quality) and the size of the retrieval

buffer. To clarify the meaning of the buffer consumption rate, a term called buffer-empty-duration is introduced. The buffer-empty-duration, expressed as T_c, is the time required for a consuming process to empty one of the retrieval buffers. T_c is an application dependent parameter. For example, to retrieve a video stream from the disk to display, buffers with a fixed size are allocated. Obviously, a faster display rate will give a shorter T_c. If we use the size of a buffer (B) divided by T_c, then, the buffer consumption rate (R_c) for that consuming process is obtained. Note that our consumption rate is different from the real data transfer rate between two memory addresses(i.e. from the retrieval buffer to the display buffer in the display station), and the later one is assumed to be much faster.

The main purpose of using buffers is to coordinate the transfer process and the consuming process to avoid discontinuity during the retrieval. Therefore, to guarantee continuous retrieval, data must be in the buffer before, or at the time, it is retrieved by the consuming process. On the other hand, if the transfer rate R_{dt} is faster than the buffer consumption rate (R_c), the control of the transfer process must be applied to avoid data loss which may happen when a buffer overflows.

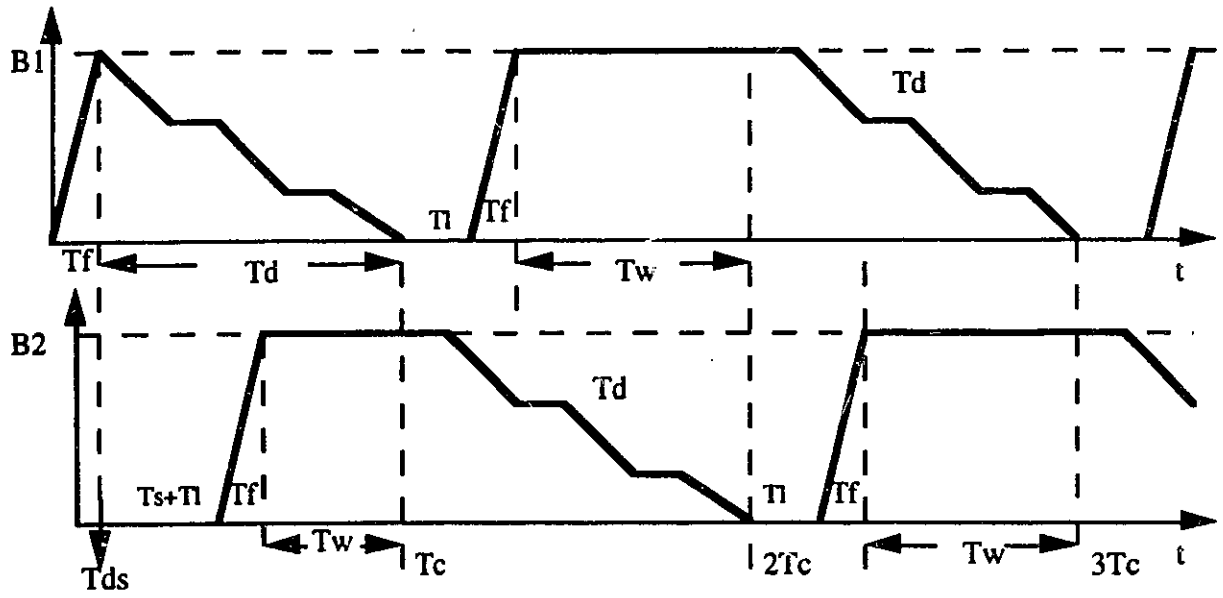
We will now follow a data transfer process to see how data is transferred from the disk to one of the retrieval buffers. In a movable-head disk system, the total time to transfer a data block from the disk to a buffer is the sum of the seek time, rotational latency, and block transfer time. To transfer a data block, the disk drive must first mechanically position the read/write head on the right track. The time it takes to do this is called the seek time. Following that, there is a delay – called rotational latency – until the beginning of the block rotates into the position under the read/write head. Finally, time is needed to transfer the data into a buffer; this is call block transfer time. In our storage model, if only one media stream is being retrieved, there is no seek time for all media blocks but the first one. The rotational latency from one media block to the successive media block is constant for all but the first media block. This is because that our media gaps of one stream are designed to have the same size. A media block may contains several

physical data blocks which are stored contiguously, so the rotational latency among them is eliminated.

Notations T_s, T_l and T_f are used in the following equations. T_s denotes the average seek time when the disk head is switching from one position to another, T_l is the rotational latency during the period that a buffer is filled, and T_f is the total time needed to transfer contiguous media blocks to fill out a buffer completely. Next, we will illustrate the relationships between T_c, R_{dt}, T_s, T_l and T_f during the real-time retrieval of one continuous media stream.

To make our analysis more clear, we assume a pair of buffers is allocated as retrieval buffers for the given stream. The consuming process starts at the time that one of the two buffers is initially filled completely full by the transfer process. Following, the consuming process retrieves the data from this buffer according to the required rate. While the buffer is being emptied by the consuming process, the transfer process will locate and read the following data blocks from the disk and fill the other buffer. When there is no empty buffer, the transfer process stops serving this request until a buffer becomes empty. If there are not other requests to serve before it restarts filling the retrieval buffer, the transfer process will perform an idle delay (i.e. when the disk head is not doing any reading or writing). We call this idle delay the transfer-idle-period(T_w). Since the disk transfer rate is faster than the buffer consumption rate, the transfer-idle-period occurs once every buffer-empty-duration. In figure 4.2, we explore the occupation of two retrieval buffers versus time, so the functions of both the transfer process and the consuming process are depicted.

In figure 4.2, the small sharp slope in each buffer-empty-duration(T_c) denotes to real data transfer rate between one of the retrieval buffer and the display buffer. Instead of showing the details of the consuming process in each buffer-empty-duration, in figure 4.4 and figure 4.5, a single slope is used to indicated the average – the buffer consumption



Tds -- Display Start Point

Tf -- Real Data Transfer Time

Tc -- Buffer Empty Duration

Tw -- Transfer Idle Period

Ts -- Seek Time

Tl -- Rotational Latency

Figure 4.2: Buffer Occupation for Retrieval of One Media Stream

rate. A retrieval of a pipelined fashion is illustrated in figure 4.2. At the time that data in buffer(B_1) is being consumed by the consuming process, the other buffer(B_2) is being filled with data blocks by the transfer process. In each buffer-empty-duration, the transfer process will, first, fill an empty buffer with data blocks from the disk, in which some rotational latency (T_l) is incurred to skip the gaps between two successive media blocks, and second, play an idle delay (T_w) waiting for the other buffer to become completely empty. Since the two retrieval buffers have the same size, T_c is constant and actually is the service cycle. Therefore, in general, T_c can be expressed as the sum of data blocks transfer time - T_f, rotational latency(seek time is only applicable for the first media block) - T_l and the transfer-idle-period - T_w. This relationship is shown in the following equation:

$$T_c = T_f + T_l + T_w \tag{4.1}$$

where

$$T_c = \frac{B_1}{R_c} = \frac{B_2}{R_c}$$

and

$$T_f = \frac{B_1}{R_{dt}} = \frac{B_2}{R_{dt}}$$

where B_1 and B_2 indicate the size of retrieval buffers. We assume that each buffer can contain up to the number of K media blocks(M) :

$$B_1 = B_2 = K * M$$

, so to fill out a buffer, the disk head has to skip $K - 1$ equal sized gaps (G). Thus, the rotational latency in each buffer-empty-duration can be formulated as:

$$T_l = (K-1)\frac{G}{R_{dt}}$$

Therefore, transfer-idle-period per buffer-empty-duration can be derived as:

$$T_w = \frac{B_1}{R_c} - \frac{B_1}{R_{dt}} - (K-1)\frac{G}{R_{dt}} \tag{4.2}$$

We are more interested in the transfer-idle-period (T_w) and the buffer-empty-duration (T_c). During T_w, the transfer process is not using the storage device either to seek or to read. It must be ensured that T_w is actually long enough to do some data retrieval for other tasks. The length of T_c is also important, because it indicates how often T_w becomes available, and, of course, increasing T_w must increase the length of T_c. The length of transfer-idle-period per buffer-empty-duration is the real indicator of how much sharing can occur. It indicates how much time of the storage device is really being freed for use by other tasks. In the following sections, we will discuss the methods which can support the real-time retrieval of multiple streams simultaneously by using the transfer-idle-period in each buffer-empty-duration.

4.2.2 Retrieving N Real-Time Media Streams

The real-time retrieval of N continuous streams simultaneously is done by the transfer process and N consuming processes. The consuming process of different stream is independent from each other and the consumption rate may be different depending on applications. However, the transfer process is shared by N retrieving streams, since all media blocks are disk residence and only one disk head is in serving. To guarantee continuous retrieval of N streams, we assume that a pair of equal sized buffers is assigned to each retrieving stream. The buffer size of each stream can be the same or different depending on its consumption rate. The scenario of simultaneous retrieval of multiple streams is shown in Figure 4.3. In order to explore some important properties, a simplified retrieval model is used for our study.

The model has the following assumptions:

- The storage device is dedicated to only one continuous media type, e.g. video.

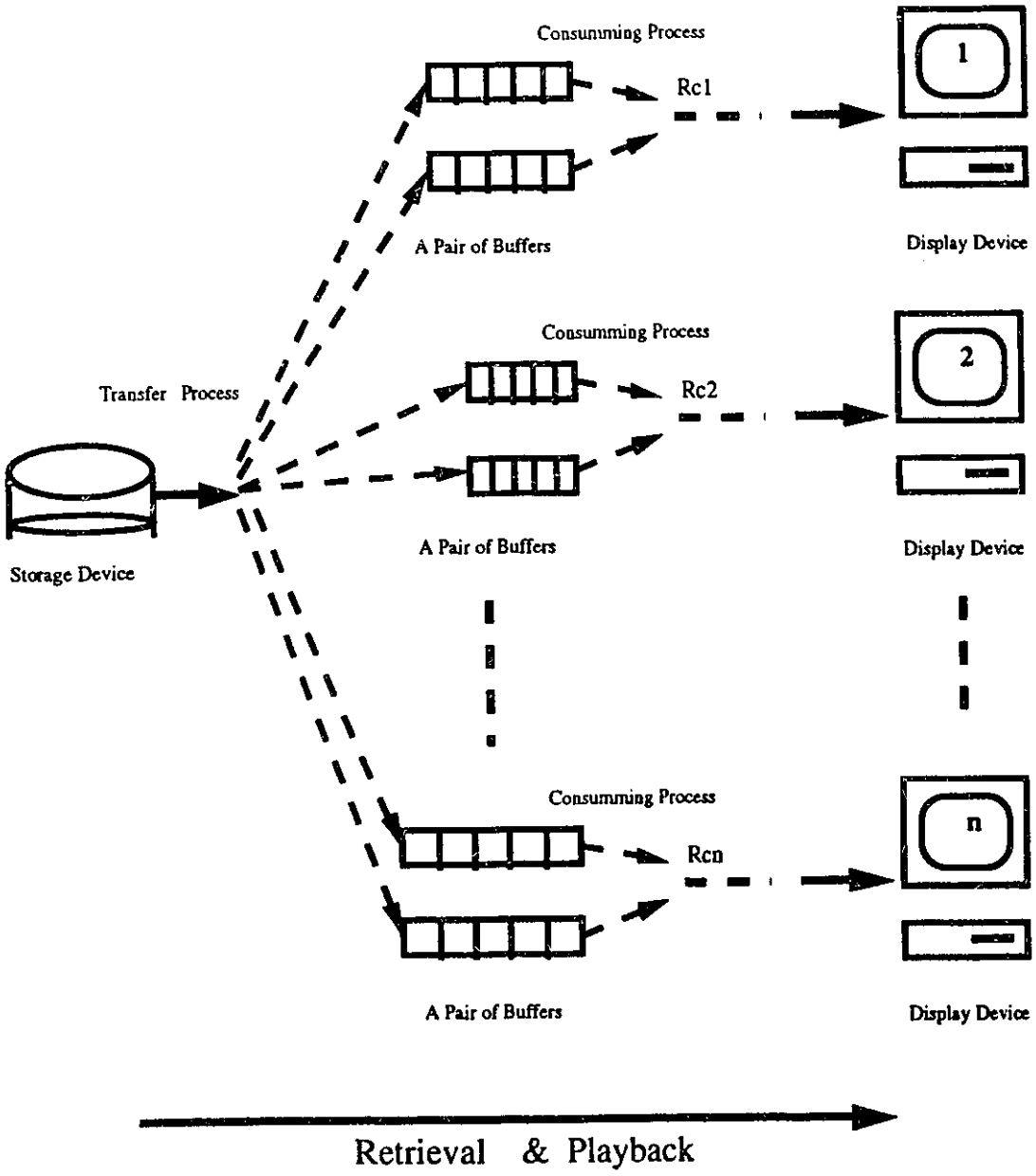


Figure 4.3: Retrieval of Multiple Media Streams

- The number of simultaneous real-time retrieval streams is N.
- The sum of the buffer consumption rates is less than or equal to the data transfer rate of the storage device:

$$\sum_{i=1}^{N} R_{ci} \leq R_{dt}$$

- The average seek time T_s is considered when the disk head switches between the media blocks of two retrieving streams.
- Two equal sized buffers are allocated for each retrieving stream. There are, in total, $2N$ retrieval buffers for N retrieving streams.
- The round robin method is chosen to be the service discipline of the transfer process when multiple streams are being retrieved simultaneously.

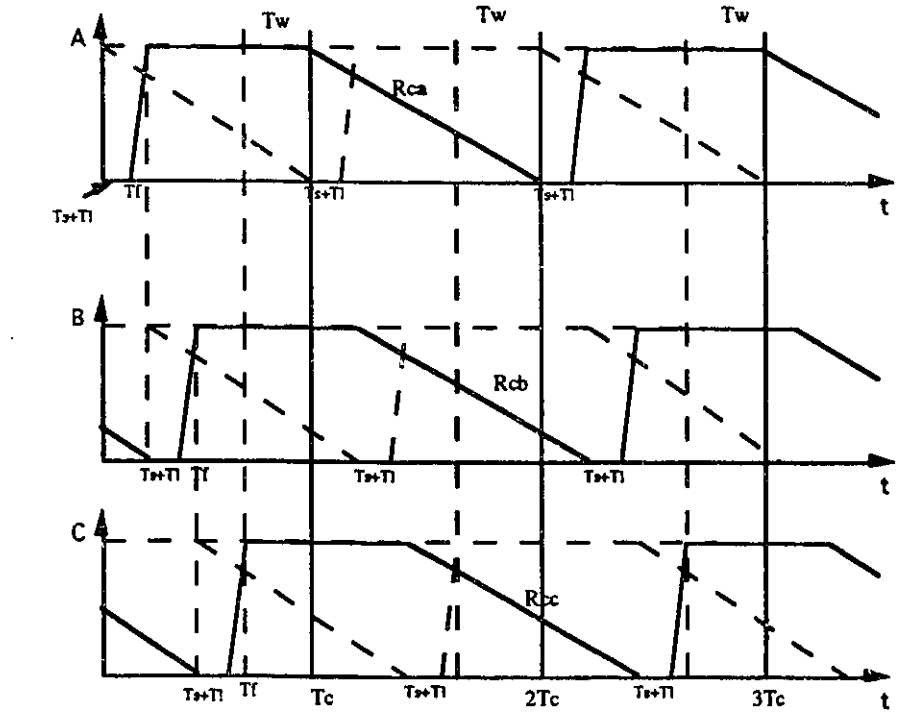
In the following, we will analyze the simultaneous real-time retrieval of multiple streams under two different situations:

- Streams with the same buffer consumption rate.
- Streams with different buffer consumption rate.

1. The Retrieval of Multiple Streams with the Same Buffer Consumption Rates.

In this situation, we assume that the buffer consumption rate of all the retrieving streams are the same, and the rate can be expressed as R_c. In total, a number of $2N$ retrieval buffers which have the same size are allocated for a set of N retrieving streams, and the size of each buffer denoted by B. Each buffer may contain up to a number of K media blocks:

$$B = KM$$



Note:

one of the two buffers for each stream

The other buffer for each stream

Buffer size: A=B=C, Consumption Rate: Rca=Rcb=Rcc.

Buffer Empty Duration: Tc = A/Rca

Figure 4.4: Buffer Occupation for the Retrieval of Three Media Streams with the Same Rate

As we discussed in the last part, in the retrieval of a single stream, to support continuous retrieval, the transfer process has to fill out one buffer, then play an idle delay in each buffer-empty-duration. Now, in order to support continuous retrieval of N streams simultaneously, the transfer process has to fill out N buffers corresponding to N retrieving streams in each buffer-empty-duration.

In figure 4.4, an example is given to show the scenario of the simultaneous retrieval of three streams. We can see that during each T_c, the transfer process has to fill out, all together, three buffers - one for each of the streams A, B, and C. In each buffer filling process, before a media block can be transferred from disk to its retrieval buffer, the transfer process has to first locate this media block on the disk, so the seek time and the rotational latency delay are incurred in this period. During each T_c, since the disk head is shared among the three retrieving streams, the total seek time will be $3T_s$ and total rotational latency will be $3T_l$.

Extending the above analysis to the case when retrieving N continuous streams simultaneously, each T_c can be expressed as:

$$T_c = T_f + NT_l^{avg} + NT_s + T_w \tag{4.3}$$

where $T_c = \frac{B}{R_c}$ and

$$T_f = N\frac{B}{R_{dt}}$$

and

$$B = KM$$

Different streams may have a different storage pattern (i.e. different gap size), so the rotational latency may be different. For a specific stream i, rotational latency - T_l equals to :

$$T_l = (K-1)\frac{G_i}{R_{dt}}$$

To simplify the equation, T_l^{avg} is used to indicate the average rotational latency during the period that a buffer is being filled:

$$T_l^{avg} = \frac{\sum_{i=1}^{N}(K-1)\frac{C_i}{R_{dt}}}{N}$$

If we let L_{max} be the total disk latency when transfer the data from the disk to a buffer in each T_c, equation 4.3 can be simplified further.

If $L_{max} = T_s + T_l^{avg}$ then equation 4.3 can be rewritten as:

$$\frac{B}{R_c} = N\frac{B}{R_{dt}} + NL_{max} + T_w \tag{4.4}$$

From the above equation, the transfer-idle-period (T_w) can be derived as:

$$T_w = (\frac{B}{R_c} - \frac{NB}{R_{dt}}) - NL_{max} \tag{4.5}$$

In equation 4.5, since $R_{dt} \geq NR_c$, so item $(\frac{B}{R_c} - \frac{BN}{R_{dt}}) \geq 0$. The T_w per T_c has to be zero or positive in order to guarantee the continuous retrieval. Thus, we can see that to increase the T_w - the transfer-idle-period, we can either increase B, R_{dt} or decrease R_c, N, and L_{max}.

When B is increased, the buffer-empty-duration as well as the time spent to transfer the data from the disk to its retrieval buffer are increased. The total rotational latency is also increased, since more time is spent to skip more gaps during buffer filling period. Therefore, as a total result, increasing buffer size may not increase T_w effectively.

One of the other ways to increase T_w is to reduce the disk latency (L_{max}). The total latency per T_c - L_{max} can be decreased by minimizing the seek time or by decreasing the rotational latency. The seek time overhead is a dominant concern in the disk latency. It is desirable for the transfer process to performance long operation (buffer more media blocks) for each stream in each service cycle. However, increasing buffer size will bring some side effect which make the total improvement less significant. Decreasing rotational

latency can be achieved through changing the data placement on the disk, for example, using the contiguous data placement method.

Decreasing R_c or increasing R_{dt} will result in the increasing of T_w. R_c is an application related parameter. To decrease R_c means to reduce the application requirements. R_{dt} is a hardware based parameter, and can only be improved by using a fast disk driver.

2. The Retrieval of Multiple Streams with Different Buffer Consumption Rates.

The simultaneous retrieving streams may not always have the same buffer consumption rate. Thus, when the different retrieving streams have different buffer consumption rates, retrieval buffers of different size have to be allocated for the different retrieving streams. The size of a retrieval buffer is determined by the buffer consumption rate of that stream. By relating the retrieval buffer size with its consumption rate, we can obtain the same T_c for all the retrieving streams. Therefore, the control of the real-time retrieval of multiple streams can be simplified.

In figure 4.5, an example of the simultaneous retrieval of two streams with different buffer consumption rates is given. We use solid lines and dotted lines to differentiate the two retrieval buffers of a stream. Since the consumption rate R_{ca} of stream A is twice the consumption rate R_{cb} of stream B, we allocate two buffers with each sized (B_a) for stream A, and two buffers with each sized B_b for stream B. The relationship between buffers of these two streams is $\frac{B_a}{R_{ca}} = \frac{B_b}{R_{cb}}$. Thus, T_c will be the same for both stream A and stream B, and it can be expressed as:

$$T_c = T_{fa} + T_{fb} + 2T_s + T_{la} + T_{lb} + T_w$$

Where $T_{fa} + T_{fb}$ is the time required to transfer data from the disk to fill a buffer for stream A and to fill a buffer for stream B. The total seek time in each T_c is shown as

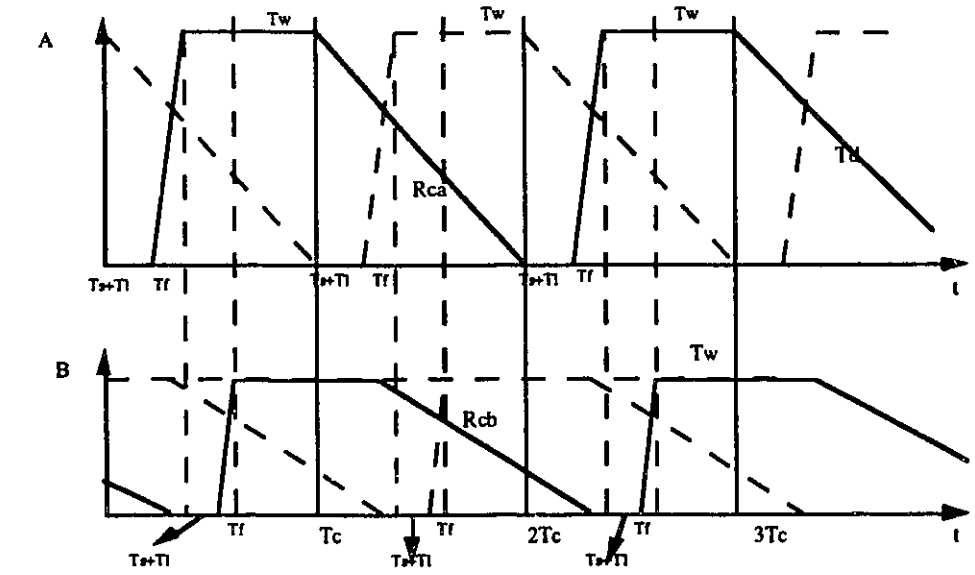


Figure 4.5: Buffer Occupation for Retrieval of Two Media Streams with Different Rates

$2T_s$. T_{la} and T_{lb} respectively are the rotational latency that is required to skip the gaps between the media blocks during T_c. If T_w in each T_c is not less than zero, both of the streams can be retrieved continuously.

In general, if there are N retrieving streams, by using the same approach, the buffer-empty-duration (T_c) can be expressed as:

$$T_c = \sum_{i=1}^{N} T_{fi} + NT_s + NT_l^{avg} + Tw \tag{4.6}$$

where the footnote i corresponds to a specific stream ith.

From the above analysis, we can find out the conditions when the number of N streams can be retrieved simultaneously. The condition can be expressed as $T_w \geq 0$, or as:

$$T_c - (\sum_{i=1}^{N} T_{fi} + NT_s + NT_l^{avg}) \geq 0 \tag{4.7}$$

Since the buffer size is different from stream to stream, T_f and T_l are different. If we assume that each retrieval buffer of stream ith can contain upto K_i media blocks, then T_f and T_l in each T_c are equal to:

$$T_{fi} = \frac{B_i}{R_{dt}}$$

and

$$T_l = \sum_{i=1}^{N} (K_i - 1)\frac{G_i}{R_{dt}}$$

With the details of each term, the equation can be rewritten as :

$$\frac{B_i}{R_{ci}} - (\sum_{i=1}^{N} \frac{B_i}{R_{dt}} + NT_s + \sum_{i=1}^{N} (K_i - 1)\frac{G_i}{R_{dt}}) \geq 0$$

By satisfying this condition, the number of N streams can be retrieved simultaneously without violating the real-time constraints of any of the streams.

4.3 Simultaneous Real-time Retrieval Users

We are interested in the number of simultaneous users a system can support in a given situation. We assume that each user requests to retrieve one continuous stream at a time. So the number of simultaneous retrieval streams equals the number of simultaneous users. To simplify our discussion, we use a model where the buffer consumption rates are the same for all retrieving streams. From the above analysis, we know that when transfer-idle-period T_w equals to zero, that is when the T_c equals to the total data transfer time plus the total latency, a new request of retrieval can not be accepted. At this time, the number of simultaneous retrieving streams have reached maximum, and the number of simultaneous retrieving streams N can be derived from equation 4.5.

When $T_w = 0$, we have the total number of retrieving streams:

$$N = \frac{\frac{B}{R_c}}{\frac{B}{R_{dt}} + L_{max}} \tag{4.8}$$

where B/R_c and B/R_{dt} are respectively the buffer-empty-duration and the time needed to transfer number of continuous media blocks to fill out a buffer. L_{max} is the total disk latency during the data transfer in each buffer-empty-duration.

Rewriting equation 4.8, we have

$$N = \frac{R_{dt}}{R_c}\left(\frac{1}{1 + \frac{L_{max}}{\frac{B}{R_{dt}}}}\right)$$

From this equation, we can see clearly that the upper limit of N will be $N_{limit} = \frac{R_{dt}}{R_c}$, which is the ideal case when there is no seek time and no rotational latency during the retrieval. In practice, those latency factors reduce the number of simultaneous users by a factor of $1/(1 + L_{max}/(B/R_{dt}))$.

With the current technology, the disk transfer rate is still very low compared to the total rate that is required to retrieve multiple continuous streams. For example, given the transfer rate of storage device: 5.5MBytes per second, if we want to retrieve a video stream with a frame size of 360x240 bytes in 30 frames per second, the upper limit of the number of simultaneous retrieval streams this device can support is only 2.1. With the disk latency deductions, the actual number of simultaneous real-time retrieval streams may only be one. In this case, if more users request to retrieve streams simultaneously, schemes to fully buffer or partially buffer in advance have to be used. Fully buffering in advance is to locate and transfer complete streams from the disk to buffers before the retrieval starts.

Fully buffering in advance is a completely non real-time approach with a long response time, and the buffer requirements will increase dramatically with the increasing of stream length. Partially buffering in advance means to buffer part of the streams for each user according to its retrieval rate, then consuming processes start to retrieve data from their buffers, in the mean time, the remaining media blocks of each retrieving stream will be retrieved from the disk to buffers in real-time. With the same number of simultaneous retrieval users, partially buffering in advance can reduce the response time and also can reduce the buffer requirement compared to the fully buffering approach.

However, both buffering in advance approaches need a relatively large memory space, and the response time is longer compared to the real-time retrieval approach. With the development of new storage technologies, in the future, a disk with a gigbytes transfer rate will be able to support a fair number of simultaneous real-time retrieval users. In the mean time, it can also support non real-time retrieval user by using only a small potion of the I/O bandwidth of the storage device. In the following, we will discuss some possible methods which can improve the number of simultaneous real-time retrieval users for a given storage device.

Since the upper limit of simultaneous users depends only on the data transfer rate

of the storage device and the required buffer consumption rate, the most effective way to improve the number of simultaneous users is to increase the data transfer rate R_{dt} or decrease the buffer consumption rate R_c. Since R_c is an application dependent parameter, decreasing R_c will reduce the quality of the playback. For example, in order to decrease R_c, we have to scarify video quality during its playback, which can be done by either reducing the video playback rate or reducing the video frame size.

By minimizing the rotational latency without changing the data transfer rate or the buffer consumption rate, the total number of simultaneous retrieval users can only slightly approach upper limit of the system. However, decreasing the rotational latency requires the changing of the data placement of streams on the disk which can be done only during the recording period. The rotational latency is mainly introduced by the gap between two successive media blocks of a stream, so, we can reduce the gap size during the recording of the stream. Here, we give an example: if the required minimum number of simultaneous users a server has to support is m, then during the recording of a stream, the playback rate of this stream needs to be increased m times by using R_c multiplied by m, and this increased playback rates will be used in the real-time conditions(for the details of real-time condition, see chapter 3) of the stream:

$$\frac{M + G_i}{R_{dt}} \leq \frac{M}{mR_c}$$

By doing this, the gap size between two successive media blocks is reduced, so the rotational latency is reduced. Thus, more real-time retrieval users can be supported. However, if the application mainly requires the fast retrieval of multiple streams simultaneously, we should use the storage model in which media blocks are stored contiguously.

The seek time is another main factor which reduces the number of simultaneous real-time retrieval users. To alieviate the effect of the seek time, we prefer to transfer data from the disk to a buffer as much as possible in each transfer operation, which means a bigger buffer size for each stream. In our simulation, we will analyze how the seek time

and the buffer size affect the number of simultaneous real-time retrieval users. One of the other approaches to reduce the seek time is to sort all the requests according to their disk locations and the current disk head positions before retrieval starts. However, sorting operation itself is also time consuming and will cause long response time. Currently, effective methods to reduce the seek time are still relatively unexplored, and it requires efforts in both the hardware and the software areas.

4.4 Dynamic Access Control

While serving a set of N retrieval requests, if a storage server receives a new $(N+1)$th real-time retrieval request, it has to decide if this new request can be accepted. The acceptance of the new request must not validate the real-time condition of any of the retrieving stream, and the real-time condition of this new request also needs to be guaranteed during the retrieval. From our above discussion of simultaneous real-time retrieval users, we see that whether or not a new retrieval request can be accepted mainly depends on the transfer-idle-period per buffer-empty-duration.

The upper limit of the number of simultaneous real-time retrieval users depends on the data transfer rate of the storage server and the total buffer consumption rates. If the sum of the buffer consumption rates, including the buffer consumption rate of the new request stream, is larger than the data transfer rate of the disk, then obviously this new request can not be accepted.

However, even when the total data consumption rate is less than the data transfer rate, the acceptance of the new request $(N + 1)$th still depends on the length of the transfer-idle-period per buffer-empty-duration when the server is retrieving a set of N streams. If the length of T_w satisfies the following condition, then, the $(N + 1)$th request

can be accepted. The condition of acceptance can be expressed as:

$$T_{w(N)} \geq \frac{B_{N+1}}{R_{dt}} + (K_{N+1} - 1)\frac{G_{N+1}}{R_{dt}} + T_s$$

where B_{N+1} is the size of the retrieval buffer that will be allocated for the retrieval of the $(N+1)$th request, and $B_{N+1} = K_{N+1}M$ which means that each buffer can contain a maximum of K_{N+1} media blocks. G_{N+1} is the gap size of stream $(N+1)$th.

The above condition can be explained as: if the transfer-idle-period per buffer-empty-duration is larger than the total time needed to locate the media blocks and to fill out one of the two retrieval buffers for this new stream, the request can be accepted.

If the above condition is not true, it tells us that the number of simultaneous real-time retrieval users has already reached the maximum. However, in fact, the disk bandwidth is not being totally consumed at this time, since the transfer bandwidth is still larger than the total consumption bandwidth ($T_w(N) > 0$). Although the remaining bandwidth is not enough to support a new retrieval request by using the real-time method, this bandwidth can be used to retrieve the request media stream by using fully or partially buffering in advance methods. Fully buffering in advance is not suggested except when the retrieval of the required stream has no fast response requirement or the request stream is very short. Partially buffering in advance can reduce the response time, and save memory space compared to the fully buffering method. To calculate the amount of data needed to be buffered in advance, we have to find out the length of the required stream.

If we assume that the request buffer consumption rate is $R_{c(N+1)}$, the length of the required stream is L_{N+1}, and the remaining unused data transfer bandwidth of the disk, while serving a set of N real-time retrieval users, is R_r. Then, to guarantee the continuous retrieval of this new stream, the amount of data (B_{adv}) needed to be buffered before the consuming process starts is:

$$B_{adv} = L_{N+1}(1 - \frac{R_r}{R_{c(N+1)}}) + M \tag{4.9}$$

Where the $R_{c(N+1)}$ is given by the user's request, and L_{N+1} – the length of the stream $(N+1)$th is stored as one of the attributes in the header of the stream. One more media block, M, is buffered to give some error tolerance margin. The unused transfer bandwidth – R_r can be determined by the length of the transfer-idle-period ($T_w(N)$) per buffer-empty-duration(T_c):

$$R_r = \frac{T_{w(N)}}{T_c} R_{dt}$$

and from equation 4.6, we know that

$$T_w(N) = T_c - \sum_{i=1}^{N} T_{fi} - NT_s - NT_l^{avg}$$

Through the above analysis, we can give our dynamic access control algorithm as follows:

Algorithm : *ACC.CONTL*

Input : Request $(N+1)$th— buffer consumption rate $R_c(N+1)$.

Output : Boolean: Accepted or Rejected.

Start: {

If ($R_{dt} \geq \sum_{i=1}^{N+1} R_{c(i)}$)

{

If($T_{w(N)} > 0$)

{

If ($T_{w(N)} > \frac{B_{N+1}}{R_{dt}} + (K_{N+1} - 1)\frac{G_{N+1}}{R_{dt}} + T_s$))

{ ACCEPTED}

else {

The remaining data transfer bandwidth: $R_r = T_{w(N)}/T_c * R_{dt}$

get the stream length– L_{N+1} from the header of the stream.

$B_{adv} = L_{N+1}(1 - \frac{R_r}{R_{c(N+1)}}) + M$

If (the available memory space > B_{adv})

{ACCEPTED}

else{REJECTED}

} (end else)

} (end if)

else{REJECTED}

} (end if)

else{REJECTED}

} (end.)

$T_{w(N)}$ in the algorithm is the transfer-idle-period per buffer-empty-duration when the server is retrieving a set of N streams simultaneously, and B_{N+1} is the size of one of the two buffers assigned to the new required stream, and B_{N+1} equals the number of K_{N+1} media blocks. Also, the G_{N+1} is the gap size of the required media stream.

By using the above algorithm, we can control the access to the storage device so that the continuous requirements of all the retrieving streams can be guaranteed.

4.5 Discussions

4.5.1 Retrieval Start Time

When a new retrieval request is accepted, the retrieval start time of this stream must be controlled so that the buffer empty point of this new stream will not conflict with the buffer empty point of any other retrieving streams. Thus, the contention of the storage device can be avoided.

The start time of a new stream depends on at which point in T_c the retrieval request arrives. We assumed that once the buffer filling process start, it has to fill a buffer completely before it can switch to fill one of the other buffers for another retrieving stream. Therefore, if a request arrives at a point where the time left in the current T_w is not enough for the transfer process to fill a buffer completely for this new stream, then, the retrieval of this new stream has to be delayed, and wait until the next T_w becomes available. Once a retrieval buffer is completely filled, the consuming process can start to retrieve the data from that buffer and send it to a display device.

4.5.2 Synchronization Schedule among Related Media Streams

In our analysis, we assumed that each user will require to retrieve one stream at a time. However, in real multimedia applications, a user may request simultaneous retrieval of more than one real-time media streams(e.g. video stream and audio stream) as well as non-real-time media streams(e.g. text, graphic etc.). The synchronization relationships among different media streams is represented by a timing schedule defined by the user. Requests from the differnt users, each asking for the retrieval of mixed media streams, will arrive at the storage server along with their own timing schedules. Therefore, the file server must serve not only the multiple real-time retrieval streams, but also the non-real-time retrieval streams. The service policy is such that it first guarantees the continuous retrieval of the real-time streams, and the residual disk bandwidth will be used to serve the non real-time retrieval requests. Priorities can be assigned among the different no-real-time requests for retrieval.

In our storage model, different media will be stored separately from each other on the disk. Due to the bandwidth limitation of a storage device, we suggest assigning one disk head for each real-time media(e.g. audio or video). This means that, for a system which can support audio and video media, there are at least two disk heads to serve the retrieval requests. Thus, the non-real-time requests can be served by using either the

audio head residual bandwidth or the video head residual bandwidth.

According to the different retrieval rates in a user's request, the same or different sized buffers will assigned to each individual real-time retrieval streams and to each non-real-time retrieval stream. After all the retrieval buffers are initially completely filled by transfer processes, the consuming processes will start to consume the data from the buffer according to its timing schedules. The details of satisfaction of different timing schedule for different users simultaneously at disk retrieval level is not the topic in this thesis, and it may become our future research work.

4.6 Computer Simulation

As we discussed in section 4.3, there are several factors that affect the number of simultaneous users. Affecting factors and the number of simultaneous real-time retrieval users are related in equation 4.5. The main purpose of our simulations is to study how the number of simultaneous real-time retrieval users is affected by factors such as disk latency, retrieval buffer size and the speed ratio between the disk transfer rate and the average buffer consumption rate. As one important measurement of a storage server's capacity, T_w is used in our simulation to indicate the usage of the disk I/O bandwidth. The simultaneous real-time retrieval condition(e.g. equation 4.3) is simulated on the computer using the model presented in the following section, and our simulation results are given in the later sections.

4.6.1 Simulation Model

Simulations have been performed using the model which is shown in figure 4.3. The same buffer consumption rate is assumed for all the retrieving streams. Besides the

assumptions we made for this model in section 4.2.2, we suppose that all $2N$ buffers are located inside the user's display devices, and all requests from users are video retrieval with required rate: 30 frames per second.

Parameters used in the simulations are shown in table 4.1.

Name	Symbol	Value
frame size	$FRAME$	600 sectors
average buffer consumption rate	Rc	30 *FRAME/s
disk sector size	$SECTOR$	512 bytes
media block size	M	600 sectors
transfer idle period	T_w	
average seek time	T_s	
gap size of stream ith	G_i	
number of simultaneous real-time retrieval user	N	
the size of each retrieval buffer	B	
speed ratio R_{dt}/R_c	SR	
data transfer rate of the storage device	R_{dt}	

Table 4.1. Parameters used in simulations

The gap size of a stream is determined by its real-time condition and its disk layout algorithm during its storage (chapter 3), so different streams may have different gap size. In all our simulations, the gap size of each retrieving stream is modeled as an independent poisson random variables having a certain mean value representing the average gap size of the retrieving stream. According to the probability law, the gap G_i of stream ith is a poisson random variable:

$$P\{G_i = n\} = \frac{e^{-j} j^n}{n!}$$

where j is the average value of gap size for all retrieving streams.

Another poisson random variable is the seek time. The seek time is determined by the current position of the disk head and the disk location of the required media block. Thus, the seek time (i.e. seeking a media block and switching the disk head) can be modeled by the poisson probability law:

$$P\{T_s = m\} = \frac{e^{-k}k^m}{m!}$$

Where k is the average value of seek time when the disk head seeking and switching between media blocks of two different streams during the retrieval.

4.6.2 Simulation Results and Discussions

Three sets of experiments have been performed, namely, the effect of the disk latency, the effect of retrieval buffer size, and the effect of speed ratio. They are detailed as follows:

1. Effect of The Disk Latency.

As we discussed in section 4.3, the disk latency (i.e. rotational latency and seek time) reduces the the number of simultaneous real-time retrieval users a system can support. This is verified by our simulation results.

Rotational Latency.

The rotational latency is mainly introduced by the gap between the two successive media blocks of a stream. Through our simulation, we studied the fact that how the size of the gap affects the length of the transfer-idle-period(T_w), so as to affect the number of simultaneous real-time retrieval users.

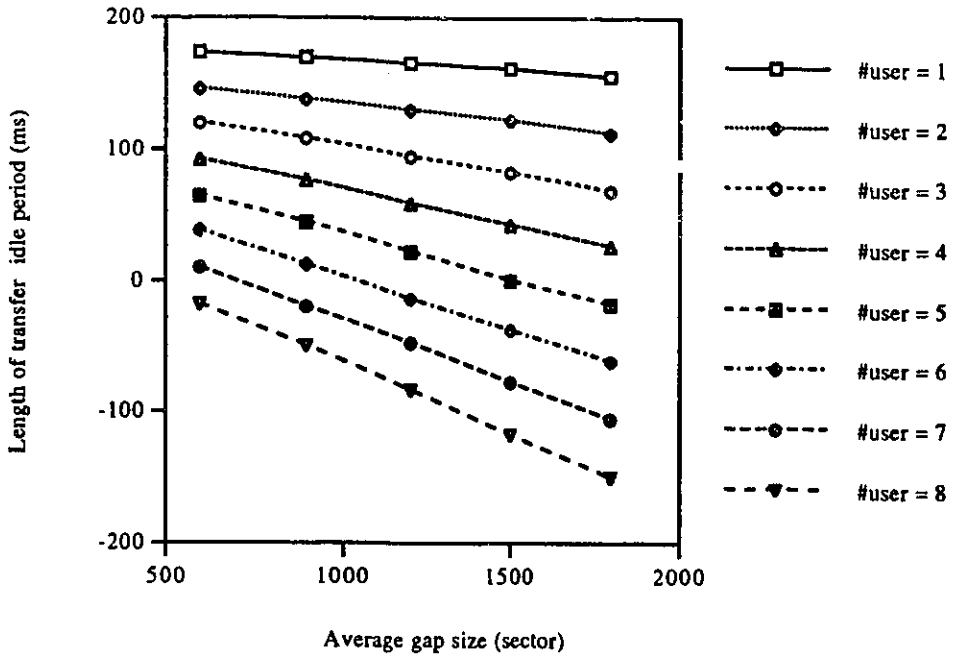


Figure 4.6: Simulation Result on Latency Effect

As shown in figure 4.6, for the given speed ratio($SR = 20$), the retrieval buffer size of one stream($B = 3600x512bytes$), the average seek time ($T_s = 9ms$), and the number of simultaneous real-time retrieval users (as shown in the figure), then, the larger the average gap size is, the shorter T_w will be. The positive value of T_w means that the storage server may be able to accept a new retrieval request. However, if T_w less than zero, the continuity of the retrieving stream may not be guaranteed. Also, the gap size effect becomes more significant when the number of simultaneous real-time retrieval users is increased. From the number of simultaneous retrieval user's point of view, given T_w equal to zero, the large gap size will decrease the number of simultaneous retrieval users the system can support.

This scenario can be explained by our equation 4.5, if we rewrite the equation as:

$$T_w = B - \frac{NB}{SR} - NT_S R_c - \frac{(\frac{B}{M} - 1)Nj}{SR}$$

then we see that both j (average gap size of all retrieving streams) and N (number of simultaneous real-time retrieval users) are in the negative portions.

Seek Time Effect.

In figure 4.7, a linear relationship between the average seek time (T_s) and the length of T_w is shown. Four plotted lines in the figure represent the number of 5, 6, 7, or 8 simultaneous retrieval users respectively. The size of retrieval buffers, the average size of the gap, and the speed ratio are given (as shown in the figure) in the simulation.

As we expected, T_w can be increased by decreasing the average seek time (T_w). The number of simultaneous real-time retrieval users that the system can support will also increase, when the average seek time decreases. For instance, when T_s is larger than 7.5ms, the storage server can support 6 simultaneous users, and when T_s drops to 5ms, 7 simultaneous users can be supported.

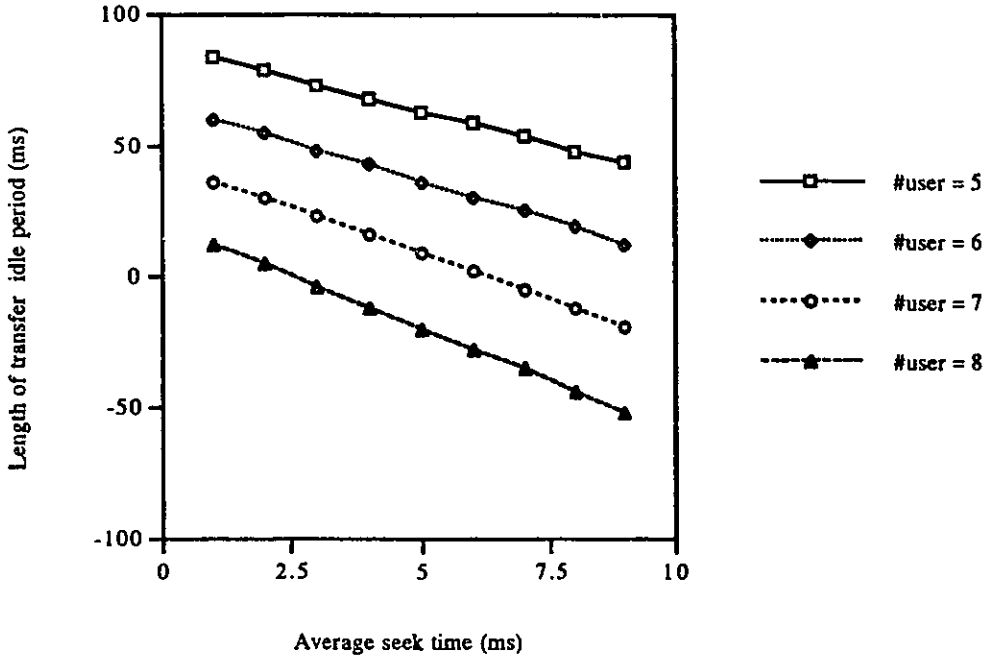


Figure 4.7: Simulation Result on Seek Time Effect

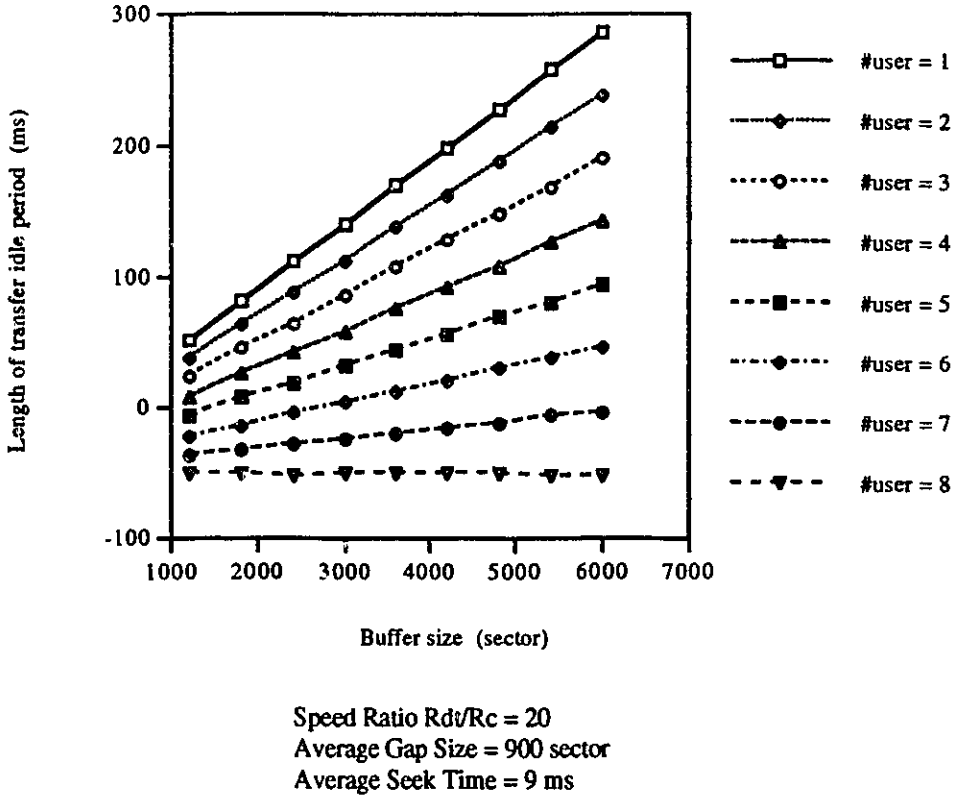


Figure 4.8: Simulation Result on Buffer Size Effect

2. Effect of Retrieval Buffer Size.

Increasing the buffer size, as we discussed in section 4.2.2, the T_c as well as the T_f and total rotational latency are increased. The overall effect can not be seen clearly from equation 4.3.

With the given average gap size, the average seek time and the speed ratio (shown in the figure 4.8), our simulation is done to find out how the T_w as well as the number of simultaneous real-time retrieval users change with the size of retrieval buffer.

The simulation result is shown in figure 4.8. We have observed that the length of T_w does increase with the size of retrieval buffer, but it is true only when the number of simultaneous real-time retrieval users is less than the maximum simultaneous real-time retrieval users the given system can support. For the given system parameters in figure 4.8, the maximum number of simultaneous real-time retrieval users is 7. This means that with the specified disk latency and the speed ratio, more than 7 simultaneous real-time users can not be supported, no matter how large the size of retrieval buffer is. As we see in the simulation result, when there are 8 simultaneous retrieval users, T_w does not increase with the buffer size.

To explain this observation by using the simultaneous retrieval control equation 4.3, we first reformat the equation to be as the following:

$$T_w = B(1 - \frac{N}{SR} - \frac{\sum_{i=1}^{N} G_i}{M * SR}) - (NT_s R_c - \frac{\sum_{i=1}^{N} G_i}{SR})$$

To guarantee the continuous retrieval of all the streams, T_w must be zero or positive. Considering the first part of the above equation which relates to the size of retrieval buffer, a positive T_w can only be obtained when

$$1 - \frac{N}{SR} - \frac{\sum_{i=1}^{N} G_i}{M * SR} > 0$$

When the above equation equals to zero, the T_w will always equal a negative number, e.g. $NT_s R_c - \frac{\sum_{i=1}^{N} G_i}{SR}$, and the size of retrieval buffer will have no effect on T_w. The T_w will decrease with buffer size increase, when $1 - \frac{N}{SR} - \frac{\sum_{i=1}^{N} G_i}{M*SR} < 0$. Neither of the last two situations can guarantee continuous retrieval of all the retrieving streams. As we discussed in section 4.5, while serving a maximum number of real-time retrieval streams, the remaining bandwidth of the storage device can be used to support either non real-time retrieval users or support real-time retrieval users by using partial buffering in advance.

When the total disk latency and the speed ratio are known in advance, these simulation results can also help us to determine the optimal size of the retrieval buffer, which

is when T_w is equal to or is slightly larger than zero.

3. Effect of Speed Ratio.

The upper limit of simultaneous real-time retrieval users is given by the speed ratio between the transfer rate and the average buffer consumption rate. Because of the disk latency, the real number of simultaneous real-time retrieval users a system can support is lower than the speed ratio. Obviously, with the same disk latency, increasing the speed ratio R_{dt}/R_c will increase the maximum number of simultaneous real-time retrieval users.

Figure 4.9 plots the relationship between the speed ratio R_{dt}/R_c versus the length of T_w. The simulation result shows, with a fixed number of simultaneous real-time retrieval users, the T_w increases when the speed ratio is increased. The maximum number of simultaneous real-time retrieval users a system can support also increases as the speed ratio goes up, as shown in figure 4.10. This result is obtained by simulating the equation:

$$1 - \frac{N}{SR} - \frac{Nj}{M * SR} = 0$$

, where j is the average gap size of the streams.

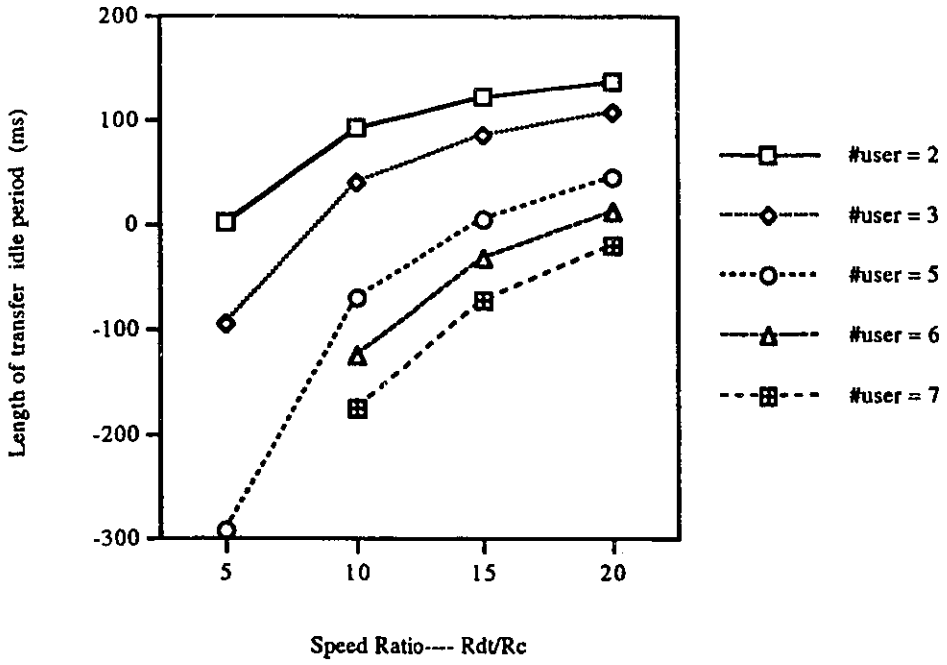


Figure 4.9: Simulation Result on Speed Ratio Effect

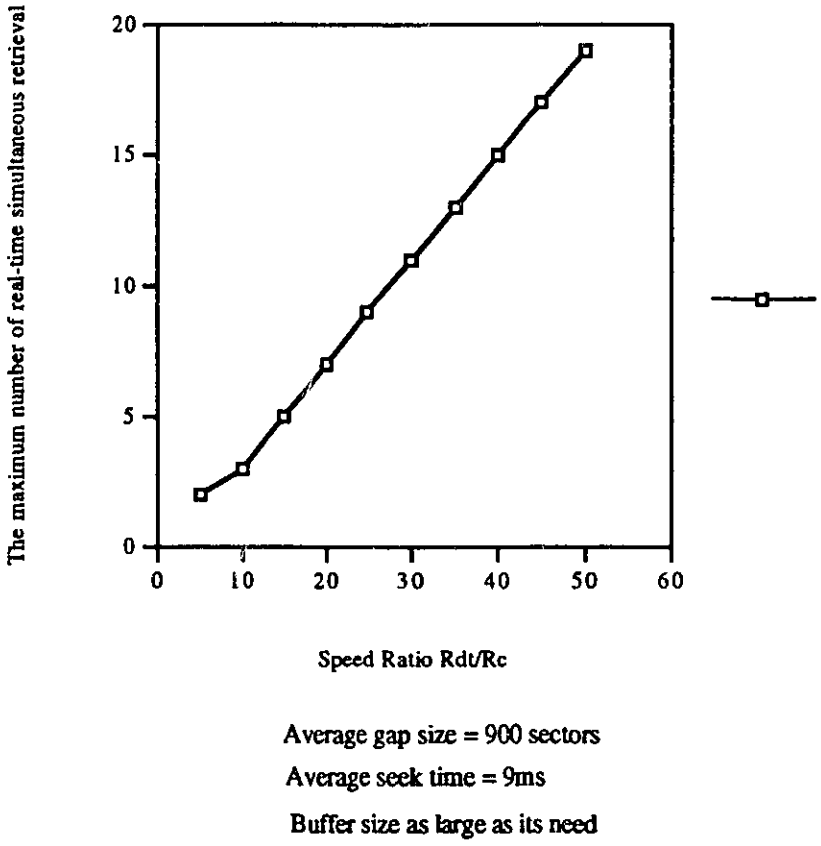


Figure 4.10: The Speed Ratio Vs. The Maximum Number of Simultaneous Real-Time Retrieval users

4.7 Summary

In this chapter, we have discussed our supporting scheme for simultaneous real-time retrieval of multiple streams. A complete analysis of the simultaneous real-time retrieval as well as a dynamic access control algorithm were presented. The simulation results have shown the relationships between the number of simultaneous real-time retrieval users, disk latency, I/0 bandwidth and buffer space. By serving a number of simultaneous real-time retrieval users along with some non-real-time retrieval users, the disk I/O bandwidth can be used more efficiently. The method we addressed here, although only for a single disk head storage system, is valuable and can be extended to multiple disk heads or to a multiple disk driver's system for a higher I/O bandwidth. However, how to reduce the seek time, the major reduction factor, more efficiently when serving multiple simultaneous users is still relatively unsolved, and it will become one of our next research directions. The other work in the future is to extend our scheme to a system which has multiple disk drivers or multiple disks.

Chapter 5

Conclusions

5.1 Summary of the Thesis

In this thesis, our storage and retrieval model for multimedia data has been presented. Considering the requirements of interactive multimedia applications(e.g multimedia database), we chose to use amagnetic disk as our main storage device. In order to integrate discrete and continuous media on the hard disk, we designed our storage and retrieval model, as well as our special storage pattern for continuous media streams. In the storage model, continuous media such as audio and video is stored in separate locations on the disk. Discrete media (e.g. text, or images) may be interleaved with audio, video or other continuous media. Our storage pattern and our merging and layout algorithms were proposed to store continuous media objects and to guarantee the real-time requirements of all the streams. The temporal relationships (i.e. synchronization information) which tie different kinds of data are stored as attributes in the header of each stream file or in the header of each data object.

Focusing on the applications which require considerable editing and updating, we developed our own storage structure for multimedia data, a structure which can reduce the data reallocation during editing and can minimize the buffer requirement during the continuous retrieval. Different from Rangan' model, our merging and layout algorithms

were developed by using the central merging method. In the merging of a new stream, the real-time constraint is guaranteed for each media block of this new stream. So, there is no need to buffer a finite number of blocks in advance in order to preserve the continuity properties on the average over a finite number blocks of this stream(as occurs in Rangan's model). Therefore, our algorithms help reduce the buffer requirements as well as the complexity of calculation during the retrieval of the streams.

Based on our storage model, a control scheme for the simultaneous retrieval of multiple media steams was introduced in chapter 4. This control scheme can guarantee the continuity of the retrieving streams. Our dynamic access control algorithm was developed by combining the simultaneous control scheme with the partial buffering scheme (the detailed analysis of the buffer requirement of a stream can be found in [24]). This dynamic control algorithm can be used to check if there are sufficient I/O bandwidth and memory space for the storage device to serve another retrieval request.

Our simulation results verified the design and analysis of our storage model and retrieval control scheme.

5.2 Suggestion for Future Research

Multimedia file system is the basic and one of the important topics in the area of multimedia information and communications. However, the problem has not turned out to be clear; nor easy to solve. Because of the integration of continuous media, and the involvement of a synchronization problem, the current file system is being challenged. With the development of new multimedia technologies, such as the multimedia data transmission scheme and system architecture, new file systems have to be adapted so that they are able to support new and different multimedia systems.

Currently, the model and algorithms for multimedia data storage and retrieval have been well studied. Nevertheless, the implementation of these model and algorithms requires further efforts.

The study of this thesis focused on the performance of the magnetic disk with a single disk head, therefore, the retrieval performance(i.e. access time, number of simultaneous users, and etc.) may not turn out to be very attractive. This is mostly limited by the current storage devices which have low I/O bandwidth and long seek time.

The improvement of the performance of the storage device is of great research interest in both hardware and software areas. One of the solutions is to develop algorithms to reduce the seek time, which is the biggest overhead during the retrieval. A few algorithms have already been proposed. For example , a storage server can sort all the requests according to their disk locations and the position of the disk head before initializing the retrieval process. However, the sorting operation itself is time consuming and will increase the system response time. Quantization of retrieval buffers may reduce the total seek time by asking the transfer process to fill as many buffers as possible before it switches to another request. However, we did not find any research report discussing this technology. A further development of our algorithms by using quantized retrieval buffers is recommended.

The best solution for improving the retrieval performance of a multimedia file system will be the usage of distributed methods. Through the collection of the I/O bandwidth of multiple disk-heads or multiple disk drivers, the total I/O bandwidth of the file system will be improved dramatically. Thus, the file system will be able to support more simultaneous multimedia users, and minimize the delay of storage access. Extensions of our dynamic access control algorithm and the simultaneous retrieval scheme in a distributed environment are areas of considerable interest.

The other important issue that is not addressed in this thesis, and may be of interest in the future research, is development of a retrieval schedule for data objects in a distributed environment. The retrieval schedule depends on the synchronization requirements of data objects. As we known, the presentation of data objects is controlled by the user defined scenarios. In order to meet the presentation scenarios at the user end, a data delivery schedule is derived according to the condition of the network for each involved media object. This data delivery schedule specifies the time when a media object should

be delivered to the network. The data delivery schedule is then sent as part of the request to the multimedia storage server for the retrieval of data objects. The required media objects may reside on the same or different disks in a distributed computing system, and some of the required media streams may have real-time requirements. So the storage server must be able to schedule and control the access of the storage device to meet the data delivery deadlines and to guarantee the continuity of the retrieval streams. An algorithm for deriving the retrieval schedules needs to be developed. This algorithm must be able to relate together the disk I/O bandwidth, the location of media objects, the latency of the storage device, and the the data delivery deadline of each data object.

Bibliography

[1] E. A. Fox, "The coming Revolution II," *Communications of the ACM, July 1989, vol. 32, pp794-801.*

[2] Larry Press " Personal Computer" *Communications of the ACM,July 1989, pp784-788.*

[3] N.mackay and G.Davenport."Virtual Video Editing In Interactive Multimedia Applications" *Communications of the ACM,July 1989, pp802-810.*

[4] K.A.Frenkel " The Next Generation of Interactive Technologies" *Communications of the ACM, July 1989, pp872-881.*

[5] G.D.Ripley "DVI: A Digital Multimedia Technology" *Communications of the ACM,July 1989, pp811-822.*

[6] C.Yu,WeiSu,DinaBitton,etc."Efficient Placement of Audio Data On Optical Disks for Real-time Applications" *Communications of the ACM,July 1989, vol 32, pp862-871*

[7] Michael Tinker "DVI Parallel Image Compression" *Communications of the ACM,July 1989, pp844-851.*

[8] D.F.Dixon "Life Before The Chips:Simulating Digital Video Interactive Technology" *Communications of the ACM,July 1989, pp824-831.*

[9] A.Lippman and W.Butera " Coding Image Sequences For Interactive Retrieval" *Communications of the ACM, July 1989, pp852-860.*

[10] D.J.Moore "Multimedia Presentation Development Using the Audio Visual Connection" *IBM SYSTEM JOURNAL, VOL29.NO.4,1990, pp494-508.*

[11] Richard Comerford, "The Multimedia Drive",*IEEE Spectrum, April 1994, pp77-83.*

[12] Harrick M.Vin and P.V.Rangan, "Designing File Systems For Digital Video and Audio" *Tech. Rep. No. cs91-191, Department of Computer Science and Engineering,University of California,San Diego, Feb,1991.*

[13] Harrick M.Vin and P.V.Rangan, "Designing A Multi-User HDTV Storage Server" *Tech. Rep. No. cs92-225, Department of Computer Science and Engineering,University of California,San Diego, Jan,1992.*

[14] D.P.Anderson,R.Govindan,G.Homsy "Digital Audio And Video in A Network Window System" *Department of Computer Science and Engineering,University of California,San Diego , Feb.1990.*

[15] Harrick M.Vin,T.Kaeppner and P.V.Rangan, "A Testbed For Managing Digital Video And Audio Storage" *Department of Computer Science and Engineering,University of California,San Diego. April,1991.*

[16] D.P.Anderson,R.Govindan,G.Homsy "A Window-Based Editor For Digital Video and Audio". *Department of Computer Science and Engineering,University of California,San Diego, June,1991.*

[17] D.B.Terry and D.Swinehart, "Managing Stored Voice In The Etherphone System",*Computer Science Laboratory, Xerox Palo Alto Research Center*

[18] "Video Compression Makes Big Gains" *IEEE Spectrum , Oct.,1991, pp9-14.*

[19] E.A.Fox,Polytechnic Institute and State University. "Advances In Interactive Digital Multimedia Systems", *IEEE Computer, October, 1991, pp9-21.*

[20] "Multimedia" *BYTE, Feb,1990, pp15-20.*

[21] A.Karmonch,L.O.Barbosa,and N.Georganas,M.Goldberg "A Multimedia Medical Communications System" *IEEE Journal VOL.8,NO.3, April 1990, pp325-339.*

[22] R.Steinmetz "Synchronization Properties In Multimedia System" *IEEE Journal VOL.8, April,1990, pp401-412.*

[23] D.P.Anderson,G.Homsy. "A Continuous Media I/O Server and Its Synchronization Mechanisms", *IEEE Comp. Spec. Issue on Multimedia Info. Sys. Oct. 1991, pp51-57.*

[24] Jim Genmmell and Stavros Christodoulakis, "Principles of Delay-Sensitive Multimedia Data Storage and Retrieval," *ACM transactions on Information Systems, Vol.10,No.1, January 1992, pp51-90.*

[25] J.Emery and A.Karmouch, "A multimedia Document Architecture and Rendering Synchronization Scheme", *Procceding, Second International Conference on Broadband Island, June, 1993, pp59-67.*

[26] 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.*

[27] 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.*

[28] Karmouch, A. "A Multimedia Information and Communications System: MEDIABASE." *Proc. Multimedia Commun.'93 conference, Alberta, April 13-16, 1993, pp234-244.*

[29] Richard Staehli and Jonathan Walpole, "Constrained Latency Storage Access",*Computer, March 1993, pp44-53.*

[30] Fouad A. Tobagi and et.al., "Streaming RAID: A Disk Array Management System For Video Files", *ACM Multimedia, June, 1993, CA, USA. pp393-400.*

[31] Shahram Ghandeharizadeh and Luis Ramos, "Continuous Retrieval of Multimedia Data Using Parallelism", *IEEE Transactions on knowledge and Data Engineering, Vol.5, No. 4, August 1993, pp658-669.*

[32] David P. Anderson, Yoshitomo Osawa and Ramesh Govindan, "The Continuous Media File System", *Summer 92 USENIX, June 8 - June 12, 1992, pp157-164.*

[33] Harrick M.Vin and P.venkat Rangan, "Techniques for Efficient Storage of Digital Video and Audio," *Computer Commun. Vol 16, pp168-176, March 1993.*

[34] Scoot Wallace, "Managing Mass Storage", *BYTE, March 1994, pp78-89.*

[35] Alan E. Bell, "Critical issues in high density magnetic and optical data storage", *Proc. SPIE, VOL. 27, No.6, June, 1984.*

[36] Richard Comerford, "The multimedia drive", *SPECTRUM, April, 1994. pp77-84.*

[37] Bernard Cole, "The technology frame work", *IEEE Specirum, March 1993, pp32-39.*

[38] Srinivas Ramanathan and Venkat Rangan, "Architectures for Personalized Multimedia", *IEEE Multimedia, spring, 1994, pp36-43.*

[39] S. Sibbs, D.Tsichritzis, A. Fitas, et al, "MUSE: A Multimedia Filing System", *IEEE Software, 4(2):4 - 15, March, 1987.*

[40] B.C. Ooi, A.D. Narasimhalu, et al, "Design of a Multi - media File Server using Optical Disks for Office Applications", *IEEE Computer Society Office Automation Symposium, Gaithersburg, MD, pp157-163, April, 1987.*

[41] R.H. Thomas, et al., "Diamond: A Multimedia Message System Built on a Distributed Architecture", *Computer, 18(12): 65-78, December 1985.*

[42] Anderson, D.P et al., "Abstractions for continuous media in a network window system.",*Proceedings of the International Conference on Multimedia Information System 91, Singapore, pp273-298, 1991.*

[43] Anderaon, D.P. et al., "Support for Continuous Media in the DASH system", *Proceedings of the International Conference on Distributed Computing Systems, pp54-61, 1990.*

[44] Abdot, C. "Efficient Editing of Digital Sound on Disk.", *J. Audio Eng. Soc. 32, 6, pp394-402, 1984.*

[45] Park, A., and English, P. "A variable Rate Strategy for Retrieving Audio from Secondary Storage.", *Proceeding of the International Conference on Multimedia Information Systems 91, Singapore, pp135-146, 1991.*

[46] Li Li, A. Karmouch, N.D. Georganas, "Real-time Synchronization Control in Multimedia Distributed Systems", *ACM Computer Communication Review, vol. 22, no. 3, pp79-86. 1993.*

[47] L. Li, A. Karmouch, N.D. Georganas, "Multimedia Teleorchestra with Independent Sources: Part1-Temporal Modeling of Collaborative Multimedia Scenarios," *ACM Multimedia Systems, vol.1, No.4, Feb. 1994, pp13-19.*

[48] L. Li, A. Karmouch, N.D. Georganas, "Multimedia Teleorchestra with Independent Sources: Part2-Synchronization Algorithms." *ACM Multimedia Systems, vol 1, No.4, Feb. 1994, pp78-84.*

[49] A. Karmouch, R. Wang and T.Yap, "Design and Analysis of a Storage and Retrieval Model for Continuous and Discrete Media" *Technical Report 941109, Dept. of Electrical Engineering, University of Ottawa, Feb. 1994.*

[50] C.A.Ellis, S.J.Gibbs, and G.I Rein, "Groupware — Some Issues and Experiences", *Communications of The ACM*, Vol.34, No.1, January 1991, pp38-58.

[51] L. Lamont and N. D. Georganas, "Protocols for real-time multimedia conferencing", *Proc. Can Conf. on Elec. & Comp. Engg.*, Toronto, Sept.1992.

[52] V. Witana and A. Seneviratne, "Operating System Support for Multimedia Applications" , *ICC Multimedia Communications'92 Australia, pp53-61.*

[53] S. Loeb, "Delivering Interactive Multimedia Documents over Networks" *IEEE Communications Mag. May, pp22-33, 1992.*

[54] T. D. C. Little and A. Ghafoor, "Network Considerations for Distributed Multimedia Object Composition and communication", *IEEE Network Mag., Nov. 1990, pp32-49.*

[55] B. Furht, "Multimedia Systems: An Overview", *IEEE Multimedia 94, 1994, pp47-59.*

[56] 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 Germany, pp341-350.*

[57] C. Fung and M. Pong, "MOCS: An Object-Oriented Programming Model for Multimedia Object Communication and Synchronization", *IEEE Multimedia Computing, Boston (to preview), 1994 HongKong.*

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

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

[60] H. Sadamoto and S. Matsushita, "Synchronization Control between Text and Voice for Multimedia Message Communication", *Proc. ICCC 90 International Conf. on Computer Communication, 1990 Japan, pp191-197.*

[61] K. Tavindran and V. Bansal, "Delay Compensation Protocols for Synchronization of Multimedia Data Streams", *IEEE Trans. on Knowledge and Data Engineering. Vol. 5, No. 4. August 1993, pp574-589.*

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

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

[64] 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, pp213-219.*

[65] D. Ferrari, A. Gupta, M. Moran and B. Wolfinger, "A Continuous Media Communication Service and its Implementation", *GlobeCom'92, pp220-224.*

[66] 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, pp1395-1404.*

[67] Sunil Sarin, "Computer-Based Real-Time Conferencing Systems", *IEEE Computer. October, 1985, pp33-45.*

[68] Chaim Ziegler and Gerald Weiss, "Multimedia Conferencing on Local Area Networks", *IEEE, Computer, September, 1990, pp52-61.*

[69] John Robinson, Eliot Rubinov, and et al., "A Multimedia Interactive Conferencing Application for Personal Workstations", *IEEE TRANSACTIONS ON COMMUNICATIONS. VOL. 39, NO. 11, NOVEMBER 1991, pp1698-1708.*

[70] Thomas D. C. Little and Arif Ghafoor, "Synchronization and Storage Models for Multimedia Objects", *IEEE Journal on Selected Areas in Communications, VOL. 8. No. 3. April 1990, pp413-427.*