Statistical Admission Control for MPEG Streams with Non- overflow Guarantees

by

Rita Dilek

B. A., Brandeis University, 1974

M.A., Harvard University, 1977

M.S.W., Simmons College School of Social Work, 1981

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia

August 1998

Abstract

Most of the existing analyses on admission control and buffer management for continuous media streams assume fixed rate data compression. This assumption is invalid for MPEG streams, which have variable bit rate (VBR) compression. With the increased acceptance of VBR compression and the development of new, more efficient VBR techniques, there is a need for new models, analyses and algorithms for streams with variable compression. The Central Limit Theorem of Statistics has been proposed for use at the granularity level of data streams to handle admission control of concurrent variably compressed streams. The implication here is that the total amount of buffer space required by all streams in the system at a particular point in time is approximately normally distributed. In this thesis we develop a model for MPEG streams that applies the Central Limit Theorem at the finer granularity level of frames. This gives a stronger and more general result: that the amount of buffer space required for each stream approximates a normal distribution. Using this model we develop several admission control algorithms that provide user-selectable, individual non-overflow guarantee levels by computing the amount of exclusive buffers needed to provide these guarantees. Experimental results indicate that the buffer space overhead required to support individual guarantees is fairly small and worthwhile. We also investigate the feasibility of providing an additional $q\%$ system-wide non-overflow guarantee on top of the individual guarantees through the use of shared buffers. Experimental results indicate that the buffer space overhead is again quite small but very successful in enhancing the reliability and quality of service to the user.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my advisor, Prof. Raymond Ng, who was everything a thesis supervisor should be and more. Without his careful guidance, the interesting and thought-provoking discussions, and his support, encouragement and patience, this project would not have been possible. I also thank Prof. Norm Hutchinson for his prompt and thoughtful reading of this work and his interesting questions. I would also like to express my appreciation to the faculty and staff of the Computer Science Department at U.B.C., who manage to create a supportive and stimulating environment for the students and are always helpful and accommodating.

My special thanks are also due to the Crane Resource Centre, Mr. Paul Thiele, and the innumerable readers who generously gave their time and energy to make my studies at U.B.C. possible. Additional thanks go to the following individuals: Dale Peterson for mathematical advice, Shirley Nalevykin for many years of reading, and Elizabeth Szabo for her patient help with proofreading and editing.

RITA DILEK

The University of British Columbia

August 1998

Chapter 1

Introduction

1.1 Video on Demand Systems

The term *multimedia* refers to the simultaneous presentation of multiple media such as text, graphics, video, audio and animation. In recent years, as the storage and processing capabilities of computers have increased, and networking and I/O technologies have improved, the use of digital multimedia has been gaining in popularity. The number and types of multimedia applications is on the rise. Multimedia is used in various areas including education, entertainment and business or scientific presentations. Computer games, computerized courses used in distance education, news on demand and video on demand systems are some examples of applications which are gaining in importance. Cable companies are investing large amounts of money to install the equipment that will give them the capability to show digital movies as part of video on demand systems. Considering all these developments, providing effective multimedia support for computer systems has become a topic of great interest and value.

A Video on Demand (VOD) system consists of a multimedia server serving multiple clients, whether local or remote. Each client or user can request to view one of a number of digital movies available to the server. The system may or may not also provide VCR-like capabilities such as pause, rewind or fast forward.

Digital movies are an example of what is called *continuous multimedia (CM)*. Continuous multimedia is delay sensitive. It has continuity and real time requirements. By this we mean that once display has been started, it should not be stopped until the end of the data stream has been reached. In addition, the display must be continued at a particular rate. For instance, if the requested rate is 30 frames per second, dropping down to 10 frames per second would not be acceptable.

The goal of a VOD server is to serve as many users as possible (maximizing throughput), while ensuring that the continuity and real time requirements of the requested movies are maintained. When a new user requests a movie, the server must assess the current usage of system resources such as memory and disk bandwidth. It must then decide whether this request can be granted without violating the continuity and real time requirements of all the users already in the system. If the requirements of this new user can be met while maintaining those of already existing users, the new user is admitted into the system. Otherwise, it is forced to wait until enough resources become available. This process is known as *admission control.*

It is clear that effective admission control mechanisms are essential to a multimedia server. The server must be able to admit the maximum possible number of users, so as not to waste resources. But it must also be capable of maintaining the continuity and real time requirements of the multimedia streams.

Another factor which affects the capabilities of a multimedia server is the fact that digital multimedia makes heavy demands on computer resources such as storage and disk bandwidth. This has increased the importance of data compression. Traditionally, digital data has been compressed using fixed rate compression. More recently, however, variable bit rate compression (VBR) has become popular.

Of greatest relevance to the topic area of this thesis are studies relating to buffer allocation. But most of the available studies are concerned with multimedia streams compressed using fixed rate compression schemes. To make up for this lack, this thesis focuses on admission control of continuous multimedia streams which have been compressed using variable bit rate compression techniques. In the rest of this chapter, we first discuss the importance of compression (and especially VBR compression) in economizing resources for multimedia systems. We then discuss some of the difficulties that arise when allocating resources for variably compressed multimedia. We then briefly review the admission control needs of continuous multimedia streams and discuss how VBR compression complicates the issue. Finally, we review the statistical approach of Vin et al., which attempts to solve some of these problems, and then discuss the capabilities not made available by their approach. The chapter is concluded by a brief discussion of the problems addressed and contributions made by this thesis.

1.2 Data Compression: Fixed versus Variable Rate Systems

Digital multimedia makes heavy demands on computer resources such as storage, memory and disk bandwidth. This has increased the importance of data compression, which enables the system to save on not only permanent storage but disk bandwidth and memory allocation as well. Here we assume that data would be transferred from storage and placed in memory buffers in its encoded form. The user process would be responsible for decoding the buffered data at its own rate.

Even compressed multimedia can be extremely resource-intensive. For example, a two hour movie, needing to be displayed at a rate of 30 frames per second, would take 2 gigabits, even in a compressed form.[1] At this same rate of 30 frames per second, a movie encoded by the J-PEG method would require disk bandwidth (and memory buffers) of 7 megabits per second. The need to economize on resources has led to the development of more efficient compression schemes.

The earliest data compression schemes employed fixed rate compression. This means that the size of each compressed frame is a fixed percentage of that of the original frame. With fixed rate compression, it is easy to predict resources needed during playback, such as disk bandwidth and memory.

Variable bit rate (VBR) compression schemes such as MPEG are one attempt to increase compression efficiency. Their main characteristic is that the sizes of compressed frames are not a fixed fraction of their original size. Instead, they can vary widely based on the content of frames and the methods by which they are compressed.

Since its arrival, MPEG has quickly established itself as one of the standards for compression [20, 35]. MPEG, based on delta compression, is a scheme that uses VBR compression (see Section 3.1 for more details). In general, when compared with schemes such as JPEG that use a fixed compression rate for a stream, variable bit rate compression schemes have the advantage of being typically more effective

[1]In this thesis we consider only video compression.

in their compressions. For instance, at the rate of 30 frames per second, an MPEG-encoded stream requires only 1.5 mbits/sec versus the 7 mbits/sec required by JPEG streams. The advantages of such improved compression efficiency are obvious: demands on resources are greatly decreased, allowing more streams to be stored, and more users to access the system at the same time.

On the other hand, there are disadvantages to the use of variable bit rate compression. To achieve satisfactory results, a fixed frame rate (usually 30 frames per second) must be maintained. But since compressed frames vary widely in size, it is difficult to predict exact resource needs. So the allocation of resources, such as buffer space and disk bandwidth, is more complicated for streams using VBR compression than for streams using fixed rate compression. For instance, if the average frame size is used as the basis for allocation, overflows may occur. Overflows can be defined as situations where the actual amount of compressed data exceeds the amount of allocated resources. If the maximum frame size is used instead, while overflows are avoided, resources may be over-allocated and severely under-utilized, as the actual size of a frame may vary greatly from the maximum frame size. Yet another approach is to record the size of each and every compressed frame. But this may involve a significant amount of bookkeeping and overhead, particularly for long streams such as movies (at rate of thirty frames per second, a two hour movie contains 216 000 frames).

1.3 Admission Control of VBR-compressed Continuous Multimedia

As discussed in the previous section, it can be difficult to predict the exact resource needs for a variably compressed stream. However it is essential that some prediction be made so that the server can determine how much to allocate to a given stream.

We recall that the goal of admission control for a multimedia server dealing with continuous multimedia is to maximize throughput while ensuring that continuity and real time requirements of all streams in the system are met. Consequently service interruptions are not acceptable. Frames must be supplied to the user at the prescribed rate, and there should be no degradation in quality. This last condition is easy for streams using fixed rate compression, but quite difficult to achieve for streams using VBR compression. Since exact resource needs cannot be predicted, some estimation must be performed. Recall from the previous section that this is quite complicated: If allocation is made on the basis of average frame size, overflows may occur, so that some data is lost and degradation of quality results. If the maximum frame size is used as a basis for allocation, each stream is given a lot more resources than it really needs most of the time, so that the system throughput decreases. The other possibility of recording the size of every compressed frame can give rise to unacceptable amounts of bookkeeping and overhead. It is therefore essential to find a solution to the problem that steers somewhere between these extremes and yet allows the system to take advantage of the considerable benefits of VBR compression.

1.4 Statistical Admission Control

Given its nature, a variably compressed stream can be modeled statistically. Vin et al. use a random variable b_i to represent the total disk bandwidth (and similarly, the total buffer space) required by a variably compressed stream St_i over a certain time period [49]. By applying the Central Limit Theorem of statistics [6, 9], the total buffer space required by n streams $b_1 + \ldots + b_n$ approximates a normal distribution, for sufficiently large values of n. With the normal distribution, a server can provide statistical non-overflow guarantee on a per system basis (e.g., 99% non-overflow guarantee).

Intuitively, statistical guarantees can be explained as follows: The assumption made, is that not all the streams will require the maximum amount of resources at the same time. Some may require the average, and others require amounts ranging between the minimum and the maximum. Therefore, the total requirement for resources can be estimated assuming a normal distribution, in the expectation that the high values will usually be balanced out by the low.

When trying to provide statistical performance guarantees, the fundamental question is:

> *What is a minimized amount of resources required to guarantee non-overflow q% of the time?* [2]

We call this the *non-overflow guarantee (NOG) problem.* In practice, q% need not be 100%, and smaller values of q may also make sense. In many cases, a user may trade off the highest quality for a lower cost. While Vin et al. [50] suggest policies

[2]We do not use the word "minimum" because, given the statistical nature of the problem, a proof of optimality may be too hard to achieve. Instead, we weaken the question by the word "minimized".

on how to choose frames to discard when overflows do occur, they do not address the issue of selecting *how many* frames to discard from *which* of the streams. Thus, there is an issue of fairness involved. One simple way to ensure fairness is to force every stream to discard a similar number of frames. This solution would work reasonably well if all streams are equal or similar. However, if the streams vary in their characteristics, such as their average frame sizes or priorities, this solution is too simplistic.

1.5 Problem Definition

In this section we briefly describe the problems addressed by this thesis. As mentioned in the previous section, by applying the Central Limit Theorem of statistics [6, 9], Vin et al. develop a framework that provides a non-overflow guarantee on the total amount of resources required by all the active streams in the system [49]. In other words, the non-overflow guarantee is provided at the granularity level of the entire system. This framework is too coarse to support non-overflow guarantee on a per stream basis. Hence, without some additional schemes to ensure fairness to all users, this framework cannot guarantee a reliable level of performance to any given user. On the other hand, it could be highly desirable to have individual non-overflow guarantees (NOGs) provided to each user. User needs vary. A given user might want to select a NOG level based on stream characteristics such as average frame size or a balance between his/her tolerance for degradation in quality and the cost of different levels of NOG.

1. The first problem addressed by this thesis is whether the non-overflow guarantee problem can be solved directly at the granularity level of individual

streams. More specifically, is it possible to develop a framework which will enable the system to provide an individual non-overflow guarantee to each user, so that a user can specify the desired non-overflow guarantee level $p\%$ directly for his/her particular stream? This means that the stream is guaranteed not to overflow its allocated resources $p\%$ of the time, and a user can select this value based on his/her own needs and preferences. In order to do this we need a statistical model not only on the level of the aggregation of n streams, but also on the finer granularity level of each individual stream. The key question here is how to provide a statistical model for the size of MPEG compressed frames, which would enable the system to calculate resource needs and allocate them in a way that would ensure individual non-overflow guarantees. This would be desirable as it ensures fairness.

2. Assuming this is feasible, the next problem addressed is whether this can be achieved at a reasonable cost or whether providing individual guarantees may prove too expensive. The term "reasonable cost" may appear vague, and what may be acceptable for one system may not necessarily be so for another. Here we use the term to mean that no more than 5% additional resources need be allocated to provide individual guarantees than would be required for system level guarantees.

3. Assuming that providing individual non-overflow guarantees is feasible, a natural question to then ask is whether a further $q\%$ system-wide guarantee can be supported, on top of the individual guarantee ($q\% \geq p\%$). This could be done via a pool of shared buffers, which could be used by streams overflowing their allocated resources. The goal here is to decrease the number of overflows

and/or data lost by each stream even beyond what is achieved by the individual non-overflow guarantee. Such an additional system-wide guarantee can be highly desirable as it can help to enhance reliability and the quality of service to the user.

4. Lastly, since the general goal is to enhance quality of service at as low a cost as possible, this work investigates the cost of providing such additional system-wide guarantees. Again, we take 5% additional resource use as "reasonable cost" when evaluating the performance of different algorithms.

1.6 Contributions of this Thesis

The key contributions of this work are as follows.

1. By making use of a specific version of the Central Limit Theorem, we develop a statistical model that approximates the total size of several adjacent compressed frames as normally distributed. This allows us to determine the amount of buffer space [3] necessary to provide a p% non-overflow guarantee, for any p value chosen by the user.

2. We develop the admission control Algorithm CLT(E) that supports individual non-overflow guarantee on a per stream basis, through the use of buffer space allocated exclusively to the stream. More specifically, Algorithm CLT(E) applies our statistical model to compute the required amount of buffer space for a p% individual guarantee.

3. Our experimental results indicate that CLT(E) is effective and that individual

[3] The focus of this work is on buffer space management. But at the modeling level, disk bandwidth allocation can be dealt with in a similar way.

non-overflow guarantees can be provided at a very low cost–less than 5% extra buffer space.

4. We develop several admission control algorithms that are capable of providing both a $p\%$ individual guarantee and an additional $q\%$ system-wide guarantee ($q\% \geq p\%$). This guarantee is implemented through the use of buffer space shared by all streams that overflow their exclusive buffers. Two of these algorithms are for the homogeneous case, that is, the case where all streams have identical characteristics such as the average and distribution of frame sizes, and the selected p value is also the same. The others are for the heterogeneous case, that is, the case where stream characteristics and p values differ from stream to stream.

5. We evaluate the performance of these algorithms by simulation experiments. Our experimental results indicate that the two homogeneous case algorithms, while they have their relative advantages and disadvantages, are both very effective in providing both kinds of guarantees at a reasonable cost. Our experimental results also help us identify among the many heterogeneous case algorithms, those which are both efficient in providing the two types of guarantees and acceptable in terms of cost to the system. In other words, the additional system-wide guarantee can be provided at a very low cost, and is effective in enhancing the quality of service delivered to the user.

1.7 Overview

The rest of this document is organized as follows: Chapter 2 reviews related works including papers on resource management of continuous multimedia and papers on admission control of multimedia streams. It also gives a description of the major characteristics of MPEG streams which affect the way they must be modeled. Chapter 3 discusses the Central Limit Theorem (CLT). It then presents a statistical model for the buffer space requirement of a variably compressed stream, based on a specific form of the CLT. Chapter 4 describes Algorithm CLT(E) which uses exclusive buffers to implement individual non-overflow guarantees. Chapters 5 and 6 present algorithms which use shared buffers to provide additional system-wide guarantees. Chapter 5 presents Algorithms CLT(Sn) and CLT(Sm) which were developed for the homogeneous case, where all streams have the same characteristics and guarantee levels. Chapter 6 discusses the algorithms developed for the heterogeneous case, where streams have different characteristics and non-overflow guarantee levels. Chapter 7 presents the simulation experiments used to analyze the behavior of the algorithms, gives details of the implementation and discusses experimental results. Our conclusions and possibilities for future work are presented in Chapter 8.

Chapter 2

Background and Related Works

This chapter provides background information necessary for understanding the material in this work. In addition some works relating to the topic of this thesis will be reviewed. More specifically the following topics will be covered:

1. We discuss in greater detail the nature of continuous multimedia and the requirements that a CM server must fulfill.

2. We review some of the work relating to the design of CM servers. We discuss issues that arise in this design and some of the schemes that have been proposed for solving the problems.

3. We review works concerned with the design and implementation of CM servers that handle compressed data. Most of these studies are concerned with data compressed at a fixed rate. However some discuss issues that must be addressed when dealing with variable bit rate (VBR) compression. Studies about CM servers handling VBR compressed data fall into two categories: those providing deterministic quality of service (QoS) guarantees and those presenting

statistical approaches. We discuss how these studies relate to the topic of this thesis and how they differ from our approach.

2.1 Requirements of Continuous Multimedia Streams

A number of studies identify the characteristics of continuous multimedia data, and analyze the requirements for the storage and retrieval of such data. In this section we review these works and discuss some of the issues they address.

Gemmel and Christodoulakis [15] describe CM data as "delay sensitive". By this they mean that there are "real time deadlines for the presentation of successive units of the data". CM data can be thought of as consisting of a sequence of individual units of data which must be displayed continuously in time. These units can be pictures or frames for video or animation, or audio samples in the case of digital audio. In order to be of an acceptable quality to users, media units must be presented at a precise pre- determined rate (usually the rate at which they were recorded). If audio samples are not presented at the correct rate, pops or sound distortions are likely to occur. Similarly, if video or animation is displayed at the wrong rate, motion can appear jerky or otherwise unnatural. To avoid such problems and to provide acceptable QoS to users the system must ensure that all real time deadlines are met and that media units are presented at the correct rate, neither too fast nor too slow. Continuous presentation of the data is also essential. Service interruptions are not acceptable. Therefore, works in the field agree that CM data must meet continuity and real time requirements [3, 5, 7, 8, 10, 13, 15, 14, 16, 34, 26, 33, 44, 41, 42]. [1]

[1] As these works point out, the same requirements apply to both the storage and the retrieval and presentation of CM data. But this thesis will only focus on the retrieval aspects.

There are two other major difficulties that a continuous multimedia system must handle: synchronization and coping with massive amounts of data. Synchronizing the various strands of a multimedia stream is an important and complex task. The several channels for multi-channel audio must be coordinated and synchronized. The sound track for a video movie must not only be presented at the correct rate but must also correspond to the appropriate frames or pictures in the movie.

A number of works are concerned with synchronization of CM data. Synchronization can be achieved via time stamps and the relative position of data in a stream, as with the MPEG system [20]. Rangan et al [42] propose a method of synchronization for the case where CM data is not displayed by an integrated system, but by unrelated pieces of equipment ("media phones") receiving data directly from a server. Brief feedback messages are sent to the server by the individual display units. Using this feedback information, the server can determine if resynchronization is necessary and adjusts its transmissions accordingly. Since synchronization is not related to the topic of this thesis we limit our coverage of the issue to this brief discussion.

The final important problem which must be addressed when working with continuous multimedia is that massive amounts of data may need to be stored, retrieved, and transmitted [42, 16, 33]. As pointed out in the previous chapter, a two hour digital movie displayed at the rate of thirty frames per second contains two hundred and sixteen thousand frames. Frames can vary in size depending on the requirements of the application. But if we take as an example uncompressed frames consisting of 516 by 640 pixels and 24 bits per pixel, we see that a digital movie, even without considering sound track and any other associated media, could be gigabytes in length. These facts serve to emphasize the crucial need for effective data compression.

Handling CM data can make heavy demands on computing resources such as disk bandwidth, memory, CPU and storage. Data compression can reduce the problem by one or more orders of magnitude. As pointed out previously, even compressed data can make heavy demands on resources but the problem is greatly reduced. If a system has a fixed amount of resources, the number of clients that can be served is greatly increased when compressed data is used. Variable bit rate compression is much more efficient than fixed rate compression. This maximizes the use of server resources so that the increase in overhead caused by VBR data is worthwhile.

To summarize, in this section we have discussed the three main requirements of continuous multimedia data: the need to meet continuity and real time requirements, the needs for synchronization, and the fact that large amounts of data may need to be stored, retrieved and transmitted. These requirements and characteristics affect the way in which CM servers must be designed in order to meet the QoS needs of clients. In the next section we review some of the studies which address these issues.

2.2 Design of Continuous Multimedia Servers

In the previous section, we discussed the three main requirements and characteristics of continuous multimedia data, namely continuity and real time requirements, synchronization of media strands, and the large size of data streams. In this section we review some of the works which address these issues when designing continuous multimedia servers.

We consider the implications of CM characteristics in multimedia server design. The system needs to retrieve and transmit data at the correct rate to provide acceptable QoS to the user, since media units must be displayed at the correct rate and there must be no interruptions. In addition, the proper synchronization of multiple strands of data has to be performed. The large amounts of data associated with each CM stream necessitate the high speed retrieval, transmission, and/or storage of the data and make the use of compression schemes highly desirable. All these facts are true when the system deals with a single user requesting a CM stream. However, the storage capabilities of a system and the disk transfer rates are higher than a single stream can use. Consequently, the usual architecture is to have continuous multimedia servers that can service multiple clients simultaneously. Clients can be local or remote, the latter type being connected to the server via high speed networks. The CM server stores continuous multimedia data on disks or other storage devices and handles client requests for the retrieval of the stored data. It retrieves the data from disk or other devices and places the data in buffers for the clients. Clients then access the data from the buffers at an appropriate real time rate [16, 14]. The aim of CM servers is to serve as many clients as possible while still maintaining the QoS requirements of individual users.

As discussed in the previous chapter, a CM server must use admission control policies. When a new client makes a request, admission control policies enable the server to decide whether or not to admit the client into the system. If the request can be served while maintaining the continuity and real time requirements for all clients already in the system, then the new client is admitted for service. Otherwise, the new client must wait until enough resources become available. Two factors are therefore important in the design of CM servers.

1. For effective admission control policies to be in place, the system must be able to accurately estimate the resource needs of a given client and the total resource needs of a large number of clients.

2. The system must also have mechanisms to minimize these resource needs by such means as more efficient disk scheduling, placement of data on disk, and more efficient buffer management.

Below we discuss the approaches taken by some of the works in the field.

We recall that CM data consists of media units such as frames or audio samples and must be displayed as a sequence of such units. However there is not always a one to one correspondence between media units and storage blocks. The data can be stored in blocks containing more than one media unit, or alternatively a video frame can be stored in several disk blocks. Retrieval must be performed in blocks based on the organization of data blocks on disk and the disk scheduling performed by the system. At the same time it is essential for data to be available to the client based on the display schedule of the application. Hence retrieval of CM data is usually "bursty". This means that retrieval can be at an irregular rate. But since retrieval must keep ahead of consumption by the client, one or more blocks may be retrieved for a stream before they are ready to be used. For this reason, servers must place retrieved data into buffers which the clients access depending on the playback rate required by their applications.

Servers usually process client requests in *rounds*. A round or *cycle* is a period of time during which the server processes the needs of each client exactly once. Enough data must retrieved for each client before each round in order to meet client's needs during that entire round. The number of clients that can be served at any one time is limited by the availability of certain resources such as disk bandwidth and buffer space. So in order to admit as many clients as possible, the server must minimize the resources allocated to each client while still meeting its continuity and real time requirements.

The disk bandwidth and buffer requirements for a client can be analyzed based on its playback rate, as well as, the average seek time, rotational latency, disk transfer times, and the amount of total buffer space available to the system. A number of disk scheduling algorithms have been proposed to improve the efficiency of the disk operations of the server. But these may involve tradeoffs in terms of buffer space requirement. The most traditional algorithm is the "round robin" algorithm which processes client requests in a fixed order during each time period. This can be inefficient in terms of disk usage because it may require larger disk head movements. An alternative is the scan scheduling algorithm. This involves starting the disk head at one end and moving the head in one direction, retrieving blocks belonging to the different clients as they are encountered. When the head reaches the other end, the direction is reversed and the same process is repeated. Since the amount of head movements and hence seek time is greatly reduced, this algorithm can significantly decrease the round length. However, the order in which each client's data will be encountered on the disk cannot be pre-determined. It is possible for a given client to be served at the beginning of one round and not again until the end of the next. Thus, this algorithm requires twice as many buffers as the round robin algorithm. On the other hand, round lengths for the scan algorithms are usually shorter, because of the decrease in seek time. Consequently, smaller buffers may be required for each stream, and the total amount of buffer space may not necessarily be twice the round robin amount [16].

To exploit the potential benefits of both of the above disk scheduling algorithms, a new algorithm which is a compromise between the two has been devised. This is the *grouped sweeping scheduling algorithm (GSS)* [12]. For this scheme the clients in the system are partitioned into several groups. The groups are processed in round robin order but within each group the scan algorithm is performed. This scheme is equivalent to the scan algorithm if only one group is used and is equivalent to the round robin algorithm if each group contains only one client. One potential drawback to this scheme is that determining the composition of each group may require complex computations [34].

In addition to disk scheduling algorithms, a variety of data placement schemes have been proposed for improving the efficiency of disk operations. These can be used in conjunction with appropriate disk scheduling algorithms to decrease disk bandwidth requirements for CM files. The most common data placement solution is contiguous placement of data belonging to a file. This method eliminates intra-file seeks. It works well with read-only systems, but is inefficient for read-write systems requiring a great deal of data copying for inserts and deletes [16]. When large amounts of data are transferred from contiguously placed data, disk rotation times and inter-track seeks also become significant. To deal with these problems, files can be distributed on large arrays of inexpensive disks [16]. Multiple copies of an entire file can be stored redundantly on several disks, or alternatively, pieces of a file can be distributed across a number of disks. This last approach uses two different techniques: data striping, where disk heads are synchronized, with consecutive blocks being read in parallel; and data interleaving, where the disk heads are not synchronized, so that reads of consecutive blocks are not simultaneous but staggered. Since this thesis is mainly concerned with buffer management we do not cover disk scheduling and data placement schemes in any detail. However, for the purpose of this thesis it is assumed that clients will be served in round robin order, and that data will be placed on disks contiguously or in clusters to minimize disk operations.

Another variable that needs to be minimized is the start up latency for a new stream requesting service. The safest way to ensure continuity and real time requirements would be to buffer the entire stream before starting the display to the client. But this would lead to very high start up latencies for long streams such as movies. It also wastes buffer space resources. In a multi-user system, this means that fewer streams can be processed. A balance needs to be achieved between the continuity and real time needs of the stream, and the availability of resources such as disk bandwidth and memory buffers. This is why the method of processing clients in rounds has been generally adopted. It allows the server to share resources equitably among all the clients in the system. In the case where a new client requests admission to the system, enough data must be buffered for the new stream as well as for all existing clients for one round or cycle before the client can be admitted for service. Assuming the client to be otherwise admissible, no more buffer space than this is needed to start processing. Thus startup latency is minimized without compromising continuity requirements.

To summarize, works concerned with continuous multimedia server design emphasize the following points:

1. Efficient storage and correct placement of data on disk to ensure high speed retrieval and manipulation of the data.

2. Compression of data to ensure economizing of resources.

3. Efficient disk scheduling techniques that can meet the needs of continuous multimedia data.

4. Admission control algorithms that enable a CM server to serve as many clients as possible while meeting their continuity and real time requirements.

In the next section we review some of the admission control policies that have been proposed for CM servers.

2.3 Admission Control for CM Servers

As has been discussed earlier, the role of an admission control policy is to enable the CM server to serve as many clients as possible while ensuring that their continuity and real time needs are met. This means that the server must allocate enough resources to each client to prevent service interruptions or a decrease in QoS. At the same time, the server must not allocate too many resources to any one client as this would mean that fewer clients could be accepted. In this section we discuss some of the admission control policies that have been proposed for CM servers.

A number of authors point out that the most efficient way for a CM server to serve multiple users is for the server to proceed in cycles or rounds [41, 15]. For each client, enough multimedia data must be retrieved from disk and placed into memory buffers to meet its needs during that entire round. This involves calculating each client's disk bandwidth requirements for a given time period, based on the client streams play back rate, average disk seek times, rotational latency, and disk transfer rates. In addition, the total system requirements for memory buffers and disk bandwidth, and the appropriate cycle time must be calculated.

When considering admission control, many studies assume the availability of appropriate amounts of buffer space in the system. But the amount of buffer space available can have an effect on disk utilization. Thus, it is important to take both these factors into account when making admission control decisions. As Yang [53] points out, faster disk rates imply that smaller amounts of buffers are needed. Conversely, larger buffers, in conjunction with contiguous or clustered data, improve disk utilization, since fewer seeks are needed.

Some of the earlier studies on admission control [15, 41] consider simple cases where all client streams have the same playback rates and the same size data blocks. Later studies consider servers processing more varied client needs. Here client streams can have different playback rates, and data blocks need not be of a uniform size. Rangan et al [42] propose a "quality proportional multi-subscriber servicing"(QPMS) scheme. In this scheme data read for each client during any given round is proportional to the client's playback rate. Gemmel et al [16]. point out that this appears to be the most efficient way to service multiple clients during rounds. However, Rangan et al's [42] approach is somewhat awkward when new clients are admitted into the system. They note that when a new client is admitted the cycle length may need to change and therefore enough data must be buffered for each existing client as well as the new client before the new client can start being serviced. They make the appropriate changes by adding a single block to the requirements of each client, during each cycle, which leads to fairly long transition periods.

Ng and Yang [31, 32] have an improved approach which considers both disk bandwidth and buffer resources when calculating client needs and cycle lengths. They agree with other researchers that in order to minimize buffer requirements, the disk reading time for each stream during a round must be proportional to its consumption (playback) rate. They are able to significantly reduce buffer requirements by allowing buffer sharing among streams. In this scheme, buffers whose data has been consumed are freed for use by other streams. They also propose a method for decreasing startup latencies and improving on transition periods when new streams are admitted by pre-fetching appropriate amounts of data for streams waiting for admission. However, all the above studies are concerned with either uncompressed data or data compressed at a fixed rate.

Recall that variable bit rate compression provides much greater efficiency. But VBR compressed data is more difficult to process since the amount of data for any given frame can vary widely. Two ways have been proposed for dealing with this problem. One is to store and transmit VBR compressed data in packets that add padding to equalize the size of data corresponding to a certain length of time. Since frame sizes can vary greatly, this method can waste a lot of resources. The other method for handling the variability in frame sizes is to store and retrieve the data in its VBR compressed form.

Two approaches have been proposed for admission control schemes that handle VBR compressed streams: deterministic and statistical. Deterministic approaches ensure that all continuity and real time requirements for the streams are fully met. This necessitates calculating resources based on worst case expectations.

Statistical approaches provide the user with probability based QoS guarantees. Peaks in the data consumption rate of one stream can be balanced by troughs in those of other streams, hence, the system can afford to allocate much less than the maximum rate to each stream and more clients can be served simultaneously. Overflows and data loss do occur but many users are willing to tolerate some degradation in quality, especially if it can be offset by a reduction in cost. Some statistical approaches base their calculations on general information about statistics for the given media. Other approaches provide greater accuracy by using data rate histograms or frame or block size information for the entire stream.

Deterministic admission control policies must guarantee that resource needs are fully met at all times. This means basing calculations on worst case assumptions. Disk bandwidth and buffer space must be allocated assuming the largest frame size, and the maximum consumption rate for resources. Pan et al. [34] have tried to optimize deterministic admission control in two ways. They developed a "time-scale dependent buffer inventory based dynamic scheduling scheme" (σ-BIDS). Each client has dedicated buffers allotted to it. Instead of retrieving a fixed amount for a given stream during each round, Pan at al. keep an inventory of the buffers allocated to the stream and only top up what has been consumed. This avoids two potential problems. The first is the loss of data due to buffer overflow. Second, in the absence of such a topping up policy, if the system is to keep up with data retrieved at the maximum consumption rate it would be necessary to constantly increase buffers allocated to a stream. The other contribution of these authors is the attempt to smooth out the maximum consumption rate (MCR) by using a larger time scale. They note that the MCR can be more than 10 times the average rate in MPEG, if 1/30 sec (one frame length time interval) is considered. So using the MCR can be too conservative. But the MCR can be smoothed out considerably using a longer cycle length of, for example, 5 to 20 seconds. So this method requires computations of the MCRs for each stream over various time intervals and somewhat large buffers (150-600 frames).

Like the above study, Biersack et al.'s approach [7, 8] also involves gathering detailed data about the stored streams. But their admission control algorithm is statistical rather than deterministic. They compute data rate histograms for each stored stream at the time of storage. They are able to make precise estimates of the total resource needs of the system by performing histogram convolutions. They then estimate the probability of overflows when a new client requests admission to the system, by convolving the new client's histogram with the system load histogram to obtain the data rate histogram for the new system load. Biersack et al. note that this statistical approach works for medium sized servers since they have fairly accurate information about the behaviour of each client stream. Their experimental results show that this statistical approach can nearly double the number of clients admitted by the system when compared to deterministic approaches. But the actual amount of gain is greatest when the ratio of the largest data block to the mean bit rate is the highest. This is because when there are wide variations in the bit rate, more sharing of resources can occur. The system is able take advantage of resources not used by streams whose needs are low during a given round, in order to accommodate streams whose demands are higher.

Another statistical approach is the dynamic QoS scheduling discussed in [48]. This study adopts an optimistic resource allocation strategy. The user specifies a minimum and maximum acceptable QoS level. The system uses average resource needs calculations for the admission control policy. When a new application requests admission, average figures are considered. However, this could lead to frequent overflows. The study tries to solve this problem by dynamic QoS scheduling. Applications can be given service within the QoS range specified. Clients with the most stringent requirements are given the highest priority. The server dynamically adjusts a client's QoS based on current load. This policy could lead to unpredictable shifts in quality or even possible discontinuities for clients.

Another statistical approach is that presented by Vin et al. [49]. This approach models the disk bandwidth requirements for a continuous multimedia stream over a certain period of time as a random variable. The Central Limit Theorem of probability theory is then applied, so that the sum of disk bandwidth requirements for all streams in the system can be said to approximate a normal random variable. System load predictions can then be made using a normal distribution. This approach is an improvement in quality over the "optimistic" approach, which allocates resources based on average requirements. It also needs much less record keeping than the MCR recording and data rate histogram methods. However, it has two drawbacks. First, since the Central Limit Theorem is applied to the entire system, it only gives accurate predictions when there are a large number of clients. Second, since it works at the granularity level of streams, it predicts the entire system load and therefore cannot provide user-selectable QoS guarantees. Our own approach, which works at the granularity level of frames, is an attempt to solve both these problems. Our model allows clients to specify individual non-overflow guarantee levels, and works well even when the system has only one or two streams.

2.4 Summary: This Thesis in the Context of its Background

This chapter has presented background information necessary to the understanding of the material in this thesis. It has briefly reviewed works in the area of continuous multimedia, CM server design, and admission control for CM servers. As this thesis is concerned with CM compressed using the variable bit rate method, this chapter has also covered works dealing with admission control for VBR streams. In this section we briefly review the important points covered in this chapter and discuss how they relate to the work of this thesis.

Continuous multimedia has three major characteristics which affect the design of CM servers. It has continuity and real time requirements. As well, CM servers must be able to synchronize the various strands of data comprising a stream and handle data in large quantities. The aim of a CM server is to service as many clients as possible, while maintaining their continuity and real time requirements, or providing an acceptable quality of service. This necessitates minimizing resources allocated to each client. Admission control policies are needed to decide if a new client can be admitted for service without violating the requirements of clients already in the system.

Admission control is complicated when dealing with variably compressed data, because it is hard to make precise estimates of resource needs for VBR compressed streams. But the extra amount of overhead required for handling VBR is worthwhile, as the savings in resources is much greater than with fixed rate compression.

Admission control with VBR data can be deterministic or statistical. Deterministic admission control involves making worst-case estimates of client needs, which can be very wasteful of server resources. By contrast, statistical admission control provides probability-based quality of service guarantees to clients. Several statistical admission control algorithms were reviewed. Some of these estimate resource needs based on average consumption rate, which can lead to overflows and unpredictable degradation of quality to the user. Other approaches provide better QoS but involve a great deal of record keeping, such as, frame size information or data rate histograms for every stream. Vin et al.'s algorithm uses the Central Limit

Theorem, to model the total resource requirements for all the streams in a system as a normal random variable. Total resource needs are then estimated using the properties of a normal distribution. This algorithm works best with a large number of clients and can only provide system-wide QoS guarantees, rather than guarantees to individual clients.

The algorithms in this thesis are an attempt to solve some of the problems presented by the above approaches. They provide better QoS than the average resource allocation policy but still avoid the cumbersome task of recording of detailed data about each stream. Similar to Vin's approach, we use the Central Limit Theorem but individual MPEG streams are modeled as normally distributed random variables, so that it is possible to provide user-selectable QoS guarantees. In addition, some of our algorithms work regardless of the number of clients in the system. In chapter 3, we discuss the Central Limit Theorem, and how this theorem and the characteristics of MPEG can be used to produce statistical models of MPEG streams. In chapter 4, this statistical model will be used in developing admission control policies which provide individual QoS guarantees to clients.

Chapter 3

Statistical Modeling of MPEG Streams

In the previous chapter we discussed admission control algorithms which attempt to deal with continuous multimedia data compressed with the variable bit rate method. MPEG is one of the most well known standards for variable bit rate compression and many studies consider its characteristics when presenting their statistical approaches. In the first section of this chapter we will give an overview of the characteristics of MPEG which assist in the statistical modeling of streams and the estimation of resource needs for clients. In the second section we give a general discussion on the Central Limit Theorem. Then we present a statistical model of MPEG streams which makes use of this theorem.

3.1 MPEG Video Compression

In this section, we will discuss those characteristics of the MPEG compression system which assist in modeling MPEG compressed streams. MPEG takes advantage of certain characteristics of the human visual and auditory systems, and the properties of continuous media in order to improve compression efficiency.

MPEG selectively compresses visual and audio data in greater or less detail, depending on what is most perceivable by human senses. The visual system is less sensitive to high frequencies, so high frequency data is compressed much more than the lower frequencies. Similarly, audio information not perceivable by the human ear, can be ignored or compressed much more coarsely. In this work we concentrate only on the video component of MPEG compression.

The MPEG standard for video compression takes advantage of the fact that adjacent frames in a continuous video stream are often similar [20, 35]. The time interval between frames can be very small, for example, one thirtieth of a second or less. Usually the background in a frame will remain static. As the time interval is so small, persons and objects in the foreground are likely to move only slightly from one frame to the next. To take advantage of the similarity of adjacent frames, MPEG uses two types of compression. Some of the frames, called I frames, are "intra-coded". This means that reconstruction of I frames depends entirely on the information they keep. However, most MPEG frames are "delta" compressed. Their reconstruction depends not only on the information they keep, but also on the information kept by some previous and/or subsequent frames. Simply put, delta compressed frames primarily encode changes between frames and depend upon nearby frames for information that has not changed. This avoids the repetition of redundant information and greatly increases compression efficiency.

MPEG provides two kinds of "delta" compressed frames. First, there are the predictive P frames, which are coded with reference to a previous I or P frame. Second, there are the bi-directional B frames, which are coded with reference to a previous *and* a subsequent I or P frame.

The basic unit of MPEG video is the single picture or frame. MPEG compresses each frame individually. There are several layers of coding. Each frame consists of slices (horizontal strips in a frame). Slices consist of macro blocks, which in turn consist of four 8*8 blocks of luminance pixels and two 8*8 blocks of chrominance pixels. All variable bit rate coding is done at the macro block level. The P and B frames have motion compensation applied to them; each frame is processed block by block. Each block is first transformed using Discrete Cosine Transformation (DCT) which yields a matrix. Coefficients from this matrix are then quantized yielding sequences of zeros and ones. Then all frames are further compressed using Huffman coding for the sequences of zeros and ones. If a code exists for a given pattern this code is transmitted yielding variable rate compression. If there is no code for a given pattern it is transmitted at a fixed rate.

As is clear from the above description of the process, all MPEG frames are variably compressed. The P frames use some intra-coding and also some inter-frame coding, while the B frames only use inter-frame coding. This enables the B frames to convey a lot more information with smaller amounts of data.

The main advantage of a B frame is that it is the most highly compressed and the smallest in size of the three types of MPEG frames. By contrast, an I frame is the least compressed and largest in size of the 3 types. Indeed, the B frame is the main reason why an MPEG compressed stream typically requires less space than the same stream compressed by JPEG or other schemes that give constant compression rates. However, a major disadvantage of a B frame is that it is incapable of supporting random access. This is because a B frame only makes sense in the context of the I or P frame used to encode it. The loss of an I frame would make all B frames referencing it unusable. So to view a particular B frame the appropriate I and P frames also need to be retrieved. This limits flexibility in access when performing VCR-like operations.

In order to strike a healthy balance between compression efficiency and flexibility in access, the video part of an MPEG stream usually contains limited numbers of B frames in between I and P frames. The needs of the application determine the proportions of I, P and B frames, though there are usually many more delta-compressed frames than intra-coded frames. In a majority of the MPEG streams we encountered, a high percentage (70-80%) of compressed data consist of delta-compressed B and P frames. MPEG streams generally consist of frame sequences of the form: IBBBPBBBIBBBPBBBI... or IBBPBBPBBPBBI... The exact order and number of B frames in between I and P frames varies from application to application, and is determined at compression time. In general, compression efficiency is increased when there is a higher number of B frames compared to I or P frames. Furthermore, according to [20], the higher the number of B frames compared to I and P frames, the smaller is the correlation of B frames with their referenced I and P frames. This factor will assist in the statistical modeling of MPEG streams which will be discussed in section 3.3.2.

To summarize, in this section we have described the MPEG method for video compression. MPEG takes advantage of the similarity between adjacent frames in continuous video to increase compression efficiency. MPEG streams have intra-coded I frames and delta coded B and P frames, which primarily encode the differences between nearby frames. All three types of MPEG frames use variable rate compression. It is therefore possible to model the three types of frames using the same statistical principles. Given the importance of the role played by B frames, and the fact that P and I frames can be modeled in a similar way, we simplify the discussion in this work by focusing on B frames only. In the next section we will discuss the Central Limit Theorem which can be used to provide a statistical model for MPEG streams.

3.2 The Central Limit Theorem

The term "Central Limit Theorem" (CLT) refers to a set of theorems in probability theory, which are concerned with the asymptotic behaviour of a sum of random variables. Intuitively, the theorem states that, provided certain conditions are satisfied, as the number of random variables gets sufficiently large, the probability distribution of their sum approaches a normal distribution, regardless of the probability distributions of the individual random variables. There are many versions of the Central Limit Theorem, as the topic has been the subject of active research in probability theory for many years. In the following, we give an overview of the standard versions of the CLT. For more details, see [6, 9].

Let a and σ be real numbers with $\sigma > 0$. Let $N(a, \sigma^2)$ be the probability distribution with density: $\frac{1}{\sigma\sqrt{2\pi}}e^{-(x-a)^2/(2\sigma^2)}$. For $n = 1, 2, ..,$ let Φ_n be a finite set of random variables on some probability space and let $S_n = \sum_{x \in \Phi_n} x$. For each positive number τ, let x_τ be defined as: $x_\tau = x$, if $|x| \leq \tau$, but $x_\tau = 0$ otherwise. Then let $S_{n,\tau}$ be defined as $\sum_{x \in \Phi_n} x_\tau$. Further assume that for each n, Φ_n is an independent set of random variables. Assume also that for all $\epsilon > 0$ and some $\tau > 0$, the following conditions hold: (i) $lim_{n \to \infty} \sum_{x \in \Phi_n} P(|x| > \epsilon) = 0$; (ii) $lim_{n \to \infty} E(S_{n,\tau}) = a$; and (iii) $lim_{n \to \infty} V(S_{n,\tau}) = \sigma^2$. Then $N(a, \sigma^2)$ is the weak limit of the probability distributions of the S_n.

We can paraphrase the above formalization of the Central Limit Theorem as follows. Let S be the sum of many small independent random variables. If the first and second moments of S and the Gaussian distribution $N(a, \sigma^2)$ coincide, then S is approximated by $N(a, \sigma^2)$ – regardless of what the original distribution is. Some of the above conditions are difficult to verify. However, there are a number of alternative versions of the CLT which allow some of the necessary conditions to be replaced by equivalent conditions. The most well-known alternatives are the Lindeberg condition and the Lyapounov condition.

1. The Lindeberg Condition:

   In the theorem above, we allow the case $\tau = +\infty$ under the assumption that $x \in \Phi_n$ are square integrable. Assume also that $E(x) = 0$. Then the above condition (i) of the CLT can be replaced by:

   $$lim_{n \to \infty} \sum_{x \in \Phi_n} E((x - x_\epsilon)^2) \quad = \quad 0$$

   for all $\epsilon > 0$.

2. The Lyapounov Condition:

   The Lindeberg condition is implied by the following condition:

   $$lim_{n \to \infty} \sum_{x \in \Phi_n} E(|x|^{2+\delta}) \quad = \quad 0$$

   for some $\delta > 0$.

The main point to note here is that independence of random variables alone is not sufficient for the CLT to hold, and the satisfaction of other conditions, such as the Lindeberg condition or the Lyapounov condition, is required. However, these conditions are often difficult to verify for any given distribution. Fortunately, there is the following Lindeberg-Levy theorem, which defines a general class of random variables that satisfy the Lindeberg condition and the Lyapounov condition.

Theorem 1 (Lindeberg-Levy Central Limit Theorem [6, 9]) Let $X_1, \ldots, X_n$ be independent variables having the *same distribution* with mean μ and finite positive variance σ^2. Then for sufficiently large n, $(X_1 + \ldots + X_n - n\mu)/(\sigma\sqrt{n})$ follows a standard normal distribution $N(0,1)$.

The above theorem, which is one of the most well-known and widely applied versions of the CLT, states that the sum of n identically distributed independent random variables is approximately normally distributed, as long as n is sufficiently large. For many applications, it has been found empirically that the theorem gives a good approximation even with as few as 10 to 25 random variables [22]. Finally, it is important to note that for the CLT to hold, the actual distribution is immaterial. The above theorem only requires the mean and the variance of the distribution. Consequently, the theorem is extremely useful in providing approximations to certain sampling distributions, particularly when the distributions are unknown or difficult to compute [22]. In the next section we discuss how an MPEG stream can be modeled using the Central Limit Theorem.

3.3 Statistical Model for Multimedia Streams Using the Central Limit Theorem

In the previous section we gave an overview of the CLT, which can provide approximations to the distribution of random variables. The Theorem states that the sum of n random variables is approximately normally distributed, provided that n is sufficiently large and that the random variables satisfy certain conditions. In this section, we first show how Vin et al. [49] use the Central Limit Theorem to approximate the disk bandwidth requirements for a continuous multimedia server. We point out that this model provides an approximation at the granularity level of streams. Consequently, this method can only provide non-overflow guarantees at the level of the entire system. In the following subsection we show how the CLT and the properties of MPEG streams can be used to provide an approximation at the granularity of frames. In the Chapter 4, we show that this model can be used to provide non-overflow guarantees to individual users rather than to the entire system.

3.3.1 Vin's Statistical Modeling of a CM System

In this section we review the statistical model of a continuous multimedia system proposed by Vin et al. In [49], Vin et al. use a random variable B_i to represent the total disk bandwidth required by Stream St_i over a certain period of time. Then by arguing that $B_1, \ldots, B_n$ are independent, they apply the CLT to model $B_1 + \ldots + B_n$ as a normally distributed variable. This enables them to give a good approximation for the behavior of the system when there are a large number of streams. The total disk bandwidth requirements for this system can be calculated using the mean and standard deviation for the normal random variable $B_1 + \ldots + B_n$ and the non-overflow guarantee level specified for the system.

However, a problem with this approach is that, as discussed in section 3.2, independence of random variables is not a sufficient condition for the CLT to hold. Apart from independence, the random variables need to satisfy one of the following: the Lindeberg condition, the Lyapounov condition, or the condition that the random variables are identically distributed. Since the framework of Vin et al. allows streams with different statistical characteristics to be served simultaneously, the variables

$B_1, \ldots, B_n$ need not be identically distributed. This being the case, to approximate $B_1 + \ldots + B_n$ as a normally distributed variable, the variables must satisfy either the Lindeberg or the Lyapounov conditions. However, this point is not discussed at all in [49].

Two other disadvantages of this approach should be mentioned briefly: As discussed earlier, because Vin et al. use random variables at the granularity level of streams, the non-overflow guarantee thus obtained is on a per system basis. This is not refined enough to support the approach of allowing an individual user to select a non-overflow guarantee on a per stream basis. In addition, because the CLT is applied at the level of streams, this method works best when there are a large number of streams in the system. In Section 3.3.2 we discuss a more refined model of MPEG streams which enables us to provide user selectable non-overflow guarantees. Since, for this model, the CLT is applied at the level of frames, a single stream's resource requirements can be represented as a normal random variable. Hence, the model works when there are only a few, or even a single stream in the system.

3.3.2 Statistical Model for MPEG Streams Proposed by this Thesis

In the previous section we discussed Vin et al.'s Statistical Model for a continuous multimedia system which enables good approximations for resource needs on a per system basis. Since this model is at the granularity level of streams, the approximations of resource requirements are only reasonably accurate when a large number of streams are involved. In this section we present a more refined model for MPEG streams, which works at the granularity level of frames, and makes it possible to provide good approximations of resource requirements at the individual stream level. The first step towards supporting the latter kind of guarantee is to use a random variable $s_{i,j}$ to denote the size of the j-th compressed B frame of Stream St_i. [1] Then our hope is to apply the Central Limit Theorem to approximate $B_i = s_{i,j} + \ldots + s_{i,j+k}$ as a normally distributed variable. In the following, we examine whether it is valid to do so.

First, our random variables $s_{i,j}, \ldots, s_{i,j+k}$ are identically distributed. This is the case because these random variables correspond to B frames of the same stream, but not of different streams. One may ask what actual distribution the random variables follow. [35] presents empirical evidence, both positive and negative, of a gamma distribution. But the results are not conclusive. However, as discussed in Section 3.2, for the CLT to hold, the actual distribution does not matter. So long as the random variables follow the same distribution, all that is needed are the mean and the variance of the distribution. Both can be computed easily at the time the stream is compressed.

Second, as stated in Theorem 1, the CLT requires that the random variables be independent. However, recall from Section 3.1 that a B frame is obtained with reference to a previous and a subsequent I or P frame. Thus, two adjacent B frames may be correlated directly in their contents and indirectly in their sizes. In other words, $s_{i,j}$ and $s_{i,j+1}$ cannot be assumed independent. Fortunately, there are some versions of the CLT which allows the independence requirement to be relaxed. For the purpose of this thesis it suffices to state here that the Central Limit Theorem still holds if the random variables far apart from each other are nearly independent. Refer to [6, 9] for formal details. In other words, even if $s_{i,j}, \ldots, s_{i,j+k}$ are not

[1] Strictly speaking, a similar approach can be applied to I and P frames as well. But as discussed in Section 3.1, the number of B frames typically far exceeds the numbers of I and P frames. In addition, the proportion of data stored in delta-coded B and P frames is usually high (70-80%), and all three types of frames can be approached in a similar fashion. So, to simplify our discussion, here we deal with B frames only.

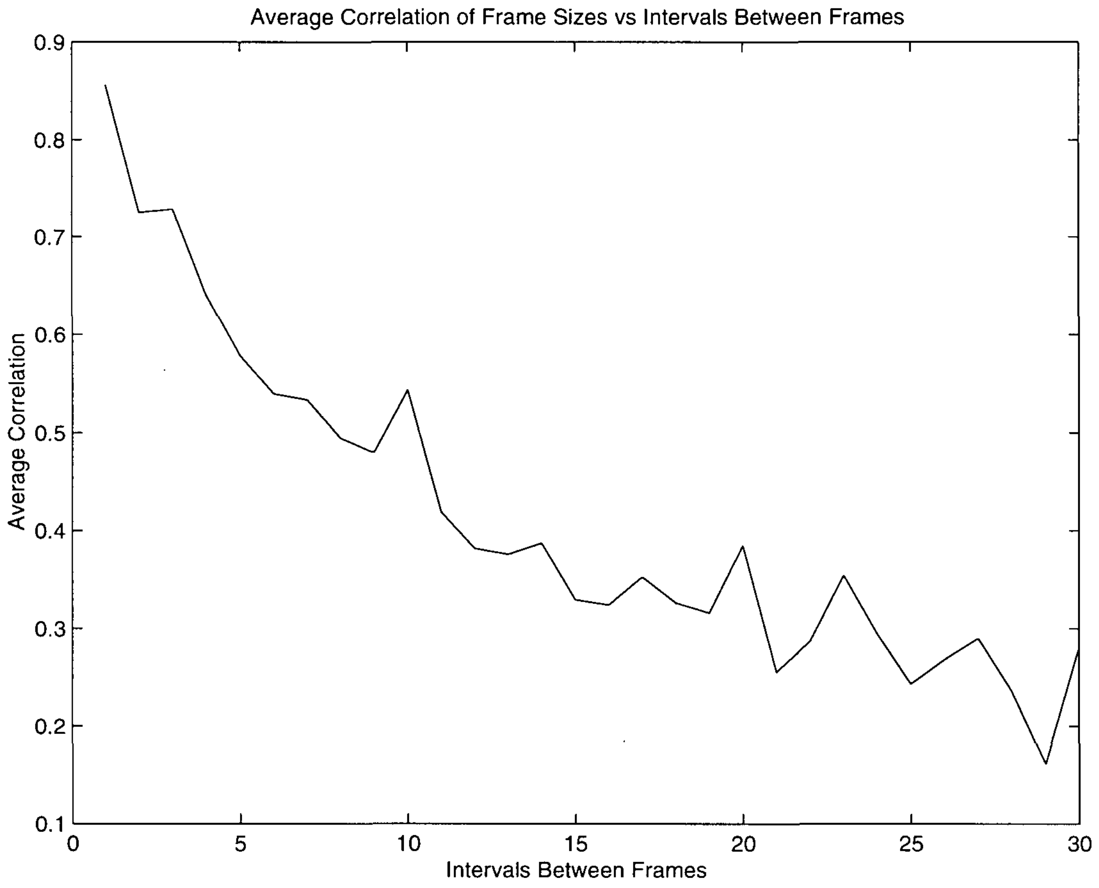


Figure 3.1: Average Degree of Correlation with Different Values of k

entirely independent, it is sufficient to have the degree of dependence between $s_{i,j}$ and $s_{i,j+k}$ decrease as k increases.

To verify the above condition, we performed experiments by acquiring from a variety of sources (e.g., Internet) 30 MPEG streams. Figure 3.1 shows the average degree of correlation between $s_{i,j}$ and $s_{i,j+k}$ as k increases. (This is the average over all B frames in several randomly picked streams.) Indeed, for adjacent B frames (i.e., $k = 1$), the degree of correlation is quite high, i.e., exceeding 0.7. But as k

increases, the degree decreases. The rate of decrease varies from stream to stream. In Figure 3.1, when $k = 10$, the degree of correlation drops to around 0.4, which is an acceptable level of "near independence". Throughout the rest of this paper, we use k_0 to denote the minimum value of k for the correlation to drop below a certain threshold. As will be shown in Section 7.2.1, our experimental results show that $k_0 = 10$ can already give very good behaviour for buffer allocation.

To summarize, in this section we presented a statistical model for MPEG streams. We applied a special form of the Central Limit Theorem that relaxes the independence assumption imposed on the random variables. We have shown that, $B_i = s_{i,j} + \ldots + s_{i,j+k_0}$ can be approximated as a normally distributed variable. Since the sum of normal variables is a normal variable [6, 9], $B_1 + \ldots + B_n$ is normally distributed—even when $n = 2$. This strictly generalizes the model of Vin et al., for in their case, n needs to be sufficiently large for the Central Limit Theorem to be in effect. Note that having normally distributed random variables also enables us to sum B_i's corresponding to different types of frames. In the next chapter we use the model we just developed in calculating resource needs for individual streams. We also develop an admission control algorithm that makes use of the statistical model to provide for user-selectable non-overflow guarantee levels on a per-stream basis.

Chapter 4

Providing Individual Non-overflow Guarantees: Algorithm CLT(E)

In the previous chapter, we presented a statistical model for MPEG streams, using a special form of the Central Limit Theorem that relaxes the independence assumption imposed on the random variables. In this chapter, we first review the issues that are important in developing admission control algorithms for continuous multimedia servers. We then present the Algorithm CLT(E) that uses this statistical model to provide for individual non-overflow guarantees. In Chapter 7, we will present experimental results evaluating the effectiveness of CLT(E), particularly on the issue of whether or not it would be too expensive to provide individual guarantees.

4.1 Issues in Developing Admission Control Algorithms

In this section, we review the questions that must be addressed when developing admission control algorithms for servers that serve multiple clients requesting continuous multimedia. The servers are assumed to have a fixed amount of resources: CPU, disk bandwidth, memory buffer space. In general, the purpose of an admission control algorithm is to determine, given that there are already n active streams in the system, whether or not there are enough resources to admit the $(n+1)$st stream. For systems that support only streams with constant compression rates, the typical goal of an admission control algorithm is to maximize throughput, while guaranteeing that the continuity requirements of clients are fully satisfied. The latter is rather easy to do for streams with constant compression rates. See, for example, [41, 31, 32]. The situation is far more complicated for MPEG streams, which use variable bit rate compression. Since the exact resource needs for a VBR compressed stream cannot be predicted, the system must perform some estimation. There are issues which should be considered when deciding what approach to adopt when making such estimations. On the one hand, the allocation of resources based on the minimum compression rate (maximum frame size) will lead to the waste of resources. Keeping in mind the goal of maximizing throughput, this is not acceptable to the system. On the other hand, allocation based on the average compression rate may lead to frequent data overflows, possible discontinuities and severe degradation in quality. This is not acceptable to the user. And if the system were to keep track of the exact size of each frame of each stream, the overhead involved in admission can be substantial, again not acceptable to the system.

We recall from Chapter 2 that admission control algorithms dealing with

VBR-compressed data can have either deterministic or statistical approaches. Deterministic algorithms guarantee that all resource needs will be fully met. They allocate resources based on a worst case estimate of resource needs, hence using maximum frame sizes in their calculations. Statistical algorithms, on the other hand rely on the fact that peaks and troughs in the data consumption rate can often balance each other. They provide a probability-based quality of service guarantee. One way of expressing such a statistical QoS guarantee is exemplified by the non-overflow guarantee problem introduced in Chapter 1. The system calculates a "minimized " amount of resources required to guarantee that the stream does not overflow its allocated resources p% of the time. This non-overflow guarantee need not be 100%.

Given variable compression rates within one stream, the question is how much guarantee on the continuity requirements is desirable or practicable. With regard to the question of how much non-overflow guarantee is desirable, the approach of this thesis is to let the user, rather than the admission control algorithm, decide what an acceptable level to the user is. The job of the admission control algorithm is then to deliver the specified level of guarantee or quality, while trying to maximize throughput.

Below we will present Algorithm CLT(E), which supports the above notion of user-selectable non-overflow guarantee by the allocation of exclusive buffers to each user.

4.2 Algorithm CLT(E)

In this section we present Algorithm CLT(E), which provides user-selectable non-overflow guarantees to clients using the CLT-based statistical model of MPEG streams developed in the last chapter. This algorithm allocates exclusive buffers to each client admitted to the system.

The approach taken by Algorithm CLT(E) is to allow the non-overflow guarantee to be made at the finer granularity level of streams rather than at the coarser granularity of the entire system. Algorithm CLT(E) does so by first using the statistical model developed in the previous chapter to compute the amount of buffer space B_p^e that gives the $p\%$ guarantee requested by the user. If there is more than B_p^e buffer space available, CLT(E) activates the stream and enforces the individual guarantee by ensuring that all of the B_p^e buffer space can only be used exclusively by this particular stream. Hereafter, B_p^e is referred to as the "exclusive buffers" for the stream. It is important to note here that the $p\%$ guarantee cannot be insured if an algorithm such as Vin et al's is used, where total system requirements are calculated using the properties of a normal distribution, and buffers are shared by all streams. As an example, consider the case where the joint system requirements for providing $p\%$ guarantee to all clients is 16 buffers. If the system has 5 streams where providing the $p\%$ guarantee to each stream requires 4 buffers, the system cannot enforce the $p\%$ to every stream in situations where all streams require their 4 buffers. Hence, the additional amount of resources needed to provide exclusive buffers is essential if each client is to receive the guaranteed QoS.

The details of Algorithm CLT(E) are as follows (where "CLT" stands for the Central Limit Theorem and "E" stands for exclusive).

Algorithm CLT(E)

Let St be the stream considered for admission. Let $p\%$ be the non-overflow guarantee level requested for St. Let μ and σ be the mean and the standard deviation of the size of the frames of St.

1. From the standard normal probability table [22], find z_p such that $prob(Z \leq z_p) = p\%$, where Z denotes a standard normal random variable.

2. Compute the amount of the exclusive buffer space required:

$$B_p^e \quad = \quad k_0 * \mu + z_p * \sigma\sqrt{k_0}, \qquad (4.1)$$

   where as introduced in Section 3.3.2, k_0 denotes the number of adjacent frames needed for the Central Limit Theorem to be applicable.

3. If the remaining buffer space of the system is less than B_p^e, the stream St must wait. Otherwise, St is admitted with an allocation of B_p^e buffer space reserved solely for the use of St.

The following explains the steps of Algorithm CLT(E). As discussed in Section 3.3.2, the total size of k_0 successive frames, $s_{i,j+1} + \ldots + s_{i,j+k_0}$, can be approximated as a normally distributed random variable. Then by Theorem 1, $(s_{i,j+1} + \ldots + s_{i,j+k_0} - k_0 * \mu) \,/(\sigma\sqrt{k_0})$ follows a standard normal distribution. Thus, to find B_p^e such that $prob((s_{i,j+1} + \ldots + s_{i,j+k_0}) \leq B_p^e) = p\%$, it is sufficient to consider and manipulate $prob((s_{i,j+1} + \ldots + s_{i,j+k_0} - k_0 * \mu) \,/(\sigma\sqrt{k_0}) \leq z_p) = p\%$. Re-arranging the terms in the latter probability statement gives the definition of B_p^e in Step 2 of the above algorithm.

Algorithm CLT(E) (as well as our other admission control algorithms to be introduced later) adopts the simple policy of considering admissions on a FIFO fashion. While this policy may not be optimal, it does ensure fairness in the simplest way. Furthermore, once a stream is determined to be admissible, Algorithm CLT(E) reserves the appropriate amount of buffer space for the stream. This amount of buffer space is released only when the stream exits the system.

As discussed earlier, Algorithm CLT(E) considers B frames only. Because a typical MPEG stream consists of mainly B frames, this is a reasonable simplification for easier presentation. For the sake of brevity, the more complete version of Algorithm CLT(E), which uses a similar approach to deal with I and P frames, will not be covered in this thesis. But it should be noted here that the exclusive buffers for a stream in fact consist of the sum of the B_p^e allocations, calculated separately for its I, P, and B frames. Also since the three types of frames are usually in a fixed proportion to one another within a stream, the k_0 values for B and P frames may need to be revised upwards to maintain the correct ratio of I to P to B frames.

Furthermore, as presented above, Algorithm CLT(E) regards buffer space as the only resource. In reality, there are other resources, such as disk bandwidth, that should also be considered during admission control. Again, the consideration of these resources is omitted for brevity. But it should be clear that the allocation of disk bandwidth for MPEG streams can be handled in exactly the same way as the allocation of buffer space detailed in the above algorithm.

Notice that as far as buffer allocation is concerned, the focus of this thesis is on deciding how much buffer space is required to support a certain level of non-overflow guarantee. In situations where the allocated buffer space is less than the actual space required by a certain group of frames, it is necessary to decide which frames or which parts of the frames can be dropped to fit into the allocated buffer space. We do not address this issue here in this thesis; we only refer the readers to [50] that proposes an interesting method to choose parts of the frames to be discarded. In Section 8.2.1, we will discuss related implementation issues.

It is important to note that Algorithm CLT(E) allows a heterogeneous mix of streams – that is, with different values of μ, σ, k_0 and p – to be active simultaneously in the system. More details will be given Chapter 6, but we note here that this is due to the fact that the amount of exclusive buffer space is calculated based on random variables representing the sizes of frames in the same stream. Since frames of a given type in the same stream have identical compression characteristics, the condition that the random variables must be identically distributed is satisfied. The Central Limit Theorem is applied on an individual stream basis and separately to each type of frame. Therefore streams with different characteristics and guarantee levels do not affect the application of the CLT(E) Algorithm.

In this section we have presented Algorithm CLT(E) which allocates exclusive buffers to client streams consisting of MPEG encoded data. CLT(E) estimates resource needs of streams based on the user-selected non-overflow guarantee level. It uses the Central Limit Theorem and the statistical model of MPEG streams developed in the previous chapter when performing its calculations. If there are enough resources in the system to cover the calculated needs of a new stream, the client will be admitted. Otherwise, the client must wait until enough resources are freed up. Our experimental results show that CLT(E) is able to provide these individual non-overflow guarantees at a reasonable cost. In the next chapter we present algorithms which can enhance user satisfaction by providing system-wide non-overflow guarantees in addition to the exclusive buffers allocated by CLT(E).

Chapter 5

Providing Additional System-wide Non-overflow Guarantees: Algorithms CLT(Sn) and CLT(Sm)

In the previous chapter, we introduced Algorithm CLT(E), a statistical admission control policy that provides user-selectable non-overflow guarantees to individual streams via the allocation of exclusive buffers. Under this algorithm, the quality of service received by the user is exactly what the user asks for. That is to say, if the $p\%$ individual guarantee turns out to be unsatisfactory, the user can only blame his/her own choice. In this chapter, we study how the system can enhance user satisfaction and the quality of service by providing additional non-overflow guarantees. The key question that needs to be answered is whether or not such additional guarantee can be provided at a low cost, while still being effective in reducing overflows.

The approach we take is to provide a system-wide non-overflow guarantee level $q\%$ (e.g., 99%), on top of the exclusive guarantee provided to each individual stream by Algorithm CLT(E). This is achieved by providing shared buffers, which can be used by any stream that overflows its exclusive buffers. At admission control time, the system computes two buffer space requirements B_p^e and B_q^s for the stream seeking admission. The first quantity, B_p^e, is the amount of exclusive buffer space which would provide a $p\%$ non-overflow guarantee, as requested by the user. This is given by Equation (4.1). The second buffer space requirement, B_q^s, is the amount of buffer space that must be added to the shared pool to maintain the system-wide non-overflow guarantee. (Here "s" stands for "shared".) The sum of these two requirements is the total amount of buffer space that needs to be available in order for this stream to be admitted. If it does not exceed remaining available buffer space, the stream will be admitted. Otherwise, the stream must wait.

In the remainder of this chapter, we present two different ways to compute the amount of additional space to be added to the shared buffer pool. The two different ways give rise to two different admission control algorithms called CLT(Sn) and CLT(Sm). In Chapter 7, we will present experimental results evaluating the effectiveness of these two algorithms.

An important fact to note is that in this chapter, we only consider the situation where every stream in the system has identical compression characteristics and requires the same individual non-overflow guarantee level. We refer to this as the *homogeneous case.* In Chapter 6, we will discuss whether and how these two algorithms can be generalized to deal with situations where streams can have different compression characteristics and/or request different levels of individual guarantee. This will be referred to as the *heterogeneous case.*

5.1 Algorithm CLT(Sn)

In this section we describe one of the two homogeneous case algorithms which provide a pool of buffers shared by all streams in the system, in addition to the exclusive buffers provided by Algorithm CLT(E). As noted above, when a stream is considered for admission, two quantities are calculated for estimating its resource needs. The first is the quantity B_p^e, which gives the amount of exclusive buffers provided by CLT(E). The second is the amount of buffers to be added to the shared buffer pool provided for the system.

For the purpose of computing the amount of additional space to be added to the shared buffer pool, we define an overflow random variable Ov_i for each stream St_i as follows:

$$Ov_i = \begin{cases} 0 & \text{if } B_i \leq B_p^e \\ B_i - B_p^e & \text{otherwise} \end{cases} \tag{5.1}$$

where, as defined and studied in Section 3.3.2, B_i is a normally distributed random variable that represents the buffer space required by stream St_i for several adjacent frames, and B_p^e is the amount of exclusive buffers for the stream. The meaning of Equation (5.1) is as follows. When the amount B_i of buffer space required is less than the amount of exclusive buffers, there is no overflow caused by the stream. However, when B_i exceeds the exclusive buffer space, the extra amount is the amount of overflow buffer space required.

Ov_i has an interesting probability density function. For the value 0, the probability density is $p\%$. This is because, by the definition of Equation (4.1), there is $p\%$ chance that $B_i \leq B_p^e$. Then for all positive values of $B_i - B_p^e$, the density function is exactly the same as the tail of the normal distribution to the right of z_p.

Though not standard, this density function is still a valid one as it integrates to 1.

As motivated above, the idea of providing an additional system-wide non-overflow guarantee level, say $q\%$ (e.g,, 99%), is achieved through the use of overflow buffers shared by all streams. If there are n active streams in the system, the amount of shared overflow space required is given by $Ov_1 + \ldots + Ov_n$. In order to provide a $q\%$ overall guarantee, it is imperative that the distribution of $Ov_1 + \ldots + Ov_n$ be known. It is obvious that all the Ov_i variables are independent of each other. In addition, given the fact that in this chapter, we are dealing with streams with identical characteristics and non-overflow guarantee requirements, all the Ov_i variables are identically distributed. Thus, the version of the Central Limit Theorem described by Theorem 1 is again applicable. In other words, for sufficiently large n, $(Ov_1 + \ldots + Ov_n - n\mu_{ov})/(\sigma_{ov}\sqrt{n})$ follows a standard normal distribution, where μ_{ov} and σ_{ov}^2 are the mean and variance of Ov_i.

We can now define the quantity B_q^s of buffer space needed to add to the shared buffer pool if the number of active streams is to increase from n to $n+1$. For n streams, the amount of shared buffer space to give an overall non-overflow guarantee level of $q\%$ is given by:

$$B_{q,n}^s = n\mu_{ov} + z_q * \sigma_{ov}\sqrt{n}$$

Similarly, for $n+1$ streams, the corresponding amount is:

$$B_{q,n+1}^s = (n+1)\mu_{ov} + z_q * \sigma_{ov}\sqrt{n+1}$$

Thus, the additional amount of shared buffer space required to admit one more stream into the system is:

$$B_q^s = B_{q,n+1}^s - B_{q,n}^s = \mu_{ov} + z_q * \sigma_{ov}(\sqrt{n+1} - \sqrt{n}) \tag{5.2}$$

Below we show the steps of Algorithm CLT(Sn). This algorithm performs admission control and buffer allocation with both exclusive and shared overflow buffers as described above. The letter "S" stands for "shared", and the letter "n" represents the fact that all n active streams are involved in the computation of the amount of shared overflow buffers.

Algorithm CLT(Sn)

Let St be the stream considered for admission. Let $p\%$ be the individual non-overflow guarantee level requested for St. Let $q\%$ be the system-wide non-overflow guarantee level.

1. Compute the amount of the exclusive buffer space B_p^e as defined in Equation (4.1).

2. Compute the amount of additional shared overflow buffer space B_q^s as defined in Equation (5.2).

3. If the remaining buffer space of the system is less than $B_p^e + B_q^s$, the stream St must wait. Otherwise, St is admitted with an allocation of B_p^e buffer space reserved solely for the use of St, and with B_q^s buffer space added to the shared overflow buffer pool.

Algorithm CLT(Sn), as defined above, requires that a stream be admitted only if there are enough exclusive and shared buffers available. Strictly speaking, the additional system-wide guarantee may be implemented as a bonus, rather than a necessity. In other words, a variant of CLT(Sn) may admit a stream even if there is not enough shared buffers. The reason why CLT(Sn) is defined in the way shown above is that this allows for a fair comparison between CLT(Sn) and other algorithms. See Chapter 7 for details.

5.2 Algorithm CLT(Sm)

In this section we discuss Algorithm CLT(Sm). Like Algorithm CLT(Sn), this algorithm allocates exclusive buffers to each stream and additional buffers that can be shared among streams, as discussed above. The difference between CLT(Sm) and CLT(Sn) is that they use different methods for calculating the amount of shared buffers.

The CLT(Sm) way to compute the amount of additional space to be added to the shared buffer pool aims to estimate more precisely the number of streams, out of the n active streams, that actually overflow their exclusive buffers. (Hereafter, we use m to denote this number.) More specifically, let $pr(i)$ denote the probability that exactly i of n streams overflow, for all $0 \leq i \leq n$. Then to provide a $q\%$ system-wide non-overflow guarantee level, m can be defined as the *smallest* positive integer that satisfies the constraint:

$$pr(0) + pr(1) + \ldots + pr(m) \quad \geq \quad q\% \tag{5.3}$$

To compute m as defined above, it is necessary to observe that $pr(i)$'s follow a binomial distribution, with $1 - p\%$ being the probability that each stream overflows. Thus, $pr(i)$ is given by:

$$pr(i) \quad = \quad \left(\begin{array}{c} n \\ i \end{array} \right) (1 - p\%)^i \, (p\%)^{n-i} \tag{5.4}$$

For large values of n, computing the value of m through Equations (5.3) and (5.4) could be computationally intensive. But for a given value of $p\%$, we can pre-compute the m value for every value of n, say from 1 to some maximum number depending on the capability of the system. In this way, no computation of m is needed by the admission controller.

Like the case analyzed in Section 5.1, the next step is to set up overflow variables. But there are two key differences between the current and previous cases. First, instead of n overflow variables, here there are only m such variables, as the analysis given in the previous paragraph dictates that there are at most m streams that actually overflow. More importantly, unlike the overflow variable Ov_i defined in Equation (5.1), here the overflow variable Ov'_i $(1 \leq i \leq m)$ need not deal with the situation when there is no overflow (i.e., $B_i \leq B^e_p$). In other words, Ov'_i is simply defined as:

$$Ov'_i \quad = \quad B_i - B^e_p \tag{5.5}$$

As discussed in Section 5.1, the probability density function of Ov'_i corresponds to the tail of the normal distribution to the right of z_p. But caution must be taken because, as it stands, this density function is not valid, for it only integrates to $1 - p\%$, instead of 1. Thus, it is necessary to normalize the density function with $1 - p\%$.

Like the situation for $Ov_1 + \ldots + Ov_n$, the sum $Ov'_1 + \ldots + Ov'_m$ can also be approximated by a normal distribution, since the Central Limit Theorem applies in both cases. As in the derivation of Equation (5.2), when additional buffer space needs to be added to the shared buffer pool, the additional amount is given by:

$$B^{s'}_q \quad = \quad \mu_{ov'} + z_q * \sigma_{ov'}(\sqrt{m+1} - \sqrt{m}) \tag{5.6}$$

where $\mu_{ov'}$ and $\sigma^2_{ov'}$ denote the mean and variance of the variable Ov' defined in Equation (5.5).

We are now ready to present Algorithm CLT(Sm), another homogeneous case algorithm that performs admission control and buffer allocation with both exclusive and shared overflow buffers. Unlike Algorithm CLT(Sn) introduced before, Algorithm CLT(Sm) bases its computation on only the m streams that may actually overflow.

Algorithm CLT(Sm)

Let St be the stream considered for admission. Let n be the number of active streams in the system, and m be the number corresponding to n as defined by Equation (5.3). Let $p\%$ be the individual non-overflow guarantee level requested for St. Let $q\%$ be the system-wide non-overflow guarantee level.

1. Compute the amount of the exclusive buffer space B_p^e as defined in Equation (4.1).

2. Compute the value m' based on Equations (5.3) and (5.4), with n in Equation (5.4) replaced by $n + 1$.

3. If m' is the same as m, no additional shared buffer space is needed, i.e., $B_q^{s'} = 0$. Otherwise, compute $B_q^{s'}$ as given in Equation (5.6).

4. If the remaining buffer space of the system is less than $B_p^e + B_q^{s'}$, the stream St must wait. Otherwise, St is admitted with an allocation of B_p^e buffer space reserved solely for the use of St, and with $B_q^{s'}$ (if greater than zero) buffer space added to the shared overflow buffer pool.

As discussed earlier, Step 2 can be optimized by pre-computing for all values of n, the corresponding m values. Step 2, then, becomes a simple table lookup. As opposed to Algorithm CLT(Sn), the current algorithm enjoys the advantage that as indicated in Step 3, there are times when no additional shared buffer space is needed. [1] This saves administrative overhead.

[1] In fact, the larger the value of n, the more likely it is that m stays the same rather than

Thus far, we have developed three different admission control and buffer allocation algorithms: CLT(E), CLT(Sn) and CLT(Sm). While all three provide individual non-overflow guarantees by allocating exclusive buffer space to each stream, the latter two also deliver additional system-wide non-overflow guarantee by allocating overflow buffer space to be shared by all streams. Two important questions to consider at this point are:

1. How much extra buffer space do CLT(Sn) and CLT(Sm) require, in addition to the space allocated by CLT(E)?

2. How well does the extra space reduce overflows and data loss. i.e. is the extra space justified?

In Chapter 7 we present experimental results that answer these questions.

Finally, we also recall that Algorithm CLT(Sn) and CLT(Sm) are both based on the assumption that all streams have the same compression characteristics (the same mean and standard deviation for frame sizes) and all request the same non-overflow guarantee level. Applications where this occurs are not common but do exist. An example could be the case where a number of students are viewing the same multimedia course material. Students could be viewing different parts of the same stream, so that the stream characteristics would be the same, and the guarantee level could be determined by the system. The more general case is when viewers request different movies and request different guarantee levels based on their tolerance for quality degradation or their wish to reduce costs. Though several people could be watching the same stream, streams with a variety of characteristics could be active in the system at any given time. In the next chapter we discuss issues that must

increasing by 1, when n increases to $n+1$.

be addressed when streams have different characteristics and different non-overflow guarantee levels. We also describe heuristic techniques developed for admission control policies handling a heterogeneous mix of streams.

Chapter 6

Supporting Heterogeneous Requirements

As discussed in Chapter 5, algorithms CLT(Sn) and CLT(Sm) consider only the situation where every stream in the system has identical compression characteristics and requires the same individual non-overflow guarantee level $p\%$. In contrast to this, as mentioned in Chapter 4, algorithm CLT(E) works with a heterogeneous mix of streams, which can have different compression characteristics and individual non-overflow guarantee levels. In this chapter, we first discuss issues involved in applying our algorithms to a heterogeneous mix of streams. We explain why Algorithm CLT(E) works in both the homogeneous and the heterogeneous cases. Then we outline how algorithms CLT(Sn) and CLT(Sm) can be extended to deal with the most general situation, where different streams in the system may differ in:

- their compression characteristics: mean μ, standard deviation σ, and the value k_0; and/or
- their requested individual non-overflow guarantees $p\%$.

The main complication in extending our algorithms to deal with the most general situation is that differing characteristics and/or requested guarantee levels may invalidate the application of the Central Limit Theorem in our computation of exclusive or shared buffer space. This is because random variables must be identically distributed in order to apply Theorem 1. To check whether this identical distribution condition is satisfied, we take a close look at where the Central Limit Theorem is applied in each of our three algorithms.

6.1 CLT(E): Applies in the Heterogeneous Case

As discussed in Chapter 4, Algorithm CLT(E) applies in both the homogeneous and heterogeneous cases without any modifications. The reasons for this are presented below. As formalized in Equation (4.1), the calculation of the amount of exclusive buffer space is based on random variables representing the sizes of different frames in the *same* stream. Frames of the same stream have the same compression characteristics [1]. Thus, the identical distribution condition is satisfied, and the application of the Central Limit Theorem is not affected at all by the presence of streams having different compression characteristics and requesting different individual guarantees. In other words, different sets of values of μ, σ, k_0 and p from different streams can be used independently and correctly with Equation (4.1). Hence, as presented in Chapter 4, CLT(E) can be readily applied to handle heterogeneous requirements without any change to the algorithm.

[1] As mentioned before, we divide B, P, I frames into three different groups. Frames within each group have the same compression characteristics.

6.2 CLT(Sn): Heuristics to Homogenize Overflow Variables

As discussed in Chapter 5, Algorithm CLT(Sn) assumes that all streams have the same compression characteristics and non-overflow guarantee levels. In this section we first consider how this algorithm can be extended to work with a heterogeneous mix of streams. We then describe some admission control policies which make use of these extensions.

Algorithm CLT(Sn) applies the Central Limit Theorem in two different instances: to compute the exclusive and the shared buffer space requirements. As shown in Section 6.1 above, no change is required in the computation of exclusive buffer space to handle heterogeneous case requirements. Unfortunately, the same cannot be said about the computation of shared buffer space. As formalized in Equation (5.2), the computation of the shared buffer space requires that all overflow variables Ov_i be identically distributed. For streams having different characteristics and requesting different individual guarantees, their B_i's and B_p^e's may be different, causing their overflow variables Ov_i's to have different distributions as well.

In addition to the above, there is another complication due to differing individual guarantee levels. In Chapter 5, we discussed how to upgrade the individual guarantee of $p\%$ with an additional system-wide guarantee to $q\%$ to enhance user satisfaction at a reasonable cost. The situation becomes more complex when users can request different p values. For instance, if $p_1 = 60\%$ and $p_2 = 90\%$, it does not make sense to provide a universal system-wide guarantee of $q = 99\%$ to both. This is essentially an issue of fairness, because the upgrade to p_1 is much larger than that to p_2. Thus, we further enrich our framework by allowing different q levels to be defined based on the corresponding p levels. More specifically, for any individual guarantee level p_i chosen by a user, the system can choose to upgrade p_i to q_i ($q_i \geq p_i$) with the following semantics. The stream can use:

- its exclusive buffers up to a level corresponding to p_i, and
- shared buffers, but *no more than* a total level corresponding to q_i.

This is formalized in the following definition of an overflow variable, which extends the one defined in Equation (5.1):

$$Ov_i'' = \begin{cases} 0 & \text{if } B_i \leq B_{p_i}^e \\ B_i - B_{p_i}^e & \text{if } B_{p_i}^e < B_i \leq B_{q_i}^e \\ B_{q_i}^e - B_{p_i}^e & \text{otherwise} \end{cases} \tag{6.1}$$

As discussed at the beginning of this section, different overflow variables Ov_i'' can have different distributions, means and standard deviations. Thus, the Central Limit Theorem and Equation (5.2) cannot be applied directly. In order to deal with this complication, we "homogenize" all overflow variables Ov_i'' by the heuristic of taking the average mean and average standard deviation over all Ov_i'' 's. More precisely, let the means and standard deviations of $Ov_1'', \ldots, Ov_n''$ be $\mu_1, \ldots, \mu_n$ and $\sigma_1, \ldots, \sigma_n$ respectively. Then we homogenize $Ov_1'', \ldots, Ov_n''$ by *assuming* that all of them are identically distributed with mean $\mu_{ov} = avg\{\mu_1, \ldots, \mu_n\}$ and standard deviation $\sigma_{ov} = avg\{\sigma_1, \ldots, \sigma_n\}$. With this heuristic, Equation (5.2) and Algorithm CLT(Sn) operate as before, with $q = avg\{q_1, \ldots, q_n\}$. The homogenizing can also be done using the max or min functions instead of taking the average of the means, standard deviations and q_i's.

Algorithm CLT(Sn) can therefore be extended as follows:

Heuristic CLT(n)

Let St be the stream considered for admission. Let $p\%$ be the individual non-overflow guarantee level requested for St. Let $q_{St}\%$ be the non-overflow guarantee level that the system assigns for additional system-wide guarantees, as corresponding to the user-selected p. Let n be the number of streams already in the system.

1. Compute the amount of the exclusive buffer space B_p^e as defined in Equation (4.1).

2. Homogenize the q values for all the streams in the system using some function such as avg, min or max.

3. Homogenize the overflow variables. Assume they are identically distributed with mean μ_{ov} and standard deviation σ_{ov} as discussed above. (μ_{ov} can be the minimum, average or maximum of the overflow means, depending on the particular heuristic being used. Similarly σ_{ov} can be the minimum, average or maximum of the standard deviations of the overflow variables.)

4. Compute the amount of additional shared overflow buffer space B_q^s as defined in Equation (5.2).

5. If the remaining buffer space of the system is less than $B_p^e + B_q^s$, the stream St must wait. Otherwise, St is admitted with an allocation of B_p^e buffer space reserved solely for the use of St, and with B_q^s buffer space added to the shared overflow buffer pool.

6.3 CLT(Sm): Complication Due to Binomial Calculation

The heuristic described above for CLT(Sn) obviously applies to CLT(Sm) as well. But for CLT(Sm), there is an extra complication due to the binomial computation of the value of m. As formalized in Equation (5.4), the computation of m relies on a fixed p. There are limited applications where this condition holds. e.g. when the p and q values are fixed by the system. But the more general case is where each user can select a different p value. It is not clear how multiple p values can be used in the binomial computation. One heuristic approach is to divide all streams into distinct groups according to their p values, and apply CLT(Sm) separately on each group. Since the m value (number of streams likely to overflow), is determined by the p guarantee level, and the number of users n in the group, the m value for each group can be calculated. The tails for each group can then be homogenized as described in the previous section, and the algorithm CLT(Sm) can be extended in a similar way. Future extensions to this project should include evaluating how effective this approach could be, as well as investigating other ways to extend CLT(Sm) to handle multiple p values.

To summarize, in this chapter, we have discussed how the admission control algorithms developed in the previous chapters can be extended to work when the streams in the system have different characteristics and non-overflow guarantees. Algorithm CLT(E) works without modification, since the Central Limit Theorem is applied to a group of frames which are identically distributed. For algorithms CLT(Sn) and CLT(Sm), the allocation of exclusive buffers remains unmodified, as it works in exactly the same way as with algorithm CLT(E). However, the overflow variables are not identically distributed. We therefore homogenize these by adopting the heuristic approach of using the average, minimum or maximum values of the overflow means and standard deviations. We also "homogenize" the q system-wide non-overflow guarantee level in a similar way. Preliminary experimental results indicate that our heuristic approach works well in extending CLT(Sn) to handle heterogeneous requirements. We have experimented with a variety of heuristics, such as using the average, minimum or maximum of the overflow variables and q values or various combinations of these. The goal is to find a good balance between minimizing the buffer space overhead for providing the two kinds of non-overflow guarantees and minimizing the frequency of overflows. In the next chapter, we give experimental results showing that our heuristic approach to CLT(Sn) can provide appropriate QoS guarantees at a reasonable cost.

Chapter 7

Experimental Results

In this section, we present experimental results evaluating the effectiveness of algorithms developed in the previous chapters. We evaluate these algorithms on:

1. their buffer space requirements
2. their behaviours with increasing numbers of concurrent users
3. their frequencies of overflows
4. the amounts of data loss

We verify whether the algorithms in fact provide the non-overflow guarantee promised. We also compare our algorithms with an existing statistical algorithm that provides buffers shared by all streams and does not support individual non-overflow guarantees. This is done to check whether or not individual guarantees are too expensive to provide.

7.1 Overview of the Implementation and Experimental Setup

In this section we discuss the experimental setup used to evaluate the algorithms developed in previous chapters. We describe our simulation experiments, and briefly discuss how data was acquired to test our algorithms.

7.1.1 Experimental Data

Data for our simulation experiments was acquired by collecting MPEG videos from various Internet sites, including university ftp sites and Space archives. Our object was to gather movies typical of the variety of MPEG videos that are generally encountered. We did not actually make the videos. It was difficult to find videos of any great length, but about 30 were collected, representing various characteristics. The videos differ widely in their compression characteristics, the number of frames they contain, the number of B frames between adjacent I or P frames, and the mean and standard deviation for the frame sizes. The movies also differ widely in the variability of B frame sizes. This can be seen in comparisons between the maximum and average frame sizes, or in the ratio of $\sigma : \mu$. For example, one of the videos has 1,171 B frames, with 4 B frames between I frames and with a mean and a standard deviation of 3,511 and 634 bits (after compression) respectively. Its $\sigma : \mu$ ratio is 1:5.56. The ratio of its maximum to minimum frame size is 3.84:1, and its maximum to average frame ratio is 2.14:1. Another video has 2,591 B frames, with 10 B frames between I frames, and a mean and standard deviation of 1,367 and 764 bits. Its $\sigma : \mu$ ratio is 1:1.79. Its maximum to minimum frame size ratio is 11.75:1, and its maximum to average frame size ratio is 3.92:1. The experimental results to be presented are based on a subset of the videos collected. However, the observations and conclusions that will be drawn are applicable to other videos as well.

7.1.2 Simulation Experiments

All the algorithms were implemented in C. The mean μ and standard deviation σ used in computing Equation (4.1) were determined easily for each stream, with the usual mean and standard deviation calculations. However, there is much greater complexity in the computation of $\mu_{ov}, \mu_{ov'}, \sigma_{ov}, \sigma_{ov'}$ for the overflow variables introduced in Equations (5.1) and (5.5). These required mathematical integration on the tail of a standard normal curve. Mathematica was used to handle this task. Since sample sizes varied widely among streams, it was necessary to select a uniform statistical confidence level, so that meaningful comparisons could be made. Thus all means and standard deviations were determined based on a confidence interval of 95%.

In order to evaluate whether or not individual non-overflow guarantees and exclusive buffers are too costly to provide, we needed some basis for comparison. For this purpose we implemented another algorithm, hereafter referred to as the "Joint" algorithm, that is a variant of the algorithm proposed by Vin et al [49]. The Joint algorithm essentially applies the Central Limit Theorem to all the streams together, rather than to each stream individually. With this algorithm, the full amount of the allocated buffer space is shared by all the streams, without any support for individual non-overflow guarantee.

To evaluate the effectiveness of the different algorithms, a simulation was designed and implemented in C. Using a modified version of the MPEG Statistical Analyzer developed at the University of California at Berkeley, files were created containing the sizes and sequence of each type of frame in the MPEG streams in our collection. Buffer allocation requirements for each algorithm were calculated using the data in the frame size files. The simulation then used the frame size files to evaluate the behavior of the algorithms, calculating the frequency and amount of overflow for each algorithm. Note that all calculations of buffer allocation and data loss were made in terms of bits, rather than number of blocks. The decisions about selecting an appropriate block size are left to an actual implementation and will be discussed in Section 8.2.1.

Our experiments try to simulate a scenario where multiple users are served simultaneously. We performed experiments both for the homogeneous case where all users are viewing the same stream and for the heterogeneous case where users access streams with different characteristics and guarantee levels. In the homogeneous case experiments, all users access the same stream and have the same guarantee level. But they need not view the same part of the stream at the same time, and the viewing time lengths may vary. To simulate this condition, our program randomly assigns a starting frame to each user as well as the number of frames (nframes $\leq$ total number of frames in the stream) during which the user will be active. If a user reaches the end of the stream before covering its assigned number of frames, it cycles back to the beginning of the stream and continues from there until it has read the appropriate total number of frames. If a client's run terminates before the end of the simulation, another user is started by the same procedure to keep the number of users constant for the evaluation of the algorithm. The heterogeneous case algorithms work in a similar way. However, for these experiments users are also randomly assigned to their streams and guarantee levels.

7.2 Evaluation of Homogeneous Case Algorithms

7.2.1 Relative Buffer Requirements

We compare the four homogeneous case algorithms on the amount of buffer space they require. We are interested in how buffer space requirements vary with the number of concurrent users and with the number of adjacent B frames used in the computation. First, holding the number of users constant, we consider the effects of varying the number of frames. The two graphs shown in Figure 7.1 are based on the buffer requirements for 100 concurrent users (i.e., $n = 100$). The first graph is for an individual guarantee level of $p = 90\%$ (for which $m = 27$), and the second is for $p = 80\%$ (for which $m = 41$). In both cases, the q value is set at 99.9%. The x-axes of the two graphs represent the value of k_0, which as introduced in Section 3.3.2, denotes the minimum value of k so that the average correlation between frame $s_{i,j}$ and $s_{i,j+k}$ drops below a certain threshold. In both graphs, k_0 varies from 1 to 30. The y-axes represent the buffer requirements of the algorithms relative to that of CLT(E). This explains why in both graphs, the curves for CLT(E) are flat lines at 100%. [1] An analysis of the graphs gives the following key findings:

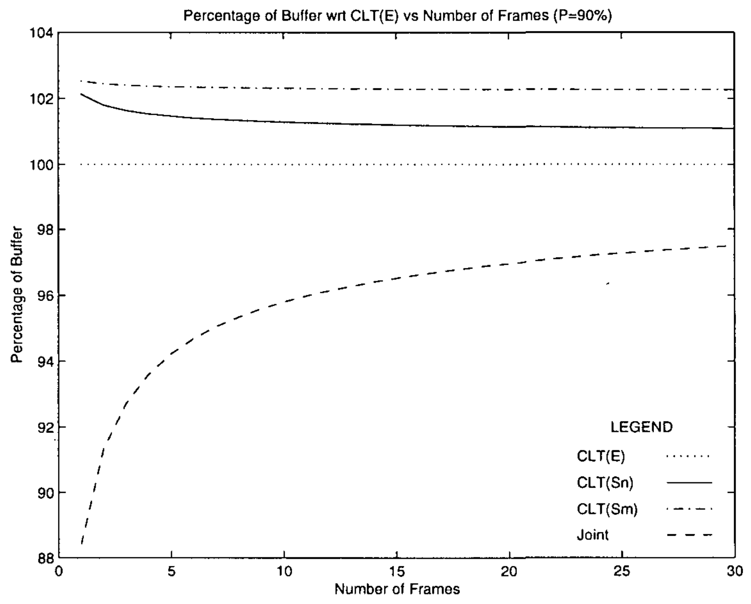
1. *The Cost of Supporting Individual Non-overflow Guarantees*

   Recall that the Joint algorithm, CLT(Sn), and CLT(Sm) all provide a $q\%$ (99.9% in the simulation experiments summarized in this figure) system-wide guarantee. But only the latter two algorithms also provide an individual guarantee of $p\%$ (80% or 90% in the figure). Thus, the difference in buffer space requirement between Algorithms CLT(Sn), CLT(Sm) and the Joint algorithm represents the amount of extra buffer space needed to support individual non-

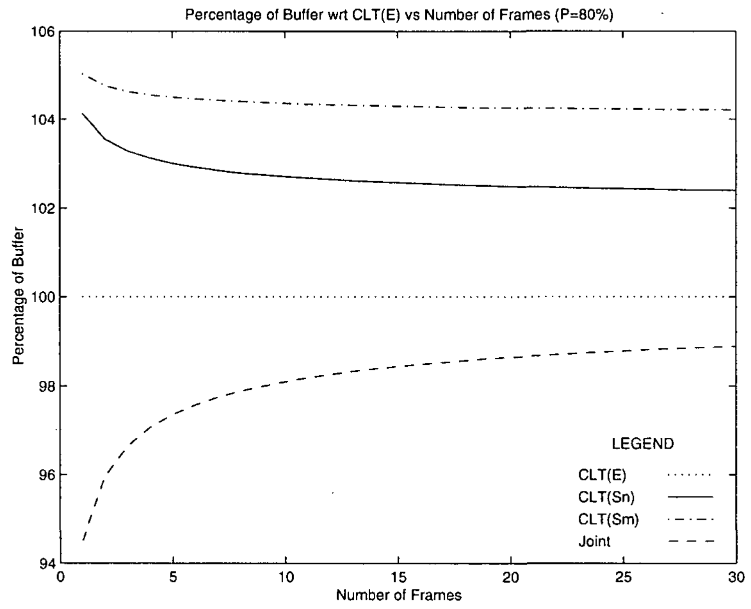
[1] If the y-axes were to represent buffer requirements in absolute terms, all curves would be increasing with increasing values of k_0.

overflow guarantees. In both graphs in Figure 7.1, the amount of extra space needed is less than 5%. This strongly indicates that individual guarantees can be provided with a very low buffer space overhead. Note that the difference in buffer space requirements between Algorithm CLT(E) and the Joint Algorithm is even less than 5%. But care must be taken when interpreting this difference. The key observation is that CLT(E) provides either a 80% or 90% individual guarantee. But the Joint algorithm provides a 99.9% system-wide guarantee. Thus, the quality of service delivered by these two algorithms is rather different. And the curves for CLT(E) and the Joint algorithm in the graphs should not be compared directly.

2. *The Cost of Supporting an Additional System-wide Non-overflow Guarantee* As discussed in Chapters 4 and 5, Algorithms CLT(E), CLT(Sn) and CLT(Sm) all provide a p% individual guarantee. But only the latter two algorithms also provide an additional system-wide guarantee of q%. Therefore, the difference in buffer space requirement between Algorithms CLT(Sn), CLT(Sm) and Algorithm CLT(E) represents the amount of extra buffer space needed to support the additional system-wide guarantee. In Figure 7.1a, the extra amount is slightly more than 1% for CLT(Sn), and around 2% for CLT(Sm). In Figure 7.1b, the extra amounts are around 3% and 4% for CLT(Sn) and CLT(Sm) respectively. Thus, the two graphs strongly suggest that providing an additional system-wide non-overflow guarantee is very inexpensive. In general, as the individual guarantee p% drops, the buffer requirement for CLT(E) drops, and the gaps between CLT(E) and the two other algorithms grow. However, as will be shown in Section 7.2.4, CLT(Sn) and CLT(Sm) are much more effective in reducing overflows and data loss. A more detailed comparison between



(a) $p = 90\%$



(b) $p = 80\%$

Figure 7.1: Buffer Utilization vs Number of Frames

CLT(Sn) and CLT(Sm) will be given later.

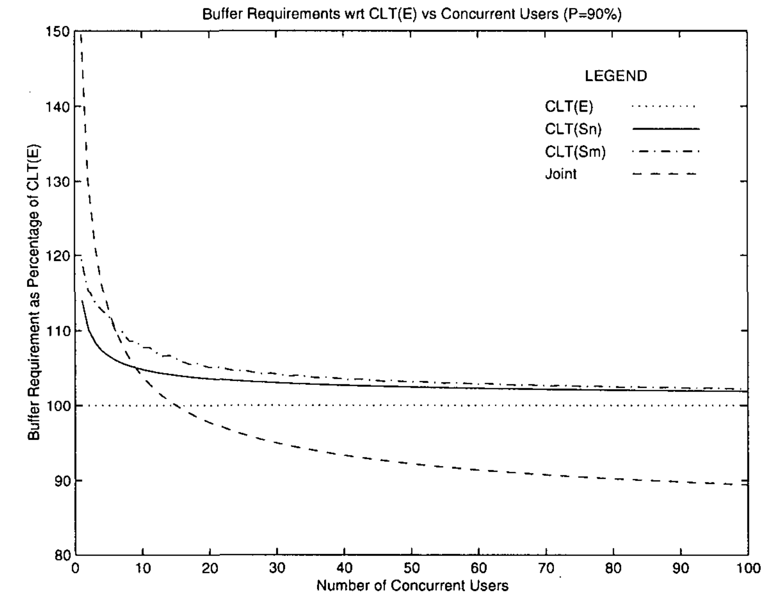
3. *The Value of k_0*

   As the value of k_0 increases, the gaps between CLT(E), CLT(Sn) and CLT(Sm) remain the same. This strongly suggests that there is little reason to use a value of k_0 larger than 10.

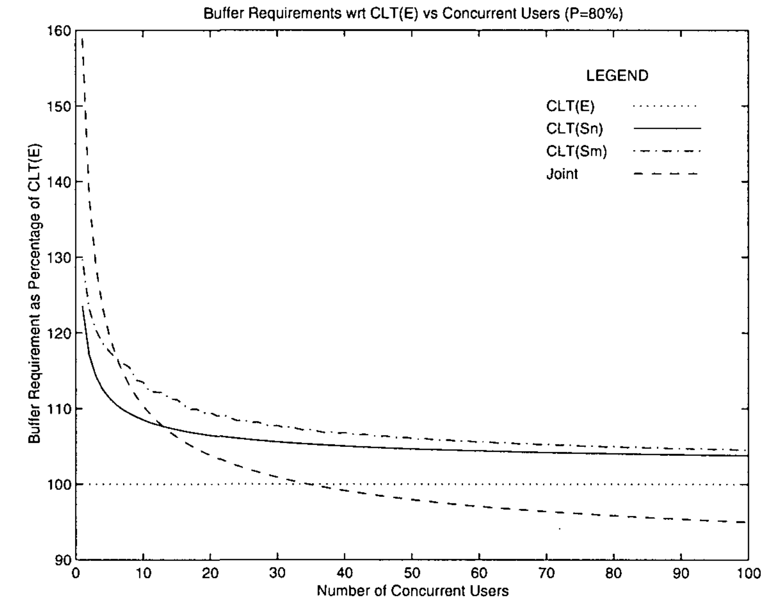
7.2.2 Relative Buffer Requirements with Respect to Number of Concurrent Users

The two graphs in Figure 7.1 are based on 100 concurrent users (i.e., $n = 100$). For the graphs in Figure 7.2 $k_0 = 10$. Figure 7.2 shows how the buffer requirements of Joint, CLT(Sn), and CLT(Sm) vary relative to that of CLT(E), as the number of concurrent users changes. The x-axes of the graphs represent the number of users, ranging from 1 to 100. And the y-axes are the same as those in Figure 7.1.

By definition, since they provide shared buffers in addition to the exclusive space allocated by CLT(E), CLT(Sn) and CLT(Sm) always use more buffers than CLT(E). But the graphs in Figure 7.2 show that as the number of concurrent users increases, the gaps between CLT(E) and the two algorithms narrow. Initially for $n = 20$, the differences can be as large as 7%. But when n increases to 100, the differences are already within 5%. For larger values of n, the gaps will become negligible. This trend is due to the fact that the more concurrent users there are in the system, the more sharing can take place in the overflow buffer pool. In other words, the amount of overflow buffer that is required to provide the system-wide guarantee becomes less in comparison to the total amount of exclusive buffers. Hence, the provision of the system-wide guarantee becomes even more attractive.



(a) $p = 90\%$



(b) $p = 80\%$

Figure 7.2: Concurrent Users

In comparison with CLT(E), the Joint algorithm also requires less buffer space as the number of concurrent users increases. Similar to the situation for CLT(Sn) and CLT(Sm), larger values of n provide greater possibilities for buffer sharing. Hence for these three algorithms the buffer requirements in absolute terms do not grow linearly with n. The buffer percentage of Joint relative to CLT(E) drops as n increases. It is interesting to note from Figure 7.2 that for a small number of users (e.g., $n \leq 40$), CLT(E) actually requires less buffer space than the Joint algorithm, even though CLT(E) provides an individual non-overflow guarantee that the Joint algorithm is not capable of. However, this observation should be treated with caution, as the guarantee levels provided by the two algorithms are not the same.

7.2.3 Service Guarantees to Individual Users: Evaluation of CLT(E)

It is important to determine whether or not our algorithms actually deliver the service guarantees promised to users. In later sections, we will present a comparative analysis of the algorithms in terms of percentage of overflow cycles and amounts of data loss. But in this section, we focus only on a performance evaluation from the point of view of the user. We consider two performance measures for Algorithm CLT(E): frequency of overflows experienced by individual users and the amounts of data loss when overflows do occur. Recall that CLT(E) allocates buffers exclusively to each user. Hence, the behaviour of the algorithm is independent of the number of users in the system, as long as system capacity is not exceeded. It also works regardless of whether users access the same stream or a heterogeneous mix of streams. So the questions that need to be addressed here are:

1. Does CLT(E) provide the promised guarantees to individual users, as seen in

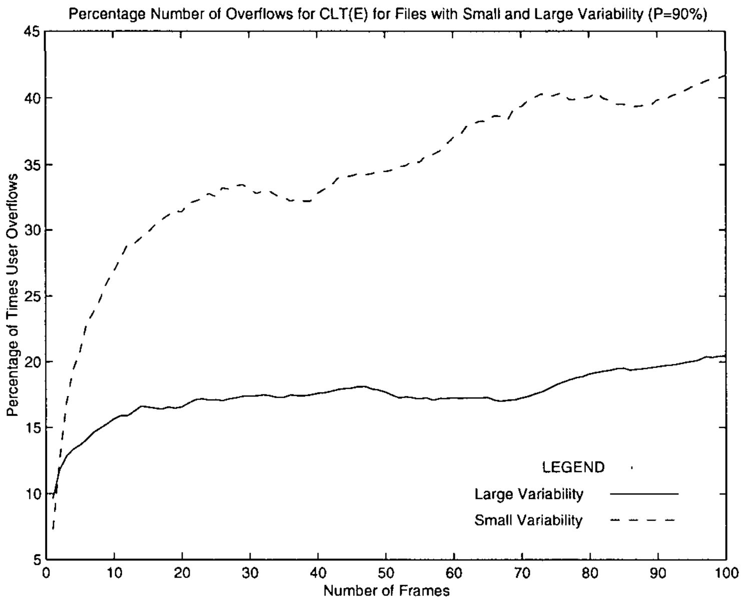
the percentages of overflows and amounts of data loss they experience?

2. How does the algorithm behave with streams of different characteristics and variability?

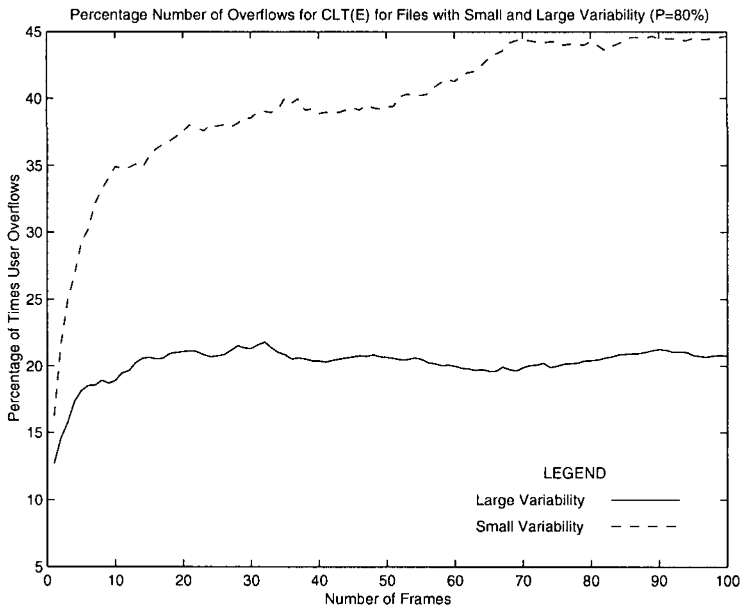
3. How does the algorithm's behaviour change with varying values of k_0[2]?

We investigated the frequency of overflows and the amounts of data lost for streams of large and small variability, as the value of k_0 changes. Our observations show reasonable results, but not as positive as we had hoped. Figure 7.3 shows the frequency of overflows for a high and a low variability stream for $p = 80\%$ and $p = 90\%$. Here, the frequency of overflows is the percentage of cycles in which the stream experiences an overflow. Figure 7.4 shows the average amount of data lost by an overflowing stream. The following trends can be observed from these graphs: For $p = 80\%$ the percentage of overflows for our stream with high variability increases for low values of k_0. It then stabilizes at around 20% when $k_0 \geq 12$, as could be predicted from the guarantee level. For the low variability stream, the percentage of overflows again increases, and eventually stabilizes, but not at a level corresponding to the p value. Instead, the percentage of overflows seems to stabilize around the m value corresponding to the p level (the number of streams out of 100 that could be expected to overflow, given the p guarantee level). However, it is essential to note that the number of frames for our low variability stream is much smaller than for the high variability stream. So the reliability of these results above $k_0 = 10$ is somewhat questionable. For $p = 90\%$, the low variability stream shows a similar trend. However, the percentage of overflows for the high variability stream stabilizes

[2]k_0 is the number of consecutive frames which are grouped together when computing buffer allocations. As introduced in Section 3.3.2, k_0 denotes the minimum value of k so that the average correlation between frame $s_{i,j}$ and $s_{i,j+k}$ drops below a certain threshold



(a) $p = 90\%$



(b) $p = 80\%$

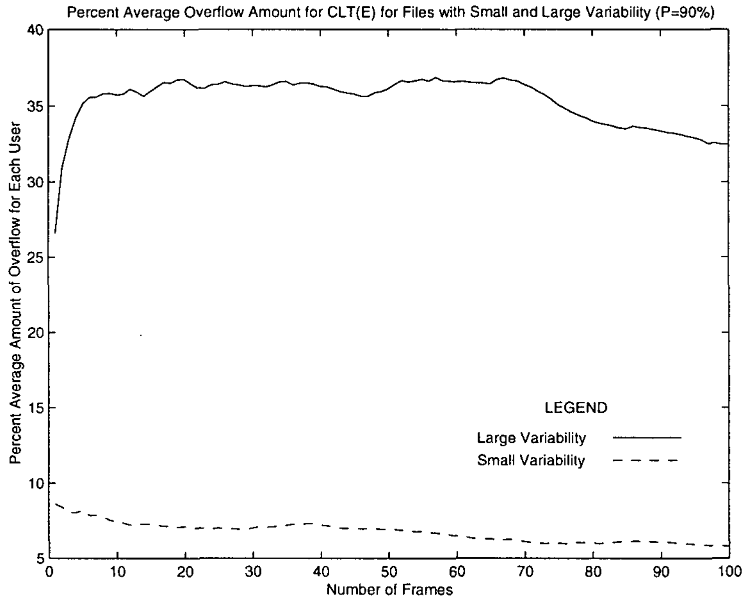
Figure 7.3: Frequency of Overflow for CLT(E)

at a level somewhat higher than could be expected from the p value: around 16% for $10 < k_0 \leq 20$ and 17-18% when $k_0 > 20$.

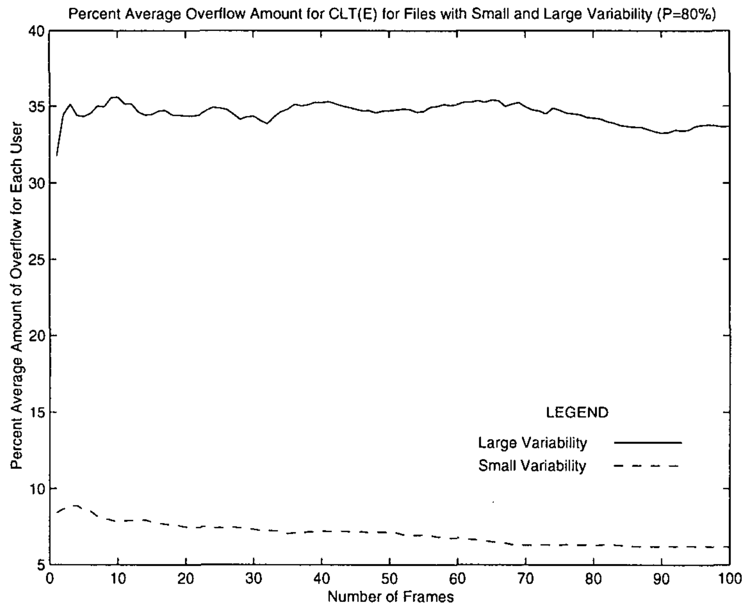
Figure 7.4 shows the opposite trend with respect to the amount of data loss in overflow situations. With the small variability stream, the amount of data loss shows a slow but steady decrease as k_0 increases, and always stays within the amount of data loss that could be expected from the guarantee level. By contrast, the high variability stream, whose frequency of overflows is closer to what could be expected from the guarantee level, consistently loses a much higher percentage of data than predicted from the p level. This result is understandable. For a file with large variability, if an overflow occurs, it is reasonable that the amount of data loss would also be high.

We obtained somewhat better results (actually quite positive) when frames were taken in groups of k_0, and the mean and variance calculated based on the groups of frames. But this procedure is similar to smoothing the data consumption rate discussed in Chapter 2, which uses a different algorithm, so that it would not be a fair assessment of the use of the CLT. In addition, the problem is still not entirely solved because when the number of frame groups is increased, the same pattern emerges as with frame groups of size one.

It was difficult to determine the exact causes of our problem. One obvious explanation may be the smallness of our samples. As k_0 increases, the reliability of our results is bound to decrease. This is due to the fact that, as frames are taken in larger and larger groups, fewer cycles are possible before the stream is exhausted. For instance, at 25 frames, our high variability stream can only provide 103 full cycles, while our low variability stream yields no more than 46. Increasing the number of cycles without regard to the stream length would not improve reliability,



(a) $p = 90\%$



(b) $p = 80\%$

Figure 7.4: Percentage Amount of Overflow for CLT(E)

since the simulation would simply have to go through the same stream over and over again. Hence, our experiments would have to be repeated with much larger samples in order to assess the influence of this factor.

Another possible explanation for the results in this section may be the fact that, since frame sizes cannot have negative values, and in fact, must be considerably larger than 0 bytes to store any meaningful information, the frame size samples yield fairly skewed distributions, with long tails to the right. Consequently, the minimum frame size may be comparatively quite close to the mean, while the maximum frame size can be several times larger. Because of this, the standard deviation may not be capturing as much of the stream characteristics as we would like. This may have led us to somewhat underestimate resource needs, when computing how much buffer space to allocate for each of our algorithms. In fact, a small amount of additional buffer space does bring frequencies of overflow and data loss to well within acceptable levels for CLT(E), but since we have not ruled out the possible effects of the smallness of the sample size, we cannot be sure which explanation is the correct one.

7.2.4 Effectiveness in Handling Overflows: CLT(Sn) vs CLT(Sm)

We have seen that the three homogeneous case algorithms we developed are ranked CLT(E), CLT(Sn) and CLT(Sm) in ascending order of buffer space requirement. We have also seen that CLT(E) is able to provide user selectable guarantees for the amounts of overflows. In this section we also consider the frequency and severity of situations where not enough buffer space has been allocated. This is an important measure of performance because these situations are undesirable for a number of reasons. First, overflows and data loss cause degradation of quality for the user, as considered in the previous section. Second, handling overflows can result in

	$p = 90\%$	$p = 80\%$
CLT(Sm)	56.8%	5.1%
CLT(Sn)	94.9%	61.0%

Table 7.1: Frequencies of Overflow

considerable system overhead. So it is useful to consider how well our algorithms avoid such situations. We define an *overflow round or cycle* as follows: For CLT(E), we say that an overflow round occurs if any one of the concurrent users runs out of its exclusive buffers. For CLT(Sn) and CLT(Sm), we say that an overflow round occurs if any one of the concurrent users runs out of its exclusive buffers, and is not allocated enough buffer space from the shared pool due to the total consumption of the shared pool. For the Joint algorithm, an overflow cycle occurs when any one of the users runs out of buffer space due to the exhaustion of the shared buffer pool. Note that all our measurements are in absolute terms. An overflow cycle occurs if any one of the concurrent users loses any data. [3]

Table 7.1 shows the percentage of times an overflow occurs, for $p = 90\%$ or 80% and $n = 100$. An analysis of the table produces the following observations:

1. *On Varying the Individual Non-overflow Guarantee p*

   For a large p, such as 90%, most of the allocated buffer space is designated as exclusive buffer space. Thus, the percentage of shared buffer space is low. For this reason, once a user has exhausted its exclusive buffer space, there is a good chance that relief cannot be obtained from the shared buffer pool. In contrast, for a low value of p, the percentage of shared buffer space is high.

[3]This makes our results not totally comparable to some other studies, where overflow cycles are defined differently. e.g. Vin at al first discard an appropriate percentage of frames based on the guarantee level. They then define an overflow round as one in which the system resources are still insufficient despite the discarded frames.

	$p = 90\%$	$p = 80\%$
CLT(Sm)	0.5%	0.38%
CLT(Sn)	0.91%	0.61%

Table 7.2: Average Percentage of Data Loss for an Overflowed Stream

Even though it is expected that more users compete for the shared buffer space (i.e., as shown above, $m = 27$ for $p = 90\%$, and $m = 41$ for $p = 80\%$), the competition is still less intense than that for a large p. Table 7.1 exemplifies this fact. For both CLT(Sn) and CLT(Sm), as p drops from 90% to 80%, the frequency of overflow cycles drops accordingly. The drop for CLT(Sm) is more dramatic than that of CLT(Sn).

2. *On Comparing CLT(Sm), CLT(Sn) and CLT(E) for a Fixed p*

   For a fixed p, the frequency of overflow cycles for CLT(Sm) is always lower than that of CLT(Sn). This is simply due to the fact that, as shown in Figures 7.1 and 7.2, CLT(Sm) requires more buffer space than CLT(Sn). By the same reasoning, the frequency of overflow cycles for CLT(E) is the highest among the three homogeneous case algorithms. In most cases, the frequency of overflow cycles for CLT(E) is 100%. Recall that whenever an overflow occurs, administrative overhead is incurred, and some user data are lost (even though it is by the user's choice that a value $p < 100\%$ is selected). Thus, for a small additional amount of buffer space (cf: Figures 7.1 and 7.2), it seems worthwhile to reduce the frequency of overflow cycles to well below 100%. Hence, CLT(Sm) looks the most attractive.

Table 7.2 shows the average percentage of data loss by an overflowed user (i.e., a user that has exhausted its exclusive buffers, and is not allocated enough buffer space from the shared pool due to the total consumption of the shared pool). As in the situation of the frequency of overflow, the percentages of data loss for both algorithms drop as p decreases. And as before, CLT(Sm) performs better than CLT(Sn). However, the key observation from Table 7.2 is that even for CLT(Sn), the average amount of data loss in an overflow is very small, less than 1%. This strongly indicates that even for $p = 80\%$, both CLT(Sn) and CLT(Sm) are capable of delivering excellent non-overflow guarantee to all users.

7.3 Heterogeneous Case Algorithms

For the heterogeneous case, we investigated the behaviour of the Joint, CLT(E) and CLT(Sn) extensions discussed in Chapter 6. We did not study extensions of CLT(Sm) for the heterogeneous case. The simulations were performed as for the homogeneous case, but with the following differences:

1. *Number of Streams Used:*

   During the simulation, each user was randomly assigned to one of two streams. For the sake of consistency, and to make the results easier to compare, the two streams were the same as the high and low variability streams studied in the homogeneous case.

2. *Guarantee Levels:*

   Users were randomly assigned to one of two $p\%$ guarantee levels: 80% and 90%.

3. *Maximum Data Utilization Based on Guarantee Levels:*

   As mentioned in Chapter 6, we added an additional condition to ensure fairness: Each user had the user selectable $p\%$ guarantee. But in addition, the system enforced a maximum buffer utilization limit for each user. This was calculated using a $q\%$ guarantee level based on the user-selected $p\%$ level. A user requesting 80% guarantee was allowed to use only up to a 95% guarantee level from shared buffers in an overflow situation. Data above this limit was discarded even if additional buffer space was available. A user requesting 90% was allowed up to 99%. This scheme has some similarity to Vin et al.'s method, though they discard data above the minimum $p\%$ level, while our method allows data to be kept up to a maximum $q \geq p\%$ level.

4. *Homogenizing the Overflow Variables (Tails of the Normal Distributions):*

   The overflow variables were homogenized as described in Chapter 6. Various combinations of the maximum, average or minimum values for mean and standard deviation were used in homogenizing the overflow variables. However, in homogenizing the system guarantee level q to calculate the shared buffers, the average of the q values was used for all algorithms.

5. *Comparisons with Joint:*

   For each algorithm using shared buffers, a corresponding version of the Joint Algorithm, using maximum, minimum or average for mean and standard deviation, was also evaluated as a comparison. Again the averaged q value was used in calculating shared buffers.

6. *Buffer Utilization Histograms:*

   Buffer utilization histograms were produced for each algorithm, to show the proportion of the cycles in which various percentages of allocated buffers were used.

7.3.1 Buffer Allocation for Heterogeneous Case Algorithms

Many similar trends were observed regarding buffer allocation in the heterogeneous case as had been observed in the homogeneous. In this section, we review these trends, point out the similarities and differences between the homogeneous and heterogeneous cases, and then give a comparative analysis of the various heterogeneous case extensions for CLT(Sn) and Joint with regard to buffer allocation.

1. The relative buffer requirements of the various CLT(Sn) extensions with respect to CLT(E), dropped as the number of concurrent users increased. For instance, the algorithm using the maximum for both the mean and standard deviation of the overflow variables required 5.85% more buffers than CLT(E) for 25 users, 4.53% more for 50 users and 3.97 % more for 100 users. This reflects the fact that the Central Limit Theorem was used in computing buffer allocations, so that the amount of buffer space does not grow linearly with the number of users. Also, as with the homogeneous case, more sharing can occur when there is a larger number of users.

2. Table 7.3 shows the relative buffer requirements of the various CLT(Sn) extensions with respect to CLT(E) and also with respect to the corresponding versions of Joint. These results are for 100 concurrent users and $k_0 = 10$ frames. Note that all results are given in terms of percentages, and not in absolute values. For the CLT(E) comparison, the value for CLT(E) is 100%. For the comparison to the Joint algorithm, the value for the corresponding version of Joint is 100%. Also note that the mean and standard deviation refer to overflow variables for the CLT(Sn) extensions and to actual frame sizes for the Joint. As can be expected, the various extensions of CLT(E)

Means	*SDs*	*CLT(Sn) ext.*	*Joint ext.*
max	max	103.97	–
max	avg	103.58	76.52
avg	min	102.40	111.62
max	min	103.45	–
avg	max	102.65	–
avg	avg	102.52	111.66
min	min	101.38	190.82

Table 7.3: Buffer requirements re. CLT(E) and Joint for Extensions of CLT(Sn)

can be ranked in terms of buffer requirements according to whether they use maximum, average or minimum values for the mean and standard deviation.

3. Similar trends are observed with the various versions of the Joint algorithm, both in their behaviours with increasing numbers of concurrent users, and in their comparative ranking. However, as can be seen from table 7.3, the variability among the different versions of Joint is much greater than among the CLT(Sn) extensions. This is because the minimums, maximums or averages of whole frame sizes are used in the Joint algorithms, while the CLT(Sn) extensions only use the overflow variables, which are a good deal smaller. In addition, the majority of buffers allocated by the CLT(Sn) extensions consist of exclusive buffers. Thus, the effect of shared buffers on the absolute resource requirements is relatively small. These facts make the various extensions of Joint less useful for comparison purposes, and perhaps other heuristics for extending Joint should be considered.

Means	*SDs*	*CLT(Sn) ext.*	*Joint ext.*
max	max	0.00	–
max	avg	0.00	0.00
avg	min	0.10	0.00
max	min	0.00	–
avg	max	0.00	–
avg	avg	0.00	0.00
min	min	51.60	100.00

Table 7.4: Percentage of Overflow Cycles for Heterogeneous Case Algorithms

7.3.2 Overflows and Actual Shared Buffer Utilization for CLT(Sn) Extensions for the Heterogeneous Case

Our observations produced the following results:

1. All our algorithms, except the min-min version were successful in keeping the frequency of overflows low.

2. The amount of data loss was also very low or 0, except in the case of the min-min algorithm.

3. The buffer utilization histograms showed that the shared buffers for the max-max algorithm were frequently under-utilized. In over 90 % of the cycles, between 20 and 60 % of allocated shared buffers were used. In less than 3 % of the cycles, buffer utilization fell between 60 and 80 % of allocated resources, and none of the cycles utilized more than 80 % of shared resources. Hence, this algorithm would waste buffer space.

It is interesting to consider why the CLT(Sn) extensions perform so much better in the heterogeneous case than CLT(Sn) did in the homogeneous case. Three factors contribute to this result: First, the amounts of buffers allocated in the heterogeneous case are generally higher than in the homogeneous case. In fact, except for the min-min extension, the amounts are comparable to CLT(Sm), which, as we saw earlier, allocated more buffers and hence performed considerably better than CLT(Sn). The second factor is that, in the heterogeneous case, we are using more than one stream, and at two different guarantee levels. As we noted in the homogeneous case, both CLT(Sn) and CLT(Sm) performed better for $p = 80\%$ than for $p = 90\%$. So, having a heterogeneous mix of streams means that more sharing and better performance is possible. The third and very important element is the maximum buffer utilization limit imposed when streams access the shared buffer pool. Because of this cap on the use of shared resources, a stream that overflows by a very large amount over its exclusive allotment can expect to lose some data even after accessing the shared pool. But it will not be able to consume so much of the shared buffers that several other streams overflow as well. This is the main reason why even the min-min extension performs better than the homogeneous case CLT(Sn) algorithm, though the heterogeneous mix also contributes to the positive result. On the other hand, the cap on resource use also leads to the under-utilization of the shared pool in the max-max case. So, the effects and desirability of this factor must be carefully considered if it is to be used in an actual implementation.

7.4 Summary of Experimental Results

In the homogeneous case, we have compared CLT(E), CLT(Sn), CLT(Sm) and the Joint Algorithm on their buffer space requirements, their behaviour with increasing numbers of concurrent users, their frequencies of overflows, and the amounts of data loss. There are several major observations. First, CLT(E), CLT(Sn) and CLT(Sm) are effective in delivering individual non-overflow guarantee at a low cost. The buffer space overhead is less than 5%.

Second, CLT(Sn) and CLT(Sm) are effective in delivering a system-wide guarantee on top of the individual guarantee provided by CLT(E). The extra amount of buffer space required by CLT(Sn) and CLT(Sm) is very small–less than 5%. Also, as the number of concurrent users increases, the percentage of additional space overhead decreases. The additional shared buffer space appears to be more than worthwhile as it considerably reduces the frequency of overflows and the amount of data loss.

While CLT(Sn) appears to deliver good performance, its only drawback is that the frequency of overflow can be high – even though the amount of data loss is still very low. This increases administrative overhead. Since a small amount of additional buffer space is available (e.g., 2% more), CLT(Sm) is even more attractive than CLT(Sn). The small amount of extra space helps to further reduce both frequency of overflow and amount of data loss.

Finally, we have investigated the performance of CLT(Sn) as extended for the heterogeneous case. We found that the algorithm works quite effectively, except in the min-min case, which produces too many overflows, and the max-max case, which wastes buffer resources.

Chapter 8

Conclusions and Future Work

8.1 Summary

With the increasingly widespread use of multimedia, providing systems support for continuous multimedia servers has become a very important task. The server's job is to serve as many users as possible while maintaining their continuity and real time requirements. Recently, this task has become further complicated by the increased popularity of variable bit rate compression over the more traditional fixed rate compression. VBR compression greatly economizes on computing resources, but complicates the task of the CM server, as resource needs are difficult to predict.

A number of admission control schemes have been proposed for CM servers dealing with VBR compressed streams. Many suffer from the disadvantages of inaccurate predictions or a great deal of system overhead. Vin et al. propose a statistical admission control algorithm that uses the Central Limit Theorem to model the disk bandwidth utilization of a CM server at a particular point in time, as a normal random variable. This allows them to offer QoS guarantees at the granularity level of the entire system. Unfortunately, this method cannot provide adequate QoS to *individual* users. In this thesis, we study how to provide non-overflow guarantees to MPEG streams, which use VBR compression. One of the main contributions of this thesis is the support of non-overflow guarantee at the granularity level of individual streams. By applying a specific version of the Central Limit Theorem, we develop a statistical model that approximates the total size of several adjacent compressed frames as normally distributed. This enables us to compute the amount of exclusive buffers needed to provide a $p\%$ individual non-overflow guarantee, for any p value chosen by the user. All our admission control algorithms are capable of providing individual guarantees. Our experimental results indicate that the buffer space overhead required to support individual guarantees is very small (less than 5%).

We also investigate the feasibility of providing an additional $q\%$ system-wide non-overflow guarantee, on top of the individual guarantees. For the homogeneous case, algorithms CLT(Sn) and CLT(Sm) represent two different ways of providing such an additional guarantee through the use of shared buffers. Our experimental results indicate that the buffer space overhead required to provide the additional system-wide guarantee is again very small (less than 5%). But this small amount of extra buffer space is very effective in reducing the frequency of overflows and the amount of data loss. Thus, both CLT(Sn) and CLT(Sm) are very successful and economical in enhancing the reliability and the quality of service delivered to the user. CLT(Sm) requires a little more buffer space than CLT(Sn) (about 2%), but the extra space helps to further reduce the frequency of overflow and associated administrative overhead, as well as, the amount of data loss.

Finally, we study the behaviour of our algorithms in the most general situation, where streams can have different characteristics and guarantee levels. Our algorithm CLT(E), which provides individual non-overflow guarantees to users, works without modification in this situation. Algorithms CLT(Sn) and CLT(Sm) can be extended by homogenizing the overflow variables, thus making the Central Limit Theorem applicable. We studied the behaviour of various extensions of CLT(Sn), and were able to determine which extensions work best in terms of both reducing the amount of extra buffers, and minimizing overflows and data loss.

8.2 Future Work

Future work on the topic of this thesis might include the following:

1. Extending the algorithms to deal with other resources such as disk bandwidth.

2. Evaluating various heuristic approaches that extend CLT(Sm), as discussed in Chapter 6.

3. Developing an actual implementation of the algorithms, or incorporating them into an already existing CM server implementation.

4. Investigating whether the techniques and algorithms in this thesis can be applied to other types of compressed CM data, such as audio.

5. Extending the algorithms to deal with situations where the video data must be synchronized with other types of compressed data.

The first two are reasonably routine extensions of the work in this thesis. Developing an efficient CM server implementation is a more complex task. Implementation issues are discussed in the next section. The fourth problem involves studying the characteristics of specific types of compressed data to see if the Central Limit Theorem can be applied. The issue of synchronization will not be covered here as it is beyond the scope of this thesis.

8.2.1 Implementation Issues

When incorporating the algorithms in this thesis into a new or existing CM server implementation, it is essential to consider a number of issues. One key implementation issue is the handling of overflows. Two elements are involved in this task. First, when the user's data overflows allocated buffers it may be necessary to discard parts of the frames. Vin et al. [50] cover how to select the parts of the frames to be discarded. There is also the issue of exactly how to accomplish this task with a minimum amount of overhead. One obvious way is to read data from disk to a system area that can give absolute non-overflow guarantee. In this area we can verify whether the actual amount exceeds the allocated amount. If so, then a selection module must be invoked to choose parts of the data to discard.

Once the overflow data has been discarded the next step is to copy the data from the system area to the user space. However, this kind of copying may be too costly. A possible way to deal with this problem is not to designate any particular part of the main memory as the system area. After reading data from disk, if there is no overflow, the address of the buffer space is simply returned to the user. Otherwise, the unwanted parts of the frames can be discarded by freeing the appropriate sections of the buffer space. The selection module then returns the list of addresses and sizes of the remaining pieces of buffer space to the requesting user. This, however, can cause serious memory fragmentation problems. Thus, some amount of memory management, such as packing, will be essential in order to reclaim the wasted space.

The choice of block size is also an important issue. If the block size is small, then the overflowed data can be discarded as units of blocks. This would simplify the kind of overflow data management discussed above. However, for multimedia data, the block size should not be too small, as this would involve to much overhead for the system. A good balance needs to be struck to minimize the total overhead.

8.3 Conclusions

In this thesis, we have developed a statistical model for MPEG streams using the Central Limit Theorem. This has enabled us to represent the resource requirements for a stream as a normal random variable and thus to provide user selectable non-overflow guarantees. We have also presented algorithms which provide additional system wide guarantees using shared buffers. Our simulation experiments have shown that all these algorithms are effective in providing the non-overflow guarantees at a reasonable cost.

The algorithms presented here have only dealt with buffer allocation. To make them more complete, they should be extended to deal with other server resources such as disk bandwidth. It is also essential to deal with other types of data such as audio and to consider the problems that might arise in synchronizing various types of multimedia data. All these points will be important when developing an efficient implementation for a CM server that incorporates these algorithms.

Bibliography

[1] A. M. Alqaed and C. H. Chang. "A Hybrid Equivalent Capacity Admission Control Algorithm for ATM Congestion Control". In *Global Data Networking, 1993 First Symposium*, pp. 54–59, 1993.

[2] D. Anastassiou. "Digital Television". *Proc. IEEE*, Vol. 82, No. 4, pp. 510–519, April 1994.

[3] D. P. Anderson. "Metascheduling for Continuous Media". *ACM Transactions on Computer Systems*, Vol. 11, No. 3, pp. 226–252, August 1993.

[4] D. P. Anderson and G. Homsy. "A Continuous Media I/O Server and its Synchronization Mechanism". *Computer*, Vol. 24, No. 10, pp. 51–57, October 1991.

[5] D. P. Anderson, Y. Osawa, and R. Govindan. "A File System for Continuous Media". *ACM Trans. on Computer Systems*, Vol. 10, No. 4, pp. 311–337, Nov. 1992.

[6] A. Araujo and E. Gine. *The Central Limit Theorem for Real and Banach Valued Random Variables.* Wiley. New York, 1980.

[7] E. Biersack and F. Thiesse. "Statistical Admission Control In Video Servers with Variable Bit Rate Streams and Constant Time Length Retrieval". In *Proc. of the 22nd EUROMICRO Conference*, pp. 633–639, 1996.

[8] E. Biersack, F. Thiesse and C. Bernhardt. "Constant Data Length Retrieval for Video Servers with Variable Bit Rate Streams". In *Proc. of the Third IEEE International Conference on Multimedia Computing and Systems, 1996*, pp. 151– 155, 1996.

[9] P. Billingsley. *Probability and Measure*, 2nd edition, John Wiley & Sons. New York, 1986.

[10] A. Campbell, G. Coulson, F. Garcia and D. Hutchison. "A Continuous Media Transport and Orchestration Service". In *ACM Conference Proc. of Communications Architectures and Protocols*, pp. 99–110, 1992.

[11] E. Chang and A. Zakhor. "Variable Bit Rate MPEG Video Storage on Parallel Disk Arrays". *Community Networking Integrated Multimedia Service, 1994 Workshop (first)*, pp. 127–137, 1994.

[12] M. Chen, D. Kandlur and P. Yu. "Optimization of the Grouped Sweeping Scheduling with Heterogeneous Multimedia Streams". In *Proc. First International Conference on Multimedia, ACM-Multimedia*, pp. 235–242, 1993.

[13] S. Christodoulakis and P. Triantafillou. "Research and Development Issues for Large-Scale Multimedia Information Systems". *ACM Computing Surveys*, Vol. 27, No. 4, pp. 576–579, December 1995.

[14] J. Gemmell. "Multimedia Network File Servers: Multi-channel Delay Sensitive Data Retrieval". In *Proc. ACM-Multimedia*, pp. 243–250, 1993.

[15] J. Gemmell and S. Christodoulakis. "Principles of Delay-Sensitive Multimedia Data Storage and Retrieval". *ACM Transactions on Information Systems*, Vol. 10, No. 1, pp. 51–90, January 1992.

[16] J. Gemmel, H. M. Vin, D.D. Kandlur, P.V. Rangan and L.A. Rowe "Multimedia Storage Servers: A Tutorial and Survey". *IEEE Computer*, Vol. 28, No. 5 pp. 40–49, May 1995.

[17] S. Guo, and N. D. Georganas. "Resource and Connection Admission Control in Real-Time Transport Protocols with Deterministic QoS Guarantees". *Infocom '95*, Vol. 3, pp. 1095–1102, 1995.

[18] B. Jabbari, F. Yegenoglu, Y. Kuo, S. Zafar, Y. Zhang. "Statistical Characterization and Block-Based Modeling of Motion-Adaptive Coded Video". *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 3, No. 3, pp. 199–207, June 1993.

[19] M. Kawashima, Cheng-Tie Chen, Fure-Ching Jeng and S. Singhal. "Adaptation of the MPEG Video-Coding Algorithm to Network Applications". *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 3, No. 4, pp. 261–269, August 1993.

[20] D. Le Gall. "MPEG: a Video Compression Standard for Multimedia Applications". *Communications of ACM*, Vol. 34, No. 4, pp. 47–58, 1991.

[21] D. Makaroff, G. Neufeld and N. Hutchinson. "An Evaluation of VBR Admission Algorithms for Continuous Media File Servers". *ACM Multimedia '97*, pp. 143–154, November 1997.

[22] W. Mendenhall, D. Wackerly and R. Scheaffer. *Mathematical Statistics with Applications*, 4th edition. PWS-Kent Publishing Company. Boston, 1990.

[23] G. Miller, G. Baber and M. Gilliland. "News On-Demand for Multimedia Networks". *ACM Multimedia 93*, pp. 383–392, 1993.

[24] S. Nachum. "Preemption-Based Admission Control in Multimedia Multiparty Communications". *Infocom '95*, V. 2, pp. 827–834, 1995.

[25] K. Nahrstedt and J. M. Smith. "The QOS Broker". *IEEE Multimedia*, Vol. 2, No. 1, pp. 53–67, Spring 1995.

[26] K. Nahrstedt and R. Steinmetz. "Resource Management in Networked Multimedia Systems". *IEEE Computer*, Vol. 28, No. 5, pp. 52–63, May 1995.

[27] G. Neufeld, D. Makaroff and N. Hutchinson. "Design of a Variable Bit Rate Continuous Media Server". In *Proc. 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 375–378. April 18–21, 1995.

[28] G. Neufeld, D. Makaroff and N. Hutchinson. "The Design of a Variable Bit-Rate Continuous Media File Server for an ATM Network". In *IS&T/SPIE Multimedia Computing and Networking* San Jose, CA. pp. 370–380, January 1996.

[29] G. Neufeld, D. Makaroff and N. Hutchinson. "Server Based Flow Control in a Distributed Continuous Media File Server". In *Proc. 6th International Workshop on Network and Operating System Support for Digital Audio and Video*, April 1996.

[30] R. T. Ng and R. Dilek. "Statistical Modeling and Buffer Allocation for MPEG Streams". In S. M. Chung, editor. *Multimedia Information Storage and Management.* Kluwer Academic Publishing, Norwell, MA. pp. 147–162, 1996.

[31] R. Ng and J. Yang. "Maximizing Buffer and Disk Utilizations for News On-Demand". In *Proc. 20th International Conference on Very Large Data Bases*, pp. 451–462, 1994.

[32] R. Ng and J. Yang. "An Analysis of Buffer Sharing and Prefetching Techniques for Multimedia Systems". *ACM-Springer Journal of Multimedia Systems*, Vol. 4, No.2, pp. 55–69, 1996.

[33] B. Ozden, R. Rastogi and A. Silberschatz. "Research Issues in Multimedia Storage Servers". *ACM Computing Surveys*, Vol. 27, No. 4, pp. 617–620, December 1995.

[34] Huanxu Pan, Lek-Heng Ngoh, and A. A. Lazar. "A Time-scale Dependent Disk Scheduling for Multimedia-on-demand Servers". In *Proc. of Third IEEE International Conference on Multimedia Computing and Systems, 1996*, pp. 572–579.

[35] P. Pancha and M. El Zarki. "A Look at the MPEG Video Coding Standard for Variable Bit Rate Video Transmission". In *Proc. Eleventh Joint Conference on Computer Communications of the IEEE Computer and Communication Societies*, pp. 85–94, 1992.

[36] P. Pancha and M. El Zarki. "Prioritized Transmission of Variable Bit Rate MPEG Video". *Globecom92: IEEE Global Telecommunicationss Conference*, 0092, pp. 1135–1139, 1992.

[37] P. Pancha and M. El Zarki. "Bandwidth Allocation Schemes for Variable-Bit-Rate MPEG Sources in ATM Networks". *IEEE Transactions on Circuits and Systems For Video Technology*, Vol. 3, No. 3, 0693, pp. 190–198, 1993.

[38] P. Pancha and M. El Zarki. "MPEG Coding for Variable Bit Rate Video Transmission". *IEEE Communications Magazine*, Vol. 32, No. 5, pp. 54–66, May 1994.

[39] P. V. Rangan, S. S. Kumar and S. Rajan. "Continuity and Synchronization in MPEG". *IEEE Journal on Selected Areas in Communications*, Vol. 14, No. 1, pp. 52–60, January 1996.

[40] P. V. Rangan, S. Ramanathan and S. Sampath Kumar. "Feedback Techniques for Continuity and Synchronization in Multimedia Information Retrieval". *ACM Transactions on Information Systems*, Vol. 13, No. 2, pp. 145–176, April 1995.

[41] P. V. Rangan and H. M. Vin. "Designing File Systems for Digital Video and Audio". In *Proceedings of the Thirteenth ACM Symposium on Operating Systems Principles*, Vol. 25, No. 5, pp. 81–94, 1991.

[42] P. V. Rangan, H. M. Vin and S. Ramanathan. "Designing an On-Demand Multimedia Service". *IEEE Communications Magazine*, Vol. 30, No. 7, pp. 56–64, July 1992.

[43] A. L. N. Reddy and J. Wyllie. "Disk Scheduling in a Multimedia I/O System". *ACM Multimedia Conference*, pp. 225–233, 1992.

[44] L. Rowe and B. Smith. "A Continuous Media Player". In *Proc. 3rd Intl. Workshop on Network and OS Support for Digital Audio and Video*, pp. 376–386. November 1992.

[45] H. Samet. "General Research Issues in Multimedia Database Systems". *ACM Computing Surveys*, Vol. 27, No. 4, pp. 630–632, December 1995.

[46] P. J. Shenoy, P. Goyal and H. M. Vin. "Issues in Multimedia Server Design". *ACM Computing Surveys*, Vol. 27, No. 4, pp. 636–639, December 1995.

[47] N. Shroff and M. Schwartz. "Video Modeling Within Networks Using Deterministic Smoothing at the Source". *INFOCOM94*, pp. 342–349, 1994.

[48] H. Tezuka, H. Fujita, and T. Nakajima. "A Processor Reservation System Supporting Dynamic QOS Control". *Real-Time Computing Systems and Applications, 1995 Second International Workshop*, pp. 224–231, 1995.

[49] H. Vin, P. Goyal, Al. Goyal and An. Goyal. "A Statistical Admission Control Algorithm for Multimedia Servers". In *Proc. ACM-Multimedia*, pp. 33–40, 1994.

[50] H.M Vin, A. Goyal, A. Goyal, and P. Goyal. "An observation-Based Admission Control Algorithm for Multimedia Servers". *Multimedia, International Conference*, pp. 234–243, 1994.

[51] A. Vogel, B. Kerherve, G. von Bochmann, and J. Gecsei. "Distributed Multimedia and QOS: a Survey". *IEEE Multimedia*, Vol. 2, No. 2, pp. 10–19, 1995.

[52] C. A. Waldspurger and W. E. Weihl. "Lottery Scheduling: Flexible Proportional-Share Resource Management". In *Proc. of the First USENIX*

Symposium on Operating Systems Design and Implementation, pp. 1–11. November 1994.

[53] Jinhai Yang. "Maximizing Buffer and Disk Utilization for News on Demand". MSc. Thesis. University of British Columbia, 1994.

[54] C. Yu, W. Sun, D. Bitton, Q. Yang and R. Bruno. "Efficient Placement of Audio Data on Optical Disks for Real-Time Applications". *Communications of ACM*, Vol. 32, No. 7, pp. 862–871, 1989.

[55] H. Zhang and D. Ferrari. "Improving Utilization for Deterministic Service in Multimedia Communication". *Multimedia, 1994 International Conference*, pp. 295–304, 1994.