Design and Performance of a Continuous Media Server for a High-Speed Network

by

Bikash Sonah

B. Sc. (Electrical Engineering) University of Manitoba

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF APPLIED SCIENCE in THE FACULTY OF GRADUATE STUDIES DEPARTMENT OF ELECTRICAL ENGINEERING

We accept this thesis as conforming to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

Dec. 1994

© Bikash Sonah, 1994
In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Electrical Engineering

The University of British Columbia
Vancouver, Canada

Date 09/Jan/95.

DE-6 (2/88)
Abstract

In this thesis we present two multimedia architectures and a data retrieval model for supporting simultaneously multiple clients requesting files of different playback rates. After reviewing some theoretical background from previous research, we study the performance of the architectures in terms of the maximum number of concurrent video streams they can support using circular SCAN disk scheduling policy. We present new techniques, namely, the maximum operation set and the Rate-based Buffer Weighting Scheme, not only to adapt the system to streams of different playback rates but also to relieve the processing load on the server. Next, we devise a new way of designing the Admission Control for both architectures. The new Admission Control is simple, effective and efficient. Finally we compare the merits of the architectures in terms of throughput, contention, client buffer size and reliability.
Contents

Abstract .................................................................................. ii

List of Figures ........................................................................ vii

Acknowledgment ..................................................................... viii

Chapter 1 Introduction ............................................................. 1

1.1 Objectives ........................................................................ 1

1.2 Motivations ....................................................................... 2

1.3 Contributions .................................................................... 3

1.4 Assumptions ...................................................................... 3

Chapter 2 Theoretical Background ........................................... 5

2.1 Safe Operation Set ............................................................. 5

2.2 Quality Proportional Multi-Subscriber Scheme ................ 7

2.3 Rounding Operation ......................................................... 8

Chapter 3 System Modelling .................................................. 10

3.1 System Architecture ......................................................... 10

3.2 File Storage ...................................................................... 12
<table>
<thead>
<tr>
<th>Chapter 6</th>
<th>Design of New Admission Control: Architecture II</th>
<th>48</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Stream Admission</td>
<td>48</td>
</tr>
<tr>
<td>6.2</td>
<td>Choosing the Rate-based Buffer Weight Constant</td>
<td>48</td>
</tr>
<tr>
<td>6.3</td>
<td>Drive Maximum Operation Set Size</td>
<td>49</td>
</tr>
<tr>
<td>6.4</td>
<td>Maximum Operation Set Size</td>
<td>51</td>
</tr>
<tr>
<td>6.5</td>
<td>Admission Control</td>
<td>53</td>
</tr>
<tr>
<td>Chapter 7</td>
<td>Further Discussions</td>
<td>57</td>
</tr>
<tr>
<td>7.1</td>
<td>Comparing Architectures</td>
<td>57</td>
</tr>
<tr>
<td>7.2</td>
<td>Admission Control Additional Features</td>
<td>58</td>
</tr>
<tr>
<td>7.3</td>
<td>Limitations</td>
<td>59</td>
</tr>
<tr>
<td>7.4</td>
<td>Relaxing the Assumptions</td>
<td>60</td>
</tr>
<tr>
<td>Chapter 8</td>
<td>Related Work</td>
<td>62</td>
</tr>
<tr>
<td>8.1</td>
<td>Storage Techniques For Digital Continuous</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>Multimedia</td>
<td>62</td>
</tr>
<tr>
<td>8.2</td>
<td>Disk Arrays</td>
<td>63</td>
</tr>
<tr>
<td>8.3</td>
<td>Streaming RAID</td>
<td>63</td>
</tr>
<tr>
<td>8.4</td>
<td>Caching Strategies</td>
<td>64</td>
</tr>
<tr>
<td>8.5</td>
<td>Admission Control Algorithms</td>
<td>65</td>
</tr>
<tr>
<td>8.6</td>
<td>Statistical Control Admission</td>
<td>66</td>
</tr>
</tbody>
</table>
## List of Figures

| Figure 3.1 | Server Servicing Multiple Clients | 10 |
| Figure 3.2 | Server Architecture | 11 |
| Figure 3.3 | Node Architecture | 11 |
| Figure 3.4 | Figure Data Retrieval Model | 15 |
| Figure 3.5 | Double Buffering | 19 |
| Figure 3.6 | Rounding Operation | 19 |
| Figure 3.7 | Seek Time versus Distance Travelled | 22 |
| Figure 3.8 | Interfile Seek Scenarios | 22 |
| Figure 4.1 | Absorption Elasticity due to Slack Time | 30 |
| Figure 4.2 | Calculating the Maximum Playback Constant | 32 |
| Figure 4.3 | Performance of Architecture I | 35 |
| Figure 4.4 | Performance of Architecture II | 37 |
| Figure 5.1 | Effectiveness of Admission Control | 47 |
| Figure 6.1 | Effectiveness of Admission Control (Random Files Requests) | 55 |
| Figure 6.2 | Effectiveness of Admission Control (Severe Contention) | 56 |
Acknowledgment

I would like to thank my supervisor, Dr Mabo Ito, and my co-supervisor, Dr Gerald Neufeld, for giving me the freedom, the guidance and the encouragement to investigate into issues where I felt room for improvement exists. I should also thank my supervisors for their time and efforts they took in understanding the materials to the depth I did and in helping me in the correction and editing of the literature.

I am also grateful to Mark McCutcheon, Rob Ross, Jeffrey Chow, John Tanlimco and Peter Meisl for their assistance whenever I wanted it. Their information at hand and their suggested books and reading materials were greatly appreciated.

Finally, I would like to thank The Lord, without Whose intelligence and resources, this project would not concretize. This project is more one of insights than one of hard work or past experience.

This work was funded by Canadian Institute for Telecommunications Research.
1 Introduction

1.1 Objectives

In recent years, the technological world has witnessed an increase in speed of CMOS technologies, in transmission bandwidth and in storage capacity. Moreover, services that were analog are now stored in digital form. These technological improvements and the advantages of statistical sharing have increased efforts towards integrating services. However, disk data rates have not increased as rapidly and have become a central issue in designing multimedia systems.

This thesis focuses on the design and performance analysis of a multimedia server built from existing drive systems. This multimedia server is intended to service simultaneously a large number of clients requesting different media of different rates. The multimedia server should guarantee that each stream be provided data at an average rate equal to its playback rate so that playback proceeds at real-time. Our objectives are to:

- to design two disk layout architectures that can support multiple client streams simultaneously.
- to analyze performance and identify bottlenecks.
- to develop techniques for supporting concurrent streams of different playback rates.
• to design simple Admission Control schemes for both architectures.
• to compare the two architectures.

1.2 Motivations

This thesis addresses many problems found in previous work. These problems are listed as follows:

• An analysis is made on a multimedia server assuming a high disk rate and considering only disk parameters. However, to achieve this high disk rate, other architectural components such as buses or intermediate buffers come into play. These components introduce parameters which must be considered to reflect a more realistic world.

• The existing Admission Control scheme guarantees no starvation in steady state only. However, when a new stream is accepted, its absorption may result in starvation. Moreover the elasticity added in the system is most suitable for the unique case in which all streams are the same.

• When a new stream is accepted, the data load to be retrieved for existing streams is likely to change, resulting in a need for transitions. Transitions constitute a heavy processing load on the server.

• The maximum number of streams admitted by the Admission Control may not correspond to the maximum throughput at the server. For instance, any
client may limit the performance of the server to a value below its maximum throughput, by subscribing with a small buffer.

1.3 Contributions

The scope of the thesis is as follows. After reviewing some theoretical background reached by previous research we present two multimedia server architectures, built from existing technologies, and a data retrieval model, which is simplified but realistic. We analyze the performance of the architectures in terms the number of concurrent video-rate streams they can support. We also present a new technique (that is, finding the maximum operation set) that makes absorption of accepted streams more elastic. Third, we present a new scheme, called the Rate-based Buffer Weighting Scheme, in which the load of data retrieved for existing streams remain fixed even though a new stream is accepted or an old stream dies. This scheme eliminates the necessity for transitions. Fourth, we devise a new way of designing a simple yet effective Admission Control for the server architectures presented. The maximum number of streams admitted corresponds to the maximum throughput at the server. Finally, the merits of both architectures are compared.

1.4 Assumptions

Our study is based on the following assumptions:
• The granularity of a stripe is chosen to be one track. Therefore, zero-latency read is possible.

• A linear seek-time-versus-distance-travelled model is used.

• Delays at switches and propagation delays are not considered.

• Blocks are retrieved without errors.
2 Theoretical Background

2.1 Safe Operation Set

The server operates in cycles. In each cycle, the server delivers a load of data in a burst for each stream into its buffer, a load of data being the number of blocks the server retrieves for each stream. While a stream is playing back one load of data, the server ensures delivery of another load of data within the cycle.

Suppose the server has admitted $n$ clients. The server services each stream $S_1, S_2, ..., S_n$ in a service round [1] [2] by reading $k_1, k_2, ..., k_n$ media blocks of the files $F_1, F_2, ..., F_n$ with constant playback rates $R_1, R_2, ..., R_n$. The set $\{S_i, i = 1..n\}$ is known as the stream set. The set $\{k_i, i = 1..n\}$ is known as the operation set, where $k_j$, the number of blocks to be retrieved for stream $j$, is called the operation for that stream.

Let the size of buffer at the client $i$ be denoted as $B_i$. Then a feasible operation set is one in which

$$k_i U \leq B_i \quad i = 1..n$$

(2.1)

where $U$ is the size of the media block. The number of blocks the server reads for each client should not overflow the buffer available at the client. The above equation is known as the feasibility criterion.
Let the time for the server to retrieve $k_i$ blocks for the stream $i$ be denoted as $L(k_i)$ and the cycle time as $\tau$. Then a **continuous** operation set is one in which the time to perform the operation set $l$ is

$$l = \sum_{i=1}^{n} L(k_i) \leq \min_i \left\{ \frac{k_i U}{R_i} \right\} = \tau \quad (2.2)$$

The quantity $k_i U / R_i$ is the playback duration that the server gives to client $i$ in each cycle. The above equation simply says that the time for retrieving blocks for all the streams should be less than the minimum playback duration of all the streams. The stream that has the minimum playback duration is called the **critical** stream, because it determines the cycle time $\tau$ and the operation set. The above equation is also known as the **continuity criterion**.

If $w_i(t)$ denotes the number of blocks remaining at the buffer of client $i$, where $0 \leq w_i(t) \leq k_i$ over time $t$, then the remaining playback duration $W_i(t)$ at time $t$ for that stream is given by

$$W_i(t) = \frac{w_i(t) U}{R_i} \quad (2.3)$$

A stream is said to **starve** if $W_i(t) = 0$ during any cycle. A **safe** operation set is one in which no stream starves. Past literature [1] made the following very important claim about a safe operation set:

**Claim**

A feasible and continuous operation set is a safe operation set.
Proof

When the client buffer is filled, the client gets the maximum playback duration. Therefore, using the feasibility and continuity criteria,

\[ W_i^{max}(t) = \frac{B_i}{R_i} \geq \frac{k_i U}{R_i} \geq \sum_{i=1}^{n} L(k_i) = l \]

Now, because the client gets a load of data every cycle, \( W_i(t) > 0 \) during any service round.\( \rightarrow \)

2.2 Quality Proportional Multi-Subscriber Scheme

Recall that the server retrieves \( k_1, k_2, ..., k_n \) block units for the streams \( S_1, S_2, ..., S_n \) with playback rates \( R_1, R_2, ..., R_n \). The simplest operation set would be a fixed number of block units, say \( K \), for all the streams no matter what how different their playback rates are. By substituting \( K \) in the continuity criterion, that is,

\[ \sum_{i=1}^{n} L(K) \leq \min_{i} \left\{ \frac{K U}{R_i} \right\} = \frac{K U}{\max_{i} \{ R_i \}} \]

we notice that the streams whose playback rates are smaller than \( \max_{i} \{ R_i \} \) have more data blocks retrieved in a cycle than they really need. This simple operation set results in the server admitting fewer streams than optimum.

The Quality Proportional Multi-Subscriber Scheme (QPMS) [2] finds this optimum operation set by proposing that the higher the playback rate of a stream,
the more data block units the server should retrieve for that stream. In other terms,

\[ k_i \propto R_i \text{ or } k_i = \kappa R_i \]  

(2.4)

where \( \kappa \) is a constant of proportionality which we will call the **playback constant**.

The continuity criterion therefore becomes

\[ \frac{I}{\min_i \left\{ \frac{\kappa R_i U}{R_i} \right\}} = \kappa U = \tau. \]  

(2.5)

Furthermore, the playback durations the clients get are equalized and are given by

\[ \frac{k_i U}{R_i} = \frac{\kappa R_i U}{R_i} = \kappa U \quad i = 1..n. \]  

(2.6)

From (2.5) and (2.6), each client gets a playback duration equal to the cycle time.

This is the optimization that QPMS brings to the operation set.

### 2.3 Rounding Operation

If the playback constant is a fraction, so are the operations \( k_i = \kappa R_i \quad i = 1..n \).

Since we are reading whole values of \( U \), in our model we shall make the operations integral, that is,

\[ k_i = \left\lfloor \kappa R_i \right\rfloor \quad i = 1..n \]  

(2.7)

and modify the feasibility and continuity requirements accordingly.

This modification is still correct because the playback duration each stream gets is definitely greater or equal to the cycle time, that is,

\[ \frac{\left\lfloor \kappa R_i \right\rfloor U}{R_i} \geq \kappa U = \tau. \]  

(2.8)
In other words, the number of blocks retrieved for a stream is *a little* more than what is really needed. The server shall rectify this by rounding down the operation to \([\kappa R_i]\) in some cycles along the retrieval process so that on average the client gets \(\kappa R_i\) media blocks per cycle [2].
3 System Modelling

3.1 System Architecture

As shown in Figure 3.1, the multimedia server services several clients across an ATM network. The server architecture is shown in Figure 3.2. The nodes are identical and communicate with the manager microprocessor via an ethernet bus. The node architecture is shown in Figure 3.3. In the next section, we present two variations of the server architecture. The major difference among them is in the way the files are stored.

![Server Servicing Multiple Clients](Figure 3.1 Server Servicing Multiple Clients)
Atm Fibre Link  
Peak Rate: 140 Mbps  
Design Rate: 124.4 Mbps  

Figure 3.2 Server Architecture

Figure 3.3 Node Architecture
3.2 File Storage

Architecture I (Striping) Each file is striped along $N$ disks in a contiguous fashion. The striping granularity is a whole track length, $U_T$, which will be called a block sub-unit. For example, the first stripe of a file is on the first drive at some cylinder location, the second one on the second drive at the same cylinder location, . . . , the $N^{th}$ one on the $N^{th}$ drive at the same cylinder location, the $(N + 1)^{th}$ one back on the first drive but on the next cylinder location, and so on. Therefore, the size of media block unit is

$$U = NU_T$$

(3.1)

The reason for choosing the striping granularity to be a whole track length is to make zero-rotational-latency read possible. As soon as the head reaches the desired cylinder location, the disk controller (DC) can start reading and buffering data in its random access memory.

Other drives can be hooked in parallel to the SCSI buses to mainly increase the file storage capacity of each node. Each node may therefore have several disk arrays each of $N$ disks. Indeed, a row of drives can be viewed as a single large drive.

Architecture II (No Striping) In Architecture II, files are not striped among disks but are stored in a contiguous fashion. The size of the media block unit
in this case is

\[ U = U_T \]  

(3.2)

3.3 Data Retrieval Model

To retrieve a media block from the node, the node microprocessor sends to all the SCSI hosts a command to read a track or block sub-unit. The transfer of a block sub-unit from a particular drive across the buses and the intermediate buffers to the client is summarized in Figure 3.4. Some typical parameters are listed in Table 3.1. The drive parameters are from the drive system Seagate ST12400N/ND.

First, the microprocessor requests the VME bus. Upon getting the bus, the microprocessor sends a read command to a particular SCSI host and releases the bus. In turn, the SCSI host requests the SCSI bus, gets it, sends a read command to a particular DC and disconnects from the SCSI bus. The DC makes a seek, and buffers a block sub-unit at its memory. It then arbitrates the SCSI bus and the SCSI host allows a reconnection. The transfer of the block sub-unit from the DC's buffer to the SCSI host's buffer then takes place.

Upon receiving a block sub-unit, the SCSI host interrupts the microprocessor. An interrupt latency is encountered before the microprocessor responds. The microprocessor saves the context, and goes into polling the SCSI hosts using a rotating priority scheme. Upon identification of the interrupt source, the
microprocessor can initiate another read at that SCSI host. It then initializes registers of the DMAC at that SCSI host and releases control of the VME bus to the DMAC. A DMA burst transfer takes place from the SCSI host's buffer to the main memory, after which the microprocessor is again interrupted.

Once the block sub-unit reaches the main memory and provided the buffer at the ATM interface is not full, the microprocessor initializes the DMAC at the ATM interface for another DMA burst transfer from the main memory to the buffer at the ATM interface. To signal the end of transfer, the ATM interface interrupts the microprocessor. The ATM interface transfers the block across the switch. After the transfer, the ATM interface interrupts the microprocessor to indicate the availability of buffer.

<table>
<thead>
<tr>
<th>Minimum Seek, $S_m$</th>
<th>1 ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Seek, $S_M$</td>
<td>19 ms</td>
</tr>
<tr>
<td>Revolutions Per Minute</td>
<td>5411</td>
</tr>
<tr>
<td>Track Capacity, $U_T$</td>
<td>43 kB</td>
</tr>
<tr>
<td>Cylinders, $C_M$</td>
<td>2621</td>
</tr>
<tr>
<td>Interrupt Latency</td>
<td>200 $\mu$s</td>
</tr>
<tr>
<td>Saving Context</td>
<td>200 Instructions</td>
</tr>
<tr>
<td>Polling a Device</td>
<td>2 Instructions</td>
</tr>
<tr>
<td>Cell Header Overhead</td>
<td>5 B</td>
</tr>
</tbody>
</table>

Table 3.1 System Parameters
Figure 3.4 Figure Data Retrieval Model
3.4 VME Interrupt bus

Whether or not the files are striped among several drives, the VME bus must be fair to all the SCSI hosts attached to it, or else the one with the lowest priority interrupt may have a poor access to the VME bus. Unfortunately, the VME interrupt bus, which is made of seven interrupt lines, has a fixed priority scheme. Moreover, the other available option, daisy chaining, is "biased" to the devices located early on the daisy chain.

However fairness can be established by software by making all SCSI hosts interrupt on a single interrupt line and by using a rotating-priority polled interrupts scheme. While software polling scheme is usually slow and inefficient, polled interrupts scheme is not. Once an interrupt occurs, polling is used only to identify which device has pulled the interrupt line low.

Other devices contending for the VME, such as the ATM interface, the Ethernet interface, can interrupt the microprocessor on other interrupt lines. The interrupt lines must be properly allocated to the devices and properly disabled by the microprocessor such that

- no device can preempt a DMA burst transfer once it starts.
- the microprocessor manager has a fast access to the VME bus and thus to the node microprocessor for communication purposes.
3.5 Client Buffering Requirements

A media block unit is striped among several disks in parallel. When a media block unit is being retrieved, the media block sub-units (or stripes) may reach the client buffer in a random chronological order. This represents a synchronization problem if clients have FIFO buffers. We circumvent this synchronization problem by using random-access memory (RAM) at the client.

Moreover, because the data retrieval model is quite complex, predicting exactly when in a cycle the media block sub-units may reach the client buffer is impossible. Therefore a stream may starve even though it is given a playback duration equal to or slightly greater than the cycle time.

Noting that within a cycle the client shall receive a load of data in accordance to the continuity criterion, we can avoid the danger of starvation by using double buffering at the clients. That is, the RAM buffer size at the client should be twice the load of data it gets in a cycle. We suggest that the client should start playback only at the beginning of the second cycle, after it has buffered its first load of data. While the client is playing back one load of data (stored in the first buffer), it has enough buffer space (the second buffer) to buffer another load of data which is expected within the cycle. At the end of the cycle, the first buffer becomes free, while the second buffer continues the playback. Continuity of data is thus preserved as can be seen in Figure 3.5.
A client may require one extra block of buffer because of rounding operations. Because operations are rounded up, a client effectively receives a playback duration \textit{à little} greater than the cycle time, that is, a client obtains an extra fraction of a block. This extra data accumulates cycle after cycle. When close to one block of data is in this way accumulated, the server then rounds the operation down. The extra block of buffer is needed to accommodate the extra block a client accumulates.

Figure 3.6 depicts the importance of an extra block of buffer at the client. The unrounded operation is 3.8. The rounded-up and rounded-down operations are 4 and 3 respectively.
Figure 3.5 Double Buffering

<table>
<thead>
<tr>
<th>cycles</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td># blocks retrieved</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td># blocks played back</td>
<td>0</td>
<td>3.8</td>
<td>3.8</td>
<td>3.8</td>
<td>3.8</td>
<td>3.8</td>
</tr>
<tr>
<td># blocks in excess</td>
<td>0.2</td>
<td>0.4</td>
<td>0.6</td>
<td>0.8</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td># blocks at end of cycle</td>
<td>4</td>
<td>4.4</td>
<td>4.6</td>
<td>4.8</td>
<td>4</td>
<td>4.2</td>
</tr>
<tr>
<td>rounding up/down</td>
<td>up</td>
<td>up</td>
<td>up</td>
<td>up</td>
<td>down</td>
<td>up</td>
</tr>
</tbody>
</table>

Figure 3.6 Rounding Operation
However, without loss of generality, we shall continue to use single buffering in the analysis.

### 3.6 Circular Scan Disk Scheduling Policy

In this policy [6], the disk head generally scans in one direction, from the outermost toward the innermost tracks, reading tracks for the streams in the order of the scan direction. In the next cycle, the scan direction is reversed, that is from the innermost toward the outermost track. This policy is called Circular Scan (CSCAN). Overlapping files may cause a slight deviation from the above policy.

At the beginning of each cycle, the server reorders the streams in accordance to the cylinder location of their reads within each disk-array. The order in which streams are serviced changes with cycles for the following reasons. Firstly, the scan direction switches back and forth from cycle to cycle. Secondly, because of different operations and because files are located on different platters, the head position for a higher-rate stream may pass the position for a lower-rate stream. Stream set reordering poses no starvation threat if double buffering is used at the clients.

The following two sections aim at finding the worst-case interfile seek for CSCAN disk scheduling policy.

### 3.7 Interfile Seek

The measured seek-time-versus-distance curve typically [4] looks like that
shown in Figure 3.7. We consider a simple model where seek time is linear with the distance that the head travels; after an initial start-up time $S_m$. Our simple model uses the minimum seek time and the maximum seek time given in the drive specifications.

An interfile seek is a seek from the last track that the head has just read for the previous stream to the first track that the head should read for the next stream. There are two scenarios for interfile seeks as shown in Figure 3.8. In the first scenario, the tracks that the head should read for any two streams $i-1$ and $i$ do not overlap in cylindrical location. After performing the operation for stream $i-1$, the head travels a few cylinders, say $d_i$, along the scan direction to the desired track at which it can perform the operation for stream $i$.

In the second scenario, the tracks that the head should read for two streams overlap in cylindrical location. This overlap can occur because the files are physically located on different platters and may overlap vertically. Overlap can also occur because the two streams request the same file with very small phase difference. In both cases, as the head finished the operation for stream $i-1$, it has to retract a few cylinders (of course smaller than or equal to $k_{i-1}$) in the direction opposed to the scan direction before performing the operation for stream $i$.

If $d$ denotes the distance in cylinders, then from Figure 3.5 the interfile seek
is given by

\[ S_{f,i} = \begin{cases} 
S_m + \Delta S_{d=d_i}, & \text{no overlap} \\
S_m + \Delta S_{d \leq k_{i-1}}, & \text{overlap}
\end{cases} \quad (3.3) \]

Figure 3.7 Seek Time versus Distance Travelled

Figure 3.8 Interfile Seek Scenarios
3.8 Worst-case Interfile Seek

Let $\theta$ be a subset of the operation set whose operations involve seeks with no overlap. Assume the node has only one disk array. The total time a drive spends on seeking in a cycle is given by

$$T_S = \sum_{i=1}^{n} [S_{f,i} + (k_i - 1)S_m]$$

$$= \sum_{i \in \theta} S_{f,i}^{(no\ overlap)} + \sum_{i \in \theta'} S_{f,i}^{(overlap)} + \sum_{i=1}^{n} (k_i - 1)S_m$$

$$= \sum_{i \in \theta} (S_m + \Delta S_{d=d_i}) + \sum_{i \in \theta'} (S_m + \Delta S_{d\leq k_i}) + \sum_{i=1}^{n} (k_i - 1)S_m$$

$$= \sum_{i \in \theta} \Delta S_{d=d_i} + \sum_{i \in \theta'} \Delta S_{d\leq k_i} + \sum_{i=1}^{n} k_i S_m$$

The total number of cylinders the head travels over in a cycle can never be greater than the maximum number of cylinders of the drive, plus twice the sum of the cylinders the head retracts. Thus,

$$T_S \leq \Delta S_{d=C_M} + 2 \left( \sum_{i \in \theta'} \Delta S_{d\leq k_i} \right) + \sum_{i=1}^{n} k_i S_m$$

$$\leq \Delta S_{d=C_M} + 2 \left( \sum_{i \in \theta'} \Delta S_{d= k_i} \right) + \sum_{i=1}^{n} k_i S_m$$

$$\leq \Delta S_{d=C_M} + 2 \left( \sum_{i=1}^{n} \Delta S_{d= k_i} \right) + \sum_{i=1}^{n} k_i S_m$$

$$= (S_M - S_m) + 2 \left( \sum_{i=1}^{n} \frac{k_i}{C_M} (S_M - S_m) \right) + \sum_{i=1}^{n} k_i S_m$$

$$= \left\{ \sum_{i=1}^{n} S_m + (S_M - S_m) + 2 \left( \sum_{i=1}^{n} \frac{k_i}{C_M} (S_M - S_m) \right) \right\} + \sum_{i=1}^{n} (k_i - 1)S_m$$

23
If $M'$ denotes the number of disk-arrays from which streams are requesting files, then by inspection,

$$T_S \leq \left\{ M'(S_M - S_m) + \sum_{i=1}^{n} S_m + 2 \left( \sum_{i=1}^{n} \frac{k_i}{C_M} (S_M - S_m) \right) \right\} + \sum_{i=1}^{n} (k_i - 1) S_m$$

(3.4)

Let $\gamma_j$ be a subset of the stream set whose streams correspond to the files physically located in the disk-array $j$ and let $f(\gamma_j)$ denotes the first stream, in the scan direction, of the subset $\gamma_j$. Then, a worst-case model for the interfile seek is

$$S_{f,i} = \begin{cases} S_m + (S_M + S_m) + \frac{2k_i}{C_M} (S_M - S_m), & i = f(\gamma_j), \ j = 1..M \\ S_m + \frac{2k_i}{C_M} (S_M - S_m), & \text{otherwise} \end{cases}$$

(3.5)

### 3.9 Determining Time to Perform Operation Set

The time to perform the operation set is determined by simulation. The buses, namely, the SCSI buses, the VME bus and the ATM fibre link, and the intermediate buffers, namely, the SCSI host's buffer, the main memory and the ATM interface's buffer are modelled as resources. The queue model for the queues at the resources, except the SCSI bus, is First-Come-First-Serve. The queue at the SCSI bus is a priority queue because the SCSI bus has a fixed priority scheme.

The transfer of a media block sub-unit from the drive to the client is modelled as a process named Read_a_block. Some important parameters that the process uses are:

- the interfile seek time and the minimum seek time
the time a block sub-unit is read and buffered at the DC

- the time durations for a block sub-unit be transferred from the DC to the SCSI host, from the SCSI host to the main memory, from the main memory to the ATM interface, and from the ATM interface across the ATM switch to the client.

- interrupt latencies

At the beginning of the simulation, \( N \) Read_a_block processes are created because of \( N \) parallel drives. During the simulation, without going into details, each of those processes will recursively generate at appropriate events a total of \( \sum_{i=1}^{n} k_i \) identical Read_a_block processes inclusive. Processes make requests for the resources, suspend themselves for some amount of time, are suspended if resources are held, are reawakened if resources become free. The simulation ends when all Read_a_block processes die. \( l \) is the time from the beginning of the simulation to the end of the simulation.

If we use the worst-case model for the interfile seek, the \( l \) obtained from simulation is also worst-case. Therefore, we need to determine \( l \) by simulation only once to test if the operation set is continuous.
4 Performance Analysis

4.1 Admission Control

Architecture I If a node has $n$ streams running with the two safety requirements satisfied, that is,

$$k_i U = \kappa R_i U \leq B_i \quad i = 1..n,$$

and

$$\sum_{i=1}^{n} L(\kappa R_i) \leq \min_{i} \left\{ \frac{\kappa R_i U}{R_i} \right\} = \kappa U,$$

and if a new stream $S_{n+1}$ with playback rate $R_{n+1}$ subscribes, then the server can accepts the stream only if a new playback constant $k'$ exists such that the two safety requirements are still satisfied, that is,

$$k'R_i U \leq B_i \quad i = 1..n + 1,$$

and

$$\sum_{i=1}^{n+1} L(\kappa' R_i) \leq \kappa' U.$$

Architecture II In Architecture II, the drives in fact operate independently, each servicing a small set of streams. In other words, each drive $w$ has a stream set of $n_{d_w}$ streams and an associated playback constant $\kappa^{(w)}$ depending on $n_{d_w}$. Each drive can be thought of as a “mini-node” within a node.
Say a new stream requests a file located in some drive \( W \). The node can take the new stream only if there exists a new playback constant \( \kappa^{(W)}' \) such that, like in Architecture I, the two safety requirements are still satisfied, that is,

\[
\kappa^{(W)}' R_i U_T \leq B_i \quad i = 1..n_{dW} + 1,
\]

and

\[
\sum_{i=1}^{n_{dW}+1} L(\kappa^{(W)}' R_i) \leq \kappa^{(W)}' U_T,
\]

\[
\sum_{i=1}^{n_{dW}} L(\kappa^{(w)} R_i) \leq \kappa^{(w)} U_T, \quad w = 1..M N, \quad w \neq W.
\]

4.2 Minimum Operation Set

Past research [1] [2] has found the minimum safe operation set by finding the minimum playback constant with the two safety requirements satisfied. In this case, the server is busy almost throughout the service round either seeking or buffering data at the drive. In other words, the time to perform the operation set is very close to the playback duration that each stream gets or the cycle time.

This technique is not adequate when absorption of newly accepted streams is to take place. As a new stream is accepted, the playback constant, thus the cycle time and the time to perform the new operation set, will increase. In the transition, starvation is likely to occur as insufficient data is available for the existing streams to play over the time to perform the new operation set.
Past research [3] added elasticity to the system by modifying the continuity criterion as follows:

$$I = \sum_{i=1}^{n} L(k_i) < \sum_{i=1}^{n} L(k_i + 1) \leq \min_{i} \left\{ \frac{k_i U}{R_i} \right\} = kU = \tau.$$  

The above equation states that whether the server retrieves for stream $i$, $k_i$ or $k_i + 1$ media blocks, the continuity criterion is not violated. The elasticity from this modification is shown in the following example.

Say as a result of a new subscription, the operation for a particular stream changes from 6 to 10 media blocks. Because of the above modification, the server can safely retrieve for that stream 7 media blocks a cycle instead of 6. Getting a block of data in excess in every cycle, the client finds itself with 10 media blocks after 4 consecutive cycles. (In other words, the client’s playback duration is increased until it becomes comparable with the new cycle time.) After the playback duration for all existing streams have been adjusted in the same way in parallel, the server can then absorb the new subscriber (or perform the new operation set) in the fifth cycle.

This method has a few disadvantages. First, this technique is well suited for a unique case where all streams have the same playback rate. Second, in a transition only one new stream can be absorbed. Third, the transition can span through many cycles.
4.3 Maximum Operation Set

To make absorption of new streams elastic, we suggest finding the maximum safe operation set by finding the maximum playback constant $n_{max}$ which satisfies the two safety requirements. The clients get the maximum playback duration that their buffers allow. In this scenario, if the system is not loaded, the server is busy for as long as it performs the operation set and is slack during the rest of the cycle time (See Figure 4.1). The slack time $s$ is given by

$$s = \tau - l.$$  \hfill (4.1)

This slack time makes the absorption process very elastic. A newly accepted stream is absorbed by placing its operation in the ordered operation set and performing the new operation set. Data continuity is not impaired at existing streams because double buffering is used. Recall that with double buffering, because each existing stream always plays back the load of data it received in the previous cycle, exactly when in a cycle the server performs the operations of the streams does not matter (See Figure 3.5).
4.4 Finding the Maximum Operation Set

The following claims (with informal proof) help in finding the maximum playback constant.

**Claim 3.1**
Increasing the playback constant, thus the operation set, increases the continuity criterion of the set.

**Proof**

The extra playback duration each client get from an extra block is larger than
the time the server retrieves that extra block. Thus when the playback constant increases, so does the slack time $s$.

**Claim 3.2**

Increasing the playback constant, thus the operation set, decreases the feasibility criterion of the set.

**Proof**

As $k_i = \kappa R_i \ i = 1..n$ increases, the quantity $B_i - k_i U \ i = 1..n$ decreases.

To calculate the maximum playback constant, the following steps are taken:

- Find the maximum feasible operation set, or find $\kappa = \kappa^{max}$ such that

$$[\kappa^{max} R_i] U \leq B_i \ i = 1..n.$$ (4.2)

- Then, check if with $\kappa = \kappa^{max}$ the set is continuous.
- If it is not, there is no feasible-continuous operation set. From claim 3.2, if for $\kappa = \kappa^{max}$ the set is not continuous, there is no feasible-continuous set.

From claim 3.1, we can use the binary search concept to find $\kappa^{max}$. We start with a high $\kappa$ (corresponding to non-feasible set, say 1.0) and a low $\kappa$ (corresponding to feasible set, say 0.0). The range (0.0, 1.0) is binary sliced repeatedly until a given tolerance is reached. As a result of setting the low $\kappa$ to 0.0 with a sufficiently low tolerance, we are guaranteed that a feasible set will always be found. Then $\kappa^{max}$ is used to test if the set is continuous. The algorithm is summarized in the flowchart given in Figure 4.2.
Figure 4.2 Calculating the Maximum Playback Constant
4.5 Performance

To analyze the performance of the architectures, we assume for simplicity that all streams are compressed video. Our aims are to investigate the relationships amongst the number of concurrent streams, the number of parallel drives and the client buffer size, to identify bottlenecks and to determine if the ATM switch can be used to the maximum.

Architecture I (striping)  If the number of parallel drives $N$ is 1, the VME bus and the ATM fibre link will not be utilized as long as the drive is seeking and buffering data at its DC. If $N$ is increased, other drives in the array can use that idle time for data transfer. Increasing $N$ in this fashion corresponds to a virtual increase in data retrieval rate and to increased utilization at the resources. We shall increase $N$ until one of the resources, the VME bus or the ATM fibre link, saturates. The point at which saturation occurs will correspond to the maximum number of streams the node can support. Figure 4.3 shows the following.

- The number of concurrent streams increases with increasing client buffer size until a limit, caused by the speed of the drive arrangement, is reached.
- As $N$ increases from 1 to 5, the virtual speed of the drive arrangement increases resulting in an increased number of concurrent streams.
- When $N$ is 5, the node can support 79 concurrent streams. This maximum is reached because the ATM fibre link (at its designed bandwidth) saturates and
becomes the bottleneck. This means that the ATM switch, acting mainly as a concentrator for the five identical nodes, can be used to the maximum. A value of $N$ greater than 5 does not increase the number of streams supportable.
Figure 4.3 Performance of Architecture I
Architecture II (No striping) In this architecture files are not striped among drives. All drives operate independently. The speed of a drive is high enough to service a small set of streams. Each drive competes for the buses and the intermediates buffers.

- When file requests are random, the same performance is obtained as with striping, that is a maximum of 79 video streams are supported, with the ATM fibre link reaching saturation. However, the maximum performance is obtained with a smaller buffer size at the clients.

- When all streams request the same file or files stored in the same drive, thus contending for the same drive, the performance is reduced considerably as only one drive is operating. The drive speed is so low that even though client buffer is increased, performance is not improved.
Figure 4.4 Performance of Architecture II
5 Design of New Admission Control: Architecture I

5.1 Problems of Dynamic Playback Constant

Recall that the maximum playback constant was previously calculated as follows:

- Find the maximum feasible operation set, or find $\kappa = \kappa_{\text{max}}$ such that $\kappa_{\text{max}} R_i U \leq B_i \quad i = 1..n$.
- Then, check with $\kappa = \kappa_{\text{max}}$ if the operation set is continuous.

From above, $\kappa_{\text{max}}$ can be seen to depend on the buffer size available at each and every client. Therefore, as the set of streams changes either because a new stream subscribes or because an old stream dies, $\kappa_{\text{max}}$, and thus the operations, also change.

A changing $\kappa_{\text{max}}$ is highly undesirable because of the following two reasons. First, when $\kappa_{\text{max}}$ changes, the server needs an adjustment phase. For example, suppose a stream subscribes and is accepted, and then $\kappa_{\text{max}}$ increases. In general, the server should not immediately absorb the new stream even though there exists plenty of slack time. In the adjustment phase, the server has to use the available slack time to adjust the playback duration of existing streams to the new cycle time. It is only after this adjustment that the server can absorb the new stream.
In the special case where slack time is large, the adjustment and the absorption can take place within a single cycle.

Secondly, letting streams subscribe with a client-specified buffer size can reduce performance of the server. If a stream subscribes with a "stupidly" small buffer, the server may accept and absorb it but with the new $\kappa_{max}$ becoming small. Because $\kappa_{max}$ is small, the cycle time, thus the slack time, is also small. Because the slack time is small, the critical stream constrains the server into accepting a small number of streams, even though the other streams have large buffers.

Can we remove the dependence of the playback constant on the size of client buffers?

5.2 Rate-based Buffer Weighting Scheme

QPMS suggested that the operation for a client should be proportional to its playback rate. It is natural then that the amount of data being buffered at the client might be also proportional to the client's playback rate. We, therefore, introduce the Rate-based Buffer Weighting Scheme RBWS (an accompaniment to QPMS) in which each client should subscribe with a buffer also proportional to its playback rate. Can such a buffer constraint on clients fix the playback constant? We make the following claim:

Claim

If each stream subscribes such that
its operation is proportional to its playback rate (QPMS), that is,

\[ k_i \propto R_i, \]

or more precisely,

\[ k_i = \lceil \kappa R_i \rceil \]

its buffer is also proportional to its playback rate (RBWS), that is,

\[ B_i \propto R_i \quad \text{or} \quad B_i = \beta R_i, \quad (5.1) \]

or more precisely,

\[ B_i = \left\lceil \frac{\beta R_i}{U} \right\rceil U, \quad (5.2) \]

then, assuming the set is continuous, the playback constant does not change with the streams in the stream set. \( \beta \) is a proportionality constant which we call the rate-based buffer weight constant.

**Proof**

To prove this, without loss of generality, let us drop the ceiling and floor functions. Given \( k_i = \kappa R_i \) and \( B_i = \beta R_i \), then the feasibility criterion leads to:

\[ k_i U \leq B_i \quad i = 1..n \]

\[ \kappa R_i U \leq \beta R_i \quad i = 1..n \]

\[ \kappa \leq \frac{\beta}{U} \quad i = 1..n. \quad (5.3) \]
We thus notice that if $\beta$ is fixed,

- the playback constant does not depend on the buffer size at the clients.
- the playback constant does not change with the number of subscribers in the stream set.—

To make absorption of streams elastic, we choose the playback constant to be maximum. Therefore, from (5.3), the maximum playback constant is given by

$$
\kappa^{\text{max}} = \frac{\beta}{U}.
$$

(5.4)

5.3 Choosing the Rate-based Buffer Weight Constant

The rate-based buffer weight constant, along with the number of parallel drives among which files are striped, should be judiciously chosen such that the maximum number of streams accepted corresponds to the saturation of a resource in the node. In our case, it is the ATM fibre link that becomes the bottleneck.

When the CSCAN disk scheduling policy is used and when all streams are video streams, the ATM fibre link nearly becomes saturated when the number of parallel drives $N \geq 5$ and the client buffer $B_v \geq 1$ MB. The region $\{N \geq 5, B_v \geq 1 \text{ MB}\}$ is the region where the buffer-rate weight constant should be picked.

With a view to keeping the client buffers small we somewhat arbitrarily choose the point $\{N = 6, B_v = 1 \text{ MB}\}$ to fix $\beta$. At this point, if $R_v$ denotes the playback
rate of a video file, the rate-based buffer weight constant is given by

\[ B_v = \beta R_v, \quad (5.5) \]

or more precisely,

\[ \beta = \left[ \frac{B_v}{U} \right] \frac{U}{R_v}, \quad (5.6) \]

where \( B_v = 1 \, MB \) and \( R_v = 175 \, kBps \).

**5.4 Maximum Operation Set Size**

The size of the operation set \( P \) is the total number of blocks that the server/node retrieves in a cycle for all the streams in the stream set. Thus,

\[ P = \sum_{i=1}^{n} k_i. \quad (5.7) \]

The maximum size of the operation set \( P^{max} \) is the maximum number of blocks that the server retrieves in total for the stream set, where the maximum is due to the saturation of the ATM fibre link. At the point where \( \beta \) was picked, the maximum number of video rate streams \( n_v \) that the node can handle is 79. Thus,

\[ P^{max} = \sum_{i=1}^{n_v} k_i \]

\[ = k_v n_v, \quad \text{homogeneous set} \]

\[ = \left[ \kappa^{max} R_v \right] n_v \]

\[ = \left[ \frac{\beta}{U} R_v \right] n_v, \quad \text{from (5.4)} \]
\[
\begin{align*}
&= \left\lfloor \frac{B_v}{U} \frac{U}{R_v} R_v \right\rfloor n_v, \quad \text{from (5.6)} \\
&= \left\lfloor \frac{B_v}{U} \right\rfloor n_v \\
&= \frac{B_v}{U} n_v & (5.8)
\end{align*}
\]

When all subscribers are video streams, the maximum operation set size is known. The quantity \( P^{max} \) is therefore an important quantity as it is the basis for a new way of designing the Admission Control. This quantity can now be used for predicting whether a node can accept a video stream or not. Basically, the node can accept a new video subscriber only if the new operation set size is less than or equal to \( P^{max} \). Is the importance of \( P^{max} \) restricted to the homogeneous case, where all subscribers are video streams? Is \( P^{max} \) just as important for the heterogeneous case, where subscribers can have any playback rate? We make the following claim:

**Claim**

Provided the following conditions hold, namely,

- the CSCAN disk scheduling policy is used,
- the chosen \( N \) is unchanged,
- the maximum playback constant is used,
• the buffer-rate weight constant is fixed at the value chosen for the homoge-
neous case,

then, however heterogeneous the stream set is, the ATM fibre link is bound to
saturate when the operation set size reaches the maximum operation set size.

**Heuristic Proof**

The following two arguments give an intuitive proof of the claim:

□ Whatever the stream set is, the worst case model for the interfile seek, given
by (3.5), is *almost* the same because the quantity \( \frac{2k_i}{C_M}(S_M - S_m) \) is very small
relative to \( S_M \) and \( S_m \). This means that whatever the stream set is, the worst
case arrival pattern of media block sub-units from a row of drives at the ATM
fibre link does not change much.

□ Recapitulate that a row of drives are not utilizing the resources as long as

• they seek
• they buffer data at the DC’s.

Denote \( T_B \) as the total time that a row of drives spend on buffering data at
the DC’s in a cycle. A track of data takes one revolution time to be buffered
at the DC. If \( r \) is the revolution time, then

\[
T_B = r \sum_{i=1}^{n} k_i. \tag{5.9}
\]
The total time a row of drives spend on seeking in a cycle is given by (3.4).

Adding $T_S$ and $T_B$ gives

$$
T_S + T_B \leq M'(S_M - S_m) + \sum_{i=1}^{n} S_m + 2 \sum_{i=1}^{n} \frac{k_i}{C_M} (S_M - S_m) + \sum_{i=1}^{n} (k_i - 1)S_m + r \sum_{i=1}^{n} k_i
$$

$$
= M'(S_M - S_m) + 2 \sum_{i=1}^{n} \frac{k_i}{C_M} (S_M - S_m) + \sum_{i=1}^{n} k_i S_M + r \sum_{i=1}^{n} k_i
$$

$$
= M'(S_M - S_m) + \left[ \frac{2(S_M - S_m)}{C_M} + S_m + r \right] \sum_{i=1}^{n} k_i
$$

$$
= (const) + (const) \sum_{i=1}^{n} k_i. \quad (5.10)
$$

During this time $T_S + T_B$ other identical rows of drives working in parallel help to saturate the ATM fibre link during a cycle. Now, if the ATM fibre saturates for a particular homogeneous operation set whose size is equal to greater than $P_{max}$, then the ATM fibre will saturate for any heterogeneous operation set whose size is equal to or greater than $P_{max}$. The reason is that, from (5.10), the bound on $T_S + T_B$ depends on the size of the operation set, not on the size of a particular homogeneous stream set.

The above claim is very useful for designing the Admission Control for Architecture I.

## 5.5 Admission Control

Let the CSCAN policy be used. Let $N$, $k_{max}$, $\beta$, and $P_{max}$ be unchanged.
If a node is currently servicing $n$ streams, then the node can accept and absorb a new stream $j$ of playback rate $R_j$ only if

- it subscribes with a buffer of size

$$B_j \geq \left\lceil \frac{\beta R_j}{U} \right\rceil U,$$

- and the new operation set size is less than or equal to $P^{max}$, that is

$$\sum_{i=1}^{n+1} k_i \leq P^{max}. \tag{5.11}$$

In the simulation Figure 5.1, streams subscribe with different playback rates randomly between 10 kBps and 500kBps. The following points are noted:

- As the number of subscribers increases, the time to perform the operation set increases. Streams are accepted as long as this time is less than the cycle time.
- The proposed Admission Control can effectively predict whether the node can accept a new stream or not. At the point where the Admission Control rejects a subscriber, that is at the point where operation set size exceeds $P^{max}$, the time to perform the operation set also exceeds the cycle time.
- The point at which the Admission Control is about to reject a stream corresponds to saturation of the ATM fibre link. The Admission Control is therefore efficient.
Figure 5.1 Effectiveness of Admission Control
6 Design of New Admission Control: Architecture II

6.1 Stream Admission

In Architecture II, data is not striped among parallel disks. The drives operate independently, each servicing a small set of streams and competing for the resources and intermediate buffers. Therefore a stream can be rejected because of two reasons. First, one of the resources (in our case, the ATM fibre link) saturates and becomes the bottleneck.

Second, the speed of a drive under contention is so low that even though no resource saturates and even though client buffer is large, the drive cannot accommodate another stream. In this case, we shall say that the drive is limit stressed.

6.2 Choosing the Rate-based Buffer Weight Constant

Like Architecture I, Architecture II employs both QPMS and RBWS. The rate-based buffer weight constant should be wisely picked so that the maximum number of streams accepted corresponds to saturation of a resource, or in case of severe contention, to a drive being limit stressed.

From Figures 4.3 and 4.4, the region \( \{B_v \geq 0.5 \text{ MB}\} \) is a good region to pick \( \beta \). \( \beta \) should be as small as possible so that client buffers do not need be
large. We therefore pick the rate-based buffer weight constant as

\[ \beta = \left[ \frac{B_v}{U} \right] \frac{U}{R_v} \]

where now \( B_v = 0.5 \, MB \) and \( U = U_T \).

### 6.3 Drive Maximum Operation Set Size

Each drive services a small set of streams. The operation set size for a drive \( w \) servicing \( n_{d,w} \) streams is the number of block units the server retrieves from the drive, and is given by

\[ P^w = \sum_{i=1}^{n_{d,w}} k_i. \quad (6.1) \]

At the point where \( \beta \) was picked, the maximum number of video streams \( n_{v,dr} \) a drive can support in case of severe contention is 15. This limit results because the drive is limit stressed.

The drive maximum operation set size \( p^{max,dr} \) is the maximum number of blocks that can be retrieved from a particular drive which is limit stressed. It is given by (5.8), modified slightly, as

\[ p^{max,dr} = \left[ \frac{B_v}{U_T} \right] n_{v,dr}. \quad (6.2) \]

This quantity is an important quantity as it can be used to predict how many video streams a drive can handle. Like before, we ask if this quantity is just as important in predicting whether or not a drive can take a stream of any playback...
rate. We make the following claim, somewhat similar as before:

Claim

Provided the following conditions hold, namely,

- the CSCAN disk scheduling policy is used,
- the maximum playback constant is used,
- the rate-based buffer weight constant is fixed at the value chosen for the homogeneous case,

then, however heterogeneous the stream set for a drive is, that drive is bound to become limit stressed when its operation set size reaches the drive maximum operation set size.

Heuristic Proof

We give two arguments (similar to those in Section 5.4) to support the claim:

□ Whatever the stream set for a drive, the worst case model for the interfile seek is almost the same.

□ If \( n_{d_j} \) denotes the number of streams that drive \( j \) is servicing, the total time the drive spends on seeking and buffering data at the DC is given from (5.10) as

\[
T_S + T_B \leq (S_M - S_m) + \left[ \frac{2(S_M - S_m)}{C_M} + S_m + r \right] \sum_{i=1}^{n_{d_j}} k_i = (\text{const}) + (\text{const}) \sum_{i=1}^{n_{d_j}} k_i
\]  

\tag{6.3}

\]
Again, because $T_S + T_B$ depends on the operation set size for the drive, rather than on the individual operations, the latter is bound to become limit stressed if its operation set size reaches $P_{\text{max,dr}}$, irrespective of whether its stream set is homogeneous or heterogeneous.

### 6.4 Maximum Operation Set Size

The stream set for the node is the union of the stream sets of the drives in the node. If the number of drives in the node is $NM$, the size of the operation set for the node is given by

$$P = \sum_{w=1}^{NM} P^w = \sum_{u=1}^{NM} \sum_{j=1}^{n_{dw}} k_j = \sum_{i=1}^{n} k_i$$  \hspace{2cm} (6.4)

where $n = \sum_{w=1}^{NM} n_{dw}$.

At the design point where $\beta$ was picked, the maximum number of video streams $n_v$ the node can handle is 79. This limit is due to the saturation of the ATM fibre link. The maximum operation set size given by (5.8), with proper modification, becomes

$$P_{\text{max}} = \frac{B_v}{U_T} \dot{n}_v.$$  \hspace{2cm} (6.5)
As before, we ask whether this quantity can be important in predicting admission of a stream of any playback rate. We make the following claim:

**Claim**

Provided the following conditions hold, namely,

- the CSCAN disk scheduling policy is used,
- the maximum playback constant is used,
- the rate-based buffer weight constant is fixed at the value chosen for the homogeneous case,

then, *however heterogeneous the stream sets are*, the ATM fibre link is bound to saturate when the operation set size for the node reaches $P_{\text{max}}$.

**Heuristic Proof**

Notice that since the SCSI bus rate is higher than the ATM fibre link rate, if a bottleneck is to occur, it would be due to the ATM fibre link.

For both the homogeneous and heterogeneous cases, the rate-based buffer weight constant, and thus the cycle time, is the same. Therefore, for both cases, when the ATM fibre link is not saturated, the number of media blocks sent across it in a time slice of duration equal the cycle time is constant for a particular operation set size. This occurs even though the cycles for any drive are unsynchronized with those of the other drives in the node.

As the operation set size increases until $P_{\text{max}}$ is reached, the competition for
and queueing at the resources increases steadily, until the ATM fibre link reaches saturation. If for the homogeneous video-stream case the ATM fibre link saturates while handling $\text{P}^{\text{max}}$ block units for a given cycle time, it should saturate while handling $\text{P}^{\text{max}}$ block units for any heterogeneous case provided the cycle time is of the same duration as for the homogeneous case.

The above claims are important in the design of the Admission Control for Architecture II.

6.5 Admission Control

Let the CSCAN policy be used. Let $\kappa^{\text{max}}$, $\beta$, $\text{P}^{\text{max},\text{dr}}$, and $\text{P}^{\text{max}}$ be unchanged. Suppose a node is currently servicing $n$ streams. Suppose each drive $w$, $w = 1..MN$ is servicing $n_{d_w}$ streams such that $n = \sum_{w=1}^{NM} n_{d_w}$. Then the node can accept and absorb a new stream $j$ requesting a file physically located in some drive $W$ and having playback rate $R_j$, only if

- it subscribes with a buffer of size
  \[
  B_j \geq \left\lceil \frac{\beta R_j}{U_T} \right\rceil U_T,
  \]
- the new operation set size for the drive is less than or equal to $\text{P}^{\text{max},\text{dr}}$, that is
  \[
  \sum_{i=1}^{n_{d_w}+1} k_i \leq \text{P}^{\text{max},\text{dr}},
  \]
- the new operation set size for the node is less than or equal to $\text{P}^{\text{max}}$, that is
  \[
  \sum_{i=1}^{n+1} k_i \leq \text{P}^{\text{max}}.
  \]
In the simulations of Figure 6.1 and 6.2, streams subscribe with playback rates between 10 kBps to 500 kBps. For Figure 6.1, streams request files in a random fashion whereas for Figure 6.2 streams contend for files located on a single drive. Streams are accepted as long as no starvation is detected. We notice the following:

- The new Admission Control can effectively predict whether a new stream can be accepted. The number of streams accepted by the new Admission Control is very close to the number of streams derived from simulation with no starvation.

- For Figure 6.1, the point at which the new Admission Control rejects a stream corresponds to the saturation of the ATM fibre link.

- For Figure 6.2, the throughput at ATM fibre at the point at which the new Admission Control rejects a stream is very close to the ATM fibre link throughput at which rate-based buffer weight constant is picked. At this point the drive was said to be limit stressed.
Figure 6.1 Effectiveness of Admission Control (Random Files Requests)
Figure 6.2 Effectiveness of Admission Control (Severe Contention)
7 Further Discussions

7.1 Comparing Architectures

Performance Both architectures can maximize throughput at the ATM fibre link and thus at the ATM switch.

Contention In Architecture I, a node does not have a contention problem. Because the node is servicing streams in bursts every cycle, whether the streams request a particular file or different files is irrelevant. If all streams contend for the same file (taking the worse case, with no phase difference among them), the drives of the node simply need to read the same data as many times as needed, provided the time to do so is less than the cycle time.

Architecture II suffers from a contention problem. If all streams request the same file or files located on the same drive, they will contend for the same drive. Unfortunately, the drive cannot service any more concurrent streams after it becomes limit stressed.

Rate-based Buffer Weight Constant Architecture II has a smaller rate-based buffer weight constant than Architecture I. This means that smaller client buffers are required in Architecture II than in Architecture I.
Reliability  Architecture II is more reliable than Architecture I if no redundancy and disk mirroring are assumed. In Architecture II, the probability that a file request is unachievable is in fact the probability that the drive in which the file is located fails. Let this probability that a drive fails be $p$.

In Architecture I, the probability that a file request is unachievable is the probability that the disk-array in which the file is located fails. Due to striping, a whole disk-array fails if one or more drives in it fails. Probability a disk-array of $N$ drives fails is thus $1 - (1 - p)^N$ or approximately $Np$ if $N$ is large.

Obviously, $p$ is smaller than $1 - (1 - p)^N \approx Np$.

7.2 Admission Control Additional Features

One possible criticism of our system is that rounding down operations results in reduced throughput of the server. To restore the throughput to its maximum, the new Admission Control should include the following additional features.

Staggered Rounding Technique  Past research [2] proposed a staggered rounding technique. In this technique two (or more) streams can be matched, that is two streams can share efficiently the space corresponding to 1 media block. For example, a stream whose unrounded operation is 4.2 can be matched with one whose unrounded operation is 6.8. The toggling for both streams can be easily synchronized such that when the operation for one of the streams is rounded up,
the operation for the other is rounded down. Therefore the two streams use a space of 11 blocks \([4.2 + 6.8]\) instead of 12 blocks \([4.2] + [6.8]\).

**Dynamic Stream Set**  For very small playback rates, the unrounded operation may be small. For example, for an audio rate file (8 kbps), the unrounded operation is 0.14. However, when the server rounds the operation up, the audio stream receives 1 block of data, but the audio stream needs only 0.14 block per cycle. The audio stream has so much excess data that the server indeed does not need to retrieve any data for it for the next 6 cycles. When the node accepts one audio stream, it could accept six other audio streams as well. The server can keep track of this fact and create a *dynamic stream set*. In this set, seven audio streams, in seven consecutive cycles, multiplex the space that one audio stream uses.

**Non-Time-Critical Streams**  Finally, not all media are time critical. The space made free due to rounding down of operations could also be used by non-time-critical media such as text and still images.

### 7.3 Limitations

**Client Buffer Size**  Our system uses the maximum operation set. One disadvantage of this technique is that it requires large buffers at the clients even though the server has a light load. When lightly loaded, the server that uses the minimum operation set can service the streams with smaller operations for the streams,
thus with smaller buffer size at the clients. Our counter-argument is that, with a minimum operation set, a stream subscribing with a small buffer may constrain the number of streams the server can admit. Allowing streams to subscribe with a client-specified buffer size plays counter to the objective of maximizing the number of streams.

Start-up Latency The second disadvantage of our system is the start-up latency, the time between a client receiving its data and its actual consumption of the data. When the maximum operation set is used, the cycle length is as large as possible. Obviously, because of double buffering, our system will have a larger start-up time as compared to one with a smaller cycle length. Our counter-argument is that, for the server that uses the minimum operation set, the cycle length increases when the stream set increases. As already pointed out, absorption of a stream (an incremental process in this case) may span over several cycles. Moreover, newly accepted streams are absorbed one at a time and have to wait for their turn to be absorbed. Therefore, while smaller start-up latency may be true when the stream set is small, it is not when the stream set is large. In our system, as many streams as the Admission Control accepts can be absorbed within the same cycle.

7.4 Relaxing the Assumptions

Linear-Seek-Time-Versus-Distance Model In our study, a linear-seek-time-versus-distance model is used. Our simple model, which uses the minimum seek
time and the maximum seek time given in the drive specifications, introduces a small error as can be seen in Figure 3.7. However, we should point out that while our study was developed for a linear seek-time-versus-distance model, it is not restricted to that particular one. As designers, we can arbitrarily choose the minimum and maximum seek times so that our linear model can be a worse-case bound on the measured seek-time-versus-distance curve.

Network Delays In our study, all network delays were ignored. Such an assumption may sound restrictive, but turns out to be irrelevant if the same physical path/circuit (with a constant aggregate network delay) is established in every cycle between the server and a particular client. This network delay, if constant, simply delays by the same amount the arrival of all media blocks to a client and therefore does not impair the data continuity at the client.
8 Related Work

8.1 Storage Techniques For Digital Continuous Multimedia

Rangan and Vin [7] address the problem of collocational storage of media strands, which are sequences of continuously recorded audio samples or video frames, on disks. They present a model that relates disk and device characteristics to the playback rates of media strands and derives storage patterns so as to guarantee continuous retrieval of media strands. To efficiently utilize the disk space, mechanisms for merging storage patterns of multiple strands are developed by filling the gaps between media blocks of one strand with the media blocks of the other strands.

The merging process is briefly described as follows. First, for one strand (corresponding to a stream), the block size and the separation (gap) between successive blocks are determined based on the playback rate of the strand and the retrieval rate of the disk. Continuous retrieval can be guaranteed if the time to skip over a gap and retrieve another block does not exceed its playback duration. Merging a new strand is the procedure of storing the blocks of the latter in the gaps of the first strand. Several strands can be merged in this way.
8.2 Disk Arrays

A disk array [4] [8], which groups multiple drives into what appear to be a single logical volume, can narrow the gap between processors and hard disk by working the component drives in parallel. It uses data striping. Because the whole disk array fails if one or more drives fail, a disk array is less reliable than single drives.

However, reliability of disk arrays can be increased using two techniques. In one technique, called Data Redundancy, the drives store some redundant data used to compute or predict the information data stored in any failed drive, very much like the error-correction bits used to recover information bits when a Hamming-coded data frame is corrupted by channel noise. The other technique is called Disk Mirroring, in which two drives store identical information. If one drive fails, its “mirror drive” serves as back-up. Both techniques involved huge data storage overhead.

8.3 Streaming RAID

Tobagi, Pang, Baird and Gang [9] focus on the management of an array of disks. The disk arrangement is quite similar to ours. Several disks are connected to a bus to increase data throughput of the drive arrangement and a microprocessor monitors the data retrieval. Data is striped among the disks.
The data fetched is basically stored in the main memory, whereas in our system, data is stored at the clients themselves. They discuss two techniques to decrease buffering requirements. The first technique is sharing the buffers among several streams. The streams do not really utilize all their required buffers throughout the whole cycle. When a stream has consumed the data from a buffer, it can release that buffer for use by another stream. The second technique is using subgrouping and subcycling. The streams are grouped in several groups. By properly time-offsetting the cycles for the groups, buffer occupancy can be reduced.

They also discuss the issue of the start-up time. The start-up time for a stream depends on the start-up data stored, the scheduling strategy used and the number of streams in the system. To decrease start-up time, they employed separate subcycles. One subcycle involves performing the requests for active streams, while the other for the newly arrived streams.

8.4 Caching Strategies

Mechanical latencies are major causes of inefficiency in disk systems. Karedla, Spencer and Wherry [10] examine caching strategies to improve data throughput of disk systems. Three cache replacement algorithms are investigated, namely, random replacement, least recently used, and least frequently used. They may be implemented at the drive itself, the hosts' buffer or the main memory.
of the system. Wherever they are implemented, they may reduce reads if some requests are for the same file and are close temporally.

Even though those techniques virtually improve throughput of disk systems, they are ignored in our architectures. We should remember that in our system, the cause of the bottleneck is the ATM switch. While the caching strategies may relieve the work load at the drives, they do not do so at the ATM switch. Moreover, given the huge volume of data for a file, and its high playback rate, rarely will clients request the same file with minute phase difference among them. In other words, the principle of locality is not as pronounced in media data as in program data.

8.5 Admission Control Algorithms

Vin and Rangan [11] consider two storage techniques. In contiguous allocation of media blocks, the media blocks are stored contiguously and thus successive blocks can be retrieved without incurring maximum seeks. However, contiguous allocation suffers from the problem of fragmentation which occurs during insertions and deletions. In unconstrained allocation of blocks on disk, blocks of data for a particular file are not stored contiguously but are somehow “scattered over” the drive. However, latencies are enormous as reading logically successive blocks involves several maximum seeks, the result being reduced throughput of the disk system.
They formulate Control Admission algorithms for both file storage techniques. However their algorithms include only characteristics of the drives and are therefore not close to being realistic.

### 8.6 Statistical Control Admission

Due to human perceptual tolerances and inherent redundancy in continuous media streams, most clients can tolerate a brief deterioration in playback continuity and occasional loss in media information. Worst-case assumptions that characterize deterministic control admissions constrain the server into admitting a small number of clients.

Vin, Goyal, Goyal and Goyal [12] propose a statistical admission control. A new client can be accepted as long as the statistical estimation of the aggregate data rate requirement, rather than the peak rate requirement, is satisfied. This control admission may lead to violations in data continuity of some streams. To be statistically fair to all streams, they propose to distribute the violations among all existing streams within a pre-defined violation limit. Such an admission control, they found, can lead to a 200% increase in number of streams the server can support.

Our server guarantees that it will retrieve data for the streams at their peak rates. No violations in data continuity can happen.
8.7 Scheduling Policies with Batching

ATM networks are equipped with a multicast facility. This feature can be used to reduce the number of streams required by the server to support a number of clients. Requests by multiple clients arriving within a short time interval can be batched together and serviced using a single stream [13]. For example, if two clients request the same movie within a short interval, the server can delay the playback of the first client and service both requests using a single stream as opposed to two.

Several policies are selected for multicasting namely:

- **FCFS (First Come First Serve) Policy** allows all requests to join a single queue. Requests for the same movie are satisfied by the same stream.

- **MQL (Maximum Queue Length) Policy** allows all requests for a specific movie to join a separate queue. The server starts a stream for the movie with the maximum number of requests.

- **FCFS-n Policy** allows requests for “hot” movies to be batched together in a single queue, and services those requests by starting a new stream after some batch interval.

This batching technique is very useful and can be used in our server to increase the number of clients it can support, assuming that our switch has a multicasting feature.
8.8 Scheduling Policies for Multimedia Services

Daigle and Strosnider [14] consider a real-time scheduling approach to guarantee timing requirements for multimedia applications. The scheduling model services the requests in a round-robin fashion, but the order of service is not fixed for all cycles. Each request has a deadline before which it should be serviced, a priority depending on its media type, and a period within which another request must be made.

They consider several scheduling policies, namely:

- In FCFS Policy, the request first on the queue is serviced first.
- In SCAN Policy, the requests are scanned and serviced according to their cylindrical location on the disks.
- In P-FCFS Policy, requests arrive at multiple queues, one for each media type or priority.
- P-SCAN Policy is similar to SCAN Policy, except that multiple queues, one for each media type, are scanned and their requests are serviced according to the priorities of the queues.

However, in those scheduling policies, a large request may block a small request approaching its deadline. Three techniques are discussed to reduce blocking. The first one is using priority queues. A higher priority request may then virtually preempt a lower priority request. The second technique is chunking
a large request into smaller requests to be dispatched sequentially. Chunking allows chunk-level preemption. The third technique is transforming the period of all requests to a common period. This technique is basically the QPMS scheme.

8.9 Lancaster Continuous Media Server (LCMS)

The LCMS server [15] and our server are similar in several ways, namely, round-robin servicing of streams, file striping, double buffering, availability of slack time. However, the two servers differ from each other in the following ways:

- For our server, the data granularity was one track length and the operation for a stream is a multiple of it, depending on the playback rate of the stream (QPMS). For LCMS server, the data granularity depends on the playback rate of the stream, that is, the data block unit was proportional to the playback rate of the stream. However, in both servers the basic idea is the same, which is to retrieve more data for a higher-playback-rate file.

- Our server has a CSCAN disk scheduling policy. The LCMS server has a round-robin disk scheduling policy, servicing streams whose requests are closest to their deadlines. Wear is smaller in the former than in the latter.

- Our server chooses an optimum cycle length that guarantees that the maximum number of streams serviced corresponds to the maximum throughput at the
server. For the LCMS server, the choice of the cycle length is not discussed and the cycle length may not be optimum.

- Our server involves more architectural components than the LCMS server and is therefore more realistic. Moreover in spite of more architectural components, we developed a simple, effective and efficient Admission Control.

- Both servers noticed the problem of data accumulation at clients. In our server, this problem is solved by rounding down the operations and using a dynamic operation set. In the LCMS server, this problem is solved by skipping read requests for streams whose data accumulation has reached a certain limit.

### 8.10 Multimedia Storage Server (MSS)

This server [16] is similar to our server in several techniques, namely, file striping, round-robin servicing of streams, SCAN disk scheduling policy, and double buffering. However, the two servers differ from each other in the following ways:

- MSS has several secondary storage servers. Each secondary storage server stores a specific media type. For example, one of them stores MPEG files, while another one may store reduced quality video. Our server stores data of any media type.
• For MSS, the Control Admission is designed for a specific media type or for a specific secondary storage server. The Control Admission for our server is flexible in that it predicts admission of a stream of any media type.

• The Control Admission for MSS was designed using disk parameters only and is therefore not realistic. The Control Admission we designed was for a fairly realistic system model.

• The MSS server has secondary (archival) storage system, while ours does not.

8.11 Gopher-style Real-time ATM Multimedia Services (GRAMS)

GRAMS [17] and our server are similar in servicing streams in a round-robin fashion, but the order of service is not fixed for all cycles. Both servers can be configured for a distributed system. Both servers are designed for an ATM network. However, they are different in several ways, namely:

• GRAMS considers only 3 media types, real-time media (video, audio), image and data. Our server deals with any media types.

• GRAMS uses real-time scheduling approach to guarantee timing requirements of real-time media. Our server uses CSCAN disk scheduling policy. Therefore wear is greater in the former.

• GRAMS scheduler prioritizes the media types. The server schedules real-time requests (video, audio) first and services them according to their deadlines.
The requests for other media types are serviced with lower priorities because they do not have time constraints.

- GRAMS admits streams as long it can satisfy deadlines to some pre-defined deterioration limit. Our server has a Control Admission designed to guarantee data rate requirement of a stream once it is admitted.

- GRAMS implements Cache strategies to remove disk bottleneck, while our server does not.
9 Conclusions

9.1 Contributions

We have presented two multimedia server architectures and a data retrieval model and have analyzed their performance in terms of the number of concurrent video streams that they can support using Circular SCAN disk scheduling policies. One of the architectures uses file striping while the other does not. For the analysis we have also developed a fast algorithm for finding the operation set.

We have proposed finding the maximum operation set as a means of providing elasticity in the absorption of newly accepted streams. In this strategy, absorption of a stream takes place within a single cycle by using the server slack time. Moreover, several streams, whatever their playback rates, can be absorbed within a single cycle.

We have introduced the Rate-based Buffer Weighting Scheme in which a stream subscribes with a buffer size proportional to its playback rate. When this scheme is applied along with QPMS, the operations do not change with new subscriptions or terminations. This scheme thus relieves the server from transitions that are necessary otherwise.

We have also presented a new way of designing the Admission Control for both architectures. The proposed Admission Control schemes are simple yet
effective and efficient. They are simple because predicting admission of a stream is a matter of comparing quantities. They are effective because they can predict admission of a new stream correctly and accurately. They are efficient because the maximum number of streams admitted by a node corresponds to the saturation of the ATM fibre link, and hence to the full utilization of the ATM switch.

Finally, we have compared both architectures. Architecture II is more reliable and requires smaller buffer size at the clients. On the other hand, Architecture I does not present any contention problem. (While performance of Architecture II can be increased by having copies of "hot" files in different drives, duplication of files and copying of files from one disk to another result in waste of resources, namely memory space, bandwidth, and cpu time.)

9.2 Choice of Architecture

Choosing one architecture over another may depend not only on the merits and demerits of the individual architectures but also on the external factors. One external factor is the length of the files (in time). The shorter the file length, the less contention there exists, given a particular inter-subscription pattern of clients. Therefore, where the server is intended to service short clips, Architecture II might be the appropriate choice. On the other hand, if file length is long (for example a movie), contention for a hot file is highly probable, and duplication
of large files is obviously unwanted. For that type of application, Architecture I might be the choice.

9.3 Future Work

We have focused on continuous servicing of streams of different playback rates. However, a multimedia object usually consists of more than one stream (audio and video as an example). While maintaining continuity of individual streams was one challenge, maintaining inter-media synchronization is another and is the subject of future work.
Bibliography


