

# A Dynamic Scheduling Mechanism for an Effective Admission Control for Variable-Bit-Rate Video Streams \*

KyungOh Lee, Heon Y. Yeom  
Dept. of Computer Science, Seoul National University  
Seoul, 151-742, Korea  
{leeko,yeom}@arirang.snu.ac.kr

## Abstract

*For admission control in real time multimedia systems, buffer space, disk bandwidth and network bandwidth must be considered. Most admission control mechanisms developed to date have been based on the CBR data model and have used a static period length. These mechanisms do not use system resources effectively, since media data is usually encoded with VBR compression techniques. We propose an admission control mechanism based on a VBR data model, that has a dynamic period and considers both disk bandwidth and buffer space. Simulations show that our scheme can accept approximately twice as many streams as previous schemes based on CBR techniques and static time periods.*

## 1. Introduction

Multimedia systems like VOD (Video-On-Demand) systems require considerable resources and have tight real-time constraints. Multimedia data to be sent to clients must be read from disk to memory buffers before the actual transmission, and thus the maximum number of clients that the system can support depends upon both disk bandwidth and total buffer size.

In many admission control approaches, resources are reserved based on the worst-case assumption of a peak-data-rate CBR (Constant Bit Rate) data model [4, 2]. However, since video data objects are usually generated with VBR (Variable Bit Rate) compression techniques, CBR-based approaches tend to waste resources. There are some existing admission control mechanisms which use a VBR data model to reserve buffer space. However, they do not consider disk bandwidth or assume CBR data retrieval from the disk [9]. There is a close relationship between disk band-

width and buffer space, and an admission control mechanism should consider both simultaneously. Although some existing mechanisms do consider both buffer space and disk bandwidth, they are based on the CBR data model [2].

In this paper, we propose a new admission control scheme, which exploits the characteristics of VBR data and which uses a dynamic time period in scheduling. To compare the trade-offs among several scheduling algorithms, we derived the equations for the mean distance of head movement in one seek for each scheduling algorithm and analyzed the buffer requirements.

The rest of the paper is organized as follows: Related works are presented in section 2 and the disk latency and the buffer requirements of several disk scheduling schemes are discussed in section 3. Our admission control algorithm using dynamic period is proposed in section 4 and our simulation results are presented in section 5. Finally, our conclusions are given in section 6.

## 2. Related Work

Scheduling algorithms play an important role in VOD systems, where concurrent streams need to be read effectively. Several combinations of conventional disk scheduling algorithms and real-time scheduling techniques have been investigated in the recent past. The simplest of all such techniques is the round-robin (RR) scheduling algorithm, in which the order in which clients are serviced does not vary from one period to another. However, the major drawback of RR scheduling is that it does not exploit the relative positions of the media blocks being retrieved during a period.

To address the limitations of the RR scheduling algorithm, the SCAN disk algorithm has been adapted. Notice that in the case of the RR algorithm, since the order in which clients are serviced is fixed across periods, the maximum separation between the retrieval times of successive requests of a client is limited by the duration of a period. However, in the case of SCAN, the relative order for ser-

---

\*This work was supported in part by Korea Science and Engineering Foundation Grant(95-0100-23-04-3)

vice clients is based solely on the placement of the blocks being retrieved, so it is possible for a client to receive service at the beginning of one period and at the end of the next period.

The best known algorithm for real-time scheduling of tasks with deadlines is the Earliest Deadline First (EDF) algorithm. In EDF scheduling, after one media block has been accessed from the disk, the media block with the earliest deadline is then scheduled for retrieval next. Scheduling of the disk head based solely on the EDF policy, however, may yield excessive seek times and rotational latency, and may thus lead to poor utilization of server resources.

One variant of these basic algorithms combines SCAN with EDF, and is referred to as the SCAN-EDF scheduling algorithm [7]. In SCAN-EDF scheduling, the requests with the earliest deadlines are served first, but if several requests have the same deadline, then their respective blocks are accessed using the SCAN algorithm. Clearly, the effectiveness of the SCAN-EDF technique is dependent on how many requests have the same deadline. If all of the media blocks to be retrieved during a period are assigned the same deadline, then SCAN-EDF is effectively reduced to SCAN. Although SCAN scheduling schemes have shorter seek times (or latency times), they need much more buffer space.

GSS (Grouped Sweeping Scheme) scheduling divides the streams into several groups, which are each serviced RR style, while the streams in each group are serviced SCAN style [10]. If the number of groups is large, GSS acts like RR, while if the number is small, it acts like SCAN.

As observed in [3], the most natural way to process multiple streams simultaneously is to interleave the readings of the streams in a cyclic fashion. As illustrated in figure 1, each period (cycle or round) consists of a set of working slots and latency slots.

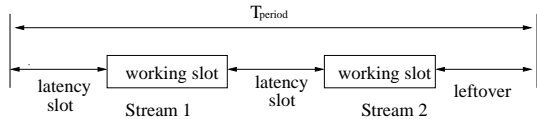


Figure 1. Periodic Stream Service

The amount of data that must be read from the disk for each stream within a working slot should be sufficient for the client for one period (cycle). After one stream has read sufficient data, a new stream starts to read the data, and there must be at least one seek to switch to another stream. The latency slot includes these seek latencies and other scheduling overhead latencies. As the service period ( $T_{period}$ ) becomes long, each stream must read more data from the disk into the buffer. However, due to the limited size of the buffer,  $T_{period}$  can not be longer than some maximum value,

$T_{max}$ . As  $T_{period}$  becomes short, every working slot also becomes short and each stream must then read all necessary data rapidly in the short working slot. This requires high disk bandwidth. Due to the upper limit of disk bandwidth,  $T_{period}$  can not be shorter than some minimum value. If the lower limit  $T_{min}$  is chosen, then the probability of starvation is less than some value. To support all active streams with the current available buffer space and disk bandwidth,  $T_{period}$  must satisfy the following inequality:

$$T_{min} < T_{period} < T_{max} \quad (1)$$

In [2], it was assumed that data-rates of streams  $R_c$ , are constant (CBR data model, based on a pessimistic approach) making it very easy to calculate the  $T_{min}$  and  $T_{max}$ . But, since a pessimistic approach was used, system resource utilization was quite low.

### 3. System Modeling

Buffer space and disk bandwidth are the most important factors that determine the number of streams that a VOD server can accept. However, full disk bandwidth can not be used, due to latency. Seek latency is the largest part of the latency time, and the number of cylinders that the disk head should move in one seek is the major part of seek latency. We therefore model the seek distance first. In our system model, we assume the network has an infinite bandwidth.

#### 3.1. The number of cylinders to be crossed in one seek

According to [1, 6] seek time is maximized, under a realistic function for the seek time, for equidistant seek positions of the  $n$  requests. The seek time function itself is assumed to be proportional to the square root of the seek distance for small distances below a disk specific constant, and a linear function of the seek distance for longer distances, which is in accordance with the studies of [8]. Thus, for given disk parameters, the maximum total seek time of a sweep can be easily computed by assuming the  $n$  seeks positions to be at cylinders  $(i \cdot c)/(n + 1)$  for  $i = 1, \dots, n$  where  $c$  is the total number of disk's cylinders, and applying seek time function. This computation yields an upper bound for the seek time. That is, if we use the following formula as the number of cylinders to be crossed in one seek, then there will be almost no starvation caused by the seek latency.

$$\text{The Number of Cylinder in one seek} = \frac{c}{n + 1} \quad (2)$$

To be adapted for GSS scheduling, equation (2) should be extended. We had better divide the streams into groups of equal size(except for the last group) to reduce the seek

time skewness among the groups. If we assume that streams are divided into  $g$  groups, then the number of members in a group will be  $\text{round}(\frac{n}{g})$  and the number of members in the last group will be

$$n - (g - 1)\text{round}(\frac{n}{g}) .$$

In a GSS scheme, SCAN scheduling is used for all streams in a group, so by (2), we get the following formula.

$$\frac{c}{\text{round}(\frac{n}{g}) + 1} \quad (3)$$

Since  $g$  is the same as  $n$  in RR scheduling, the number of cylinder in one seek will be  $\frac{c}{2}$ . Kiessling et al [5] showed that the expectation(or average) of the number of cylinders to be crossed in one seek is  $\frac{c}{3}$  in RR scheduling scheme. If we use  $\frac{c}{3}$  instead of  $\frac{c}{2}$  in modeling the seek distance, then we may utilize disk bandwidth more effectively. However, since the probability of starvation will increase much higher, we used  $\frac{c}{2}$  instead of  $\frac{c}{3}$ .

### 3.2. Comparison of the buffer usage factor for various scheduling schemes

In an RR scheme, the order in which streams are serviced does not change from one period to another. Thus, we need to load data sufficient for only one period(say  $T$ ). But in a SCAN scheme, the order of service depends on the physical positions of the logical blocks. In the worst case, the stream serviced first in the current period can be serviced last in the next period. Since we do not know the order in advance, the total amount of data able to be read into the buffer should be sufficient for 2 periods (say  $2T$ ) to guarantee problem-free playback. This means that 50% of the buffer may be wasted in the SCAN scheme, as compared to the RR scheme. In the GSS scheme, every stream in one group is serviced by SCAN scheduling, but each group is serviced by the RR scheme. If the number of groups is  $g(g < n)$ , then the total amount of data to be loaded into buffer should be sufficient for  $T + \frac{T}{g}$ . We can therefore say that we waste  $\frac{T}{T + \frac{T}{g}}$  buffer space. If  $g = 1$ , then the wastage is the same as for SCAN, and if  $g = n$ , it is the same as for RR.

## 4. Proposed Admission Control Algorithm

Most earlier works used the fixed time period and CBR data modeling. As the playback data-rate of an active stream does not vary during playback in the CBR scheme, we do not need to change the length of the period. These approaches are simple but inefficient when used with VBR data. When servicing VBR data, the optimal length of the period would vary frequently according to the playback, depending on which streams are playing or which parts are

playing. Since the static period length causes the system to accept less streams than is possible, we propose to use dynamic period length to maximize efficiency.

We should find some schedulable set of periods to accept a new stream. Assuming that current time period is  $T_i$ , if  $T_{min} < T_{max}$ (schedulable), we can take some value between  $T_{min}$  and  $T_{max}$  as the next time period,  $T_{i+1}$ , and add this period to the period set  $ST$  ( $ST$  is first initialized with empty set). There are many choices in taking  $T_{i+1}$ , but we use  $\frac{T_{min} + T_{max}}{2}$  as  $T_{i+1}$  intuitively and start the step to get  $T_{i+2}$  with  $T_{i+1}$ . If  $T_{min} > T_{max}$ (unschedulable), the new stream can not be accepted. We can accept the new stream if there is no unschedulable period until the sum of  $T_{periods}$  in  $ST$  becomes larger than the length of the playback time of the stream. Assuming that we decide to accept a stream after the  $k$ -th step,  $ST$  will be as follows.

$$ST = \{T_{i+1}, T_{i+2}, \dots, T_{i+k}\}$$

As stated in [2], the average playback rate of each stream is required to calculate the  $T_{min}$  and  $T_{max}$ . Since they used CBR data model, the playback rate is constant all the time. However, it varies dynamically from period to period. Let  $Rc_i(j)$  be the average playback data-rate of stream  $j$  at some period  $T_i$  and  $t$  be the start time of the next period. Then  $Rc_i(j)$ , the data-rate of stream  $j$  at  $T_i$ , is the sum of all frames belonging to  $T_i$  divided by  $T_i$  and the data-rate of the next period is the sum of all frames in  $[t, t + T_{i+1}]$  divided by  $T_{i+1}$ . Since  $T_{i+1}$  is not known for the current period, we can not get an accurate data-rate for  $T_{i+1}$ , so we approximate  $T_{i+1}$  using  $T_i$ . We add all the frames in  $[t, t + T_i]$  and divide that by  $T_i$  to get  $\widehat{Rc_{i+1}}(j)$ (approximation of  $Rc_{i+1}(j)$ ). Since we have  $\widehat{Rc_{i+1}}(j)$  now, we can get  $T_{min}$  and  $T_{max}$  and finally we can get  $T_{i+1}$ . However, the error from approximating  $T_{i+1}$ , can result in either starvation or buffer overflow.

To cope with these problems, we reserve a small fraction of resource to overcome starvation and overflow problems. From our simulation, 1% of disk bandwidth and 5% of buffer space was sufficient to guarantee that neither starvation nor overflow occurs. Even if we do not leave any spare resource (100% of resources are allocated), the probability of starvation or overflow is less than 0.05 as shown in section 5.

Our admission control algorithm based on VBR data modeling and dynamic period length is given below.

```
boolean AcceptanceTest(New Stream) {
/* Initialize variables */
ST = φ;
T = Length of Current Period;
AcceptNewStream=TRUE;
t = Starting Time of Next Period;
/* End Time of Current Period */
```

```

while( $\sum T_{in ST} < Playback\ Time\ of\ New\ Stream$ ) {
  Add all frame sizes of Active Streams
  and New Stream in  $[t, t + T]$ ;
  Get  $\sum Rc(j)$ ;
  /* Sum of  $Rc(j)$ s in Active Streams and New Stream */
  Get  $T_{min}$  and  $T_{max}$  using  $\sum Rc(j)$ 
  if ( $T_{min} < T_{max}$ ) {
     $T = \frac{T_{min} + T_{max}}{2}$ ;
    Put  $T$  into  $ST$ ;
     $t = T + t$ ;
  }
  else {
    unschedulable();
    AcceptNewStream = FALSE;
    break;
  }
}
return AcceptNewStream;
}

```

If AcceptanceTest() returns FALSE, the new stream can not be accepted, and we may retry at either the next time period or a few periods later. If AcceptanceTest() returns TRUE, the new stream can be serviced from the next period using the new period set obtained during the test. Time complexity of the algorithm is  $O(m \cdot n)$ , where  $m$  is the number of active streams and  $n$  is the number of frames in the stream being tested.

## 5. Experimental Evaluation

### 5.1. Assumptions

Most existing storage-server architectures employ random allocation of blocks on a disk. Since the latency between the blocks of a media object is unpredictable, this type of organization is insufficient to meet the real-time constraints of a multimedia application. Contiguous disk block allocation yields the highest effective disk bandwidth, but has the penalty of costly reorganization during data insertion and updates. However, since video data objects are seldom modified, contiguous allocation is not an unreasonable assumption in a VOD system (If contiguous allocation is impossible, then the use of blocks of the maximum size possible is best). Even if we store the data on contiguous blocks, latency is inevitable, because a VOD server must support several streams concurrently. At least one seek is necessary to switch to another stream.

In this paper, we use Round Robin, SCAN, and GSS as scheduling schemes. In modeling the disk characteristics, we use the formulas in section 2 and parameters are retrieved from a Seagate Wren 8 ST41650N disk. Head

switching time (time to change head to the another surface), seek start-up time, time to cross tracks and rotational latency are considered for the simulation. Starwars MPEG I trace data (frame size data) and Red's Nightmare MPEG I trace data are used to produce the workload. To generate different video streams, we choose one type of trace data and generate two random numbers between 0 and 1. The smaller generated number is used to indicate the starting point of the stream and the larger is used to indicate the end point. These video streams need 1.5 Mbps display data-rate (peak-rate), but actually the mean data-rate of Starwars is 670 Kbps and that of Red's Nightmare is 600 Kbps. More than 50% of resources are wasted if we use CBR data modeling (worst-case assumption). Other trace data with different characteristics can be used for similar simulations. We used an Ultra Sparc 1 workstation running Solaris 2.5 for the simulation.

### 5.2. Experiment Method

In our experiment, we have compared the performance between the VBR scheme and the CBR scheme. The VBR scheme employs the dynamic length of period and the CBR scheme employs the static length of period. We used RR (Round Robin), SCAN, and GSS (Grouped Sweeping Scheme) as the scheduling methods for the VBR scheme. RR and SCAN methods were used for the CBR scheme. Since the difference between them was negligible, only the results of the SCAN-CBR scheme are presented for simplicity. For each scheme, the average number of streams accepted was used as the performance metric. The following abstract mechanism was used for 1 million seconds to get the average number of accepted streams for each scheme. Assume that there is at least one stream waiting for service in the FIFO (first in first out) queue and that the system accepts as many streams as possible and services them. If no more streams can be accepted, the time is passed to the next period until a new stream is accepted. In some periods, certain streams may be over, and some streams may be accepted. The mean number of accepted streams in a period is computed through executing each scheme for a million seconds.

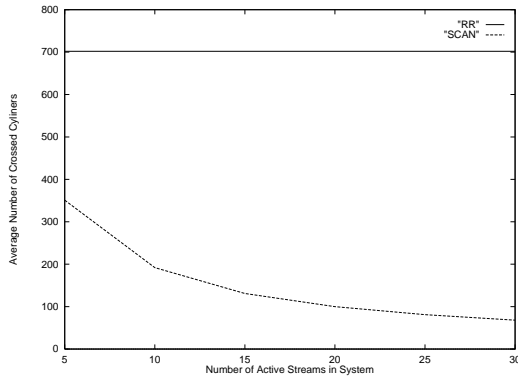
The admission control algorithm in section 4 is changed to evaluate the effect of static time period on the performance. For a given time period  $T_s$  (unchanged during a run), we can get  $T_{min}$  and  $T_{max}$ . If any one of the 3 conditions holds, the stream can not be accepted.

1.  $T_s < T_{min}$
2.  $T_{max} < T_s$
3.  $T_{max} < T_{min}$

Since a stream can be accepted in the dynamic period length scheme, unless it meets condition 3 above, the dynamic scheme shows better performance.

### 5.3. Results

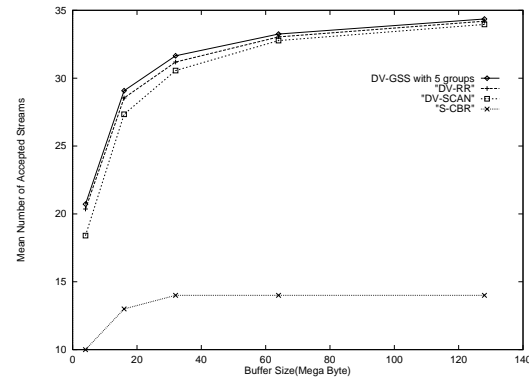
Figure 2 shows the effect of the number of active streams on the average seek distance. There is no relationship between the average seek distance and the number of active streams in the system when using the RR scheme, but there is a strong relationship when using the SCAN scheme. As the number of streams increases, the average seek distance decreases.



**Figure 2. The Affect of the Number of Active Streams on the Average Seek Distance**

Figure 3 shows the mean number of streams accepted with varying buffer size, and S-CBR means the CBR scheme with static period length, DV-RR means dynamic period length VBR scheme with RR disk scheduling, and DV-GSS means dynamic period length VBR scheme with GSS disk scheduling. The main conclusions that may be drawn from the figure are that dynamic period length VBR schemes accept almost twice as many streams as the CBR scheme, and that there is little difference among the dynamic period length VBR schemes. DV-GSS, with 5 groups, shows the best throughput among the dynamic VBR schemes. However, the improvement is very small (at best, less than 3%). While the playback data-rate ( $R_c$ ) of a stream is given at admission control time in the CBR scheme, it varies from period to period in the VBR schemes. As expected from section 2, the SCAN scheme shows some gain in disk bandwidth due to the reduced seek distance, but buffer utilization is poor compared to that of the RR scheme (50% worse). The RR scheme uses the buffer more efficiently, but much more disk bandwidth is wasted.

The major factor that affects the lower limit of period length ( $T_{min}$ ) is effective disk bandwidth (as disk bandwidth increases, the lower limit decreases), and the ma-

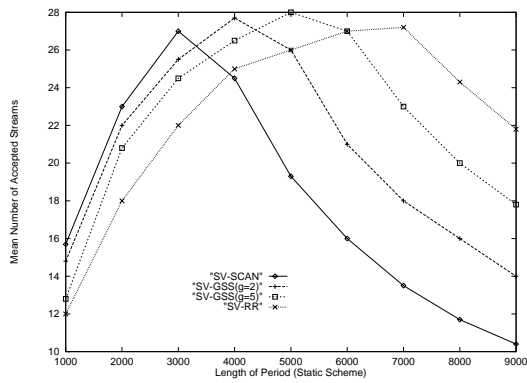


**Figure 3. Average Number of Accepted Streams for Various Strategies**

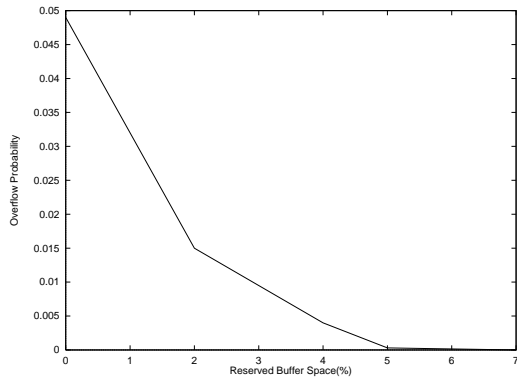
For factor which affects the upper limit of period length ( $T_{max}$ ) is the effective buffer space (as buffer space increases, the upper limit increases). Period length is also influenced strongly by the sum of the data-rates of all active streams. Even when we use the VBR characteristics for resource reservation, we cannot achieve good performance if we also use a static time period (no changes in the length of period). If we use a static time period, then the throughput varies with the length of the period (worse than the dynamic schemes in all cases). For example, with a fixed period of 2000 ms, SV-RR (static period VBR scheme with RR scheduling) accepts 18 streams, and SV-SCAN (static period VBR scheme with SCAN scheduling) accepts 23 streams (using 32M buffer space), while with a fixed period of 5000 ms, SV-RR accepts 26 streams, and SV-SCAN accepts 19 streams. It is obvious that if we use some value close to the optimal period for the fixed period, then the throughput is reasonable, but otherwise it is very poor. In our algorithm, a new stream is rejected if at least one non-schedulable period exists ( $T_{min} > T_{max}$ ). Even if we decide that the time period is close to the optimal length, the possibility of rejection is higher than with dynamic period schemes. Figure 4 shows the mean number of maximum accepted streams using a static time period with a VBR scheme (the buffer size is 32 M).

Since we use approximated values to obtain the length of the next period ( $T_{i+1}$ ), starvation or overflow can occur. In our simulation, the starvation probability is less than 0.5% (one starvation in 200 periods) and the overflow probability is less than 5%. When we reserve 5% of buffer space, we experience no overflow. While 0.5% of starvation is acceptable, if we reserve 1% of disk bandwidth, then we experience no starvation. Figure 5 shows the overflow probability with varying space resource reservation.

Even if we reserve 5% of resource, there is only a 3% degradation in throughput. This means that dynamic VBR



**Figure 4. Mean of Maximum Accepted Streams in Case of Static Period Length**



**Figure 5. Overflow Probability with Varying Space Resource Reservation**

schemes still accept twice the number of streams accepted by static CBR schemes.

## 6. Conclusions

In this paper, we have presented an effective admission control scheme for VBR data, which considers both buffer space and disk bandwidth. To compare the trade-offs among several scheduling algorithms, we have derived the equations regarding the mean distance of head movement in one seek for each scheduling algorithm and we have analyzed the buffer requirements for those algorithms. Since most existing control mechanisms use a static period scheme, the resources of the server tend to be wasted. If a static period length is far from optimal, then the throughput decreases significantly. In a dynamic time-period scheme, the period is changed dynamically, based upon the available disk bandwidth and the buffer space, so as to maximize the throughput. Our experiments show that there is little difference in

performance among VBR schemes if we use dynamic periods. When some spare resources are reserved (5% of buffer and disk bandwidth) for starvation or overflow, then VBR schemes can still support twice as many streams as a CBR scheme and experience neither starvation nor overflow. Our VBR-based scheme needs more calculation overhead than a CBR scheme, but if frame-size data is merged by some unit (say, the summation of 30 frames), then the overhead will be significantly reduced (to almost 1/30). The largest part of the calculation is the summation of frame sizes. Since CPU performance is increasing faster than that of memory or disks, optimizing memory utilization and disk bandwidth is becoming increasingly important.

We use a single disk model but disk arrays are currently used in many systems. If we use a simple disk array configuration, in which data are perfectly striped on all the disks, the entire collection of disks can be treated, logically, as a single disk unit and we can replace disk bandwidth  $R$  with  $\sum_{i \text{ in disk array}} R_i$ . We will extend our scheme in order that more complicated disk arrays can be used.

## References

- [1] E. Chang and H. Garcia-Molina. Effective memory use in a media server. In *Proceedings of International Conference on 23rd Very Large Data Bases (VLDB'97)*, pages 496–505, Athens, Greece, Aug. 1997.
- [2] H.-J. Chen and T. Little. Storage allocation policies for time-dependent multimedia data. *IEEE Trans. on Knowledge and Data Engineering*, 8(5):855–864, Oct. 1996.
- [3] J. Gemmell and S. Christodoulakis. Principles of delay-sensitive multimedia data storage and retrieval. *ACM Trans. of Information Systems*, 10(1):51–90, Jan. 1992.
- [4] J. Gemmell, H. M. Vin, D. D. Kandlur, and V. Rangan. Multimedia storage servers: A tutorial and survey. *IEEE Computer*, 28(5):40–49, May 1995.
- [5] W. Kiessling. Access path selection in databases with intelligent disc subsystems. *The Computer Journal*, 31(1):41–50, Feb. 1988.
- [6] Y.-J. Oyang. A tight upper bound of the lumped disk seek time for the SCAN disk scheduling policy. *Information Processing Letters*, 54:355–358, 1995.
- [7] A. Reddy and J. C. Wyllie. I/O issues in a multimedia system. *Computer*, 27(3):69–74, Mar. 1994.
- [8] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27(3):17–28, Mar. 1994.
- [9] Y. S. Ryu and K. Koh. A dynamic buffer management technique for minimizing the necessary buffer space in a continuous media server. In *Proc. of IEEE International Conference on Multimedia Computing and Systems*, pages 181–185, Hiroshima, Japan, June 1996.
- [10] P. Yu, M. Chen, and D. Kandlur. Design and analysis of a grouped sweeping scheme for multimedia data. In *Proc. of 3rd International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 38–49, San Diego, Nov. 1992.