An Observation-Based Admission Control Algorithm for Multimedia Servers

Harrick M. Vin, Alok Goyal, Anshuman Goyal, and Pawan Goyal

Department of Computer Sciences, University of Texas at Austin
Taylor Hall 2.124, Austin, Texas 78712-1188

E-mail: {vin,alok,anshu,pawang}@cs.utexas.edu, Telephone: (512) 471-9732, Fax: (512) 471-8885

Abstract

In this paper, we propose a novel observation-based admission control algorithm, in which a client is admitted for service by a multimedia server only if the predicted extrapolation from the status quo measurements of the storage server utilization indicate that the service requirements of all the clients can be met satisfactorily. The performance of the admission control algorithm, and hence, the number of clients admitted and serviced simultaneously are maximized by employing a disk scheduling algorithm that minimizes both seek and rotational latency incurred while accessing a sequence of media blocks from disk. The effectiveness of the observation-based admission control algorithm is demonstrated through extensive simulations.

1 Introduction

1.1 Motivation

Since the timid debut of the first usable computer almost half a century ago, the world has witnessed dramatic improvements in computer and communications technologies. Whereas breakthroughs in computer technology have stimulated the integration of digital continuous media (such as audio and video) with computing, rapid advances in communication technology have made available high-bandwidth, fiber-optic networks at modest cost. The synergy between the advances in computer and communications technologies promises to create an infrastructure in which computer systems will support a wide range of multimedia services over high-speed metropolitan area networks.

In its simplest configuration, the architecture of such multimedia services will comprise of multimedia information servers (which we will henceforth refer to as *Multimedia Servers*) connected to media capture and display sites, such as video cameras, HDTVs, telephones, and videophones, belonging to clients via high speed networks (e.g., ATM). Multimedia servers will digitally store media information such as proceedings of tele-conferences, educational documentaries, entertainment movies, advertisements, etc., on a large array of extremely high-capacity storage devices (e.g., optical or magnetic disks). On receiving a playback request from a client, a multimedia server will service the client by retrieving the chosen media segments from disk, and then transmitting them to the client's site. The retrieval can be interactive, in the sense that clients can stop, pause, resume, and edit the media information if they have permissions to do so. Clients may even be permitted to store multimedia objects at the server.

Since a media stream consists of a sequence of media quanta, such as video frames or audio samples, which convey meaning only when presented continuously in time, a multimedia server must ensure that recording and retrieval of media stream to and from disks proceed at their real-time rate. Whereas designing a dedicated, single-client multimedia server does not offer very many design choices and is relatively straightforward, the design of a multimedia sever that is capable of servicing multiple clients simultaneously poses interesting research challenges. This is because, given the maximum rate of data transfer from disks, a multimedia server can only service a limited number of clients. Hence, before admitting a new client, a multimedia server must employ admission control algorithms to decide whether a new client can be admitted without violating the continuity requirements of any of the clients already being serviced. Development of such algorithms is the subject matter of this paper.

1.2 Relation to previous work

Most of the existing work on admission control algorithms for multimedia servers has been focussed on developing techniques for providing deterministic service guarantees to clients (i.e., playback requirements are strictly met for the entire service duration) [1, 5, 9, 11, 13, 15]. However, since human perception is tolerant to brief distortions in audio and video, providing deterministic guarantees to each client is superfluous. Furthermore, the worst-case assumptions that characterize most of these techniques may needlessly constrain the number of clients that are serviced simultaneously, and hence, may lead to severe under-utilization of server resources. This is because, the average time spent in accessing a media block from disk, in practice, is significantly smaller than the corresponding worst case times. Hence, in order to improve the utilization of server resources, a multimedia server must employ an admission control algorithm which exploits the statistical variation in the access times of media blocks from disk.

1.3 Our research contributions

In this paper, we propose an *observation-based admission control algorithm*, in which a client is admitted for service only if the predicted extrapolation from the status quo measurements of the storage server utilization indicate that the service requirements of all the clients can be met satisfactorily. That is, the admission control criteria is defined using the measured characteristics of the current load on the server, rather than theoretical worst-case bounds. The main goal of our admission control algorithm is to accept enough traffic to efficiently utilize the server resources, while not accepting clients whose admission may lead to the violations of the service requirements of clients.

The performance of our admission control algorithm is critically dependent on the disk scheduling algorithm. Most of the conventional disk scheduling algorithms (such as, SCAN, Shortest Seek Time First (SSTF), etc.) have addressed the problem of optimizing the total seek time incurred while accessing a sequence of blocks from disk. The fundamental limitation of these algorithms, however, is that they optimize only the seek time, and completely ignore the rotational latency. In this paper, we present a disk scheduling algorithm that derives a sequence for accessing media blocks from disk so as to simultaneously minimize both seek and rotational latency incurred during retrieval.

Finally, since the observation-based admission control algorithm provides fairly reliable service but no absolute guarantees, simultaneous servicing of multiple clients may lead to occasional violation of the continuity requirements (i.e., media unit losses) of some of the clients. In order to enable a multimedia server to meet the requirements of as many clients as possible, we propose a technique for minimizing as well as distributing the media unit losses among multiple clients.

In order to demonstrate the effectiveness of the observation-based admission control algorithm, we have carried out extensive simulations. We present and analyze our simulation results.

The rest of the paper is organized as follows: Techniques for servicing multiple clients simultaneously are presented in Section 2. In Section 3, we present the observation-based admission control algorithm. The disk scheduling algorithm and the technique for minimizing and distributing media unit losses are outlined in Sections 4 and 5, respectively. Our simulation results are described in Section 6, and finally, Section 7 summarizes our results.

2 Servicing multiple clients

Digitization of audio yields a sequence of samples, and that of video yields a sequence of frames. We refer to a continuously recorded sequence of audio samples or video frames as a *strand*. A multimedia server must organize the storage of such media strands on disk (in terms of *media blocks*) so as to ensure their continuous retrieval at real-time rates [9]. Due to the periodic nature of media playback, a multimedia server can service multiple clients simultaneously by proceeding in *rounds*. During each round, the multimedia server retrieves a sequence of media blocks for each strand. The number of blocks of each media strand retrieved during a round is dependent on its playback rate requirement, as well as the available buffer space at the client [13]. Consequently, ensuring continuous retrieval of each strand requires that the *service time* (i.e., the total time spent in retrieving media blocks during a round) does not exceed the minimum of the playback durations of the sequences of blocks for each strand retrieved during the round. Since service time is a function of the number of blocks retrieved during a round, a server can service only a limited number of clients simultaneously. Hence, before admitting a new client, a multimedia server must employ admission control algorithms to decide whether a new client can be admitted without violating the continuity requirements of any of the clients already being serviced. A precise formulation of this requirement is presented in the next section.

2.1 Formulating the admission control problem

Consider a multimedia server that is servicing n clients, each retrieving a different media strand (say, S_1, S_2, ..., S_n, respectively). Let $\mathcal{R}_{pl}^1, \mathcal{R}_{pl}^2, ..., \mathcal{R}_{pl}^n$, denote the playback rates (in terms of bytes/sec) of strands S_1, S_2, ..., S_n, respectively. Furthermore, let $k_1, k_2, ..., k_n$ denote the number of blocks of strands $S_1, S_2, ..., S_n$ retrieved during each round.

Assume that the multimedia server is employing the SCAN disk scheduling policy, in which the disk head scans back and forth across the platter servicing requests as it passes the target cylinder. Consequently, the total amount

Symbol	*Explanation*	*Units*
T	Number of tracks on disk	-
M	Disk block size	bytes
a, b	Constants (seek time parameters)	sec
$l_{seek}(t_1, t_2)$	Seek time ($a + b * \|t_1 - t_2\|$)	sec
l_{seek}^{max}	Maximum seek time	sec
l_{rot}	Rotational latency	sec
l_{rot}^{max}	Maximum rotational latency	sec

Table 1 : Summary of disk parameters used in this paper

of time spent in servicing n requests during a round is dependent on the total seek and rotational latencies incurred while accessing $(k_1 + k_2 + ... + k_n)$ media blocks from disk. Since all the media blocks to be retrieved during a round, in the worst case, may be stored on separate tracks, the disk head may have to be repositioned onto a new track at most $(k_1 + k_2 + \cdots + k_n)$ times during each round. Hence, using the symbols for disk parameters presented in Table 1, the total seek time incurred during each round can be computed as: $(a * \sum_{i=1}^{n} k_i + b * T)$. Additionally, retrieval of each media block may incur a rotational latency of l_{rot}^{max}, and hence, the total rotational latency incurred during each round is bounded by $(l_{rot}^{max} * \sum_{i=1}^{n} k_i)$. Hence, the total service time for each round is bounded by:

$$\tau = b * T + (a + l_{rot}^{max}) * \sum_{i=1}^{n} k_i \qquad (1)$$

Consequently, ensuring continuous retrieval of each strand requires that the total service time per round does not exceed the minimum of the playback durations of k_1, k_2, ..., or k_n blocks. We refer to this as the *admission control criteria*, and can be formally stated as:

$$b * T + (a + l_{rot}^{max}) * \sum_{i=1}^{n} k_i \leq \min_{i \in [1,n]} \left(\frac{k_i * M}{\mathcal{R}_{pl}^{i}} \right) \qquad (2)$$

Notice that, for a given number of clients, satisfying Equation (2) ensures that the playback requirements of all the clients are *strictly* met for the entire service duration. Hence, a multimedia server which employs such ab admission control criteria is said to provide *deterministic* service guarantees to each client.

2.2 Discussion

In general, clients of a multimedia service can be classified into two categories: *intolerant* and *tolerant*. Whereas intolerant clients demand that their requirements be strictly met throughout the service period, tolerant clients may afford brief distortions or loss of information (especially so if it is reflected in a reduced cost of service). Additionally, the loss tolerances may vary from one client to another. Consequently, providing deterministic service guarantees is appropriate for intolerant clients, but an overkill for tolerant clients.

Furthermore, the worst-case assumptions, regarding the seek and rotational latency incurred while accessing each media block that characterize Equation (2), may needlessly constrain the number of clients that are serviced simultaneously, and hence, may lead to severe under-utilization of server resources. This is because, the average time spent in accessing a media block from disk, in practice, is significantly smaller than the corresponding worst case values. Hence, in order to improve the utilization of server resources, a multimedia server must employ an admission control algorithm which exploits the statistical variation in the access times of media blocks from disk.

Towards achieving these goals, we have developed an *observation-based admission control algorithm*, which is based on the assumption that the amount of time spent in servicing each of the clients already being serviced will continue to exhibit the same behavior, even after a new client is added into the system. Therefore, a new client will be admitted for service only if the prediction from the status quo measurements of the server performance characteristics indicate that the service requirements of all the clients can be met satisfactorily. A multimedia server that employs such an observation-based approach is referred to as providing *predictive* service guarantees to clients (A similar technique was presented by Clark et al. [3, 7] for optimizing the utilization of network resource).

Notice that the observation-based admission control algorithm offers fairly reliable service, but no absolute guarantees. The admission control decisions are based on the measured characteristics of the current load on the server, rather than theoretical worst-case behavior. Hence, the key function of the admission control algorithm is to accept enough traffic to efficiently utilize the server resources, while not accepting clients whose admission may lead to the violation of the service requirements. Details of our admission control algorithm are presented in the next section.

3 Admission control algorithm

Consider a multimedia server that is servicing n clients, each retrieving a media strand (say $S_1, S_2, ..., S_n$, respectively). Let n_d and n_p denote the number of clients that require deterministic and predictive service guarantees, respectively (i.e., $n = n_d + n_p$). Without loss of any generality, let us assume that clients retrieving strands

$S_1, S_2, ..., S_{n_d}$ require deterministic service guarantees, and those retrieving $S_{n_d+1}, ..., S_n$ are tolerant to brief distortions or loss of information. Let $\forall i \in [1, n]$, the level of tolerance for client i be characterized by the the value P_i, which denotes the percentage of media blocks of strand S_i that must be retrieved on time so as to satisfy the service requirements of the clients. Clearly, since clients retrieving strands $S_1, S_2, ..., S_{n_d}$ require deterministic service guarantees, we get: $\forall i \in [1, n_d] : P_i = 1$. On the other hand, for all the tolerant clients, we get: $\forall i \in [n_d + 1, n] : 0 < P_i \leq 1$.

As we had mentioned in Section 2, a multimedia server can service multiple clients simultaneously by proceeding in terms of rounds, retrieving a fixed number of media blocks for each client during each round. Let $k_1, k_2, ..., k_n$ denote the number of media blocks of strands $S_1, S_2, ..., S_n$ retrieved during each round. Then, assuming that $\forall i \in [1, n] : \mathcal{R}_{pl}^i$ denotes the playback rate of strand S_i, the playback duration of a round is given by:

$$\mathcal{R} = \min_{i \in [1,n]} \left(\frac{k_i * M}{\mathcal{R}_{pl}^i} \right)$$

On the other hand, the total time spent in retrieving the media blocks from disk during each round (referred to as *service time* τ) is dependent on their relative placement on disk as well as the disk scheduling algorithm. Since the relative placement of blocks can vary from one round to another, the service times may also vary across rounds. Consequently, in some rounds, τ can be greater than $\mathcal{R}$. We refer to such rounds as *overflow* rounds. Since maintaining continuity of playback for intolerant clients requires the service time to be smaller than the duration of a round, blocks of some of the tolerant clients may have to discarded (i.e., not retrieved) during such overflow rounds. Hence, if $\mathcal{K}$, $\mathcal{K} \leq \sum_{i=1}^{n} k_i$, denotes the number of media blocks accessed during a round, then the average amount of time spent in retrieving each media block during the round can be computed as:

$$\eta = \frac{\tau}{\mathcal{K}}$$

The admission control algorithm that we present in this section is based on the assumption that the average amount of time spent for the retrieval of each media block (i.e., the value of η) does not change significantly even after a new client is admitted by the server. In fact, to enable the multimedia server to accurately predict the amount of time expected to be spent while retrieving media blocks during a future round, we maintain a history of the values of η observed during the most recent W rounds (referred to as the *averaging window*). If η_{avg} and σ, respectively, denote the average and the standard deviation of η over W rounds, then the time required to retrieve a block in future rounds ($\widehat{\eta}$) can be estimated as:

$$\widehat{\eta} = \eta_{avg} + \epsilon * \sigma \tag{3}$$

where ϵ is an empirically determined constant. Clearly, a positive value of ϵ enables the estimation process to take into account the second moment of the random variable η, and hence, can be used to make the estimate reasonably conservative. The value of $\widehat{\eta}$, thus derived, forms the basis of our admission control algorithm.

In order to precisely formulate the admission control criteria, consider a scenario in which a multimedia server, receives a new client request for the retrieval of strand S_{n+1}. Let $\mathcal{R}_{pl}^{n+1}$ denote the playback rate requirement for client $(n + 1)$, and k_{n+1} denote the number of blocks of strand S_{n+1} that need to be retrieved during each round.

If the new client desires deterministic service guarantees, then before admitting the client, the multimedia server must ensure that neither deterministic nor predictive services being provided to the existing clients will be violated after the new client is admitted:

1. To verify that the deterministic service guarantees provided to n_d clients will not be violated, the multimedia server must evaluate the deterministic admission control criteria derived in Section 2.1. Specifically, the multimedia server must ensure that:

$$b * T + (a + l_{rot}^{max}) * \sum_{i=1}^{n_d+1} k_i \leq \mathcal{R} \tag{4}$$

2. The predictive service being provided to the $n_p = n - n_d$ clients will not be violated after the admission of the $(n+1)$th client, if the extrapolation from the values of η measured during the most recent W rounds indicate that the service requirements of all the $(n + 1)$ clients can be met satisfactorily. The precise formulation of this requirement is as follows:

   Since P_i denote the percentage of media blocks of strand S_i that must be retrieved on time so as to meet the requirements of client i, the average number of blocks of strand S_i that must be retrieved by the multimedia server during each round is given by $k_i * P_i$. Furthermore, since the average time spent in retrieving a media block from disk is empirically derived to be $\widehat{\eta} = \eta_{avg} + \epsilon * \sigma$, the requirements of tolerant clients will not be violated if:

$$\widehat{\eta} * \left(\sum_{i=1}^{n+1} k_i * P_i \right) \leq \mathcal{R} \tag{5}$$

   where $\forall i \in [1, n_d] : P_i = 1$, $\forall i \in [n_d + 1, n] : 0 < P_i \leq 1$, and $P_{n+1} = 1$.

Hence, a multimedia server will admit a new client client requiring deterministic service guarantees if and only if Equations (4) and (5) are satisfied.

Consider, on the other hand, the scenario in which the new client requires predictive service. Let $0 < P_{n+1} \leq 1$ denote the percentage of media blocks of strand S_{n+1} that must be retrieved on time from the multimedia server so as to satisfy the service requirements of the new client. Since admitting a tolerant client cannot violate the requirements of intolerant clients (since in the case of an overflow, media blocks of only tolerant clients are discarded), the multimedia server can admit the new client if its admission will not violate the predictive service being provided to any client. Hence, the multimedia server will admit the new client requiring predictive service only if Equation (5) is satisfied.

Notice that the admission control criteria presented above is only a heuristic which can be employed by a multimedia server to increase the number of clients that can be serviced simultaneously. The performance of such an observation-based admission control algorithm is dependent on:

- The values of η_{avg} and σ: smaller the values of η_{avg} and σ, greater is the number of clients that can be serviced simultaneously by the server. Hence, the multimedia server must employ disk scheduling algorithms that minimize the total time spent in retrieving media blocks during each round (Section 4).

- The ability of the algorithm to meet the requirements of as many clients as possible. This requires that the multimedia server employ policies for determining the minimum number of media blocks, which when discarded will yield sufficient reduction in service time so as to ensure that $\tau \leq \mathcal{R}$ (Section 5).

4 Techniques for efficient retrieval of media blocks

Consider a multimedia server that is expected to retrieve N blocks during each round. Let the position of each media block on disk be denoted as $z_i = \langle x_i, y_i \rangle$, where x_i and y_i denote the track number and the block number on that track, respectively. The goal of the disk scheduling algorithm is to find the optimal access sequence for blocks $z_1, z_2, ..., z_N$, assuming that the head is initially positioned at location $z_0 = \langle x_0, y_0 \rangle$. Formally, if $\psi(z_i, z_j)$ denotes the cost function characterizing the overhead in positioning the disk head at location z_j starting from location z_i, then a sequence for retrieving N media blocks can be considered optimal if it minimizes the sum:

$$\Psi = \psi(z_0, \omega_1) + \psi(\omega_1, \omega_2) + \cdots + \psi(\omega_{N-1}, \omega_N)$$

where $\forall i \in [1, N] : \omega_i \in \{z_1, z_2, ..., z_N\}$, and $\forall j \in [1, N], j \neq i : \omega_i \neq \omega_j$. The sequence $\omega_1, \omega_2, ..., \omega_N$ denotes the *retrieval sequence*.

Most of the conventional disk scheduling algorithms (such as, SCAN, Shortest Seek Time First (SSTF), etc.) have addressed the problem of optimizing the total seek time incurred while accessing a sequence of blocks from disk [4, 6, 10, 14]. That is, the cost function is defined as $\psi(z_i, z_j) = l_{seek}(z_i, z_j) = a + b * |x_i - x_j|$. The fundamental limitation of these disk scheduling algorithms is that they optimize only the seek time, and completely ignore the rotational latency. In most of the state-of-the-art disks, however, the values of maximum seek and rotational latencies are comparable (typically, the maximum rotational latency is about half of the maximum seek time). Consequently, disk scheduling algorithms which optimize only the seek time (e.g., SCAN, SSTF, etc.) may incur significant rotational latencies during the retrieval of a sequence of blocks from disk.

To address this limitation, we present a disk scheduling algorithm that derives the sequence for accessing media blocks from disk by simultaneously minimizing both seek and rotational latency incurred during retrieval. Specifically, we define the cost function $\psi(z_i, z_j)$ as:

$$\psi(z_i, z_j) = l_{seek}(z_i, z_j) + l_{rot}(z_i, z_j)$$

where $l_{seek}(z_i, z_j)$ and $l_{rot}(z_i, z_j)$ denote the seek time and rotational latency incurred while moving the disk head from location z_i to z_j, respectively. In what follows, we first formulate the problem of finding an optimal sequence for retrieving media blocks from disk as a graph theory problem, and then present a simple heuristic which is extremely easy to implement, and yet achieves significant performance improvements over the conventional SCAN and the SSTF disk scheduling policies.

4.1 An efficient disk scheduling algorithm

Consider a fully connected directed graph $G = (V, E)$, where $V = \{z_0, z_1, z_2, ..., z_N\}$. Let each edge $(z_i, z_j), i \neq j$ in G be labeled with weight $\psi(z_i, z_j)$. Notice that since the rotational latency incurred while moving the disk head from node z_i to z_j, in general, is not equal to that incurred while moving the disk head from z_j to z_i, the graph G contains both the edges (z_i, z_j) and (z_j, z_i), each with a different weight.

Having constructed this graph, the problem of minimizing the total cost of retrieving media blocks during a round starting from node z_0 can be reduced to the *traveling salesman problem*, a classical graph theory problem known to be NP-complete [14]. Hence, instead of deriving an optimal sequence for accessing media blocks, we present a *greedy*

```
sort all edges in increasing order of cost;
edgesPicked = 0;
G' = NIL;
While (edgesPicked < N) do
{
        select the next edge (z_i, z_j), j ≠ 0;
        if (in-degree(z_j) = 0 ∧ out-degree(z_i) = 0
           ∧ tail(z_i) ≠ z_j) then
        {
                G' = G' + (z_i, z_j);
                edgesPicked = edgesPicked +1;
                in-degree(z_j) = 1;
                out-degree(z_i) = 1;
                MaintainTail();
        }
}
```

Figure 1 : Greedy disk scheduling algorithm

algorithm which, in practice, derives near-optimal retrieval sequences.

Given the graph $G = (V, E)$, the goal of the greedy algorithm is to construct a minimum cost sub-graph $G' = (V, E')$, such that edges in E' constitute a simple path of length N starting from z_0. In order to do so, the greedy heuristic evaluates and selects edges in the increasing order of their weight. An edge $(z_i, z_j), j \neq 0$ is included in E' if it does not create a cycle amongst already selected edges, and if out-degree(z_i) $= 0$ and in-degree(z_j) $= 0$ (i.e., ensure that at most one edge is incident upon and emanating from each node). Notice that since the edges are chosen in the increasing order of weight, the heuristic may initially create a disconnected set of *fragments* (i.e., a chain of edges), which are eventually connected to form a path that connects all the nodes in the graph. Consequently, this heuristic is also some times referred to as the *multiple fragment* heuristic [2]. The algorithm terminates when $|E'| = N$. The complete algorithm for deriving graph G' is described in Figure 1.

As is evident from Figure 1, the greedy algorithm first sorts all the edges in the increasing order of weight, and then constructs G' by sequentially considering edges from the sorted array. In order to ensure that at most one edge is incident upon and emanating from each node in G', the algorithm maintains the values of in-degree and out-degree for each node. Furthermore, cycles are detected by maintaining the *tail* array (through the MaintainTail() procedure) to identify the first node in each fragment. An analysis of this algorithm will reveal that the overall complexity of the algorithm is $O(N^2 logN)$. It can be shown that the complexity of the above algorithm can be reduced to $O(N^2)$ by employing bucket sort for ordering the edges in the increasing order of weight. A detailed description of these techniques is provided in [12], and is beyond the scope of this paper.

Observe that the greedy algorithm is an off-line algorithm. Hence, it requires that the set of blocks to be retrieved from disk be known a priori. The sequential nature of audio and video playback enables a multimedia server to predict the set of blocks that need to be accessed during successive rounds. Consequently, during each round, a multimedia server can retrieve media blocks from disk while concurrently computing the retrieval sequence for the next round, and so on. Hence, employing this algorithm is not only feasible, but also efficient.

5 Enforcing predictive service guarantees

Recall that the admission control algorithm presented in Section 3 admits a new client if the predicted extrapolation from the measurements of the average time spent in retrieving a media block from disk during the last W rounds indicate that the service requirements of all the clients can be met satisfactorily. Due to the aggressive nature of this admission control criteria, the total time spent in retrieving media blocks during a round (i.e., service time τ) may occasionally exceed the duration of a round $\mathcal{R}$, yielding an overflow.

Observe, however, that since the retrieval sequence for round i is precomputed during round $(i - 1)$, an overflow can be detected before actually initiating round i. Hence, in order to ensure that the deterministic service guarantees provided to clients are not violated, a multimedia server must *discard* (i.e., not retrieve) sufficient number of media blocks of tolerant clients so as to maintain the service time within the duration of the round (i.e., $\tau \leq \mathcal{R}$). However, in doing so, the multimedia server must minimize the number of blocks discarded, as well as distribute the set of discarded blocks among the tolerant clients so as not to violate any of their requirements. In what follows, we present a technique for addressing this problem.

Consider a multimedia server that is servicing n clients simultaneously. Let $\forall i \in [1, n] : P_i$ denote the percentage of media blocks of strand S_i that must be retrieved on time from the multimedia server so as to meet the service requirements of the clients. Assuming that n_d clients require deterministic service guarantees, we get $\forall i \in [1, n_d] : P_i = 1$. Let the requirements of each of the tolerant clients be met on an average over a period of length I time units. Since $\mathcal{R}$ denotes the duration of a round, each interval of length I contains exactly $r = \frac{I}{\mathcal{R}}$ rounds. Furthermore, since k_i

blocks of strand S_i are expected to be retrieved during each round, the number of media blocks of strand S_i which can be discarded, within each interval of I time units, without violating the requirements of client i is bounded by:

$$L_i = \lfloor (1 - P_i) * k_i * r \rfloor \quad (6)$$

Consider, now, an overflow round $\mathcal{O}$, $1 \leq \mathcal{O} \leq r$. Let l_i denote the number of media blocks of strand S_i that have already been discarded during this service interval. Consequently, the number of media blocks of strand S_i that can be discarded during round $\mathcal{O}$ is bounded by $(L_i - l_i)$, based on which we define the *loss affordability* for client i as:

$$\chi_i = \frac{L_i - l_i}{L_i} = 1 - \frac{l_i}{L_i} \quad (7)$$

Clearly, as the number of blocks of strand S_i discarded during an interval I increases, the loss affordability of client i decreases. In the limit, if $l_i = L_i$ then $\chi_i = 0$. That is, if the number of media blocks of strand S_i discarded during interval I has already reached its upper limit, no more blocks of S_i should be discarded during round $\mathcal{O}$. A multimedia server will discard blocks of only those clients with $\chi_i > 0$.

Given the set of clients with $\chi_i > 0$, a multimedia server may employ various techniques to select the precise set of media blocks which can be discarded. In the simplest case, media blocks of a strand with the highest loss affordability can be discarded until no more blocks of that strand can be discarded during the round. If the resulting round continues to yield an overflow (i.e., $\tau > \mathcal{R}$), then discard blocks of the strand with the second largest loss affordability, and so on. Although extremely simple to implement, such a scheduling policy may yield lopsided loss distributions. A multimedia server can address this problem by distributing the number of media blocks that need to be discarded during round $\mathcal{O}$ among all the clients with $\chi_i > 0$.

In either case, once a media block to be discarded is determined, the multimedia server can recompute the service time τ_{new} (by determining a new retrieval sequence using the greedy algorithm). If $\tau_{new} > \mathcal{R}$, then the number of media blocks discarded can be progressively increased until the new retrieval sequence yields $\tau_{new} \leq \mathcal{R}$. However, the high computational complexity of the greedy algorithm renders this straightforward approach prohibitively expensive to implement. To avert the recomputations, a multimedia server may just approximate the reduction in service time yielded by discarding a media block located at ω_i as $\psi(\omega_{i-1}, \omega_i) + \psi(\omega_i, \omega_{i+1}) - \psi(\omega_{i-1}, \omega_{i+1})$, where ω_{i-1} and ω_{i+1} denote the predecessor and successor nodes of ω_i in the original retrieval sequence. Conceptually, this is equivalent to replacing edges (ω_{i-1}, ω_i) and (ω_i, ω_{i+1}) from graph G' by edge $(\omega_{i-1}, \omega_{i+1})$.

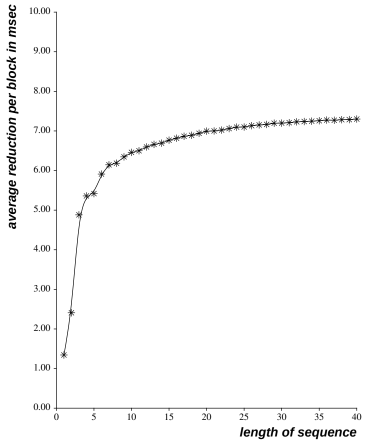

Figure 2 : Variation in the average reduction in service time yielded for each discarded media block with the subsequence length

Whereas such an approximation technique significantly reduces the computational complexity of the algorithm, it is not suitable for either one of the schemes (namely, discarding as much for one client before switching to the next client, or distributing the discarded blocks among all the clients with non-zero loss affordability). This is because, both of these schemes determine the set of blocks to be discarded based on the loss affordability of the strands, and not on the relative placement of the chosen blocks on disk. Since the greedy algorithm derives a retrieval sequence which minimizes both the seek time as well as the rotational latency, discarding an isolated block from the sequence may not yield any significant reduction in the service time (i.e., the difference $(\psi(\omega_{i-1}, \omega_i) + \psi(\omega_i, \omega_{i+1}) - \psi(\omega_{i-1}, \omega_{i+1}))$ may not be significant). For instance, if blocks ω_{i-1}, ω_i, and ω_{i+1} happen to be located on the same track, then discarding block ω_i does not yield any reduction in the service time at all. In fact, the average reduction in service time yielded for each discarded media block is higher if a sequence of n media blocks, rather than n isolated media blocks, are discarded during a round (see Figure 2). Consequently, any scheme that is based solely on the loss affordability of clients may discard a larger number of media blocks, than are necessary, during an overflow round.

To address this limitation, we now present an algorithm that selects a set of media blocks to be discarded during a round based both on the loss affordability of the clients as well as the relative placement of the selected blocks on disk. To clarify the exposition of the algorithm, let us assume that $\omega_0, \omega_1, ..., \omega_N$ ($\omega_0 = z_0$, $\forall i \in [1, N] : \omega_i \in \{z_1, z_2, ..., z_N\}$, and $i \neq j \Rightarrow \omega_i \neq \omega_j$) denotes the retrieval sequence derived by the greedy al-

Disk capacity	4 GBytes
Number of disks in the array	16
Number tracks per disk	1024
Disk block size	32 KBytes
Rate of disk rotation	3600 RPM
$l_{seek}(t_1, t_2)$	$4 + 0.02 * \|t_1 - t_2\|$ ms
Maximum seek time	24.48 ms
Maximum rotational latency	16.66 ms

Table 2 : Disk parameters assumed in the simulation

gorithm. Based on the loss affordability of each strand, the algorithm first labels each media block (i.e., ω_i's) to be retrieved during the round as either *can-be-discarded* or *can-not-be-discarded*, and then determines maximal length subsequences of can-be-discarded blocks. For each such subsequence, the server computes the average reduction in service time per node yielded by discarding the entire subsequence. The algorithm then discards as many complete subsequences as needed, in the decreasing order of the average reduction, so as to make $\tau \leq \mathcal{R}$, with possibly the last subsequence discarded partially.

If, even after discarding all the subsequences, the service time continues to be greater than the duration of the round, then the subsequence determination algorithm is repeated with all the blocks of tolerant clients marked as can-be-discarded. Whereas this would violate the requirements of some of the tolerant clients, proper choice of ϵ in the admission control criteria (Equation (5)) virtually eliminates the possibility of such an event.

Notice that the labeling operation as well as the process of deriving subsequences of can-be-discarded blocks are relatively straightforward. Hence, the algorithm not only reduces the number of media blocks that are discarded during round $\mathcal{O}$, but does so efficiently.

6 Performance evaluation

So far, we have presented an admission control algorithm and a disk scheduling technique for a multimedia server. In this section, we demonstrate their viability through simulations. The simulations were carried out in an environment consisting of a synchronous disk array with 16 disks. The characteristics of each disk are shown in Table 2. Each media strand is assumed to be striped across the entire disk array, and successive blocks of a strand are stored on the disk using the random placement model [8].

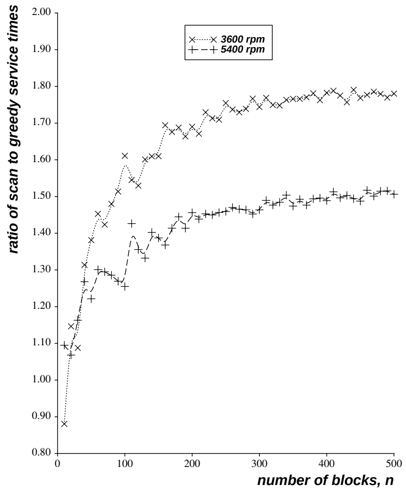

Figure 3 : Comparison of SCAN and greedy disk scheduling algorithms

6.1 Evaluation of the greedy disk scheduling algorithm

Recall that the greedy algorithm presented in Section 4 derives a sequence of accessing media blocks from disk so as to simultaneously minimize both seek and rotational latency. In contrast, conventional disk scheduling algorithms (such as, SCAN) optimize only the seek time. Figure 3 shows the variation in the ratio of the service times derived for the SCAN and the greedy algorithms with increase in the number of blocks, n, retrieved during each round. Observe that the performance of the greedy scheduling technique improves with increase in n. This is because, at smaller values of n, the total seek time incurred during their retrieval dominates the performance (and hence, SCAN performs as well as the greedy algorithm). However, as the value of n increases, the cumulative rotational latency starts dominating the total service time, thereby enabling the greedy algorithm to outperform SCAN. Figure 3 also indicates that even at higher values of rotational rate, the gain in performance yielded by the greedy algorithm is significant.

In order to demonstrate the viability of employing the greedy algorithm for determining the retrieval sequence during each round, we compared the time required to derive the sequence with the service time. Whereas the time to compute the retrieval sequence shows a quadratic dependence on the number of blocks being retrieved, the time spent in retrieving the blocks increases linearly with increase in the number of blocks. Figure 4 illustrates this behavior. It also illustrates that as long as the number of blocks being retrieved in a round is less than 650, the time to derive a retrieval sequence is less than the service time. As we will point out later, the admission control criteria

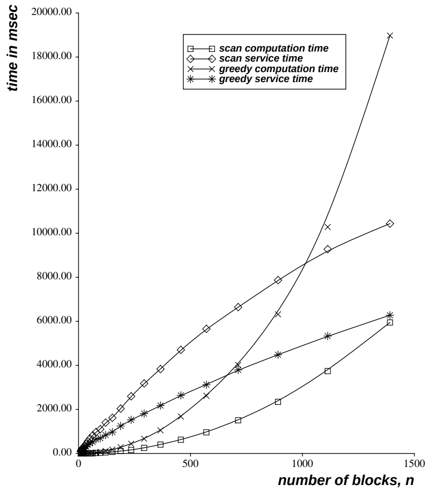

Figure 4 : Evaluation of the greedy algorithm - comparison of time to derive a retrieval sequence with the service time. The time to derive the retrieval sequence was measured on a moderately loaded SPARCstation 10 - Model 30.

restricts the value of n to be about 150. Hence, although theoretically possible, the time required to compute a retrieval sequence, in practice, will never exceed the service time. In fact, at $n = 150$, the time to compute the retrieval sequence is only about 18% of the total service time.

6.2 Evaluation of the admission control algorithm

For our simulations, we assume that client requests received by the multimedia server are categorized into three service classes: a deterministic class (i.e., $P_0 = 1$), and two predictive classes (with $P_1 = 0.99$ and $P_2 = 0.97$). Client requests in each class were generated using a poisson process, with average inter-arrival times of 2, 1, and 1 seconds, respectively. During each round, exactly one block from each of the disks was retrieved for each client (i.e., $\forall i \in [1, n] : k_i = 1$). Since the block size is assumed to be 32 Kbytes, the total amount of media information retrieved during each round is 512 KBytes. Assuming that the playback rate of each strand is 512 Kbytes/s yields $\mathcal{R} = 1$ second.

In order to enable the multimedia server to accurately predict the amount of time expected to be spent while retrieving media blocks during a future round, we maintain a history of the values of η observed during the most recent 75 rounds (i.e., the value of W was empirically determined to be 75). The values of η_{avg} and σ were computed over the averaging window. The effectiveness of the admission control criteria was measured for different values of ϵ. Figure 5 depicts the variation in the average number of clients admitted as well as the number of clients successfully serviced (i.e without the service requirements being violated) with increase in ϵ. It illustrates that for $\epsilon = 1.0$, the service requirements of all the clients are satisfied with the average number of clients being 135. In comparison, the deterministic admission control algorithm only services 47 clients, thereby demonstrating that the observation-based admission control algorithm increases the number of clients serviced simultaneously by about 200%.

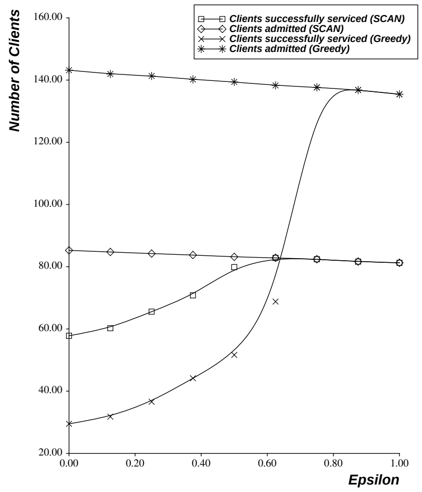

Figure 5 : Evaluation of the admission control algorithm

Figure 5 also illustrates that a multimedia server employing greedy disk scheduling algorithm can service a larger number of clients simultaneously as compared to the one that employs the SCAN disk scheduling algorithm. In fact, for $\epsilon = 1.0$ greedy disk scheduling algorithm enables the server to service about 135 clients simultaneously whereas SCAN reduces this number to 81.

7 Conclusion

Providing deterministic service guarantees to each clients may needlessly constrain the number of clients that are serviced by a multimedia server, and hence, may lead to severe under-utilization of server resources. To address this limitation, we have presented an observation-based admission control algorithm, in which a client is admitted by a multimedia server if the predicted extrapolation from the status quo measurements of the storage server utilization indicate that the service requirements of all the clients can be met satisfactorily. The main goal of our admission control algorithm is to accept enough traffic to efficiently utilize the server resources, while not accepting clients whose admission may lead to the violations of the service requirements of clients. The greedy disk scheduling algorithm as well as the technique for minimizing and distributing the discarded media blocks presented in this paper significantly improve the performance of the admission control algorithm.

We have demonstrated the effectiveness of the observation-based admission control algorithm and the greedy disk scheduling algorithm through simulations. Our simulations reveal that the greedy disk scheduling algorithm yields an 80% improvement in performance over the conventional SCAN disk scheduling algorithm. Furthermore, employing the greedy disk scheduling algorithm in conjunction with the observation-based admission control algorithm yields a server utilization of about 97%, and a 200% increase in the number of clients serviced simultaneously by the multimedia server. A prototype multimedia server, based on the algorithms presented in this paper, is being implemented at the UT Austin Distributed Multimedia Computing Laboratory. Our prototype server will employ resource reservation techniques that will integrate admission control algorithms for storage server and network, and will provide end-to-end real-time service guarantees to each client.

REFERENCES

[1] D. Anderson, Y. Osawa, and R. Govindan. A File System for Continuous Media. *ACM Transactions on Computer Systems*, 10(4):311–337, November 1992.

[2] J.L. Bentley. Experiments on Geometric Travelling Salesman Heuristics. Technical Report Computing Science Technical Report No. 151, AT&T Bell Laboratories, Murray Hill, NJ, 1990.

[3] D.D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network. In *Proceedings of ACM SIGCOMM*, pages 14–26, August 1992.

[4] E. G. Coffman and M. Hofri. On Scanning Disks and the Analysis of their Steady State Behavior. *Proceedings of the Conference on Measuring, Modeling, and Evaluation Computer Systems, North-Holland*, pages 251–263, October 1977.

[5] J. Gemmell and S. Christodoulakis. Principles of Delay Sensitive Multimedia Data Storage and Retrieval. *ACM Transactions on Information Systems*, 10(1):51–90, 1992.

[6] M. Hofri. Disk Scheduling: FCFS vs. SSTF Revisited. *Communications of the ACM*, 23(11):645–653, November 1980.

[7] S. Jamin, S. Shenker, L. Zhang, and D.D. Clark. An Admission Control Algorithm for Predictive Real-Time Service (extended abstract). In *Proceedings of Third International Workshop on Network and Operating Systems Support for Digital Audio and Video, San Diego, CA*, pages 308–315, November 1992.

[8] M. K. McKusick, W. N. Joy, S. J. Leffler, and R. S. Fabry. A Fast File System for UNIX. *ACM Transactions on Computer Systems*, 2(3):181–197, August 1984.

[9] P. Venkat Rangan and Harrick M. Vin. Designing File Systems for Digital Video and Audio. In *Proceedings of the 13th Symposium on Operating Systems Principles (SOSP'91), Operating Systems Review, Vol. 25, No. 5*, pages 81–94, October 1991.

[10] T. Teorey and T. B. Pinkerton. A Comparative Analysis of Disk Scheduling Policies. *Communications of the ACM*, 15(3):177–184, March 1972.

[11] F.A. Tobagi, J. Pang, R. Baird, and M. Gang. Streaming RAID: A Disk Storage System for Video and Audio Files. In *Proceedings of ACM Multimedia'93, Anaheim, CA*, pages 393–400, August 1993.

[12] Harrick M. Vin, Alok Goyal, Anshuman Goyal, and Pawan Goyal. An Observation-Based Approach For Designing Multimedia Servers. Technical report, Department of Computer Sciences, University of Texas at Austin, 1993.

[13] Harrick M. Vin and P. Venkat Rangan. Designing a Multi-User HDTV Storage Server. *IEEE Journal on Selected Areas in Communications*, 11(1):153–164, January 1993.

[14] C.K. Wong. Minimizing Expected Head Movement in One Dimension and Two Dimension Mass Storage Systems. *Computing surveys*, 12(2):167–178, 1980.

[15] P. Yu, M.S. Chen, and D.D. Kandlur. Design and Analysis of a Grouped Sweeping Scheme for Multimedia Storage Management. *Proceedings of Third International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego*, pages 38–49, November 1992.