

# A Deadline Driven Disk Scheduling Algorithm for Continuous Media Servers

**Sameh M. Elnikety**

Rice University  
sameh@cs.rice.edu

**Walid G. Aref**

Purdue University  
aref@cs.purdue.edu

**Mohamed S. Abougabal**

Alexandria University  
abugabal@soficom.com.eg

## Abstract

Multimedia applications require different levels of service for accessing continuous media data (e.g., video and audio). To provide such different levels of service, the underlying continuous media server must efficiently support a diverse group of activities like the recording and playback of video streams with different qualities. This translates into disk requests with different types. For example, some disk requests may have deadlines while other particular disk requests can be discarded under certain conditions (as in the case of playing back a video stream for a video-on-demand application). Existing disk scheduling algorithms do not support all the types of disk requests required by server-based multimedia applications. In this paper, we present a novel organization for the disk queue and develop a deadline driven disk scheduling algorithm that is able to efficiently handle all the different types of disk requests serviced by continuous media servers. The algorithm enhances the utilization of the disk bandwidth by maintaining one queue for all the disk request types and by optimizing the seek time. Our experimental results show that the algorithm can support 6% more users for video-on-demand and video authoring when compared with other algorithms, and can support different mixes of applications.

## 1 Introduction

Multimedia applications are growing rapidly due to many-fold increase in CPU processing power and network bandwidth, as well as the advances in mass storage devices and video compression techniques. These applications include video-on-demand, video authoring tools, news broadcasting, video conferencing, digital libraries, and interactive video games. These multimedia applications are interesting and challenging because they require special resource management to accommodate the real-time demands of these applications as well as the high bandwidth that they need.

An important component of such multimedia systems is the continuous media server that can display and record continuous media (e.g., video and audio) as

well as provide access to other types of traditional data (e.g., text and images). The complexity in the design of such a server arises due to the wide range of requirements of the multimedia applications.

Due to the high bandwidth and the real-time demands of these applications, the disk bandwidth is a precious resource. Although disk technology improves in speed and capacity, applications requirements also increase at the same pace or faster. For example, the news broadcasters and the entertainment industry have started to use high quality audio and video streams like MPEG 4:2:2 and DVCPRO which require bandwidth ranges between 20-60 Mbps.

To service a disk request, several operations take place. The disk head must be moved to the appropriate cylinder (*seek time*). Then, the portion of the disk on which the disk page is stored must be rotated until it is immediately under the disk head (*latency time*). Then, the disk page must be made to spin by the disk head (*transmission time*). During both seek time and latency time, the disk does wasteful work. Whereas, during the transmission time, the disk does useful work as the required data are transferred.

Queues build up for each disk because the inter-arrival time of the disk requests can be smaller than the time required by the disk to service a disk request. Carefully changing the order in which requests are serviced has the potential of the minimizing the amount of wasteful work and hence maximizing the disk bandwidth.

In a video-on-demand system, each user has a set-top box and is connected to the video-on-demand server through a fast network. The display of a video clip requires retrieving the data from the disks at the server in a timely fashion. Each disk request has a soft deadline associated with it. If a data block arrives too late, it will be useless and dropped immediately. Also, if a data block arrives too early, it will be dropped due to the limited buffer space at the user set-top box. Therefore, for playing-back continuous media data, each disk request must be serviced within a predetermined time range. This is in contrast to

Request symbol	Type	Dead-line	Can be lost	Applications
Rdl	Read	Yes	Yes	Video-on-demand playback
Rdn	Read	Yes	No	Editor previewing data after editing, computer vision
Rnl	Read	No	Yes	Does not really exist because if there is no deadline, then it can always be serviced
Rnn	Read	No	No	Editor downloading a media clip for editing, metadata reading
Wdl	Write	Yes	Yes	Low quality real-time recording (for example, archival of a video conferencing session)
Wdn	Write	Yes	No	High quality real-time recording (for example, from cameras in a site)
Wnl	Write	No	Yes	Does not really exist because if there is no deadline, then it can always be serviced
Wnn	Write	No	No	Editor writing data after editing, metadata updates

Table 1: Classification of disk requests for a continuous media server

traditional data where usually there is no such deadlines.

Also, in a video-on-demand system, if a request is not serviced within its deadline, it is considered *lost* and results in a possible discontinuity of display. The user may not notice this discontinuity if few requests are lost (less than 3%) and the lost requests are not relatively consecutive. So, in a system like this a loss of few requests is accepted. This is in contrast to video authoring where editors download video clips, edit them, and then upload the video clips to the server for a later broadcast. In this case, the editors expect the clips to be downloaded and uploaded perfectly without any loss of data.

From the server point-of-view, write requests need to be buffered in main memory before being written to the disk. Whereas read requests are sent immediately to the clients to be buffered or played back. This means that the write requests compete for the buffer space at the server.

This motivates classifying disk requests as having deadline or not, if it is possible to lose few of them under certain conditions, and if the requests are read requests or write requests. Table 1 illustrates all the types of the disk requests that should be serviced by a continuous media server for different applications [AKN97, AKG]. In particular, each disk request can be either a read or a write request, may have a deadline or may not, and can be lost or can not be lost.

Using this classification, users in a video-on-demand system issue only Rdl requests because users read data blocks from the server and each disk request has a deadline. Also, it is possible to lose a very small number of requests while meeting a certain Quality of Service (QoS). In a video authoring system, editors issue Rnn requests to download media clips as they read data blocks from the server. These blocks have no deadlines but starvation must be prevented. All the data must be downloaded perfectly. After editing the video clips, editors upload them to the server. Wnn requests are issued as the data blocks are written to the server disks. Again, the data blocks have no deadlines and they should be written perfectly. It is interesting to note

that Wnl and Rnl request types do not really exist because if a request does not have a deadline, it can always be serviced.

In this paper, we propose a deadline driven disk scheduling algorithm which handles all the types of disk requests in a unified manner. All the disk requests for a specific disk are maintained in a single queue. The algorithm strives to (1) optimize the disk bandwidth by keeping the disk requests in scan order, (2) honor the timing constraints of the requests, and (3) minimize the number of lost requests.

The remaining of the paper is organized as follows. Section two surveys previous studies on related disk scheduling algorithms. In section three, the proposed algorithm is presented. In section four, the analysis of the algorithm is shown. Performance evaluation and comparison experiments are explained in section five. Finally, the paper is concluded in section six.

## 2 Related Work

The disk scheduling problem involves reordering the disk requests in the disk queue to meet certain criteria like maximizing the throughput. In the case of maximizing the disk throughput, the disk scheduling problem can be reduced to the travelling salesman problem [SV98] which is a classical graph theory problem known to be NP-complete. Thus, finding the optimal schedule for a group of disk requests is an NP-complete problem and is computationally infeasible.

When timing constraints are imposed on the disk requests, maximizing the disk bandwidth is not sufficient. In this environment, a scheduling algorithm should both maximize the disk bandwidth and honor the timing constraints of the requests.

In the context of real-time systems, the objective of disk scheduling is to satisfy the timing constraints of the requests. In general, these real-time disk scheduling algorithms (which offer deterministic guarantees) are not suitable for multimedia applications (which require statistical guarantees) because (1) the disk requests have soft deadlines rather than hard deadlines and (2) optimizing the disk bandwidth is important even at the

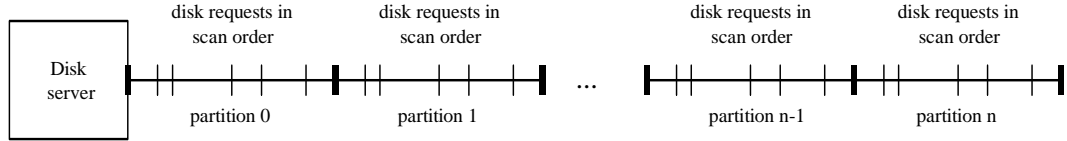


Figure 1: Disk queue organization for multi-partitioning

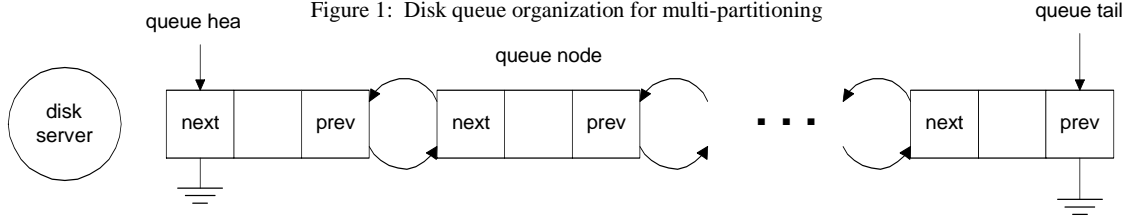


Figure 2: Double linked list representing the disk queue

cost of losing (violating the deadline) of few requests.

Previous work on continuous media servers has concentrated on retrievals in support of a continuous display. Hence, they have paid scant attention to the other applications that may involve both read and write disk requests. For example in [MSB98, SM982], it is assumed that most requests are read with deadlines (i.e., Rdl requests) and algorithms like EDF and SCAN-RealTime are used for disk scheduling.

In the context of multimedia systems, two classes of disk scheduling algorithms have been introduced. The first is termed cycle-based algorithms and the second is termed deadline driven algorithms[KNG00].

The cycle-based algorithms (e.g., Grouped Sweeping Scheduling [GKY93]), generally provide deterministic guarantees for continuous media streams. In this class, the disk bandwidth is pre-allocated to the continuous media streams. To handle traditional data (e.g., text and images) and interactive requests (e.g., video games) a two-level scheme was proposed [WR99, BBG99, SV98]. In general, the higher level corresponds to different application classes with different requirements (e.g., periodic requests with deadlines for continuous media data, interactive requests for video games, and aperiodic requests for traditional data). Some requests from different classes in the higher level are moved to the lower level and reordered to be serviced by the disk. In this class, the scheduler does not fully make use of types of the disk requests and does not have knowledge of all the pending disk requests in the higher level to give a global optimization. For example, it could postpone the service of an Rnn request to honor the deadline of a new Rdl request preemptively. Also, in these studies they did not consider the effect of having different types of write requests that compete for the buffer space.

The second class of algorithms, termed deadline driven, include EDF (Earliest Deadline First) [RW94], and SCAN-RT2 (SCAN-RealTime 2)[AKN97]. These algorithms provide statistical guarantees for the requests

as they are best-effort algorithms and do not guarantee meeting the deadline of all the requests. Generally, deadline driven scheduling algorithms provide higher disk bandwidth utilization than cycle-based disk scheduling algorithms.

EDF supports only Rdl disk requests, and SCAN-RT2 supports only Rdl and Wnn disk requests for storing digital video to the server and reading digital video for broadcasting. This is suitable only for video-on-demand systems, but does not satisfy the needs other multimedia applications.

The objective of this paper is to present a deadline driven disk scheduling algorithm that handles all the disk request types of table 1 efficiently. To the best of our knowledge, the techniques studied in this paper are novel and have not appeared in the literature.

### 3 Proposed Deadline Driven Disk Scheduling Algorithm (SCAN-RT3)

In this section we propose our deadline driven algorithm. It maintains all the disk requests in a single queue. When a new request arrives, the algorithm tries to insert the disk request such that: it is in scan order (to optimize the disk bandwidth), and its deadline and the deadlines of other pending requests are not violated.

#### 3.1 Disk Queue Organization

The algorithm maintains a single disk queue for all the types of disk requests rather than having a separate queue for each request type. This allows the algorithm to better optimize the disk bandwidth.

The algorithm handles disk request types that can afford to wait in the disk queue for a relatively long time. For example, Rnn disk requests, which do not have any deadlines, may afford to remain in the disk queue for a much longer time than Rdl disk requests, which may have stringent deadlines. This suggests the possibility of using several scan order partitions, such that in each partition the disk requests are kept in scan

Type	Deadline 1 (introduced by the bandwidth of the required multimedia stream)	Deadline 2 (introduced by the scheduling algorithm to avoid the overflow of the write buffer)	Deadline 3 (introduced by the system to avoid indefinite postponement and loss of data in the case of a system crash)
<b>Wdn</b>	Deadline enforced by the type of the multimedia stream	Computed by equation {1}	System write deadline
<b>Wnn</b>		Computed by equation {1}	System write deadline
<b>Rdn</b>	Deadline enforced by the type of the multimedia stream		System read deadline
<b>Wdl</b>	Deadline enforced by the type of the multimedia stream	Computed by equation {1}	System write deadline
<b>Rdl</b>	Deadline enforced by the type of the multimedia stream		System read deadline
<b>Rnn</b>			System read deadline

Table 2: Deadline computation for different disk request types

order (according to their cylinder numbers). This technique is called **multi-partitioning** as depicted in figure 1.

### 3.2 Data Structure

The algorithm uses a double linked list to implement the multi-partition disk queue. Each node of the double linked list represents one disk request as illustrated in figure 2.

A newly-arrived disk request is inserted anywhere in the disk queue, including both queue ends. The next disk request to be serviced is the disk request at the queue head. The double linked list is used because it is convenient for bi-directional traversal of queue nodes as well as for simple insertion, and deletion of queue nodes.

### 3.3 Priority Order of the Different Disk Request Types

The following priority order is used by the algorithm to schedule the disk requests.

(highest) Wdn, Wnn, Rdn, Wdl, Rdl, Rnn (lowest)

Generally, write disk requests are assigned a higher priority than read disk requests. Also, disk requests that can not be lost are assigned a higher priority than disk requests that can be lost. Finally, disk requests that have deadlines are assigned a higher priority than disk requests that do not have any deadlines.

Indefinite postponement is avoided by aging. This is accomplished by assigning a system wide deadline for all disk requests.

### 3.4 Computing the Deadline of a Disk Request

The deadlines assigned to the disk requests are soft deadlines because the continuous media server supports non-critical real-time tasks. The disk requests that need to be serviced have deadlines that depend on the

bandwidth of the continuous media stream used to generate these disk requests.

The write disk requests (types Wdn, Wnn, and Wdl) that are given to the continuous media server have to be buffered in the main memory buffers of the continuous media server before being written to the disk. All write disk requests compete for the buffer space. Thus, to avoid buffer overflow, a deadline for each write disk request is computed such that the buffer never overflows. The following equation is used to compute the deadline of a newly-arrived write disk request.

$$\text{deadline} = \text{present time} + (\text{free buffer space}) / (\text{arrival rate of write disk requests}) \quad \{1\}$$

Finally, the disk requests that have no deadlines (types Wnn and Rnn) are assigned an artificial deadline to prevent indefinite postponement, and avoid losing very old Wnn disk requests in the case of a system crash or power failure. This deadline is a system parameter and it is typically much less binding than other deadlines. The method of computing the deadlines of all types of disk requests is summarized in table 2 where the deadline of each disk request is assigned to the most binding deadline.

### 3.5 Algorithm

When a new request arrives, the algorithm tries to insert the request in scan order starting from the last scan order partition. If this insertion violates the deadline of the new request or the deadline of other pending requests in the request queue, the algorithm tries to insert the new request in the next partition. This continues till the partition at the queue head is reached. In this case and depending on the type of the request, the algorithm may append the request to the queue tail, reject the request and consider it lost, or insert the request in the queue and possibly violate the deadline of some disk request. When the deadlines of some disk requests are violated due to the insertion of a new disk request, the algorithm tries to reorder the requests or even reject some requests to meet the deadlines of the

remaining requests.

### Insertion of Rnn Disk Request

The algorithm tries to insert the Rnn disk request in scan order, starting from the last partition at the queue tail, such that this insertion does not cause the deadline of any pending disk request to be violated.

If this attempt fails, the Rnn disk request is appended to the end of the queue to form a new scan order partition.

### Insertion of Rdl Disk Request

The algorithm tries to insert the Rdl disk request in scan order, starting from the last partition at the queue tail, such that this insertion does not cause the deadline of any pending disk request to be violated.

If this attempt fails, the algorithm performs a series of attempts to insert the disk request in each scan order partition starting from the last partition at the queue tail.

After inserting the Rdl request in any partition, the deadlines of some disk requests would be violated. Hence, to meet the deadlines of these requests, the algorithm tries to select one or more *victim* requests of the type Rnn from locations between the queue head and the first request with a violated deadline.

The victim requests are *demoted* to the partitions near the queue tail. This takes place only if the demotion of the victim requests results in accommodating the requests whose deadlines were violated as well as the original Rdl disk request.

The algorithm uses a stack to store the original positions of the victim requests in the disk queue to undo the demotion of the victim requests if necessary.

If all these attempts fail, the Rdl disk request is rejected and considered *lost*.

### Insertion of Wdl Disk Request

The insertion of Wdl disk request is the same as the insertion of Rdl disk request.

### Insertion of Wdn Disk Request

The algorithm tries to insert the Wdn disk request in scan order starting, from the last partition at the queue tail, such that this insertion does not cause the deadline of any pending disk request to be violated.

If this attempt fails, the algorithm performs a series of attempts to insert the disk request in each scan order partition starting from the last partition at the queue tail.

After inserting the Wdn request in any partition, the deadlines of some disk requests would be violated. Hence, to meet the deadline of these requests, the algorithm tries either to *promote* them to the head of the queue or tries to select one or more *victim* requests of lower priority (types Rnn, Rdl, and Wdl) and *demotes* these lower priority requests towards the queue tail.

If it is still not possible to accommodate the newly-arrived Wdn disk request, then the algorithm rejects one (or at most two) Rdl or Wdl requests to accommodate the Wdn disk request.

### Insertion of Wnn and Rdn Disk Requests

The insertion of Wnn and Rdn disk requests is the same as the insertion of Wdn disk request.

## 4 Analysis of Algorithm SCAN-RT3

The search space of the problem is all the permutations of the  $n$  pending disk requests in the disk queue, which is  $(n!)$ . Thus, the algorithm is a best-effort heuristic and generally does not find the optimal solution. The algorithm employs loops that only traverse the disk queue. As there is a finite number of pending disk requests in the disk queue, the algorithm terminates in a finite number of steps. The algorithm inserts the disk requests in scan order and makes several attempts before rejecting a disk request that can be lost.

The measure of the time complexity of the algorithm is the number node accesses in the double linked list representing the disk queue. The time complexity of the algorithm is  $O(n^2)$ , where  $n$  is the number of pending disk requests stored in the disk queue at the time of inserting of a newly-arrived disk request. To insert a disk request in scan order such that its deadline and the deadlines of the pending disk requests are not violated,  $O(n)$  node accesses are needed. The reason for the  $O(n^2)$  node accesses is that  $O(n)$  node accesses are required to handle the disk requests whose deadlines are violated due to the insertion of the newly-arrived disk request. Table 3 summarizes the main features of the algorithm in comparison to other algorithms.

Algorithm	Applications supported	handled types	Complexity
EDF	Video-on-demand	Rdl	$O(n)$
SCAN-RT2	Video-on-demand and partial support for video editing	Rdl, Wnn	$O(n^2)$
Proposed Algorithm SCAN-RT3	Full support for all the above applications as well as those listed in table 1	Rdl, Wnn, Rnn, Wdn, Wdl, Rdn	$O(n^2)$

Table 3: Summary of relevant disk scheduling algorithms

## 5 Experiments and Results

In this section, the details of the experiments undertaken to evaluate the performance of the proposed algorithm are described.

### 5.1 System Model

Several architectures for continuous media servers were proposed (e.g. [GM98]). We based our experiments on a discrete-event simulator that models the PanaViSS II commercial video server used in [AKN97, AKG]. Users are connected to the server through a high-speed network, for example an ATM network. The video server has several disk storage units

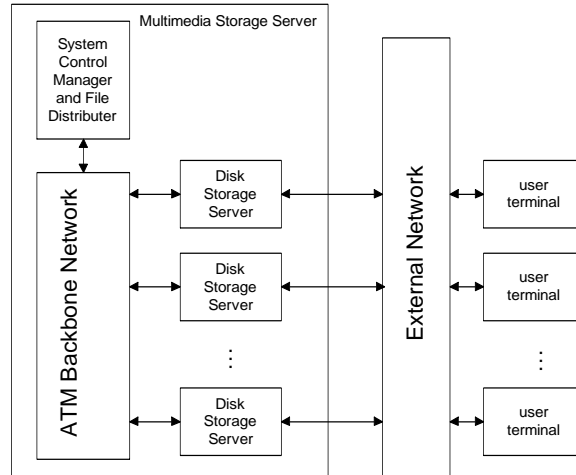


Figure 3: Architecture of the PanaViSS server

(about 15) that are fully interconnected through an ATM backbone network. The disk storage units are called disk storage servers. The number of users that can be connected to a disk storage server is around 80. So the total number of users connected to the video server is around 1200 users. The architecture of the server is outlined in figure 3.

The server supports MPEG-encoded video streams. Each stream is broken into fixed length pieces, termed media blocks. The size of a media block is the same as the size of a disk page. The media blocks for a given video stream are stored distributively through the whole file system of the video server to achieve load balancing and fault tolerance [GKR95, RW94]. The system control manager and file distributor is responsible for distributing both the media blocks throughout the whole file system, and updating the necessary metadata.

For video-on-demand, users are connected to the external network using set-top boxes that have the following functions: decoding MPEG-encoded video data, providing user interface for virtual VCR functions, and communicating with the disk storage server. For video editing, users are connected to the external network using multimedia workstations that enable the users to edit the video clips. Each disk storage server has one RAID (Redundant Array of Inexpensive /Independent Disks). The parameters of the type of the disk drive used are listed in table 4 [AKN97].

Disk Parameter	Value
Type	Quantum XP32150
No. of cylinders	3832
Tracks/Cylinder	10
Sector size	512 Bytes
Rotation Speed	7200 RPM
Average Seek	8.5 mSec
Seek cost function	$0.8 + 0.002372(d) + 0.125818(\sqrt{d})$
File Block Size	64 Kbytes
Transfer Speed	4.9 – 8.2 MBytes/sec
Disks per RAID	5 (4 data 1 Parity)

Table 4: Disk parameters

In this disk storage server, the data distribution scheme is coarse grained striping and the redundancy mechanism is declustered parity [GWH94]. The disk simulation is based on the analytic model presented in [RW94]. The simulated server provides access to MPEG1 steams. MPEG1 steams have a bandwidth of 1.5 Mbps. The system wide deadline for write disk requests and read disk requests, referred to in table 2, is selected as 10 seconds to prevent indefinite postponement, and to prevent the loss of data in the case of a system crash. For the types Wdn, Rdn, Wdl, and Rdl, the deadline introduced by the bandwidth of the multimedia stream (MPEG1 video stream) is 350 mSec as computed below.

$$(\text{disk page size})/(\text{multimedia bandwidth}) = (64k * 8) / (1.5 M) = 350 \text{ mSec}$$

The write buffer size is taken to be proportionate to the number of users who perform the writing operations, i.e., users who issue disk requests of the types Wdn, Wnn, and Wdl. Each disk has a separate write buffer.

The arrival of new users is modeled by a Poisson process. As the users access MPEG1 steams, each user issues a disk request every 350 mSec as established above. Thus, if a *time window* of 350 mSec is considered, each user must issue exactly one disk request in the time window. The Poisson process used to model the arrivals of new users is responsible for the distribution of the disk requests in the time window as shown in figure 4.

The performance metric for this system is the loss rate for Rdl and Wdl requests, and the response time (or delay time) for Wdn, Wnn, Rdn and Rnn requests. For example, in a video-on-demand system where all the requests are Rdl, we would be interested in finding the loss rate of the Rdl requests to find out the maximum

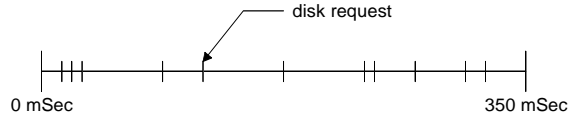


Figure 4: Window to generate the disk requests

number of users that can be supported simultaneously at a specific quality of service. As

can support up to 76 users, and algorithm SCAN-RT3 can support up to 77 users.

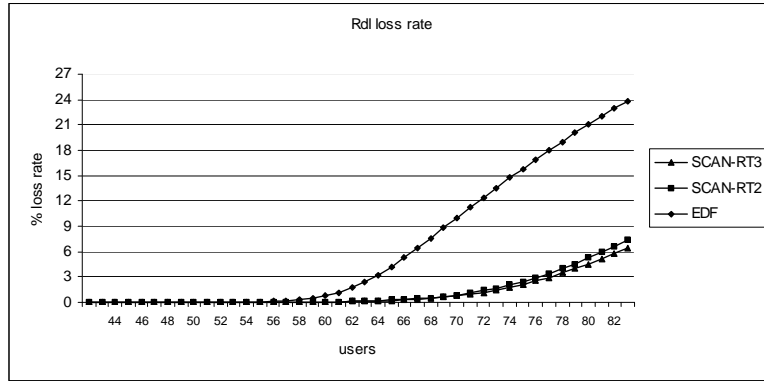


Figure 5: Rdl loss rate for algorithms EDF, SCAN-RT2, and SCAN-RT3

another example, in a video authoring system where the requests are Rnn (for downloading video clips) and Wnn (for uploading video clips), we would be interested in the response time which determines how much time the editors have to wait to download their video clips.

For each of the following experiments, the x-axis represents the number of users per disk storage server. The y-axis represents either the loss rate (as a percentage) of the disk requests or the average delay time (in seconds) taken to service a disk request.

The performance of the disk scheduling algorithms is assessed by comparing the loss rate of Rdl and Wdl disk requests, and the delay time of all the other types of the disk requests. The maximum number of users that can be supported at a 3% Rdl loss rate is considered for each algorithm. The 3% Rdl loss rate guarantees a quality of service for video-on-demand users such that only one frame of the thirty frames per second is lost.

### 5.2 Video-on-demand and Algorithms EDF, SCAN-RT2, and SCAN-RT3

Here we consider a video-on-demand environment. In this experiment, algorithms that handle Rdl disk requests are compared. The Rdl disk request type is used in video-on-demand applications, where all users are *viewers* and issue only Rdl disk requests. The difference between algorithms SCAN-RT2 and SCAN-RT3 is due to multi-partitioning only.

Figure 5 shows that at a 3% Rdl loss rate, algorithm EDF can support up to 63 users, algorithm SCAN-RT2

Therefore, SCAN-RT3 and SCAN-RT2 are superior to EDF because EDF does not try to optimize the disk bandwidth whereas both SCAN-RT2 and SCAN-RT3 reorder the requests to optimize the disk bandwidth. Also, in this case, SCAN-RT3 is marginally better than SCAN-RT2 due to the effect of multi-partitioning which gives more chances for a new Rdl request to be inserted in any of the existing partitions in the disk queue.

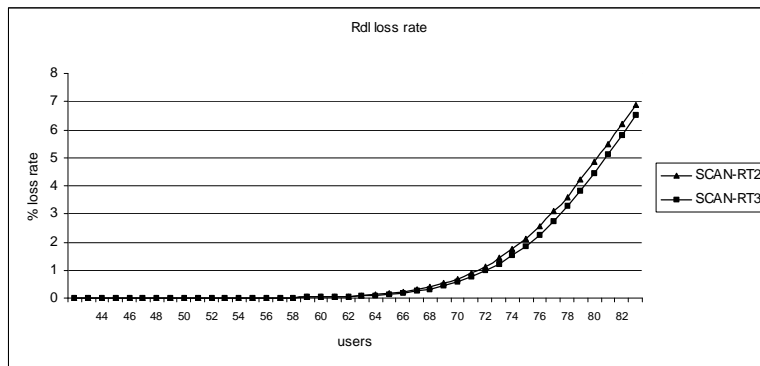


Figure 6: Rdl loss rate for algorithms SCAN-RT2 and SCAN-RT3

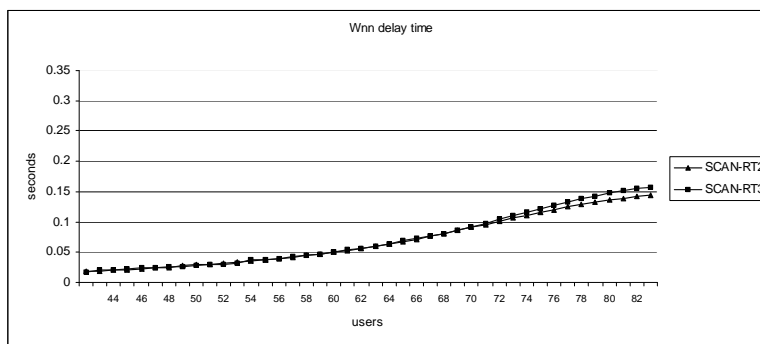


Figure 7: Wnn delay time for algorithms SCAN-RT2 and SCAN-RT3

### 5.3 Video-on-demand and Video Authoring, and Algorithms SCAN-RT2 and SCAN-RT3

In this experiment, algorithms which handle Rdl and Wnn disk requests are compared. The user mix ratio is Rdl : Wnn = 3 : 1, meaning that half users are *viewers* (issuing Rdl disk requests as in a video-on-demand application) and the other half are *editors* (issuing an equal number of Rdl and Wnn disk requests as in a video authoring application). We will assume the editors will issue Rdl requests rather than Rnn requests because algorithm SCAN-RT2 do not handle the latter type. Thus, the handling of Rdl and Wnn disk requests is only a *partial* support for combined video authoring and video-on-demand applications. The write buffer size is 80 disk pages per disk storage server. In this particular case, the difference between algorithm SCAN-RT2 and SCAN-RT3 is due to multi-partitioning only.

Figures 6 and 7 show that at a 3% Rdl loss rate, algorithm SCAN-RT2 can support up to 76 users at a Wnn delay time of 122 mSec, and algorithm SCAN-RT3 can support up to 77 users at a Wnn delay time of 134 mSec. Therefore, in this case SCAN-RT3 is marginally better than SCAN-RT2 at the cost of a slight increase in the Wnn delay time.

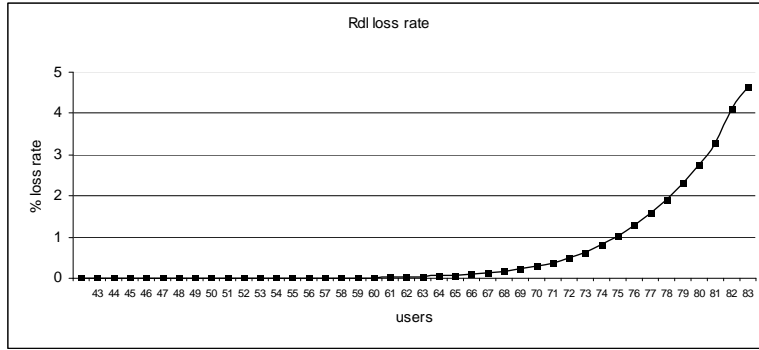


Figure 8: Rdl loss rate for algorithm SCAN-RT3

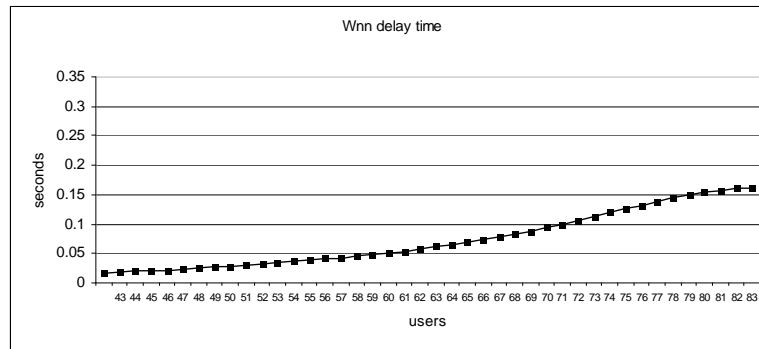


Figure 9: Wnn delay time for algorithm SCAN-RT3

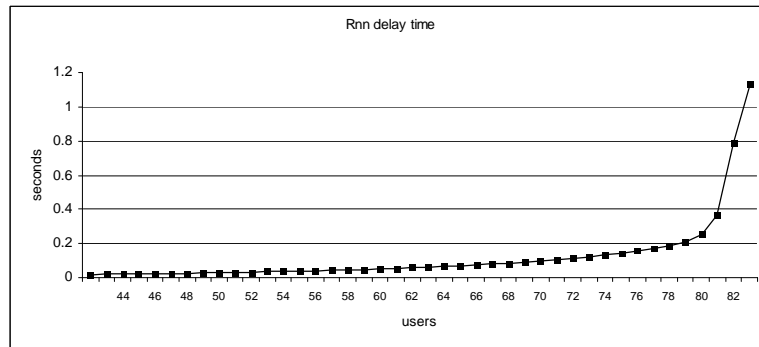


Figure 10: Rnn delay time for algorithm SCAN-RT3

## 5.4 Combined Video-on-Demand and Video Authoring Applications

In a video-on-demand application, users (*viewers*) issue Rdl disk requests. For a video authoring application, users (*editors*) issue Rnn and Wnn disk requests.

As editors buffer the media clips at their terminals, the number of the Rnn disk requests is expected to be equal to the number of the Wnn disk requests.

In the next subsection, we present the performance results when the algorithm SCAN-RT3 is used. Then in the following subsection, we compare both SCAN-RT3 (for request types Rdl, Wnn, and Rnn) and SCAN-RT2 (for request types Rdl, and Wnn only, where each Rnn request is replaced by a corresponding Rdl request).

### 5.4.1 Algorithm SCAN-RT3

The user mix ratio is viewers : editors = 1 : 1, which is Rdl : Wnn : Rnn = 2 : 1 : 1. The write buffer size is 80 disk pages per disk storage server.

Figures 8 through 10 show that at a 3% Rdl loss rate, algorithm SCAN-RT3 can support up to 80 users at a Wnn delay time of 154 mSec, and an Rnn delay time of 251 mSec.

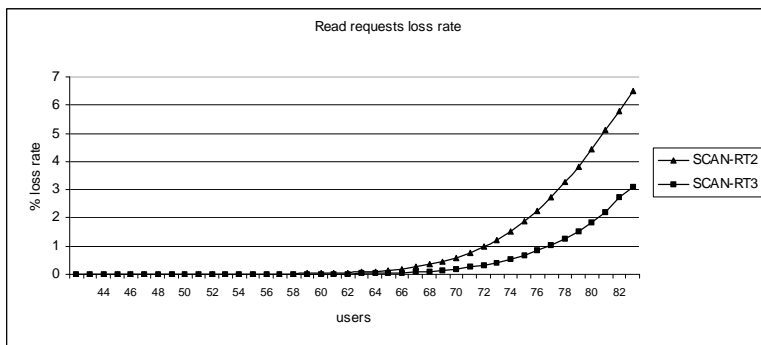


Figure 11: Read disk requests loss rate for algorithms SCAN-RT2 and SCAN-RT3

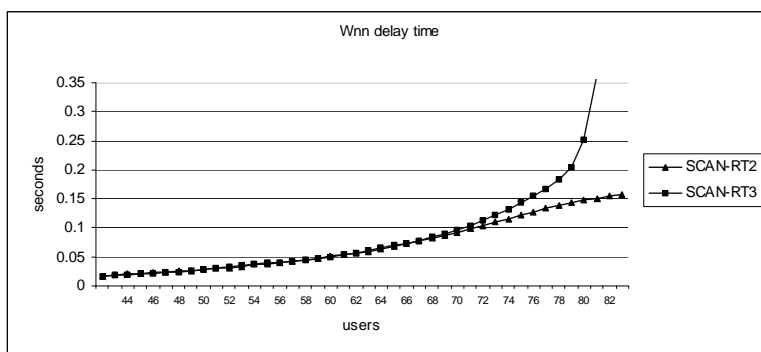


Figure 12: Wnn delay time for algorithms SCAN-RT2 and SCAN-RT3

#### 5.4.2 Algorithms SCAN-RT2 and SCAN-RT3

The user mix ratio is viewers : editors = 1 : 1 and the number of buffers is 80 per disk storage server. The first algorithm, SCAN-RT2, can only handle Rdl and Wnn disk requests. The second algorithm, SCAN-RT3, can handle Rdl, Wnn, Rnn disk requests. Thus, it fully supports all the required requirements by the two applications.

Figures 11 and 12 show that at a 3% read requests loss rate, algorithm SCAN-RT2 can support up to 77 users per disk storage server or 1155 users for the media server at a Wnn delay time of 146 mSec. Algorithm SCAN-RT3 can support up to 82 users per disk storage server or 1230 users for the media server at a Wnn delay time of 466 mSec.

The above figures reveal that algorithm SCAN-RT3 has lower loss rate for read requests than algorithm SCAN-RT2. Thus, algorithm SCAN-RT3 results in a lower loss rate for the read disk requests at the expense of a higher delay for the Wnn disk requests. Also, algorithm SCAN-RT3 is able to support the functionality required by the editors as they issue Rnn disk requests which can not be lost.

This experiment clearly shows the contribution of the proposed algorithm in this environment. (1) The algorithm SCAN-RT3 provides full functionality by satisfying the application requirements as video clips

are downloaded and uploaded without any data loss. In addition, the algorithm gives better performance. Thus, it is possible to support more users (about 6% more) by just using the proposed algorithm.

#### 5.5 Other General User Mix Ratios

In the next two subsection, we consider a setting where there are different application requirements. We will not compare the results of the algorithm to other algorithms because these algorithms do not meet the requirements of the applications.

##### 5.5.1 User Mix Ratio 1

This experiment simulates a setting where there is a mix of different applications like real-time recording, video-on-demand and video authoring. As an example, we will consider the high quality real-time recording of a football match where we can assume the user mix ratio to be *viewer : editor : high quality recording* = 8 : 1 : 1. This translates to Rdl : Wnn : Rnn : Wdn = 16 : 1 : 1 : 2. The write buffer size is 48 disk pages per disk storage server.

Figures 13 through 16 show that the algorithm can support up to 78 users at 3% Rdl loss rate with low delay times for all disk request types.

##### 5.5.2 User Mix Ratio 2

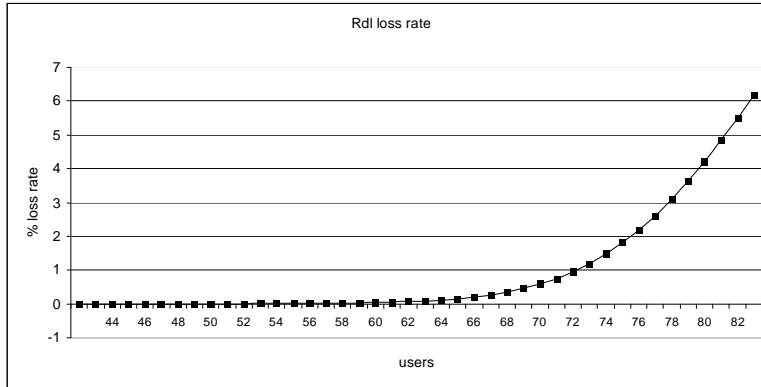


Figure 13: Rdl loss rate for user mix ratio 1

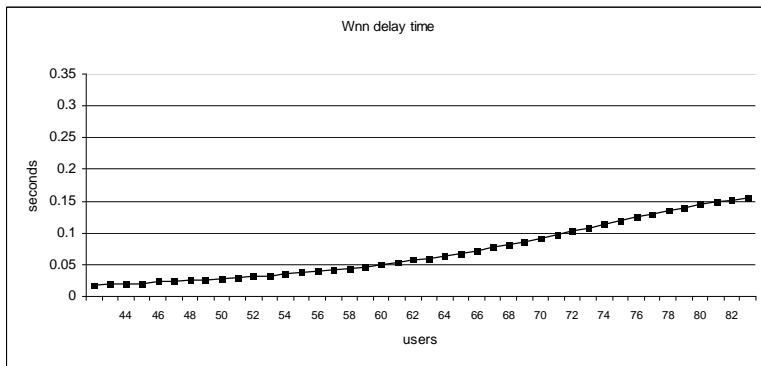


Figure 14: Wnn delay time for user mix ratio 1

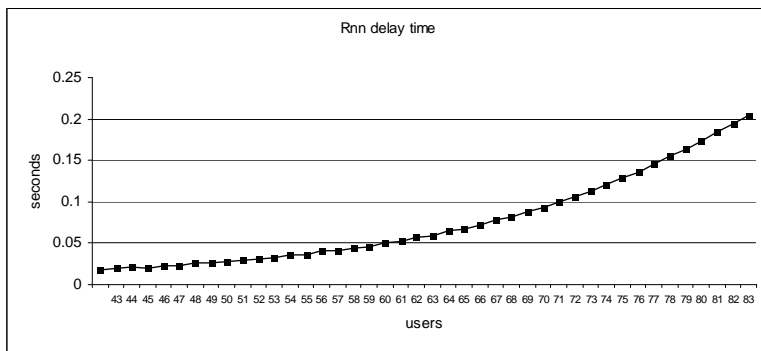


Figure 15: Rnn delay time for user mix ratio 1

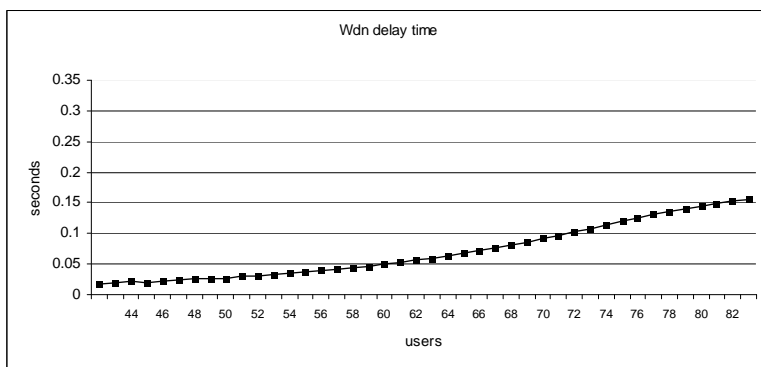


Figure 16: Wdn delay time for user mix ratio 1

In this experiment, we included all the types of the disk requests. We used a theoretical user mix ratio to show the performance of the algorithm when all the types of the disk requests depicted in table 1 are used. The user mix ratio is Rdl : Wnn : Rnn : Wdn : Wdl : Rdn = 1 : 1 : 1 : 1 : 1 : 1, and the write buffer size is 160 disk pages per disk storage server. We found that at most 63 users can be supported at a 0.0048% Rdl loss rate and 0.0046% Wdl loss rate. The delay times for Rdl, Rdn, Wnn, Rnn, Wdl, Wdn disk requests are 59.3, 59.6, 59.7, 60.4, 59.2, 59.7 mSec respectively.

## 6 Conclusions

This paper addresses the problem of disk scheduling for continuous media servers. The proposed algorithm handles all types of the disk requests, namely Wdn, Wnn, Rdn, Wdl, Rdl, and Rnn. The algorithm employs the multi-partitioning technique for the disk queue organization. Multi-partitioning is used in order to maintain several scan order partitions in the disk queue. The simulation study was conducted to assess the performance of the algorithm in comparison to other disk scheduling algorithms for continuous media servers. The simulation results indicate that the algorithm handles all the disk request types efficiently for different user mix ratios.

## References

- [AKG] W. Aref, I. Kamel, S. Ghandeharizadeh, "Disk Scheduling in Video Editing Systems", IEEE Transactions on Knowledge and Data Engineering, Accepted, To appear.
- [AKN97] W. Aref, I. Kamel, N. Niranjana, and S. Ghandeharizadeh, "Disk Scheduling for Displaying and Recording Video in Non-Linear News Editing Systems", International Conference in Multimedia and Computer Networks, San Jose, California, February 1997.
- [AMJ95] R. Alonso, V. Mani, S. Johnson, and Y-L Change, "Performance study of disk scheduling algorithms in a video server", Technical Report MITL-TR-131-94, Matsushita Information Technology Laboratory, February 1995
- [BBG99] J. Bruno, J. Brustoloni, E. Gabber, B. Ozden, A. Silberschatz, "Disk Scheduling with Quality of Service Guarantees", International Conference on Multimedia Computing and Systems, June 1999.
- [DSS94] A. Dan, D. Sitaram, and P. Shahabuddin "Scheduling policies for an on-demand video server", Proceedings of the ACM Multimedia Conference, October 1994.
- [FW95] C. Freedman, and D. DeWitt, "The SPIFFI scalable video-on-demand system", Proceedings of the ACM SIGMOD Conference, 1995.
- [GKR95] D. Gemml, H. Vin, D. Kandlur, P. Rangan, and L. Rowe, "Multimedia Storage Servers: A Tutorial", IEEE Computer, May 1995.
- [GKY93] M. Chen, D. Kandlur, and P. Yu, "Optimization of Grouped Sweeping Scheduling (GSS) with Heterogeneous Multimedia Systems", First ACM International Conference on Multimedia, August 1993.
- [GM98] S. Ghandeharizadeh, and R. Muntz, "Design and Implementaion of Scalable Continuous Media Servers", Parallel Computing, June 1998.
- [GWH94] G. Ganger, B. Worthington, R. Hou, and Y. Patt, "Disk Arrays: High-Performance, High-Reliability Storage Subsystems", IEEE Computer, March 1994.
- [GZS97] S. Ghandeharizadeh, R. Zimmermann, W. Shi, R. Rejaie, D. Lerardi, and T. Li, "Mitra: A Scalable Continuous Media Server", Kluwer Multimedia Tools and Applications, January 1997.
- [KAA99] M. Khan, A. Ghafoor, and M. Ayyaz, "Design and Evaluation of Disk Scheduling Policies for High-Demand Multimedia Servers". Proceedings of the 15<sup>th</sup> International Conference on Data Engineering, March 1999
- [KNG00] I. Kamel, T. Niranjana, and S. Ghandeharizadeh, "A Novel Deadline Driven Disk Scheduling Algorithm for Multi-Priority Multimedia Objects", International Conference on Data Engineering, San Diego, February 2000
- [MSB98] R. Muntz, J. Santos, and S. Berson, "A Parallel Disk Storage System for Realtime Multimedia Applications", International Journal of Intelligent Systems, Special Issue on Multimedia Computing System, December 1998.
- [RW94] A. Reddy, and J. Wylie, "I/O Issues in a Multimedia System", IEEE Computer, March 1994.
- [RW94] C. Ruemmler, and J. Wilkes, "An Introduction to Disk Drive Modeling", IEEE Computer, March 1994.
- [SM98] J. Santos, and R. Muntz, "Performance Analysis of the RIO Multimedia Storage System with Heterogeneous Disk Configurations", Proc. Of the 6<sup>th</sup> ACM International Multimedia Conference, 1998
- [SV98] P. Shenoy and H. Vin, "Cello: A Disk Scheduling Framework for Next-generation Operating Systems", ACM SIGMETRICS'98, 1998
- [WR99] R. Wijayarathne and N. Reddy, "Integrated QoS Management for Disk I/O", IEEE Multimedia Systems, June 1999.