WRR-SCAN: A Rate-Based Real-Time Disk-Scheduling Algorithm

Cheng-Han Tsai
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C
TEL: +886-3-574-2965
FAX: +886-3-572-3694
chtsai@cs.nthu.edu.tw

Edward T.-H. Chu
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C
TEL: +886-3-574-2965
FAX: +886-3-572-3694
edwardchu@cs.nthu.edu.tw

Tai-Yi Huang
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C
TEL: +886-3-574-2965
FAX: +886-3-572-3694
tyhuang@cs.nthu.edu.tw

ABSTRACT

Traditional real-time disk-scheduling algorithms service real-time tasks according to their deadlines. Such a priority-based algorithm, although satisfying real-time constraints, yields low disk utilization due to the excessive disk-seek time. Furthermore, it results in prolonged response time or even starvation for aperiodic tasks. In this paper, we propose a novel rate-based real-time disk-scheduling algorithm called WRR-SCAN (Weighted-Round-Robin-SCAN). WRR-SCAN guarantees to meet the deadline of a real-time task by reserving disk bandwidth according to its real-time constraints. WRR-SCAN services scheduled tasks in scan order to minimize the disk-seek time. In addition, WRR-SCAN delivers better response time for aperiodic tasks which are served in best-effort manner by priority-based algorithms. We conducted a set of extensive experiments to compare WRR-SCAN and SCAN-EDF, a priority-based algorithm studied extensively in literature. The experimental results show that WRR-SCAN reduces non-transmission overhead significantly and produces a guaranteed minimum data rate for aperiodic tasks while keeping the deadlines of real-time tasks.

Categories and Subject Descriptors

D.4.2 [**Operating Systems**]: Storage Management; D.2.8 [**Software Engineering**]: Metrics—*performance measures*

General Terms

Algorithms, Performance

Keywords

Real-time disk-scheduling, weighted round-robin, real-time multimedia servers

EMSOFT'04, September 27–29, 2004, Pisa, Italy.

1. INTRODUCTION

A real-time disk-scheduling algorithm is essential for a system where applications [2, 10, 15] issue disk I/O requests with real-time constraints. A good example is a real-time multimedia server [13, 4] that delivers isochronous data of disk I/O for each application to guarantee smooth playback. Because each application may request a different data rate and the requested data may scatter around the disk surface, the server requires a real-time disk-scheduling algorithm to schedule the available disk bandwidth to meet the real-time constraint of each application. Failing to deliver the data rate required by each real-time application may result in either buffer overflow or jittered playback. In addition, the disk-scheduling algorithm needs to provide an acceptable level of response time for aperiodic I/O tasks such as interactive or paging requests. Furthermore, to increase the disk utilization, the disk-scheduling algorithm needs to minimize its non-transmission overhead, *i.e.,* the disk-seek time and the disk-rotation time.

The SCAN algorithm has been proven to be an optimal disk-scheduling algorithm in minimizing the disk-seek time [7, 6, 1]. In SCAN, the disk head scans the disk surface and services tasks whose cylinder position falls right under the disk head. However, because the real-time constraints of tasks are not considered, SCAN cannot guarantee to meet the deadline of each task [22, 21, 3, 19]. Recently, there are several approaches that modify SCAN to take into account real-time constraints. These priority-based algorithms include SCAN-EDF [23, 24], SCAN-RT [12], and DM-SCAN [5]. Generally speaking, such a priority-based algorithm services tasks in EDF (earliest-deadline-first) order, an optimal preemptive real-time scheduling algorithm [17]. However, tasks with similar deadlines are serviced in scan order to reduce disk-seek time if none of the tasks misses its deadline. Although satisfying the real-time constraints, these priority-based algorithms degenerate to an EDF algorithm in most cases. An EDF-based disk scheduling algorithm yields low disk utilization due to the excessive disk-seek time. In addition, it delivers prolonged response time or even starvation for aperiodic tasks.

In this paper, we present WRR-SCAN (Weighted-Round-Robin-SCAN), a rate-based real-time disk-scheduling algorithm. WRR-SCAN schedules real-time tasks in midst of aperiodic tasks to deliver isochronous data for real-time tasks

and provide a guaranteed minimum data rate for aperiodic tasks. We first divide the disk bandwidth into *rounds.* During each round, the disk head moves in one direction (either inward or outward) and services tasks in scan order. We next assign each real-time task a *weight,* the maximum amount of disk-transmission time the disk head will service this task in one round. In addition, we create a virtual aperiodic server and assign this server a predetermined weight. Aperiodic tasks are queued and serviced by this server. The sum of all weights and non-transmission overhead must be less than or equal to the round length to meet the deadline of each real-time task. By reserving the disk bandwidth required for each real-time task, we guarantee its specified minimum data rate with bounded maximum response time. Furthermore, because tasks are serviced in scan order, non-transmission overhead is reduced, and therefore, the disk utilization is increased.

To demonstrate the effectiveness of WRR-SCAN, we conducted a set of experiments to compare the performance of WRR-SCAN and SCAN-EDF, a traditional priority-based disk-scheduling algorithm that has been studied extensively in literature [23, 24, 12, 5]. We evaluate each algorithm in terms of its total disk-seek distance, normalized disk-idle time, normalized non-transmission overhead, disk utilization, and throughput and response time of aperiodic tasks. The experimental results show that, by servicing each task in scan order with a reserved weight, WRR-SCAN significantly reduces the disk-seek time, increases the disk utilization, and delivers an acceptable level of response time for aperiodic tasks, in addition to keeping the deadlines of real-time tasks.

The rest of the paper is structured as follows. Section 2 describes the task model. Section 3 describes the WRR-SCAN disk-scheduling algorithm. The experimental results that compare WRR-SCAN and SCAN-EDF are presented in Section 4. Section 5 discusses related work. Finally, Section 6 concludes this paper.

2. TASK MODEL

There are two types of commonly-used real-time disk I/O tasks, rate-based tasks and deadline-based tasks. A rate-based I/O task demands a minimum rate of isochronous data to satisfy its real-time constraints. A deadline-based I/O task, represented by the total number of bytes to be transferred and its deadline, requires the disk controller to transfer the requested bytes before its deadline. Rate-base tasks are issued by real-time multimedia applications to support continuously smooth playback [20, 8]. On the other hand, deadline-based tasks are often included in embedded software to carry about disk I/O in a hard-real-time embedded system [16, 11].

Our task model denotes a rate-based task τ_i as (S_i, L_i), where S_i is its release time and L_i is the requested minimum data rate. In addition, we convert a deadline-based task into a rate-based task by setting its requested data rate as the total requested bytes divided by its deadline. Figure 1 shows the WRR-SCAN system architecture where the operating system allocates for each task τ_i a data buffer sized of B_i. The disk controller reads data from the disk to this buffer and, once filled, the operating system is notified to retrieve data from this buffer. Consequently, we further convert each task τ_i into a sequence of periodic jobs $\tau_{i,k} = (S_{i,k}, P_i, B_i)$, where $S_{i,k}$ is the release time of the k-th job and P_i is the

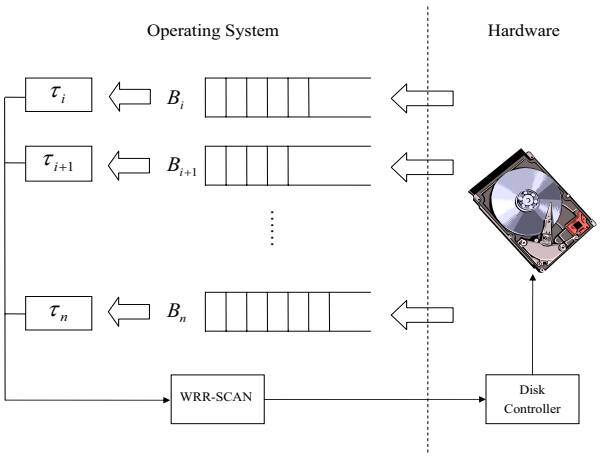

Figure 1: The WRR-SCAN system architecture

period and the relative deadline of each job, $S_{i,1} = S_i$ and $P_i = B_i/L_i$. In other words, the scheduler satisfies the requested minimum data rate L_i of τ_i by filling its data buffer at the rate of B_i bytes every P_i units of time. To prevent buffer overflow, each job $\tau_{i,k}$ is released at the beginning of its period. That is, $S_{i,k} = (S_i + (k-1) * P_i)$.

The buffer size B_i determines how efficiently WRR-SCAN services this task in each round. Because of $P_i = B_i/L_i$, a larger buffer results in a larger period that leads to less percentage of non-transmission overhead. On the other hand, a system requires minimum memory to prevent thrashing and deliver acceptable performance. A simple allocation scheme is to reserve minimum memory for system activities and allocate the rest of memory for rate-based I/O activities. Because WRR-SCAN benefits from similar period lengths, each rate-based task is next allocated a buffer according to its minimum data rate. In other words, a task with a larger L_i is allocated a buffer of larger B_i.

Without loss of generality, our task model adopts a common disk model presented in [25]. Let C denote the number of bytes in one *block*, the smallest unit of data to be read or written by the disk head. The disk model allows multiple zones of tracks. Tracks on different zones may have different numbers of blocks. We use E to denote the number of blocks in the smallest track. We assume that each B_i is a multiple of blocks. That is, $B_i = b_i * C$, where b_i is a positive integer. We use V to denote the maximum number of bytes serviced by the disk head per time unit, or the disk-transmission speed.

The disk model [25] uses a linear function of seek-distance to calculate the seek time for long seeks. Instead, the seek time for short seeks is modeled as a square-root function of seek-distance. Because the linear function dominates the square-root function at the same seek-distance, we use the linear function below to bound the disk-seek time

$$a \times d + t_e,$$

where a is a constant, d is the seek-distance, and t_e is the time to settle the disk head at the desired track. Let t_s denote the maximum time to seek the disk arm from one end of the disk to the other end of the disk, excluding the disk-settlement time. Obviously, t_s equals to a times the number of tracks of the disk. Furthermore, let t_r denote the sum

	Definition
τ_i	a real-time disk I/O task
$\tau_{i,k}$	the k-th job of τ_i
S_i	the release time of τ_i
$S_{i,k}$	the release time of $\tau_{i,k}$
P_i	the period of τ_i
B_i	the number of requested bytes per period of τ_i
C	the number of bytes in one block
E	the umber of blocks in the smallest track
V	the disk-transmission speed
t_r	the one-round disk-rotation time plus the disk-settlement time
t_s	the maximum one-trip disk-seek time
R	the round length
W_i	the weight of τ_i
W_a	the weight of the aperiodic server
$\mathcal{A}$	the disk-rotation time of R
q_i	the number of times τ_i is serviced in one period
n	the number of real-time tasks
b_i	the number of blocks in B_i
x_i	the number of blocks transferred in the length of W_i
x_a	the number of blocks transferred in the length of W_a
T_h	the time between the release of a job and the first full round
T_t	the time between the deadline of a job and the last full round

Table 1: List of notations and definitions

of the time to rotate the disk once and t_e. To simplify the presentation, we use disk-rotation time to include both disk-rotation and disk-settlement time. For ease of reference, Table 1 lists the definitions of notations used in the rest of this paper.

3. WRR-SCAN ALGORITHMS

The WRR (Weighted Round-Robin) algorithm has been applied extensively to provide QoS (Quality-of-Service) control in a system consisting of both periodic and aperiodic tasks [9, 26, 14]. In general, WRR partitions the available time (bandwidth) of a service into rounds. Each periodic task is then assigned a weight that indicates the maximum amount of time the task receives the service in a single round. The sum of weights of all tasks must be less than or equal to the round length. The server services each task in a round-robin order during each round. By adjusting the weight of a task, we control the QoS of the task accordingly.

The WRR-SCAN algorithm applies the WRR algorithm on top of the SCAN disk-scheduling algorithm to schedule real-time disk I/O tasks in midst of aperiodic tasks such as interactive and paging tasks. The WRR-SCAN disk-scheduling algorithm first partitions the disk bandwidth into rounds, and let R denote the round length. During each round, the disk head moves from one end of the disk to the other end of the disk (either inward or outward) and services tasks in scan order. On the next round, the direction of the disk-head movement is reversed. A round length R can be decomposed into three components: the disk-seek time, the disk-rotation time, and the disk-transmission time. Because

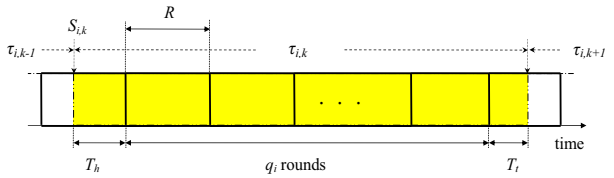

Figure 2: The relationship between the round length and the period

tasks are serviced in scan order, the disk-seek time of R is bounded by t_s. We next assign each real-time task a weight W_i that bounds the maximum amount of disk-transmission time the disk head will service this task in a single round. We note that W_i excludes any disk-seek and disk-rotation time. Consequently, the disk-transmission time of R is the sum of W_i of all tasks. Let $\mathcal{A}$ bound the disk-rotation time of R. When there is no aperiodic tasks, WRR-SCAN guarantees the requested date rate of each real-time task if

$$t_s + \mathcal{A} + \sum_{i=1}^{n} W_i \leq R,$$

where n denotes the number of real-time tasks in the system.

We create a virtual aperiodic server when aperiodic tasks exist in the system. We assign this aperiodic server a weight W_a. Incoming aperiodic tasks are queued and serviced by the server. By reserving W_a disk-transmission time in each round, WRR-SCAN provides a minimum data rate for aperiodic tasks and avoids starvation. The schedulability test for real-time tasks is modified accordingly as shown below

$$t_s + \mathcal{A} + \sum_{i=1}^{n} W_i + W_a \leq R. \quad (1)$$

In order to service tasks in scan order, WRR-SCAN needs to be aware of the disk-data location of each task. Such information is also essential in determining the bound on $\mathcal{A}$. In the following, we first describe the default WRR-SCAN algorithm, denoted by WRR-SCAN-R, designed for a system where the disk-data location of a task is randomly scattered. We next describe the WRR-SCAN-C algorithm that reduces the bound on $\mathcal{A}$ by applying a simple data-compaction scheme to place data on contiguous blocks. Finally, we present the WRR-SCAN-CA algorithm that further enhances WRR-SCAN-C by aligning release times of periodic jobs with rounds and allowing a larger round length.

3.1 WRR-SCAN-R

Weight Assignments

To determine the weight W_i of τ_i, we first calculate q_i, the number of times τ_i is serviced in one period. Figure 2 shows the execution timeline of a job $\tau_{i,k}$, where T_h denotes the time between $S_{i,k}$ and the beginning of the first full round and T_t denotes the time between the end of the q_i-th full round and the end of the period P_i. Obviously, $0 \leq T_h < R$ and $0 \leq T_t < R$. In addition,

$$P_i = q_i * R + (T_h + T_t).$$

In the worst-case scenario, the disk head misses the track in the round of T_h and the period P_i expires before the disk head reaches the track in the round of T_t. In other words,

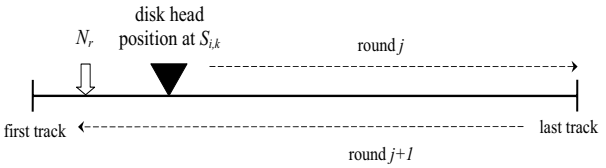

Figure 3: The worst-case response time of $\tau_{i,k}$

$\tau_{i,k}$ is serviced at least q_i times. Because $T_h + T_t < 2R$, we have

$$q_i = \frac{P_i - (T_h + T_t)}{R} > \frac{P_i}{R} - 2.$$

Therefore,

$$q_i = \lfloor \frac{P_i}{R} \rfloor - 1. \quad (2)$$

Since the disk controller transfers data in the unit of blocks, we use x_i to denote the number of blocks transferred in the length of W_i. The number of blocks transferred in P_i must be equal to or larger than b_i. Therefore, $x_i * q_i \geq b_i$. Accordingly, we obtain

$$W_i = \frac{x_i * C}{V} = \frac{C}{V} * \lceil \frac{b_i}{q_i} \rceil. \quad (3)$$

In addition, let x_a denote the number of blocks we reserve for aperiodic tasks in each round. Similarly, we can compute W_a by

$$W_a = \frac{C}{V} * x_a. \quad (4)$$

Disk-Rotation Time

When the disk-data locations of tasks are randomly scattered, the data blocks of the same task that are transferred in one round may be located in different tracks. The total number of blocks transferred in one round is

$$\sum_{i=1}^{n} \lceil \frac{b_i}{q_i} \rceil + x_a.$$

In the worst-case scenario, each of these blocks is located in a different track and reading each block incurs t_r disk-rotation time. Therefore, we can bound $\mathcal{A}$ by

$$\mathcal{A} \leq t_r * (\sum_{i=1}^{n} \lceil \frac{b_i}{q_i} \rceil + x_a). \quad (5)$$

Round Length

In addition to the lower bound of R given in Eq. (1), we also need to give an upper bound of R to guarantee that each task be serviced at least once in its period. Let a job $\tau_{i,k}$ demand a disk read starting at track N_r. Figure 3 shows the response time between $S_{i,k}$ and the time when the disk head visits track N_r. Let Δ denote the worst-case response time. Because WRR-SCAN services disk requests in scan order, we have $\Delta < 2R$. Consequently, to guarantee τ_i be serviced at least once in its period, P_i must be larger than or equal to the worst-case response time Δ. In other words, we have

$$R \leq 1/2 * P_i. \quad (6)$$

In summary, the round length R is bounded by

$$t_s + (t_r + \frac{C}{V}) * (\sum_{i=1}^{n} \lceil \frac{b_i}{q_i} \rceil + x_a) \leq R \leq 1/2 * P_i. \quad (7)$$

The WRR-SCAN-R disk-scheduling algorithm works in the following steps. We first determine x_a and W_a, the disk-transmission bandwidth reserved for the aperiodic server in each round. Based on the periods of all real-time tasks, we next choose the largest R allowed by Eq. (7). Once R is known, we then calculate q_i and W_i. If Eq. (7) is satisfied with the current selections of W_i and W_a, WRR-SCAN-R guarantees the minimum data rate specified by each real-time task and aperiodic tasks have an acceptable level of response. Otherwise, we need to choose a smaller R or reduces the bandwidth reservation for the aperiodic server and repeat the selection process. On the other hand, Eq. (7) is served as an admission control when a new real-time task is issued. The new real-time task is admitted and serviced only if it passes the acceptance test by Eq. (7). Finally, in most cases, the actual disk-rotation time would be much less than the worst-case reservation made in Eq. (7). We can allocate this unused bandwidth to the aperiodic server for better throughput and response time.

3.2 WRR-SCAN-C

To reduce the disk-bandwidth reservation on $\mathcal{A}$, we apply a simple data-compaction scheme to place the data of the same task in contiguous blocks. Reading a segment of contiguous blocks incurs only one disk-rotation time. As a result, such a compaction scheme reduces the disk-rotation overhead to one t_r per real-time task in each round. The data-compaction scheme can be easily deployed on multimedia systems whose file set contains large multimedia files. Such a multimedia file is often read-only and rarely changed. Consequently, placing data of a multimedia file in contiguous blocks is an effective approach for reducing disk-rotation overhead. On the other hand, the aperiodic server may service different tasks in one round, and these x_a blocks may still be located in different tracks. Accordingly, we modify the bound on $\mathcal{A}$ to

$$\mathcal{A} \leq t_r * (n + x_a).$$

By following the same calculations for W_i and W_a, we obtain a new bound on R

$$t_s + t_r * (n + x_a) + \frac{C}{V} * (\sum_{i=1}^{n} \lceil \frac{b_i}{q_i} \rceil + x_a) \leq R \leq 1/2 * P_i. \quad (8)$$

3.3 WRR-SCAN-CA

The calculations of q_i and W_i in Eqs. (2) and (3) are pessimistic and conservative. Such conservative calculations, although satisfying the real-time constraints, lead to over-reservations of the disk bandwidth for τ_i. Specifically, in the length of a period P_i, there are at most two rounds in which the weight W_i is reserved but not used. As a result, the disk bandwidth is not fully utilized.

To reduce the over-reservations of the disk bandwidth, we use the technique of release-time alignments to release every job at the beginning of a round, as shown in Figure 4. Such alignments are done in the conversion of a rate-based task into a sequence of periodic jobs. The new release time is reduced by $(R - T_h)$ to be a multiple of R. On the other

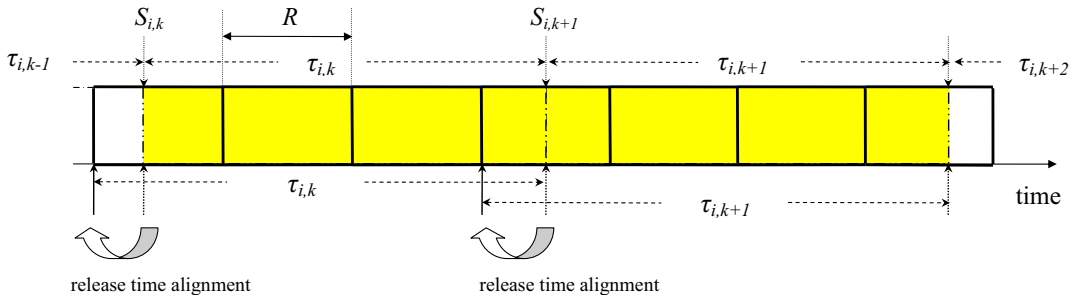

Figure 4: The illustration of release-time alignments

hand, the period and relative deadline is increased by $(R - T_h)$ to keep the same absolute deadline. Accordingly,

$$P_i' = P_i + (R - T_h) = q_i' * R + T_t,$$

and

$$q_i' = \frac{P_i' - T_t}{R} \geq \frac{P_i - T_t}{R} > \frac{P_i}{R} - 1.$$

As a result,

$$q_i' = \lfloor \frac{P_i}{R} \rfloor = q_i + 1.$$

Finally, we obtain the new weight assignment W_i' of τ_i

$$W_i' = \frac{C}{V} * \lceil \frac{b_i}{q_i + 1} \rceil. \quad (9)$$

The technique of release-time alignments may overlap the first round of a job with the last round of a previous job. To prevent buffer overflow, we need to increase the buffer size by at most $\lceil \frac{b_i}{q_i+1} \rceil$ blocks. In addition, because a job will be serviced at the first round after release-time alignments, the worst-case response time of $\tau_{i,k}$ shown in Figure 3 is now less than or equal to R. Therefore, we obtain a higher upper bound on R, $R \leq P_i \leq P_i'$. By following the same calculations of $\mathcal{A}$ given in WRR-SCAN-C, we have a new lower and upper bound on R

$$t_s + t_r * (n + x_a) + \frac{C}{V} * (\sum_{i=1}^{n} \lceil \frac{b_i}{q_i + 1} \rceil + x_a) \leq R \leq P_i. \quad (10)$$

The technique of release-time alignments reduces the lower bound on R. Consequently, given the same R, this technique allows WRR-SCAN-CA to schedule more real-time tasks at their desired data rates and, therefore, increase the disk utilization. On the other hand, the higher upper bound on R allows a larger round length that in turn allows a larger weight assignment and reduces the percentage of non-transmission overhead.

4. EXPERIMENTAL RESULTS

We conduct a set of experiments to compare the performance of WRR-SCAN and SCAN-EDF, a priority-based real-time disk-scheduling algorithm. Each algorithm is evaluated in terms of its total disk-seek distance, normalized disk-idle time, disk overhead, disk utilization, throughout and response time of aperiodic tasks. We implemented all three versions of WRR-SCAN and carefully analyze the effects of data-compaction and release-time alignment. Because a data-compaction scheme will considerably improve the performance of a disk-scheduling algorithm, as shown in the experimental results, we measure the performance of SCAN-EDF with compact data to demonstrate the effectiveness of WRR-SCAN.

Sector size		512 bytes
C		4096 bytes
Number of cylinders		1962
Tracks per cylinder		19
E		9
Revolution speed		4002 RPM
t_r		23 ms
t_s		15.7 ms
V		2.8 MB/s
Seek time	short(ms)	$3.24 + 0.400\sqrt{d}$
	long(ms)	$8.00 + 0.008d$
	boundary	$d = 383$

Table 2: The HP97560 disk model

The experiments are based on HP97560, a commonly-used disk model [25] whose parameters are given in Table 2. We simulate a workload with a set of real-time tasks and aperiodic tasks. The period P_i of a real-time task is a random variable of normal distributions with a 5000-ms mean and a 1000-ms variance. Similarly, the buffer size B_i of a real-time task is another random variable of normal distributions with a 56-KB mean and a 28-KB variance. On the other hand, the interarrival time of aperiodic tasks is controlled by an exponential random variable with a 35-ms mean to model frequent arrivals of background activities. We adjust the workload by increasing the number of real-time tasks in a task set. To simulate the characteristics of short I/O, we assume that each aperiodic task reads one block of data. The SCAN-EDF algorithm services an aperiodic task only when there are no periodic jobs. In the WRR-SCAN algorithms, all aperiodic tasks are served by a virtual aperiodic server. We reserve a fixed percentage of disk bandwidth for the aperiodic server in WRR-SCAN in order to compare fairly the three versions of WRR-SCAN.

Total Disk-Seek Distance

Figure 5 compares the four disk-scheduling algorithms at the total disk-seek distance. We apply the same workload to each algorithm for a fixed amount of time, and measure the total number of tracks the disk head seeks. The number of real-time tasks in a workload ranges from one task to 91 tasks. SCAN-EDF generates the most seek-distance which increases as the workload increases. This behavior results

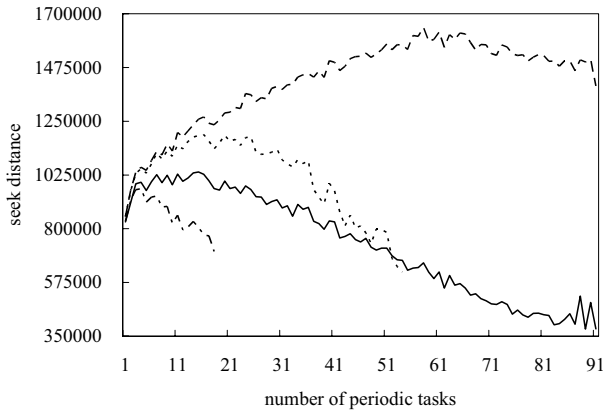

Figure 5: The total disk-seek distance

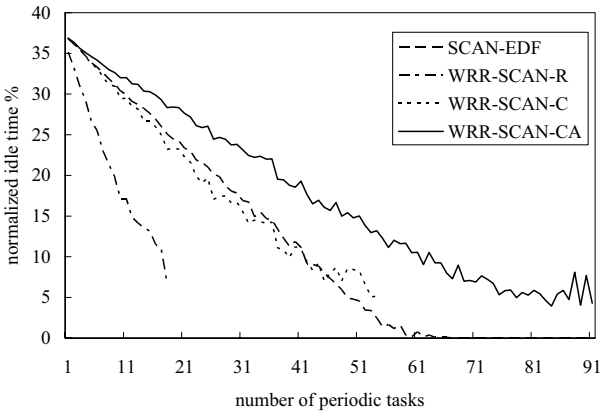

Figure 6: The normalized disk-idle time

from the EDF scheduling that vibrates the disk head to meet the deadline of each real-time task as well as serve aperiodic tasks. When the number of real-time tasks is less than 60, the workload remains relatively light such that the aperiodic server queue is empty most of the time.

In other words, most aperiodic tasks are serviced right after its release. When the number of real-time tasks is larger than 60, aperiodic tasks start to queue up, which we will show in Figure 11, as most of the disk bandwidth is taken by real-time tasks. Because serving one more real-time task will queue several aperiodic tasks due to the difference of each read size, the total disk-seek distance drops when the number of real-time tasks increases, as shown in Figure 5.

Because WRR-SCAN services tasks in scan order, its total disk-seek distance is significantly smaller than SCAN-EDF's, especially when the workload is heavy. At a light workload, only a small portion of a round is used. To increase the throughput and response time for aperiodic tasks, we reclaim unused disk bandwidth by starting a new round at the completion of all requests in the current round, instead of idling the disk head for the rest of the round. Because aperiodic tasks arrive more frequently than real-time tasks, the bandwidth reclamation approach causes WRR-SCAN to service aperiodic tasks immediately when the workload is light. On the other hand, at a heavy workload, each round is almost used up and aperiodic tasks are serviced in a batch way. Consequently, the disk seeks caused by aperiodic tasks and the total disk-seek distance decreases when the number of real-time tasks increases in all three versions of WRR-SCAN.

Since the round length used in WRR-SCAN-CA (Eq. (10)) is larger than the one used in WRR-SCAN-C (Eq. (8)), WRR-SCAN-CA allows larger weight assignments and incurs less non-transmission overhead. As a result, the total disk-seek distance of WRR-SCAN-CA is smaller than that of WRR-SCAN-C. Without data compaction, WRR-SCAN-R incurs much more disk-rotation time, compared with WRR-SCAN-C and WRR-SCAN-CA. The disk-rotation overhead consumes a significant bandwidth of a round and hinders the performance of the bandwidth reclamation approach. Similar to the effect of a heavy workload, WRR-SCAN-R services aperiodic tasks in a batch way and incurs the least total disk-seek distance. Finally, due to its strict admission test by Eq. (7), WRR-SCAN-R can only service less than 20 real-time tasks. The data-compaction scheme allows WRR-SCAN-C to service up to 55 real-time tasks. Furthermore, the release-time alignment and a larger round length allows WRR-SCAN-CA to service more than 90 real-time tasks.

Normalized Disk-Idle Time

We define the disk-idle time as the period the disk head is completely idle. The total disk-idle time is equal to the length of execution minus the sum of total disk-transmission, disk-seek, and disk-rotation time. Accordingly, the normalized disk-idle time is the total disk-idle time divided by the length of execution. Figure 6 shows the normalized disk-idle time for each disk-scheduling algorithm. In general, the normalized disk-idle time decreases as the workload increases. Due to its excessive disk-rotation time, WRR-SCAN-R has the least normalized disk-idle time. The data-compaction scheme increases the normalized disk-idle time of WRR-SCAN-C. However, because WRR-SCAN-C decomposes a job into several pieces, each of which is serviced in one round, it still incurs more disk-rotation overhead than SCAN-EDF which services a job with one disk-rotation overhead. By aligning the release time and using a larger round length, WRR-SCAN-CA further increases the normalized disk-idle time and becomes the most efficient algorithm.

Normalized Non-Transmission Overhead

The normalized disk-seek and disk-rotation time is defined as the total disk-seek time and the total disk-rotation time divided by the length of execution, respectively. Figure 7 shows the normalized disk-seek time and Figure 8 shows the normalized disk-rotation time. Figure 7 follows closely the pattern shown in Figure 5. In general, WRR-SCAN incurs more disk-rotation overhead than SCAN-EDF, as shown in Figure 8, because WRR-SCAN services a job multiple times, each of which involves at least one disk-rotation overhead. WRR-SCAN-R yields most normalized disk-rotation time because reading each block may involve one disk rotation. The data-compaction scheme and release-time alignment effectively reduces this overhead. The disk-rotation overhead of SCAN-EDF decreases when the number of real-time tasks is larger than 60 because the number of serviced aperiodic tasks decreases dramatically.

Figure 9 is the addition of Figure 7 and Figure 8, where we define the normalized non-transmission overhead as the sum of total disk-seek and disk-rotation time divided by

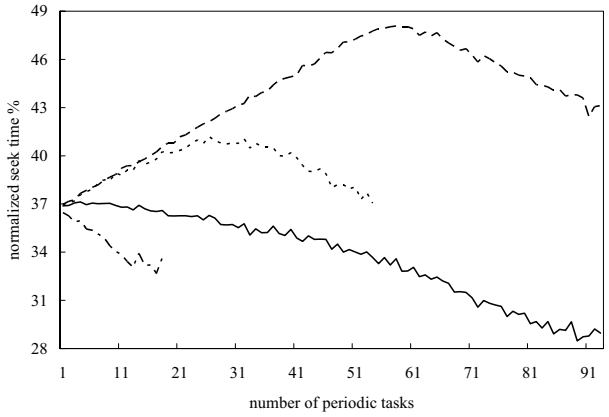

Figure 7: The normalized disk-seek time

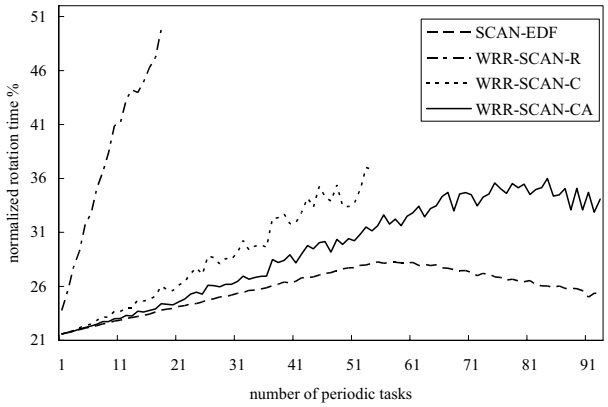

Figure 8: The normalized disk-rotation time

the length of execution. Although WRR-SCAN-R has the least disk-seek overhead, it incurs the most non-transmission overhead due to its excessive disk-rotation time. This result shows that disk-rotation, often overlooked by traditional real-time disk-scheduling algorithms, is in fact an important factor in designing an efficient disk-scheduling algorithm. Compared with SCAN-EDF, WRR-SCAN-C incurs similar overhead as its advantage on disk-seek is offset by its disadvantage on disk-rotation. Finally, by solving the over-reservation problem and serving tasks in scan order, WRR-SCAN-CA delivers the least non-transmission overhead.

Disk Utilization

We define the disk utilization as the disk-transmission time divided by the disk-service time which is equal to the sum of disk-transmission, disk-seek, and disk-rotation time. Figure 10 shows the disk utilization of the four algorithms. WRR-SCAN spends less time in disk-seek while SCAN-EDF generates less disk-rotation time. Similar to the normalized disk-idle time, the disk utilization increases as the workload increases. Among all three versions of WRR-SCAN, WRR-SCAN-CA outperforms WRR-SCAN-C which in turn is better than WRR-SCAN-R. Compared with SCAN-EDF, WRR-SCAN-CA has less non-transmission overhead and, therefore, has better disk utilization.

Throughput and Response Time of Aperiodic Tasks

We define the throughput of aperiodic tasks as the total number of bytes transferred divided by the length of execution. In addition, the response time of an aperiodic task is the time from its release to its completion. Figure 11 and 12 show the throughput and averaged response time of aperiodic tasks, respectively. Because WRR-SCAN reserves a fixed bandwidth for aperiodic tasks and services tasks in a round-robin order, all three versions of WRR-SCAN deliver a nearly constant throughput for aperiodic tasks at all workloads. On the other hand, because SCAN-EDF only services aperiodic tasks when there are no real-time tasks, the throughput of aperiodic tasks depends greatly on the workload of real-time tasks. At a light workload, the throughput remains high as there is enough disk bandwidth to service aperiodic tasks. However, when the workload increases, most of the available disk bandwidth is taken by real-time tasks and the throughput of aperiodic tasks drops considerably.

At a light workload, the response time of an aperiodic task is extremely short as it is serviced almost right after its release. SCAN-EDF delivers prolonged response time when the workload increases. WRR-SCAN will deliver bounded response time if the reserved bandwidth is larger than the requested bandwidth of incoming aperiodic tasks. Otherwise, aperiodic tasks will start to queue up and their response time is prolonged as well. In our experiments, the requested bandwidth of the aperiodic workload is larger than the reserved bandwidth. The bandwidth reclamation approach utilizes unused bandwidth to service more aperiodic tasks when the workload is light. In contrast, WRR-SCAN services tasks according to its reserved weight at a heavy workload, and therefore, the response time of an aperiodic task increases.

5. RELATED WORK

Traditional disk-scheduling algorithms, such as SCAN, SSTF, and LOOK, are designed to reduce disk-seek time and increase throughput. These algorithms do not consider real-time constraints of I/O tasks, and therefore cannot be applied directly on a real-time system. On the other hand, traditional real-time scheduling algorithms, such as Rate-Monotonic (RM) [17] and Earliest-Deadline-First (EDF) [17], address this issue without considering disk-seek time.

Several approaches that modify SCAN to consider real-time constraints have been proposed. SCAN-EDF [23, 24], the first of this kind, first sorts the tasks by their deadlines and then applies SCAN on tasks with the same deadline. SCAN-EDF degenerates to EDF when each task has a different deadline. In our simulations, SCAN-EDF and EDF deliver similar performance in most cases. SCAN-RT [12] inserts a task into the disk queue in scan order only if this insertion will not cause any existing task to miss its deadline. Otherwise, the task is appended to the end of the disk queue. Another example is the Deadline-Modification SCAN (DM-SCAN) [5]. Again, DM-SCAN first sorts tasks in EDF order. It next finds a group of tasks that can be serviced in scan order without missing any of their deadlines. DM-SCAN modifies the deadline of each task in such a group to form a new task sequence in EDF order. The same technique can be applied repeatedly to reduce exces-

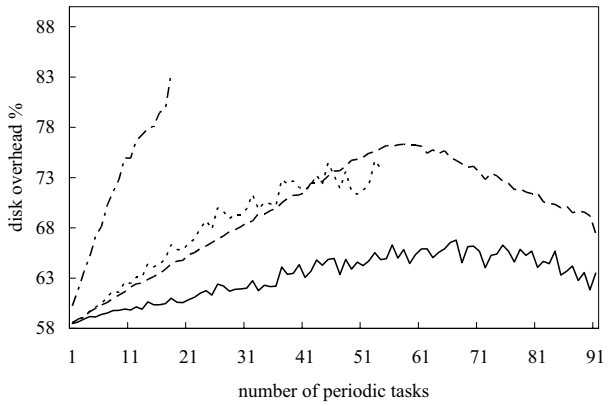

Figure 9: The non-transmission overhead

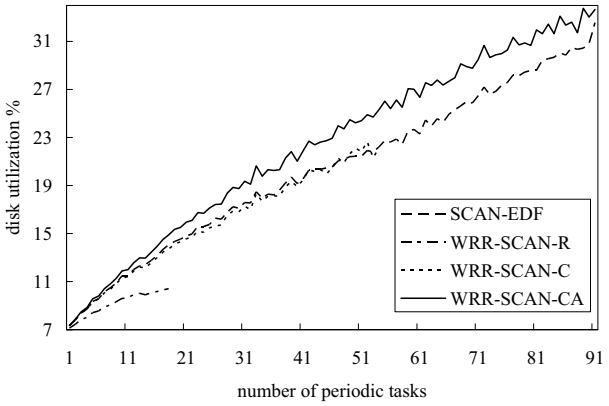

Figure 10: The disk utilization

sive disk-seek time caused by EDF scheduling. Similar to SCAN-EDF, DM-SCAN has the possibility of degeneration to EDF. In addition, the on-line performance of DM-SCAN is hindered by the complexity of its iterative algorithm.

All of the above real-time disk-scheduling algorithms are priority-based. WRR-SCAN is the first rate-based real-time disk-scheduling algorithm. The idea of WRR-SCAN was first proposed by Liu [18]. Our paper fully develops WRR-SCAN and provides extensions for performance improvement. Compared with WRR-SCAN, a priority-based algorithm incurs excessive and unnecessary disk-seek time. In addition, it provides no real-time guarantee for aperiodic tasks.

6. CONCLUSIONS

In this paper, we presented WRR-SCAN (Weight-Round-Robin-SCAN), a rate-based real-time disk-scheduling algorithm. WRR-SCAN includes WRR into the SCAN disk-scheduling algorithm to schedule real-time tasks in midst of aperiodic tasks. WRR-SCAN first divides the disk bandwidth into rounds. Each real-time task is next assigned a weight indicating the maximum amount of disk-transmission time in each round. On the other hand, all aperiodic tasks are served by a virtual aperiodic server which is also assigned a weight. The sum of all weights and non-transmission overhead must be less than or equal to the round length. Three versions of WRR-SCAN are implemented. The default WRR-SCAN, called WRR-SCAN-R, assumes random locations of data. WRR-SCAN-C reduces the disk-rotation overhead by compacting data of the same file in contiguous blocks. Finally, WRR-SCAN-CA further reduces the non-transmission overhead by aligning release times with rounds and allowing a larger round length.

We conducted a set of experiments to compare WRR-SCAN and SCAN-EDF, a priority-based disk-scheduling algorithm. Because SCAN-EDF services tasks according to their deadlines, it causes unnecessary vibrations of the disk head and excessive disk-seek distance. In contrast, WRR-SCAN delivers considerably low disk-seek distance even at a heavy workload. SCAN-EDF provides prolonged response time or even starvation for aperiodic tasks that are served in best-effort manner. Because WRR-SCAN services aperiodic tasks with a reserved bandwidth, it offers a guaranteed minimum data rate and bounded response time. Among all three versions of WRR-SCAN, WRR-SCAN-C significantly reduces the disk-rotation overhead of WRR-SCAN-R by data compaction. In addition, WRR-SCAN-CA allows a larger round length and decreases the over-reservation of WRR-SCAN-C to deliver the best performance of WRR-SCAN. In conclusion, compared with priority-based algorithms, WRR-SCAN greatly reduces non-transmission overhead, increases disk utilization, and provides an acceptable level of response time for aperiodic tasks.

Acknowledgments

The authors would like to thank Jane Liu for her valuable comments on this paper. This research was supported in part by National Science Council, R.O.C., under Grant NSC 92-2622-E-007-021-CC3 and by Institute for Information Industry, R.O.C., under the Communications Software Technology Project.

7. REFERENCES

[1] D. P. Anderson. Metascheduling for Continuous Media. *ACM Transactions on Computer Systems*, 11(3):226–252, Auguest 1993.

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

[3] D. P. Anderson, Y. Osawa, and R. Govindan. Principles of Delay-Sensitive Multimedia Data Storage Retrieval. *ACM Transactions on Information Systems*, 10(1):51–90, 1992.

[4] W. G. Aref, I. Kamel, and S. Ghandeharizadeh. Disk Scheduling in Video Editing Systems. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):933–950, November 2001.

[5] R.-I. Chang, W.-K. Shih, and R.-C. Chang. Deadline-Modification-SCAN with Maximum-Scannable-Groups for Multimedia Real-time Disk Scheduling Algorithm. In *Proceedings of the 19th IEEE Real-Time Systems Symposium*, pages 40–49, 1998.

[6] T.-S. Chen and W.-P. Yang. Amortized Analysis of Disk Scheduling Algorithm V(R)*. *Journal of Information Science and Engineering*, 8(1):223–242, 1992.

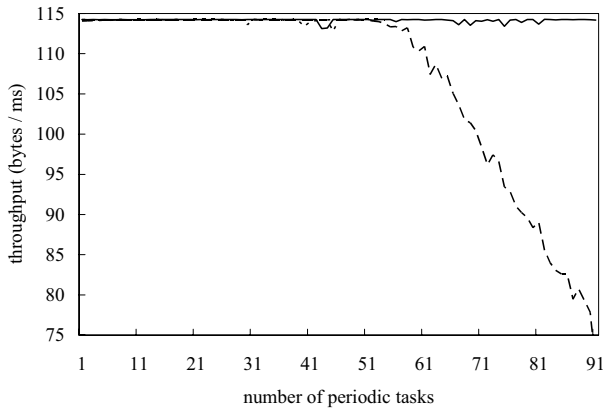

Figure 11: aperiodic throughput

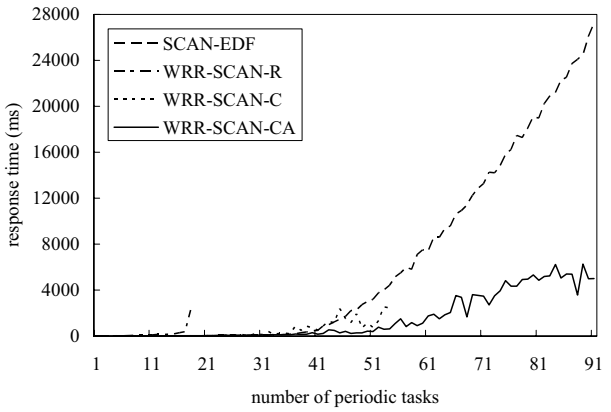

Figure 12: aperiodic response time

[7] T.-S. Chen, W.-P. Yang, and R. C. T. Lee. Amortized Analysis of Some Disk Scheduling Algorithms: SSTF, SCAN, and N-StepSCAN. *BIT*, 32(4):546–558, 1992.

[8] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling Policies for an On-demand Video Server with Batching. In *ACM International Conference on Multimedia*, pages 15–23, October 1994.

[9] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. *ACM SIGCOMM Computer Communication Review*, 19(4):1–12, September 1989.

[10] C. S. Freedman and D. J. DeWitt. The SPIFFI Scalable Video-on-Demand System. In *ACM SIGMOD International Conference on Management of Data*, pages 352–363, May 1995.

[11] T.-Y. Huang, C.-C. Chou, and P.-Y. Chen. Bounding the Execution Times of DMA I/O Tasks on Hard-Real-Time Embedded Systems. In *Proceedings of RTCSA*, pages 516–529, 2003.

[12] I. Kamel and Y. Ito. Disk Bandwidth Study for Video Servers. Technical Report TR-153-96, Matsushita Information Technology Laboratory, 1996.

[13] I. Kamel, T. Niranjan, and S. Ghandeharizedah. A Novel Deadline Driven Disk Scheduling Algorithm for Multi-Priority Multimedia Objects. In *Proceedings of Internatilnal Conference on Data Engineering*, 2002.

[14] S. S. Kanhere, H. Sethu, and A. B. Parekh. Fair and Efficient Packet Scheduling Using Elastic Round Robin. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):324–336, 2002.

[15] D. R. Kenchammana-Hosekote and J. Srivastava. Scheduling Continuous Media in a Video-On-Demand Server. In *IEEE International Conference on Multimedia Computing and Systems*, pages 19–28, May 1994.

[16] M. H. Klein and T. Ralya. An Anlysis of Input/Output Paradigms for Real-Time Systems. Technical Report CMU/SEI-90-TR-19, CMU Software Engineering Institute, July 1990.

[17] C. L. Liu and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*, 10(1):46–61, 1973.

[18] J. W.-S. Liu. Predictability of Real-time Software on Commodity Platform. In *Lecture Notes of European Summer School on Embedded and Real-Time Systems*, 2004.

[19] P. Lougher and D. Shepherd. The Design of a Storage Server for Continuous Media. *The Computer Journal*, 36(1):32–42, 1993.

[20] A. Mahanti, D. L. Eager, M. K. Vernon, and D. Sundaram-Stukel. Scalable On-demand Media Streaming with Packet Loss Recovery. In *ACM SIGCOMM*, pages 97–108, August 2001.

[21] P. V. Rangan and H. M. Vin. Designing File Systems for Digital Video and Audio. In *ACM Symposium on Operating Systems Principles*, pages 81–94, October 1991.

[22] P. V. Rangan and H. M. Vin. Efficient Storage Techniques for Digital Continuous Multimedia. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):564–573, 1993.

[23] A. L. N. Reddy and J. Wyllie. Disk Scheduling in a Multimedia I/O System. In *Proceedings of the 1st ACM International Conference on Multimedia*, pages 225–233, 1993.

[24] A. L. N. Reddy and J. Wyllie. I/O Issues in a Multimedia System. *IEEE Computers*, 27(3):69–74, March 1994.

[25] C. Ruemmler and J. Wilkes. An Introduction to Disk Drive Modeling. *IEEE Computer*, 27(3):17–28, March 1994.

[26] M. Shreedhar and G. Varghese. Efficient Fair Queueing Using Deficit Round-Robin. *IEEE/ACM Transactions on Networking*, 4(3):375–385, June 1996.