ELSEVIER

The Journal of Systems and Software 59 (2001) 143–152

The Journal of Systems and Software

www.elsevier.com/locate/jss

Reschedulable-Group-SCAN scheme for mixed real-time/non-real-time disk scheduling in a multimedia system

Hsung-Pin Chang [a], Ray-I Chang [b], Wei-Kuan Shih [c], Ruei-Chuan Chang [a,*]

[a] *Department of Computer and Information Science, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 30050, Taiwan, ROC*
[b] *Institute of Information Science, Academia Sinica, Taipei, Taiwan, ROC*
[c] *Department of Computer Science, National Tsing Hau University, Hsinchu, Taiwan, ROC*

Received 21 June 2000; received in revised form 7 October 2000; accepted 2 January 2001

Abstract

Real-time disk scheduling is important for time-critical multimedia applications. Previous approaches, such as SCAN-earliest deadline first (EDF) or DM-SCAN, applied the SCAN scheme to reschedule service sequence of input tasks to reduce tasks' service time. However, they required the input to be in EDF order. In DM-SCAN, a deadline modification scheme was employed to obtain a pseudo-EDF sequence from non-EDF ordered input tasks. Because the modified deadlines were smaller than the original ones, number of tasks that could be rescheduled is decreased and thus data throughput is reduced. In this paper, we propose Reschedulable-Group-SCAN (RG-SCAN), a new real-time disk scheduling algorithm using the concept of Reschedulable-Group (R-Group). Differing from previous approaches, RG-SCAN has no limitation on the input task's sequence. In addition, by exploiting the service time's reduction after rescheduling in each R-Group, RG-SCAN has been extended to serve mixed real-time and non-real-time workloads. As shown in experimental results, our approach can support more tasks than DM-SCAN, both real-time and non-real-time. Additionally, our approach can provide larger data throughput and offer better response time to non-real-time tasks. For example, given 30 random-generated real-time tasks, the number of non-real-time tasks that can be supported by RG-SCAN is 1.3 times that supported by DM-SCAN. In addition, our data throughput is 1.1 times DM-SCAN's.

Keywords: Real-time disk scheduling; Mixed workload; Multimedia database/file systems; Operating system

1. Introduction

In the last decade, advances in hardware technology have dramatically increased processor speeds, and this increase is expected to continue to double every year. However, although disk capacity is improved at 60–80% compounded annually, no similar advances are expected to reduce the access time of storage devices. As a result, the performance gap between processors and disks continues to increase and a computer system's performance is increasingly limited by the storage subsystem. This problem is becoming more serious with the emergence of multimedia applications (Lougher and Shepherd, 1993; Gemmell et al., 1995). Multimedia data usually consume significant disk bandwidth and many viewers have to be supported simultaneously (Dan et al., 1994). In addition, due to the rigorous timing requirements for jitter-free playback, media data must be accessed under real-time constraints (Gemmell and Christodoulakis, 1992). Therefore, how to maximize data throughput under real-time constraints poses a challenge in the design of a real-time multimedia disk scheduling algorithm (Steinmetz, 1995).

The SCAN algorithm was first proposed by Denning for scheduling conventional disk tasks (Denning, 1967). By moving the disk arm from the innermost track to the outermost track or vice versa to retrieve data block when it passes through, the SCAN algorithm minimizes the seek-time cost and has been proved as an optimal algorithm under amortized analysis and probability model (Chen and Yang, 1992; Chen et al., 1992). However, due to the lack of timing consideration, the SCAN algorithm is not suitable for scheduling real-time disk tasks. To address a task's real-time characteristic, earliest deadline first (EDF) was proposed and shown to be optimal if tasks are independent (Liu and Layland,

* Corresponding author. Tel.: +886-3-5712121-56656; fax: +886-3-5721490.
E-mail address: rc@cc.nctu.edu.tw (R.-C. Chang).

0164-1212/01/$ - see front matter
PII: S0164-1212(01)00058-9

1973; Lehoczky, 1990; Lin and Tarng, 1991). Nevertheless, for disk scheduling, the service time of a disk task depends on the previous task's track location, and the assumption that tasks are independent is not held. Actually, taking only deadlines into account without service time consideration, EDF incurs excessive seek-time costs and results in poor disk throughput (Reddy and Wyllie, 1994).

Therefore, researchers are investigating methods to combine the features of SCAN type of seek-optimizing algorithms with EDF type of real-time scheduling algorithms (Chang et al., 2000). For example, given an EDF schedule, SCAN-EDF reschedules tasks with the same deadline by SCAN to improve the data throughput (Reddy and Wyllie, 1993). Thus, the efficiency of SCAN-EDF depends upon how many tasks have the same deadlines, i.e., how often the SCAN algorithm can be applied. To increase the rescheduling probability, the DM-SCAN scheme was proposed (Chang et al., 1998). However, in both SCAN-EDF and DM-SCAN, the input tasks must be in EDF order before the seek-optimizing SCAN scheme is applied. Because the input tasks may not conform to an EDF order, a deadline modification scheme is proposed by DM-SCAN, transferring non-EDF ordered tasks into a *pseudo-EDF* sequence. (Here, "pseudo" means the tasks are ordered by the modified deadlines, not their actual deadlines.) In this way, the input tasks would always keep as an EDF sequence. In addition, making the rescheduled result to be EDF-ordered by a deadline modification scheme, DM-SCAN applies the rescheduling idea iteratively to progressively improve disk performance. Unfortunately, in order to guarantee real-time constraints, the modified deadlines are earlier than the original ones. As a result, the number of input tasks that can be rescheduled to reduce service time is decreased and the obtained data throughput is lowered.

To resolve the drawback of DM-SCAN, in this paper, we propose Reschedulable-Group-SCAN (RG-SCAN) scheme: a new real-time disk scheduling algorithm that uses the concept of Reschedulable-Group (R-Group). Given a set of real-time disk tasks, an R-Group consists of the maximum number of continuous disk tasks that can be rescheduled without violating their respective timing constraints. Therefore, by seek-optimizing tasks in R-Groups, data throughput is improved under real-time guarantees. In addition, our proposed RG-SCAN assumes no specific input sequence and thus does not require deadline modification in each iteration. Consequently, our approach is more flexible and obtains more data throughput than DM-SCAN algorithms. Furthermore, we extend the proposed approach to serve mixed real-time/non-real-time tasks in a multimedia environment. By exploiting the reduction of service time after rescheduling tasks within each R-Group, non-real-time tasks can be served to minimize response time while guaranteeing the timing constraints of real-time tasks. As presented in experimental results, our approach can support more tasks than DM-SCAN, both real-time and non-real-time. Additionally, our approach can provide larger data throughput and offer better response time to non-real-time tasks. For example, given 30 randomly generated real-time tasks, the number of non-real-time tasks which can be supported by RG-SCAN is 1.3 times that supported by DM-SCAN; and our data throughput is 1.1 times DM-SCAN's.

The remainder of this paper is organized as follows. Section 2 presents the disk service model in a real-time multimedia environment and defines the real-time disk scheduling problems. The previous DM-SCAN approach is introduced in Section 3. In Section 4, we present the definition of R-Group and our proposed RG-SCAN real-time disk scheduling algorithm. Section 5 demonstrates how RG-SCAN is extended to efficiently serve mixed real-time/non-real-time disk tasks. Sections 6 and 7 present the experimental results and the conclusion remarks, respectively.

2. Problem descriptions

2.1. Disk service model in a real-time multimedia environment

Assume that *start-time* and *finish-time* denote the actual times at which a task is started and completed, respectively. To describe the timing characteristics of a real-time task, two parameters are associated with it to determine the proper start-time and finish-time:

- *Ready time*: the earliest time at which a task can start.
- *Deadline*: the latest time at which a task must be completed.

If a task is started before its ready time, some of the resources, e.g., buffer pool, network controller, will overflow and the system will be erratic. In addition, if a task is not completed before its deadline, users will perceive glitches during the playing, which would violate the spirit of multimedia applications. Thus, to meet the real-time requirements, the start-time of a task should not be earlier than its ready time. Additionally, its finish-time should not be later than the related deadline (Jeffay et al., 1991; Stankovic and Buttazzo, 1995). A schedule of real-time tasks is said to be *feasible* if all tasks can be sequentially served according to the specified real-time requirements.

To serve a disk task, the disk-head first needs to be moved from the previous task's cylinder to the requested one by a *seek-time* cost. Then a *rotational latency* is presented for the desired sector rotated under the disk read–write head. Finally, the asked data are transferred from disk to buffer by a *transfer time*. Therefore, a

conventional disk task T_i is denoted by three parameters (t_i, l_i, b_i), where t_i is the track location, l_i is the sector number, and b_i is the data size. Assume that the schedule sequence is T_jT_i. The service time of task T_i is calculated as

$$c_{j,i} = seek\_time(abs(t_i - t_j)) + rotational\_latency(l_i) + transfer\_time(b_i). \quad (1)$$

Clearly, the service time not only depends on the requested task itself but relates to the previous one. For example, in a HP 97560 hard disk (Ruemmler and Wyllie, 1994), the service time $c_{j,i}$ with movement distance $d_{j,i} = |t_i - t_j|$ can be modeled by

$$c_{j,i} = \begin{cases} 3.24 + 0.4\sqrt{d_{j,i}}, & d_{j,i} \leqslant 383, \\ 8.00 + 0.008d_{j,i}, & d_{j,i} > 383, \end{cases} \quad (2)$$

which is piecewise non-linear, a non-decreasing concave function.

As stated above, disk tasks used to serve time-critical multimedia applications must be real-time guaranteed. Accordingly, for each disk task T_i in a multimedia environment, two more parameters are presented to characterize its real-time attributes: (r_i, d_i), where r_i is the ready time and d_i is its deadline. As disk tasks are non-preemptive, the start-time s_i and finish-time f_i of a real-time task T_i with schedule T_jT_i are thus computed by $s_i = \max\{r_i, f_j\}$ and $f_i = s_i + c_{j,i}$, respectively.

2.2. Real-time disk scheduling problem

Given a set of real-time disk tasks $\boldsymbol{T} = \{T_1, T_2, \ldots, T_n\}$ where n is the number of input disk tasks and the ith disk task T_i is denoted by $(r_i, d_i, t_i, l_i, b_i)$. The objective of a real-time disk scheduling algorithm is to find a *feasible* schedule $\boldsymbol{T}_z = T_{z(1)}T_{z(2)}\cdots T_{z(n)}$ with *maximal throughout*. The index function $z(i)$, for $i = 1$ to n, is a permutation of $\{1, 2, \ldots, n\}$. Define *schedule finish-time* as the finish time it takes to serve all input tasks according to their respective timing constraints. Clearly, this is the finish-time of the latest task $f_{z(n)}$. Therefore, the disk throughput is calculated as follows:

$$\text{Throughput} = \sum_{i=1}^{n} b_{z(i)}/f_{z(n)} \propto (f_{z(n)})^{-1}. \quad (3)$$

The obtained disk throughput is related to the inverse of schedule finish-time. If the input schedule is completed earlier, more data throughput is obtained. The data throughput improvement of scheduler $\boldsymbol{z}$ compared with scheduler $\boldsymbol{x}$ can be computed as

$$\text{Throughput improvement} = (1 - f_{z(n)}/f_{x(n)}) \times 100\%. \quad (4)$$

Therefore, the problem objective defined to maximize throughput can be achieved by minimizing the schedule finish-time. We formally formulate the real-time disk scheduling problem as follows.

Definition 1 (*Real-time disk scheduling*). Given a set of n real-time disk tasks $\boldsymbol{T} = \{T_1, T_2, \ldots, T_n\}$, where the ith task $T_i = (r_i, d_i, t_i, l_i, b_i)$, find a feasible schedule $\boldsymbol{T}_z = T_{z(1)}T_{z(2)}\cdots T_{z(n)}$ that resolves $\min_{\forall z}\{f_{z(n)}\}$ under $r_{z(i)} \leqslant s_{z(i)}$ and $f_{z(i)} \leqslant d_{z(i)}$ for $1 \leqslant z(i) \leqslant n$.

As mentioned in the preceding subsection, a task's service time depends on the related track distance between its previous task and itself. Thus, it is not fixed, but is determined by the schedule result. However, the schedule result minimizing schedule finish-time is determined by the required service time. As a result, it is hard to design an optimal scheduling algorithm for maximizing data throughput while also guaranteeing real-time constraints. This real-time disk scheduling problem has been shown to be NP-complete (Wong, 1980).

3. Related works

In past years, various real-time disk scheduling algorithms have been developed to heuristically employ a seek-optimizing SCAN scheme for an EDF schedule to reduce the disk service time. For example, the well-known SCAN-EDF scheme first schedules disk tasks with the earliest deadlines. If two or more disk tasks have the same deadline, these tasks are serviced according to their relative track locations, i.e., by the SCAN algorithm. Since only tasks with the same deadline are seek-optimized, the obtained data throughput improvement is limited. To increase the probability of applying the SCAN algorithm to reschedule input tasks, DM-SCAN proposed the concept of maximum-scannable-group (MSG) (Chang et al., 1998). An MSG is a set of continuous tasks that can be rescheduled by SCAN without missing their respective timing constraints. Given an EDF schedule $T = T_0T_1\cdots T_n$, the MSG G_i starting from task T_i is defined as the sequential tasks $G_i = T_iT_{i+1}T_{i+2}\cdots T_{i+m}$, where task T_j satisfies following criteria:

$$f_j \leqslant d_i \quad \text{and} \quad r_j \leqslant s_i \qquad \text{for } j = i \text{ to } i + m. \quad (5)$$

By iteratively rescheduling tasks within MSGs, DM-SCAN also proposes an incremental approach to progressively improve disk throughput. However, the rescheduled result will not be in EDF sequence because SCAN is applied to reschedule tasks within each MSG. Since DM-SCAN requires the input tasks based on EDF order, a deadline modification scheme is proposed to modify tasks' deadlines and transfers the rescheduled non-EDF sequence into a pseudo-EDF order. Here, "pseudo" means that the tasks are ordered by the modified deadlines. For example, given

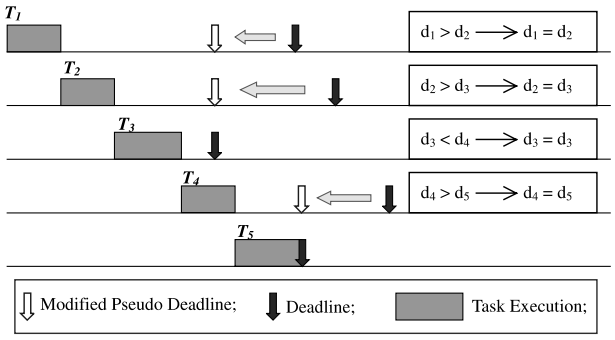

Fig. 1. A simple example to illustrate the deadline modification scheme.

the schedule sequence T_iT_j, a pseudo-deadline $d_{s(i)}$ is derived as $d_{s(i)} = \min\{d_i, d_{s(j)}\}$. Fig. 1 presents a simple example to illustrate the deadline modification scheme. The original input $\boldsymbol{T} = T_1T_2T_3T_4T_5$ is not an EDF schedule because we have $d_2 > d_3$ and $d_4 > d_5$. Traversing from the last task T_5 to the first task T_1, if any task has its deadline larger than that of its previous task, the deadline modification scheme is applied. For example, d_4 is larger than d_5 and is modified equal to d_5 in order to satisfy the EDF requirement. Following the same procedure, d_2 and d_1 are also modified. Note that, although $d_1 < d_2$ in the original input schedule, d_1 is also modified as the value of d_1 is larger than that of modified pseudo-deadline d_2. Clearly, the modified pseudo-deadlines are smaller than the original ones. In addition, although only two tasks (T_2 and T_4) violate the EDF order, three tasks (T_4, T_2, and T_1) have their deadlines modified to meet EDF sequence. Transferring the rescheduled non-EDF ordered sequence into a pseudo-EDF one by deadline modification scheme, DM-SCAN iteratively reschedule tasks from the derived pseudo-EDF schedule to obtain more data throughput.

4. RG-SCAN

From the preceding section, we can see the drawbacks of the deadline modification scheme, where the modified deadlines are smaller than the original ones. Therefore, the group size for rescheduling is narrowed down and number of tasks that can be rescheduled is decreased. Moreover, some tasks suffer from the deadline modification scheme, even though their deadlines are ordered in EDF sequence. To resolve these drawbacks, in this paper a new real-time disk scheduling algorithm called RG-SCAN using the concept of R-Group is proposed. Given *any* input tasks set, consecutive tasks that can be rescheduled under real-time constraints can be directly derived by the concept of R-Group.

Definition 2 (*R-Group* (*Reschedulable-Group*)). Given a set of real-time disk tasks $\boldsymbol{T} = T_1T_2\cdots T_n$, the R-Group G_i is defined as the maximum number of continuous tasks $G_i = T_iT_{i+1}\cdots T_{i+m}$ with each task T_k for $k = i$ to $i+m$ satisfies $f_{i+m} \leqslant \min_{k=i}^{i+m}\{d_k\}$ and $\max_{k=i}^{i+m}\{r_k\} \leqslant s_i$. Fig. 2 presents a simple example to illustrate the concept of R-Group. For example, to calculate R-Group G_2, we have $f_3 \leqslant \min_{k=2}^{3}\{d_k\} = d_3$ and $\max_{k=2}^{3}\{r_k\} = r_3 \leqslant s_2$. But $f_4 > \min_{k=2}^{4}\{d_k\} = d_3$ and $s_2 < \max_{k=2}^{4}\{r_k\} = r_4$. Therefore, $G_2 = T_2T_3$. Following the same procedure, other R-Groups can be derived as $G_1 = T_1T_2$, $G_3 = T_3$, and $G_4 = T_4T_5$, respectively. Note that, the input schedule is not an EDF sequence. The following simple example shows the different schedules obtained by DM-SCAN and RG-SCAN, respectively.

Example 4.1. Let $S = T_1T_2T_3T_4T_5$ be the input schedule (for example, the rescheduled result of the first iteration) with $(r_1, d_1) = (1, 10)$, $(r_2, d_2) = (1, 11)$, $(r_3, d_3) = (3, 15)$, $(r_4, d_4) = (6, 16)$ and $(r_5, d_5) = (5, 14)$. Their track locations are 30, 12, 56, 33 and 36. Assume that the initial disk head is located at track 0. The associated start-times and finish-times are $(s_1, f_1) = (1, 5)$, $(s_2, f_2) = (5, 7)$, $(s_3, f_3) = (7, 12)$, $(s_4, f_4) = (12, 15)$ and $(s_5, f_5) = (15, 16)$. The rescheduled results obtained using DM-SCAN and RG-SCAN are as follows:

- DM-SCAN: Since the deadlines of input tasks are not ordered incrementally, i.e., not in EDF sequence, the deadline modification scheme is applied in DM-SCAN. After applying the deadline modification scheme, the new obtained ready times and deadlines are $(r_1, d_1) = (1, 10)$, $(r_2, d_2) = (1, 11)$, $(r_3, d_3) = (3, 14)$, $(r_4, d_4) = (6, 14)$, and $(r_5, d_5) = (5, 14)$. By following Eq. (5), the obtained R-Groups, i.e., MSGs, are $G_1 = T_1T_2$, $G_2 = T_2$, $G_3 = T_3$, $G_4 = T_4$, and $G_5 = T_5$. Because $G_1 = T_1T_2$ can be rescheduled as T_2T_1 (since the initial disk head is located at track 0) to minimize the seek time, thus the output schedule is $T_2T_1T_3T_4T_5$.
- RG-SCAN: In contrast, RG-SCAN can identify R-Groups directly from *any* sequences of input tasks. Therefore, deadline modification is not needed. The

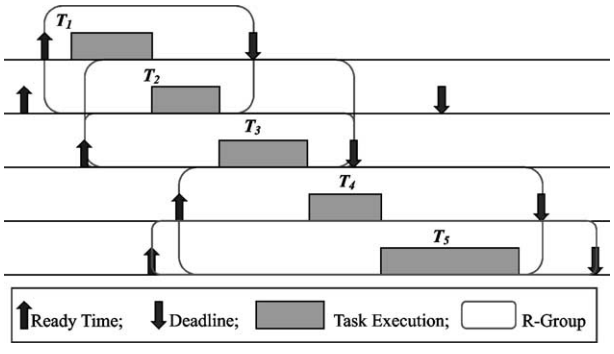

Fig. 2. An example to illustrate the identification of R-Group.

obtained R-Groups are $G_1 = T_1T_2$, $G_2 = T_2$, $G_3 = T_3T_4$, $G_4 = T_4$, and $G_5 = T_5$. Since not only $G_1 = T_1T_2$ but also $G_3 = T_3T_4$ can be rescheduled to reduce their service time, a better schedule result $T_2T_1T_4T_3T_5$ is obtained by RG-SCAN.

As shown in the above example, the R-Group $G_3 = T_3$ in DM-SCAN while $G_3 = T_3T_4$ in RG-SCAN. The RG-SCAN thus identifies more tasks for rescheduling (called *reschedulable tasks*) than DM-SCAN, which is shown in the output schedules obtained by DM-SCAN and RG-SCAN, respectively. As stated in Section 1, if more tasks are seek-optimized, more data throughput is obtained. Therefore, RG-SCAN provides more data throughput than that obtained by DM-SCAN. Fig. 3 shows the operation flows of DM-SCAN and RG-SCAN, that is, given a set of tasks, DM-SCAN requires that these tasks must be ordered in EDF sequence, whereas RG-SCAN has no such limitation. In addition, in DM-SCAN, the iterative approach of progressively improving disk throughput requires the deadline modification scheme to keep tasks in EDF order. However, RG-SCAN identifies reschedulable tasks directly from the non-EDF input tasks and deadline modification is never needed. Since original deadlines are larger than the modified deadlines, RG-SCAN has larger group size for rescheduling than that obtained by DM-SCAN, as demonstrated by Example 4.1. Therefore, our approach is more flexible and obtains more throughput improvement than DM-SCAN.

Following we prove that, if an input schedule is feasible, the refined schedule by rescheduling tasks within a R-Group does indeed improve the data throughput under guaranteed real-time requirements.

Theorem 4.1. *Given a set of feasible real-time tasks* $\boldsymbol{T_Y} = T_{Y(1)}T_{Y(2)}\cdots T_{Y(n)}$ *with R-Group* $G_i = T_{Y(i)}T_{Y(i+1)}\cdots T_{Y(i+m)}$. *Assume that* $S_i = T_{S(i)}T_{S(i+1)}\cdots T_{S(i+m)}$ *is the rescheduled result of* G_i *by seek-optimized algorithm for* $\boldsymbol{T_S} = T_{Y(1)}T_{Y(2)}\cdots T_{Y(i-1)}T_{S(i)}T_{S(i+1)}\cdots T_{S(i+m)}\cdots T_{Y(n)}$. *Then* $\boldsymbol{T_S}$ *obtains more data throughput than* $\boldsymbol{T_Y}$ *under guaranteed real-time requirements.*

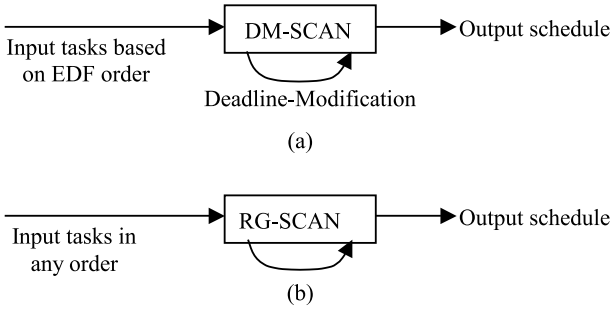

Fig. 3. The operation flows of DM-SCAN and RG-SCAN.

Proof. From the definition of R-Group $G_i = T_{Y(i)}T_{Y(i+1)}\cdots T_{Y(i+m)}$, we have $f_{Y(i)} \leqslant f_{Y(i+1)} \leqslant \cdots \leqslant f_{Y(i+m)} \leqslant \min_{k=i}^{i+m}\{d_{Y(k)}\}$. Since $S_i = T_{S(i)}T_{S(i+1)}\cdots T_{S(i+m)}$ is the seek-optimized rescheduled result of G_i, we have $f_{S(i+m)} \leqslant f_{Y(i+m)}$. In addition, for the rescheduled result in S_i, $f_{S(i)} \leqslant f_{S(i+1)} \leqslant \cdots \leqslant f_{S(i+m)} \leqslant f_{Y(i+m)} \leqslant \min_{k=i}^{i+m}\{d_{Y(k)}\} = \min_{k=i}^{i+m}\{d_{S(k)}\} \leqslant \min_{k=i}^{i+m}\{d_{S(k)}\}$. Therefore, the real-time requirements $f_{S(k)} \leqslant d_{S(k)}$ for $i \leqslant k \leqslant i+m$ is guaranteed. □

Notably, the above proof assumes that the input schedule is feasible. However, our approach may produce feasible rescheduled result even an infeasible schedule is given, although it is not guaranteed. Assume that $\boldsymbol{T} = T_1T_2\cdots T_n$ is a set of input tasks; then because each R-Group G_i is started from task T_i, there are at most n R-Groups considered $(G_1, G_2, \ldots, G_n)$. For these n R-Groups, we have the *overlapping* property, which is shown in Appendix A. In other words, these R-Groups may not be mutually exclusive. If we sequentially serve these n R-Groups, then earlier R-Group's seek-optimized rescheduled result may be destroyed by the later one's. For example, if R-Group $G_3 = T_3T_4T_5T_6$ and its seek-optimized rescheduled result $R_3 = T_5T_6T_4T_3$. From the overlapping property, assume that R-Group $G_4 = T_4T_5T_6T_7$. After rescheduling tasks in G_4, the seek-optimized result $R_4 = T_7T_5T_6T_4$. Evidently, the seek-optimized operation in G_4 destroys the previous seek-optimized order in G_3. This disturbs the algorithm's progression and makes performance of the rescheduled result to be unpredictable. As a result, we select only the *mutually exclusive* R-Groups for rescheduling and serve them in first-in-first-out (FIFO) order.

Therefore, RG-SCAN first identifies a R-Group and reschedules tasks in it by SCAN; then the next mutually exclusive R-Groups is identified and tasks within it are rescheduled. This process is repeated until the last R-Groups is achieved. However, in addition to the different identification scheme from DM-SCAN, once RG-SCAN identifies an R-Group, it reschedules tasks in the derived R-Group and immediately updates the tasks' start-times and deadlines. Therefore, the identification of remaining R-Groups is based on the more prompt value of start-times and deadlines. In addition, as will be stated in Section 5, the non-real-time tasks can thus be serviced after the identification of an R-Group to minimize their response time. The algorithm runs iteratively until convergence.

Given a set of n tasks, the time complexity of identifing n R-Groups is $O(n \log n)$. This is achieved by using an AVL tree to keep track of the minimum deadline and maximum ready time in each R-Group. In contrast, the time complexity of identifying n MSGs is $O(n)$. However, in each iteration, DM-SCAN requires $O(n \log n)$ time complexity, which is the same as the time complexity of

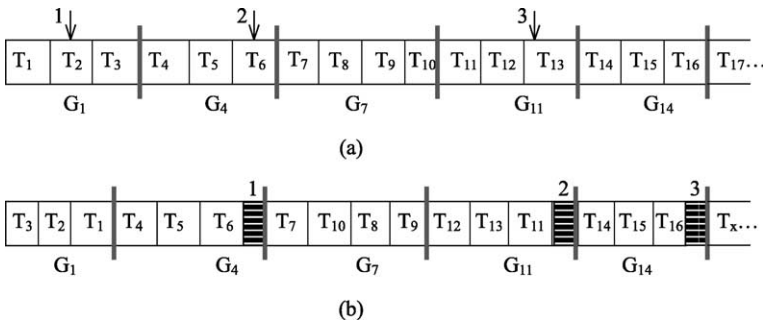

Fig. 4. A simple example illustrates the employment of R-Group to serve non-real-time tasks: (a) three non-real-time tasks arrive in the serve of R-Group G_1, G_4, and G_{11}, respectively; (b) they are served in G_4, G_{11}, and G_{14} by the service time reduction after rescheduling each R-Group.

RG-SCAN in each iteration. In other words, both DM-SCAN and RG-SCAN have the same $O(n \log n)$ time complexity in each iteration. When new tasks arrive, the time complexity of both schemes is also $O(n \log n)$.

5. Supporting non-real-time tasks

In a multimedia system, although most accesses are for media data, a few non-real-time disk requests are interposed to access conventional files. For example, in a video-on-demand (VoD) system, we first search the archive to select the desired video. After that, a continuous retrieval of selected media data is guaranteed for jitter-free playback. Although non-real-time tasks have no deadline constraints, reasonable response time has to be offered while guaranteeing the real-time tasks' timing requirements.

Intuitively, non-real-time tasks would be served after the completion of all real-time tasks. However, in this way, non-real-time tasks will be served with an undesirably long response time and at worst, be starved of service.

From Theorem 4.1, the data throughput is improved by rescheduling tasks in an R-Group using a seek-optimized algorithm. In other words, the finish time of an R-Group is advanced as Eq. (3) indicates. We obtain the "*slack*" between the advanced finish-time and the original one and use this slack to serve non-real-time tasks. Therefore, non-real-time tasks are served quickly, while still guaranteeing the real-time tasks' timing constraints. Fig. 4 shows how a R-Group is employed to serve non-real-time tasks. Once non-real-time tasks arrive, as the current R-Group is under way, we try to serve them in the next R-Group's slack. For example, if a non-real-time task T_1 arrives during the execution of R-Group G_1, it is served in the slack of the next R-Group G_4. If the slack derived from an R-Group is not large enough to sustain a non-real-time task, we continue to identify the next R-Group and the derived slack is added to the previous one, until the non-real-time task can be served.

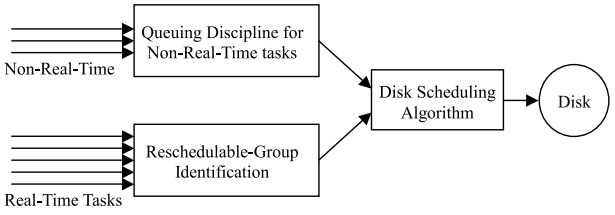

Fig. 5. Service model for serving real-time and non-real-time disk tasks.

For example, the slack derived from R-Group G_7 is too small to serve T_2. As a result, we serve T_2 by the accumulated slack derived from G_7 and G_{11}.

Fig. 5 presents the service model for serving mixed real-time/non-real-time disk workloads. Because non-real-time tasks may arrive faster than a system's capability, an isolated queue is maintained to temporarily hold them. Therefore, there are two separate queues in the system, one for real-time tasks and the other for non-real-time tasks. From the input real-time tasks' queue, we first identify tasks belonging to an *R-Group*. By rescheduling tasks within the R-Group, slack is derived from the reduction of service time. After that, one or more non-real-time tasks are taken out of the non-real-time task's queue and served within the derived slack until the schedule result is infeasible. If no non-real-time tasks arrive and the queue is empty, the finish time of the R-Group is advanced and data throughput of real-time tasks is also improved.

6. Performance evaluation

6.1. Experimental environments

In this section, the experimental results of our proposed RG-SCAN algorithm are presented to compare with previous approaches. Table 1 presents some important parameters of HP 97560, which is used as our target disk for performance evaluation (Ruemmler and

Table 1
Disk parameters of HP 97560

No. of cylinders per disk	1972
No. of tracks per cylinder	19
No. of sectors per track	72
Sector size	512 bytes
Seek-time function (ms)	$\text{Seek}(d) = \begin{cases} 3.24 + 0.4\sqrt{d}, & d \leqslant 383, \\ 8.00 + 0.008d, & d > 383 \end{cases}$
Revolution speed	4002 rpm
Transfer time	10 MBps

Wyllie, 1994). Each real-time task is assumed to ask for a track of data (36 KB in HP 97560). The ready times of real-time tasks are randomly generated and their deadlines are uniform distributed within a proper interval after their corresponding ready times. Non-real-time tasks are assumed to arrive with a Poisson distribution. The mean inter-arrival time between each non-real-time task is varied with different experiments and is described below. The request size of each non-real-time task is assumed to be 4 KB. The workloads of both real-time and non-real-time tasks are uniformly distributed over the disk surface. In all following experiments, 100 experiments are conducted with different seeds for random number generation.

6.2. Performance of real-time disk scheduling

6.2.1. Number of supported real-time tasks

First, we present the experimental results of different disk scheduling algorithms for serving real-time tasks. There are two metrics to measure the efficiency of a real-time disk scheduling algorithm: one is the number of supported tasks, and the other is the data throughput. Given a set of input tasks, the applied disk scheduling algorithm should serve as many tasks as possible. If the same tasks are served, the applied disk scheduling algorithm must finish the schedule as quickly as possible to maximize data throughput.

Given 100 experiments, Table 2 presents the minimal, maximal, and average number of real-time tasks that are supported by different disk scheduling algorithms. The number of supported tasks n is obtained by increasing the number of input tasks incrementally until the schedule result is infeasible with $n+1$ real-time tasks. On average, our proposed RG-SCAN provides 24 real-time tasks, which is better than both DM-SCAN and SCAN-EDF, which support 22 and 18 tasks, respectively. Accordingly, identifying task groups for rescheduling can serve more tasks than only rescheduling tasks having the same deadline. This is because the number of *reschedulable tasks* identified by RG-SCAN or DM-SCAN is much larger than SCAN-EDF. As a result, input tasks' service times are reduced after rescheduling and more tasks can be served before their deadlines. In addition, RG-SCAN further supports more tasks than DM-SCAN. As stated above, DM-SCAN requires modifying tasks' deadlines for the identification of reschedulable tasks. In contrast, RG-SCAN identifies reschedulable tasks by using the concept of R-Group, which demands no specific input sequence, and thus no tasks' deadlines needs to be modified. Because the modified deadlines are smaller than the original deadlines, RG-SCAN thus identifies more number of reschedulable tasks than DM-SCAN. As a result, the further reduction of tasks' service times prompts more tasks to be supported by RG-SCAN.

Table 2
The minimal, maximal, and average number of supported real-time task with different scheduling policies

Algorithms	Number of supported tasks		
	Min	Max	Average
RG-SCAN	20	27	24
DM-SCAN	20	26	22
SCAN-EDF	14	22	18

6.2.2. Data throughput improvement

If the same real-time tasks are served, then the applied disk scheduling algorithm must maximize data throughput, i.e., minimize schedule finish-time, to accommodate the huge volumes of multimedia data access. Fig. 6 shows the data throughput improvement of RG-SCAN and DM-SCAN under different number of EDF tasks. The data throughput improvement is compared with SCAN-EDF. For each bar in Fig. 6, 100 experiments were applied and the average throughput improvement is used for measurement. From the experimental results, the data throughput improvement obtained by RG-SCAN is always better than that obtained by DM-SCAN, no matter how many tasks are applied in the input schedule. This demonstrates the efficiency of RG-SCAN over DM-SCAN. For example, with 15 real-time tasks, the data throughput improved

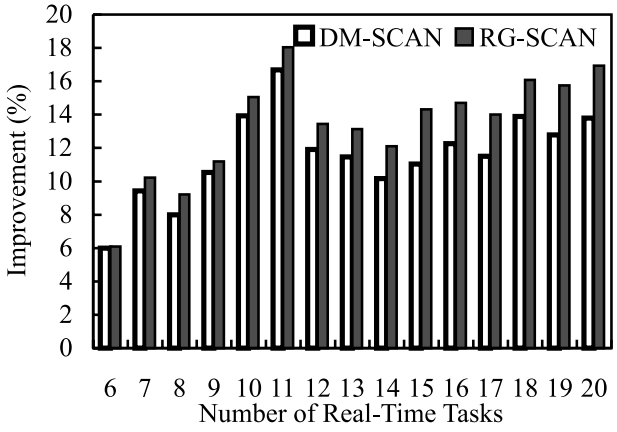

Fig. 6. The data throughput improvement of DM-SCAN and RG-SCAN under different number of input tasks.

by RG-SCAN is 1.1 times DM-SCAN's. Although input tasks are EDF-ordered, the deadline modification scheme adopted in DM-SCAN advances tasks' deadlines and restricts the group size for rescheduling. In contrast, the RG-SCAN automatically identifies reschedulable tasks within an R-Group from any input tasks set. No deadline modification is required and the obtained number of reschedulable tasks is guaranteed to be larger than DM-SCAN. Therefore, RG-SCAN can provide more data throughput than DM-SCAN.

In addition, from Fig. 6, we observe that the throughput improvement of RG-SCAN is more significant than that of DM-SCAN if more tasks are served. Because more tasks are served, the deadline modification scheme used in DM-SCAN will modify more tasks' deadlines and narrow down the group size for rescheduling. In contrast, RG-SCAN assumes no specific input sequence and no tasks' deadlines are modified. Therefore, RG-SCAN performs well with an increased number of input tasks. In multimedia system design, more and more simultaneous streams are provided to support the increasing number of client viewers. Therefore, the improvement of our proposed approach will be superior to DM-SCAN with advances in multimedia systems' design technologies.

6.3. Non-real-time task's performance

6.3.1. Number of supported non-real-time tasks

Secondly, we measure non-real-time task's performance supported by different disk scheduling algorithms in a real-time environment. Given a set of real-time tasks, a well-behaved disk scheduling algorithm should support as many non-real-time tasks as possible, while guaranteeing the timing constraints of real-time tasks. In addition, the applied disk scheduling algorithm must offer good response time for non-real-time tasks to stay below some user-tolerance threshold. Assuming that there are 20 real-time tasks, Table 3 shows the minimal, maximal, and average number of non-real-time tasks supported by RG-SCAN and DM-SCAN. The mean inter-arrival time of non-real-time tasks is assumed to be 10.1 ms, which saturates the non-real-time task's queue to avoid the occurrence of an empty queue. The queuing principle for non-real-time tasks is followed using FIFO order. Note that, by exploiting the SCAN order in R-Groups, other queuing principles exist to reduce average response time of non-real-time tasks. For example, we can select non-real-time tasks that incur minimum seek time with the tasks in an R-Group. In this paper, we select FIFO for its simplicity and fairness. Table 4 shows the same results, but with 30 real-time tasks. We also present the minimal, maximal, and average schedule finish-time of various disk scheduling schemes for comparison. Note that the schedule finish-times of RG-SCAN and DM-SCAN include both the execution of real-time and non-real-time tasks.

Tables 3 and 4 show that RG-SCAN serves more non-real-time tasks than DM-SCAN in the same real-time environment, i.e., having the same real-time tasks. Without the requirement to modify tasks' deadlines, RG-SCAN obtains larger group sizes, i.e., more reschedulable tasks, for rescheduling. The slack between advanced finish-time and original finish-time in RG-SCAN is thus larger than in DM-SCAN. Therefore, more non-real-time tasks have the chance to be served within the obtained slack by RG-SCAN. In addition, both RG-SCAN and DM-SCAN supporting non-real-time tasks have comparable schedulable finish-times with SCAN-EDF. That is, non-real-time tasks supported by RG-SCAN and DM-SCAN, together with their real-time tasks, have almost the same schedule finish-time as SCAN-EDF, which only serves real-time tasks. Especially, the average schedule finish-time of RG-SCAN under 30 real-time tasks has smaller value

Table 3
Given 20 real-time tasks, the schedule finish-time and number of supported non-real-time tasks of different approaches

Algorithm	Schedule finish-time (ms)			Number of non-real-time tasks		
	Min	Max	Avg.	Min	Max	Avg.
RG-SCAN	289.25	333.34	313.17	0	8	3
DM-SCAN	289.25	333.34	311.66	0	5	2
SCAN-EDF	285.35	338.12	309.33	NA	NA	NA

Table 4
Given 30 real-time tasks, the schedule finish-time and number of supported non-real-time tasks of different approaches

Algorithm	Schedule finish-time (ms)			Number of non-real-time tasks		
	Min	Max	Avg.	Min	Max	Avg.
RG-SCAN	522.98	552.38	537.68	7	9	8
DM-SCAN	535.36	548.88	542.12	6	7	6
SCAN-EDF	530.59	559.71	545.15	NA	NA	NA

Table 5
Given different real-time and non-real-time tasks, the obtained response time of non-real-time tasks (NRT-Tasks) by RG-SCAN and DM-SCAN

Algorithm	20 real-time tasks		25 real-time tasks		30 real-time tasks	
	NRT-tasks	Res. time (ms)	NRT-tasks	Res. time (ms)	NRT-tasks	Res. time (ms)
RG-SCAN	2	132.19	4	140.31	6	145.80
DM-SCAN	2	140.50	4	151.90	6	157.62

than that of SCAN-EDF, while RG-SCAN simultaneously supports an average of eight non-real-time tasks.

In Table 3, we support one more non-real-time task than DM-SCAN, and the number of supported non-real-time tasks is smaller than that of real-time tasks. As stated in Section 5, in a multimedia system, most disk accesses are real-time demanded for media playback. Only a few non-real-time tasks are interposed for ordinary file access. Therefore, our scheme serves 1.5 times the number of non-real-time tasks supported by DM-SCAN and we believe that this is valuable due to the small number of non-real-time tasks in a multimedia system. In addition, from Tables 3 and 4, the number of supported non-real-time tasks is increased with that of real-time tasks in systems. Therefore, the number of supported non-real-time tasks by RG-SCAN relates with the increase of real-time tasks and offers good scalability in a multimedia system. Furthermore, the techniques we propose in this paper can cooperate with other approaches (Spuri and Buttazzo, 1994; Lehoczky et al., 1987; Lehoczky and Ramos-Thuel, 1992) to further improve the performance for supporting non-real-time task.

6.3.2. Response time

Responsiveness is an important factor measuring the performance of disk scheduling algorithms for the support of non-real-time tasks. In this subsection, we present the response time of non-real-time tasks using RG-SCAN and DM-SCAN. 100 experiments are conducted for each approach and the mean inter-arrival time of non-real-time tasks is assumed to be 50.1 ms. In Table 5, using different numbers of real-time and non-real-time tasks, we present non-real-time task's average response time under RG-SCAN and DM-SCAN. From the experimental results, our proposed RG-SCAN scheme offers shorter response time than DM-SCAN. For example, with 25 real-time and 4 non-real-time tasks, RG-SCAN provides a reduction of 7.6% of DM-SCAN's response time.

As explained above, RG-SCAN has larger group sizes for rescheduling than DM-SCAN, and the service time's reduction of rescheduling tasks in RG-SCAN is larger than in DM-SCAN. Thus, RG-SCAN will obtain enough slack to quickly serve non-real-time tasks and shorten their response times. However, we observe that the response time improvement is actually not very significant compared with DM-SCAN. This is because, in some cases, the slack obtained by rescheduling the fewer real-time tasks by DM-SCAN is just large enough to serve non-real-time tasks. On the other head, RG-SCAN obtains larger group size and postpones non-real-time task's execution to wait for the completion of more real-time tasks. This limitation can be resolved by rescheduling real-time and non-real-time tasks together within an R-Group. As shown in Fig. 4, the serving of non-real-time tasks is put on the end of an R-Group. By rescheduling non-real-time and real-time tasks together, the response times of non-real-time tasks are reduced. Thus, the total service time is decreased, and more non-real-time tasks can be served. Nevertheless, this requires rescheduling by SCAN each time a non-real-time task is tested for schedulability, which increases the time complexity. However, on average, RG-SCAN still provides shorter response time than DM-SCAN.

7. Conclusions

In order to improve disk throughput, a seek-optimizing rescheduling scheme is applied as much as possible to disk requests under guaranteed real-time constraints. However, previous approaches limit their flexibility and performance since the rescheduling scheme is employed only to an EDF schedule. In this paper, we propose RG-SCAN, a new disk scheduling algorithm using the concept of R-Group, to resolve this drawback. Using this R-Group, consecutive tasks that can be rescheduled under real-time constraints are derived from *any* input tasks set, so no deadline modification is required. Since original deadlines are larger than the modified deadlines, RG-SCAN obtains a larger task group for rescheduling than DM-SCAN.

In addition, we extend the RG-SCAN algorithm to serve mixed real-time/non-real-time tasks in a multimedia environment. By rescheduling tasks within an R-Group, slack derived from the reduction of service time is used to serve non-real-time tasks to minimize their response times. The experimental results show that our approach can support more tasks, both real-time and non-real-time, than DM-SCAN. Additionally, our approach can provide larger data throughput and offer good response time to non-real-time tasks. Furthermore, our scheme offers good scalability for the support of both real-time and non-real-time tasks. Therefore,

our proposed RG-SCAN can keep pace with the explosive progress of multimedia design technology, performing much better than the DM-SCAN approach.

Appendix A

Theorem. *Given an input schedule* $\mathbf{T} = T_1T_2\cdots T_n$. *For each R-Group* $G_i = T_iT_{i+1}\cdots T_{i+m}$, *the sub-group* $T_pT_{p+1}\cdots T_{i+m}$ *of* G_i *is a part of* G_p *for* $p = i+1$ *to* $i+m$.

Proof. Assume that the smallest deadline in a R-Group $G_i = T_iT_{i+1}\cdots T_{i+m}$ is $d_q = \min_{k=i}^{i+m}\{d_k\}$.

(a) Since $\mathbf{T} = T_1T_2\cdots T_n$ is the input schedule, we have $s_i \leqslant s_{i+1}$ and $f_i \leqslant f_{i+1}$ for all i.

(b) From the definition of R-Group G_i and $G_i = T_iT_{i+1}\cdots T_{i+m}$, we have $f_{i+m} \leqslant d_q$ and $\max_{k=i}^{i+m}\{r_k\} \leqslant s_i$.

$\Rightarrow$ Therefore, from (a) and (b), it can be derived that $f_{i+m} \leqslant d_q = \min_{k+i}^{i+m}\{d_k\} \leqslant \min_{k=p}^{i+m}\{d_k\}$ and $\max_{k=p}^{i+m}\{r_k\} \leqslant \max_{k=i}^{i+m}\{r_k\} \leqslant S_i \leqslant S_p$ for $p = i+1$ to $i+m$. Thus, the sub-group $T_{Y(p)}T_{Y(p+1)}\cdots T_{Y(i+m)}$ of G_i must be a part of G_p for $p = i+1$ to $i+m$. □

References

Chang, H.P., Chang, R.I., Shih, W.K., Chang, R.C., 2000. Enlarged-maximum-scannable-groups for real-time disk scheduling in a multimedia system. In: Proceedings of the 24th IEEE International Computer Software and Application Conference (COMPSAC), pp. 383–388.

Chang, R.I., Shih, W.K., Chang, R.C., 1998. Deadline-modification-SCAN with maximum scannable-groups for multimedia real-time disk scheduling. In: Proceedings of the 19th IEEE Real-Time Systems Symposium, pp. 40–49.

Chen, T.S., Yang, W.P., 1992. Amortized analysis of disk scheduling algorithm $V(R)$. Journal of Information Science and Engineering 8 (2), 223–242.

Chen, T.S., Yang, W.P., Lee, R.C.T., 1992. Amortized analysis of some Disk-Scheduling algorithms: SSTF, SCAN, and N-Step SCAN. BIT 32, 546–558.

Dan, A., Sitaram, D., Shahabuddin, P., 1994. Scheduling policies for an on-demand video server with batching. In: Proceedings of 2nd ACM Multimedia Conference, pp. 5–22.

Denning, P.L., 1967. Effects of scheduling on file memory operations. In: Proceedings of AFIPS SJCC, pp. 9–21.

Gemmell, D.J., Christodoulakis, S., 1992. Principles of delay sensitive multimedia data storage and retrieval. ACM Transaction on Information Systems 10 (1), 51–90.

Gemmell, D.J., Vin, H.M., Kaudlur, D.D., Rangan, P.V., Rowe, L.A., 1995. Multimedia storage servers: a tutorial. IEEE Computer 28 (5), 40–49.

Jeffay, K., Stanat, D.F., Martel, C.U., 1991. On nonpreemptive scheduling of periodic and sporadic tasks. In: Proceedings of Twelfth IEEE Real-Time Systems Symposium, pp. 129–139.

Lehoczky, J.P., Sha, L., Strosnider, J.K., 1987. Enhanced aperiodic responsiveness in hard real-time environments. In: Proceedings of 8th IEEE Real-Time Systems Symposium, pp. 261–270.

Lehoczky, J.P., 1990. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In: Proceedings of 11th IEEE Real-Time Systems Symposium, pp. 201–212.

Lehoczky, J.P., Ramos-Thuel, S., 1992. An optimal algorithm for scheduling soft-aperiodic tasks in fixed-priority preemptive systems. In: Proceedings of 13th IEEE Real-Time Systems Symposium, pp. 110–123.

Lin, T.H., Tarng, W., 1991. Scheduling period and aperiodic tasks in hard real-time computing systems. In: Proceedings of ACM SIGMETRICS, pp. 31-38.

Liu, C.L., Layland, J.W., 1973. Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of ACM 20 (1), 46–61.

Lougher, P., Shepherd, D., 1993. The design of a storage server for continuous media. The Computer Journal 36 (1), 32–42.

Reddy, A.L.N., Wyllie, J.C., 1993. Disk scheduling in a multimedia I/O system. In: Proceedings of 1st ACM Multimedia Conference, pp. 225–233.

Reddy, A.L.N., Wyllie, J.C., 1994. I/O issues in a multimedia system. IEEE Computer 27 (3), 69–76.

Ruemmler, C., Wyllie, J.C., 1994. An introduction to disk drive modeling. IEEE Computer 27 (3), 17–28.

Stankovic, J.A., Buttazzo, G.C., 1995. Implications of classical scheduling results for real-time systems. IEEE Computer 28 (6), 16–25.

Steinmetz, R., 1995. Multimedia file systems survey: approaches for continuous media disk scheduling. Computer Communication 18 (3), 133–144.

Spuri, M., Buttazzo, G.C., 1994. Efficient aperiodic service under earliest deadline scheduling. In: Proceedings of 15th IEEE Real-Time Systems Symposium, pp. 2–11.

Wong, C.K., 1980. Minimizing expected head movement in one dimension and two dimensions mass storage system. Computer Survey 12 (2), 167–178.

Hsung-Pin Chang received the B.S. and M.S. degree in Computer and Information Science in 1995, and 1997, respectively, from National Chiao Tung University, Taiwan, ROC. He is currently a Ph.D. candidate in Computer and Information Science at National Chiao Tung University. His research interests include real-time system, operating system, and wireless communication.

Dr. Ray-I Chang is a member of Computer Systems and Communications Laboratory (CSCL) in Institute of Information Science (IIS), Academia Sinica, Taiwan, ROC. He earned his Ph.D. degree in Electrical Engineering and Computer Science from National Chiao Tung University in 1996, where he was a member of Operating Systems Laboratory. At CSCL of IIS, Dr. Chang has worked on the projects of real-time traffic engineering, video-on-demand server, and distributed digital library design. His current research interests include real-time and distributed multimedia systems. He has published over 50 scientific papers in the international journals and conferences. Dr. Chang is a member of IEEE.

Wei-Kuan Shih received the B.S. and M.S. degrees in computer science from the National Taiwan University, and the Ph.D. degree in computer science from the University of Illinois, Urbana-Champaign.

He is an Associate Professor in the Department of Computer Science at the National Tsing Hua University, Taiwan. His research interests include real-time systems, VLSI design automation, and wireless communication. From 1986 to 1988, he was with the Institute of Information Science, Academia Sinica, Taiwan.

Ruei-Chuan Chang received the B.S. degree in 1979, the M.S. degree in 1981, and his Ph.D. degree in 1984, all in computer science from National Chiao Tung University. In August 1983, he joined the Department of Computer and Information Science at National Chiao Tung University as a Levcturer. Now he is a Professor of the Department of Computer and Information Science. He is also an Associate Research Fellow at the Institute of Information Science, Academia Sinica, Taipei.