Scheduling Customers in a Non-Removal Real-Time System with an Application to Disk Scheduling

Shenze Chen and Don Towsley

COINS Technical Report 92-58
September 1992

Scheduling Customers in a Non-Removal Real-Time System with an Application to Disk Scheduling*

Shenze Chen *Don Towsley*

Department of Computer Science
University of Massachusetts
Amherst, MA 01003

September 3, 1992

Abstract

In this paper, we consider the problem of scheduling customers in a real-time system in which all customers are required to be serviced. Such a *non-removal* system can be distinguished from a *removal* real-time system in which customers can be removed prior to completing service. We describe and evaluate a simple paradigm for mapping policies for removal systems to policies for non-removal systems. We show that several policies known to be optimal for removal systems map into policies which are also optimal for non-removal systems. The paper concludes with an application of this paradigm to scheduling of requests with real-time constraints on disk subsystems.

*This work is supported, in part, by the Office of Naval Research under contract N00014-87-K-0796.

1 Introduction

In this paper, we study the problem of scheduling customers with deadlines in a *non-removal real-time system* (i.e., a system in which all customers must complete their services). Rather than propose and evaluate policies specifically for this system, we study the relationship between a non-removal real-time system and one in which customers may be removed prior to completing their service. We shall refer to the latter as a *removal system* (i.e., a system in which customers that miss their deadlines may be removed). We present a simple paradigm for transforming scheduling policies for removal systems to scheduling policies for non-removal systems. We show how this paradigm can be used to obtain *optimal policies* for several classes of non-removal systems (i.e., policies that minimize the fraction of customers that miss their deadlines). In addition, we apply the paradigm to the problem of scheduling I/O requests with real-time constraints to disks when requests must always be completed. In doing so, not only do we present high performance disk scheduling policies for non-removal systems but also new policies for removal type disk systems that provide better performance than those previously described in the literature.

Previous work on non-removal real-time systems has been concerned either with reducing the customer lateness and tardiness, with minimizing the customer loss ratio in the case that deadlines are not known, or minimizing the customer loss ratio when deadlines are known in the absence of arrivals. Su and Sevcik [19] looked at the problem of scheduling customers with deadlines in a queue. They showed that the *earliest due date (EDD)* rule minimizes performance parameters of expected lateness and tardiness when every customer has to be served. Several papers [9, 12, 20, 4, 13] have established that the last come first serve policy minimizes the customer loss ratio for a variety of queueing systems in the absence of deadline information. Blanc, et al., [6] has characterized the structure of optimal admission control policies for such systems. Pinedo [17] considered the problem of minimizing the number of late jobs when the processing times are exponentially distributed and the deadlines are randomly distributed. Dertouzos [8] has shown that for any set of tasks with arbitrary service times and deadlines, the EDD policy is optimal if preemptions are allowed. Both of them considered only *deterministic* scheduling policies among a set of n jobs, i.e., no new jobs are allowed into the system once the processing begins. Other work of interest can be found in [10, 2, 3, 22].

The application to disk subsystems is of interest because the scheduling policies that we present for both removal and non-removal disks are shown to outperform existing scheduling policies; namely, the window policies proposed in an earlier paper, [7], and a modified version of FD-SCAN, an algorithm first introduced by Abbott and Garcia-Molina, [1].

The remainder of this paper is organized as follows. Section 2 describes our non-removal and removal real-time system models and the paradigm for adapting policies for removal systems to

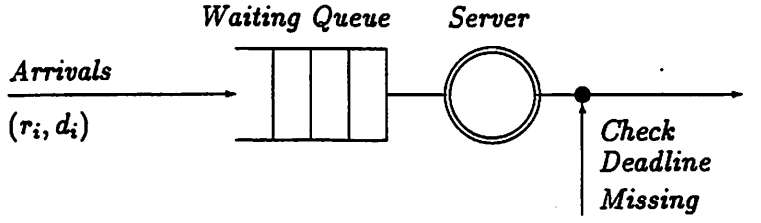


Figure 1: The Queueing Model with Time Constrained Customers

non-removal systems. Section 3 contains the proof of the optimality of earliest deadline polcies for non-removal systems. These policies are obtained by applying the new paradigm to optimal policies for removal systems. Section 4 presents an application case study in a real-time disk scheduling context where the paradigm is used. The paper is summarized in Section 5.

2 The Models and the Paradigm

In this section, we introduce the non-removal and removal real-time system models that will be studied. The remainder of the section contains a description and discussion of the paradigm that will be used to map policies for removal real-time systems to policies for non-removal real-time systems.

2.1 System Models

We study a single server system. Customers arrive to the system at times $a_1 < a_2 < \cdots$. Let $\{\sigma_i\}_{i=1}^{\infty}$ be a sequence of random variables (r.v.'s) that denote the service times of the customers. Last, let $\{r_i\}_{i=1}^{\infty}$ denote the relative deadlines for the customers. The i-th customer has deadline $d_i = a_i + r_i$, $i = 1, \cdots$. Two variations exist for both the removal and non-removal systems according to whether the deadline is to the beginning or end of service.

- **The Non-Removal System Model**

  The non-removal real-time system model we are interested in is illustrated in Figure 2.1. At the time of their arrival, customers join a queue. Customers are scheduled according to some policy that knows the deadlines of the customers in the queue. In this model, all customers have to be eventually served, even though some of them may already have missed their deadlines. When a customer completes service and leaves the system, it is checked against its deadline to see if it has missed it.

- **The Removal System Model**
  The removal system model has been studied in earlier works, [15, 16, 5]. It differs from a non-removal system in that the scheduling policies may throw away customers prior to completing service.

In both systems, the performance measure of interest to us is $L_\pi(t)$, the number of customers that have missed their deadlines by time t under policy π. When it exists, we will also be interested in the long-term loss ratio of policy π defined to be $\lim_{t\to\infty}(L_\pi(t)/t)$. In both systems, the goal is to choose π so as to minimize $L_\pi(t)$, $\forall t \geq 0$ and/or the long-term loss ratio.

2.2 The Paradigm

In this subsection, we describe a simple, but useful, paradigm for mapping policies used in removal real-time systems to non-removal real-time systems. Formally, let Σ^r be the class of scheduling policies for removal systems. A policy in this class determines when and which customer to schedule into service as well as which, if any, customer to remove from the system prior to either commencing or completing service. Policies in Σ^r have deadline information available to them, and may or may not have service time information available. Policy $\pi \in \Sigma^r$ can be thought as having two types of rules, *scheduling rules* that determine which customer and when to begin servicing it, and *removal rules* that determine which customer and when to remove it from the system. Let Σ^n denote the class of scheduling policies for non-removal systems. Policies in Σ^n behave like removal system policies except that they are not permitted to remove customers. If $\pi \in \Sigma^n$, then π is required to service all customers.

We define a function $\Phi : \Sigma^r \to \Sigma^n$ in the following manner. Consider an arbitrary policy $\pi \in \Sigma^r$. Under $\Phi(\pi)$ there are two priority classes. $\Phi(\pi)$ applies the scheduling rules of π to the higher priority customers without change, without any modification, and the removal rules of π with the modification that high priority customers are moved to a second, low priority queue rather than removed from the system. When there are no customers in the high priority queue under $\Phi(\pi)$, $\Phi(\pi)$ switches to the low priority queue and schedules customers there according to some arbitrary rule, such as FCFS.

Observe that most policies for removal systems are also directly applicable to non-removal systems by following the same scheduling rule, but without removing customers. We define the function $\Lambda : \Sigma^r \to \Sigma^n$ to correspond to this, and refer to it as the *direct mapping.*

A particular policy of interest to us is the earliest deadline policy (ED) which schedules the customer having the smallest deadline into service. It comes in several forms, depending on whether it is implemented in a removal or non-removal system. In the former, ED removes any customer that misses its deadline. In the latter, of course, no removals occur; and it is possible that Λ(ED) will schedule a customer that has already missed its deadline *even though there are customers in*

the queue that have not missed their deadlines. Previous work, [15, 16, 5], on removal systems has been concerned with establishing the optimality of ED.

We expect Φ to exhibit the following properties:

1. for any pair of policies $\pi, \pi' \in \Sigma^r$ such that $\pi \leq \pi'$, then $\Phi(\pi) \leq \Phi(\pi')$;
2. $\Lambda(\pi) \leq \Phi(\pi), \forall \pi \in \Sigma^r$.

Here $\pi \leq \pi'$, means that policy π' performs better than policy π (e.g., lower loss ratio). The first property is particularly useful because, if true, then previous optimality results obtained for removal systems, [15, 16, 5], should also hold for non-removal systems. We expect that our understanding of what constitutes a good policy in a removal system can be used to develop good policies for a non-removal system. The next two sections will provide evidence that these two properties do, indeed, hold.

3 Optimality Results

As mentioned earlier, under certain conditions, ED has been shown to be optimal for removal real-time systems in the sense that it stochastically minimizes the process of the number of the customers that miss their deadlines by time $t > 0$. We will briefly review these results and then establish analogous optimality results for Φ(ED). Results are known for two different types of systems, those in which deadlines are until the beginning of service and those in which deadlines are until the end of service.

All of our results will be based on the following stochastic ordering.

Definition 1 *Let $X, Y \in \Re^n$ be two random variables. X is stochastically smaller than Y, (denote by $X \leq_{st} Y$), iff*

$$Ef(X) \leq Ef(Y), \qquad \text{for all increasing } f.$$

In the case that $n = 1$, it is equivalent to,

$$P(X > t) \leq P(Y > t), \qquad \text{for all } t \in \Re.$$

Definition 2 *Let $\{X_t, t \geq 0\}$ and $\{Y_t, t \geq 0\}$ be two random processes with $X_t, Y_t \in \Re$. The process $\{X_t\}$ is stochastically smaller than the process $\{Y_t\}$, (denoted by $\{X_t\} \leq_{st} \{Y_t\}$), iff*

$$E[f(x_{t_1}, \ldots, x_{t_n})] \leq E[f(y_{t_1}, \ldots, y_{t_n})], \quad \text{for all increasing } f;\ t_i \geq 0;\ i = 1, \cdots, n;\ n = 1, 2, \ldots.$$

Properties of these orderings can be found in [18].

3.1 Optimality Results for Removal Systems

All of the results for removal real-time systems require the following assumptions on the arrival process, service times, and relative deadlines.

Assumption 1 *Arrivals and relative deadlines are described by an arbitrary process. Service times are assumed to be a sequence of independent and identically distributed exponential r.v.'s which are independent of the arrival times and relative deadlines.*

We define the following classes of policies

Σ_p^r : the class of policies for a removal system that never allows the server to idle while there is work in the queue and that allows preemptions. Customers resume service from the point that they were preempted. Furthermore, policies can use deadline information but not service time information when making scheduling or removal decisions.

Σ_{np}^r : the class of policies defined the same as Σ_p^r except that no preemptions are allowed.

Σ_{rp}^r : the class of policies defined the same as Σ_{np}^r except that a customer in service can be aborted and removed from the system. Such a customer cannot resume service.

The following optimality results were established in [15, 16]:

Theorem 1 *Consider a single server removal real-time system where deadlines are to the beginning of service. Under Assumption 1,*

$$\{L_{ED}(t)\} \leq_{st} \{L_\pi(t)\}, \qquad \forall \pi \in \Sigma_{np}^r.$$

Theorem 2 *Consider a single server removal real-time system where deadlines are to the end of service. Under Assumption 1,*

$$\begin{aligned} \{L_{ED}(t)\} &\leq_{st} \{L_\pi(t)\}, \qquad \forall \pi \in \Sigma_p^r, \\ \{L_{ED}(t)\} &\leq_{st} \{L_\pi(t)\}, \qquad \forall \pi \in \Sigma_{rp}^r. \end{aligned}$$

3.2 Optimality Results for Non-Removal Systems

In this subsection, we establish similar results for non-removal systems. As before, we identify two cases, deadlines to the beginning of service and deadlines to the end of service.

We define the following classes of policies

Σ_p^n : the class of policies for a non-removal system that never allows the server to idle while there is work in the queue and that allows preemptions. Customers resume service from the point that they were preempted. Furthermore, policies can use deadline information but not service time information when making scheduling decisions.

Σ^n_{np} : the class of policies defined the same as Σ^n_p except that no preemptions are allowed.

Σ^n_{rp} : the class of policies defined the same as Σ^n_{np} except that a customer in service may be preempted whenever it misses its deadline. No other preemptions are allowed.

3.2.1 Deadline to the Beginning of Service

First we consider the case where the deadline is to the beginning of service, i.e., if a customer cannot start service before its deadline, it is lost. However, once a customer begins service, it is safe and will never be lost.

The result makes use of the following assumption:

Assumption 2 *Arrivals and relative deadlines are described by an arbitrary process. Service times are assumed to form a sequence of independent and identically distributed r.v.'s which are independent of the arrival times and relative deadlines.*

Theorem 3 *Consider a non-preemptive single server non-removal real-time system, where deadlines are to the beginning of service. Under Assumption 2,*

$$\{L_{\Phi(ED)}(t)\} \leq_{st} \{L_\pi(t)\}, \qquad \forall \pi \in \Sigma^n_{np}.$$

Proof: Consider an arbitrary policy $\pi \in \Sigma^n_{np}$. Fix the arrival and service times under π. Let $t_1 < t_2 < \cdots$ denote the times that π schedules customers. Let t_i be the first time that π schedules a customer, say l, which is not the one with the earliest valid deadline. Assume customer k, $k \neq l$, is the one with the earliest valid deadline at time t_i, and that it is scheduled at time $t_j > t_i$. Construct policy π^* which follows the same scheduling rules as π except that it schedules customer k at t_i and customer l at t_j. We interchange the service times of k and l. This is allowed by the assumption that the service times are identically and independently distributed r.v.'s independent of the arrival times and deadlines.

Obviously, all customers, except for k and l, make the same contribution to the number of losses since they are scheduled exactly at the same time instances under both π and π^*. Hence, we only need to focus on customers k and l. Clearly, under π^*, k is safe and l may or may not be lost depending on whether $d_l > t_j$.

We have $L_{\pi^*}(t) = L_\pi(t)$, $0 \leq t < t_i$. We focus on $t \geq t_i$ by considering three cases.

Case 1 ($d_l < t_i$): Customer l is lost under both π and π^*. If $d_k < t_j$, then we have $L_{\pi^*}(t) = L_\pi(t)$, $t_i \leq t < d_k$ and $L_{\pi^*}(t) = L_\pi(t) - 1 < L_\pi(t)$ for $t \geq d_k$. If $d_k \geq t_j$, then $L_{\pi^*}(t) = L_\pi(t)$, $t \geq t_i$.

Case 2 ($t_i \leq d_l < t_j$): In this case $d_l > d_k$ and we have $L_{\pi^*}(t) = L_\pi(t)$, $t < d_k$; $L_{\pi^*}(t) = L_\pi(t) - 1$, $d_k \leq t < d_l$; $L_{\pi^*}(t) = L_\pi(t)$, $d_l \leq t$.

Case 3 ($t_j \leq d_l$): Here there are two subcases according to whether $d_k < t_j$ or not. If $d_k < t_j$, then $L_{\pi^*}(t) = L_\pi(t)$, $t < d_k$ and $L_{\pi^*}(t) = L_\pi(t) - 1$ for $d_k \leq t$. If $t_j \leq d_k$, then $L_{\pi^*}(t) = L_\pi(t)$ for all t.

Therefore,

$$L_{\pi^*}(t) \leq L_\pi(t), \qquad \forall t \geq 0.$$

Obviously, if π^* is not identical to Φ(ED), we can construct another policy π^{**} by applying the Φ(ED) rule at the first time π^* violates the Φ(ED) rule. This procedure can be repeated until it yields the Φ(ED) rule with $L_\pi(t) \leq L_{\Phi(ED)}(t)$. Removal of the conditioning on arrival, service and relative deadlines completes the proof. ∎

Remark. The analogous result for removal systems requires the additional restriction that service times be exponentially distributed r.v.'s whereas service times can have an arbitrary distribution in the preceding theorem. This is because the removal system is not work conserving. Hence, it is not possible to couple all service periods under two different removal system policies. The exponential assumption, howevever, allows departure instances to be coupled. This is sufficient for establishing Theorem 1.

3.3 Deadline to the End of Service

In this subsection, we consider the case that deadlines are until the end of service. We have the following result.

Theorem 4 *Consider a single server non-removal real-time system where deadlines are to the end of service. Under Assumption 1,*

$$\{L_{\Phi(ED)}(t)\} \leq_{st} \{L_\pi(t)\}, \qquad \forall \pi \in \Sigma_p^n,$$
$$\{L_{\Phi(ED)}(t)\} \leq_{st} \{L_\pi(t)\}, \qquad \forall \pi \in \Sigma_{rp}^r.$$

Proof: Consider the first result. Its proof is similar to that of Theorem 3. The difference comes from the rule that preemptions are allowed. Observe that under $\Phi(ED)$, if a customer misses its deadline while in service or there is a new arrival with a smaller deadline, then the one in service is preempted and a new one with the earliest valid deadline is scheduled. Consequently, the departure of a customer at some time t under Φ(ED) is always the one with the smallest valid deadline which arrived prior to t and was still in the system at time t. Owing to the memoryless property of the exponential service time distribution, the newly scheduled customer can take over the remaining service time of the preempted one, and therefore we can still couple the two systems at the same departure epochs. The theorem follows immediately by using the same interchange argument as that used in Theorem 3.

A similar argument can be used to prove the second relation. Note that the independence and exponential assumptions for service times are required in order to handle the preemptions that occur when customers in service miss their deadlines. ∎

Remark. There is no optimality result, either for non-removal systems or removal systems, in the case that *no preemptions (abortions) are allowed for customers in service.*

Unfortunately, the assumptions required to establish the preceding optimality results do not hold for many computer system applications. Thus, we cannot conclude that Φ(ED) is the best policy for these applications. However, the above results provide evidence that the paradigm we have introduced is useful for studying and improving the performance of different policies for these applications. In the next section, we will give such an example in real-time disk I/O scheduling applications.

4 An Application to Real-Time Disk Scheduling

In this section, we first briefly review known scheduling policies and introduce some new policies for a removal real-time disk I/O subsystem. Following this, we apply the mapping Φ, described in Section 2.2, to these policies and compare their performance to each other. We begin with a description of the disk I/O subsystem that will make the subsequent presentation clear.

Abbott and Garcia-Molina [1] were the first to study non-removal real-time disk I/O systems. In this context each I/O request arriving to the disk I/O subsystem carries a real-time constraint. If a request cannot finish the disk access before missing its deadline, it is said to be lost. The I/O subsystem is required to serve every request even though its deadline has expired. The goal is to find a policy that minimizes the fraction of requests that miss their deadlines.

4.1 Characteristics of Disk Accesses

There are several characteristics of disks that need to be identified as they may violate some assumptions required in the last section to establish the optimality of Φ(ED). First, disk service times are not independent of each other, since a request's service time depends on the current arm position which is the cylinder address of the preceding request. Second, disk service times are not exponentially distributed r.v.'s.; and, moreover, they are not independent of the scheduling policies employed. One obvious example is that service times under the *shortest seek time first (SSTF)* policy are smaller than those under the *first come first service (FCFS)* policy [11]. Finally, disk service times can be considered to be quite deterministic, i.e., once you know the current arm position, a request's cylinder address and the amount of data to be transferred, the disk service time can be estimated. Because of this property, scheduling policies can make use of the service time information to achieve a better performance. Hence, Theorems 3 and 4 are not immediately

applicable. On the other hand, because of the availability of disk service time information, we can use this information to avoid scheduling a request which may be lost during service.

We take the approach of [1] and introduce the notion of deadline feasibility. A deadline is *feasible* if it is greater than the current time plus the potential service time which consists of seek time, rotational latency, and transfer time[1]. While Abbott and Garcia-Molina, [1], assumed that the latency and transfer time are constants, we assume that the latency is uniformly distributed in the range of $[0, 16.7]$ milliseconds. Consequently, we use the average latency to estimate the feasibility of a deadline. Other more conservative or radical strategies can use the maximum or minimum latency to estimate feasibility. An even more sophisticated strategy is also possible which uses hardware similar to RPS (*rotational position sensing*) to sense the arm position under rotation; it calculates the arm's angular position after the seek to the right track and obtains the rotational latency to the destination block. Notice that it is hard to achieve a precise value of the latency because of the time needed for the calculations. However, since the time required to perform such calculations is considerably less than the rotational latency, the resulting latency estimate should be quite accurate. This strategy is justified if the rotational latency is assumed to dominate the disk access time as suggested in [14]. It requires a more intelligent disk controller and driver. We will further investigate these alternatives later in this paper.

It is interesting to observe that the feasibility check, which is specific to the disk scheduling context, introduces the option of removing a request if its deadline is found to be infeasible even though it is still valid. Finally, in most cases, we assume that the disk I/O service is non-interruptable, i.e., once a request begins a disk access, it will go through to the end. We will, however, quantify the performance benefits of providing the capability of interrupting the processing of a request which misses its deadline during service.

4.2 Real-Time Disk I/O Scheduling Policies

In this subsection, we first describe several scheduling policies for removal real-time disks. Then we obtain the corresponding policies for non-removal model by applying the paradigm Φ, or simply by using the direct mapping Λ. Specifically, we study four scheduling rules in Σ^r. Each comes in two variations according to whether they perform feasibiliy checks or not. If a feasibility check is not performed, then requests are removed when they miss their deadlines prior to completing service. If a feasibility check is peformed, they may be removed if their deadlines are not feasible. Policies using the latter removal rule will be identified by the suffix "-F".

- **The earliest deadline policy (ED)**

  ED schedules a request with the earliest deadline.

[1] We assume that each disk has its own data path and therefore the channel delay is ignored

- **The SSEDV policy**

  The SSEDV (*shortest seek and earliest deadline by value*) policy was first introduced in [7], and was shown to exhibit good performance in a removal real-time transaction system. SSEDV works as follows: all arriving requests join a waiting queue sorted by deadlines. A window of size k is defined as the first k requests in the queue. The SSEDV policy always schedules among those in the window according to a priority value calculated by $\alpha s + (1-\alpha)d$, where s is the seek distance to serve the request, d is the remaining life time to the deadline, and α is a scheduling parameter, ranging from 0 to 1, which can be used to adjust the weights placed on service time and time constraint. The smaller the priority value, the higher the priority. Observe that when $\alpha = 0$, it corresponds to ED, and when $\alpha = 1$, it applies the *shortest seek time first (SSTF)* rule to requests within the window. According to our experience, a window size of 3 and $\alpha = 0.1$ provide good performance for a wide variety of workloads and deadline settings. When the window size is equal to 1, the policy degenerates to ED.

- **The FD-SCAN policy**

  The FD-SCAN policy was first introduced in [1] for a single disk non-removal real-time system, and is based on the classical SCAN rule. A modified version of FD-SCAN for removal system was studied in [7]. In the modified version, whenever a request misses its deadline, FD-SCAN throws it away. The cylinder location of the request with the earliest feasible deadline is used to determine the scan direction. At each scheduling instance, all requests are examined to determine which has the earliest feasible deadline. After selecting the scan direction, the arm moves in that direction and serves all requests along the way. Note that neither the original version or this modified version performs a feasibility check prior to servicing a customer. Instead, the feasibility check is made only to choose the scan direction. Applying the direct mapping Λ to the modified version regenerates the original version of FD-SCAN.

- **The SSTF policy**

  This is the well-known *shortest seek time first (SSTF)* discipline, which always schedules the request closest to the current arm position.

4.3 Performance Comparisons

The performance of the policies discussed in the last subsection is evaluated in this subsection. In particular, we first briefly revisit a removal real-time disk I/O subsystem to study the benefits of including feasibility checks to different policies. We then focus on a non-removal disk I/O subsystem. We begin by studying the behavior of different variations of the earliest deadline policy, Λ(ED), Φ(ED), and Φ(ED-F). Since feasibility checks are expected to improve performance, we will then study the four policies Φ(ED-F), Φ(SSEDV-F), Φ(FD-SCAN-F), and Φ(SSTF-F). The performance

of the Λ(FD-SCAN) policy is provided for the purpose of comparison. All of the results are obtained via simulation. Each experimental result is the average over 40 runs, and within each run, 50,000 I/O requests are served. In some runs where the deadline setting is loose, each run includes 60,000 I/O requests. Ninety-five percent confidence intervals are achieved, which is less than 1% of the values of point estimates. With this number of samples, the values of the point estimates and the length of confidence intervals are observed to converge to a steady state. These confidence intervals are not shown in the graphs in order to maintain clarity.

4.3.1 Simulation Parameter Settings

In order to provide comparisons, most of the parameters used in the simulation model are taken from [1]. These parameters are summarized in Table 1. The arrival process is assumed to be Poisson with rate λ, where λ varies from 22 to 40 to represent different workloads. Disk service times are calculated as,

$$Access(n) = Seek(n) + Latency + Transfer$$

where n is the seek distance, $Latency$ is uniform in $[0, 16.7]$, $Transfer$ is a constant, and

$$Seek(n) = Acce\_Time + Disk\_Factor \times \sqrt{n}$$

The average disk access time is 25 msec. The deadline of each request is obtained as

$$Deadline = Arrival\_Time + Average\_Access\_Time + Slack\_Time$$

where $Slack\_Time$ is chosen uniformly from the range $[Min\_Slack, Max\_Slack]$. Min_Slack is set to 10 ms in our experiments. While most of the time Max_Slack is set to 100 ms, we also vary it from 50 ms to 500 ms. Under these parameter settings, the disk utilization varies from 55% to 95% (Table 2). For SSEDV-F and Φ(SSEDV-F), the default window size and scheduling parameter α are set to 3 and 0.1, respectively. The sensitivity of varying these two parameters is shown in the next subsection.

4.3.2 Performance Results

The performance metric of most interest is the loss ratio. In the following, as a complement of our previous work in [7], we first revisit a removal real-time disk system. Then we will focus on the performance comparisons of various policies of most interest in a non-removal system. The results will show that using the paradigm Φ can considerably improve the system performance, and that the disk scheduling deadline feasibility check can significantly reduce the request loss ratio. Among all of these policies, Φ(SSEDV-F) is shown to be the best, and Φ(ED-F) performs reasonably well.

Num_Tracks	1000
Acce_Time	5 ms
Disk_Factor	0.6
Latency	Uniform in $[0, 16.7]$
Transfer	1.5 ms
Avg_Access_Time	25 ms
Min_Slack	10 ms
Max_Slack	100 ms
Window_Size for Φ(SSEDV-F)	3
α for Φ(SSEDV-F)	0.1

Table 1: Simulation Parameters.

Workload (λ)	22	26	30	36	40
Disk Util.	0.5521	0.6505	0.7479	0.8817	0.9579

Table 2: Disk Utilizations.

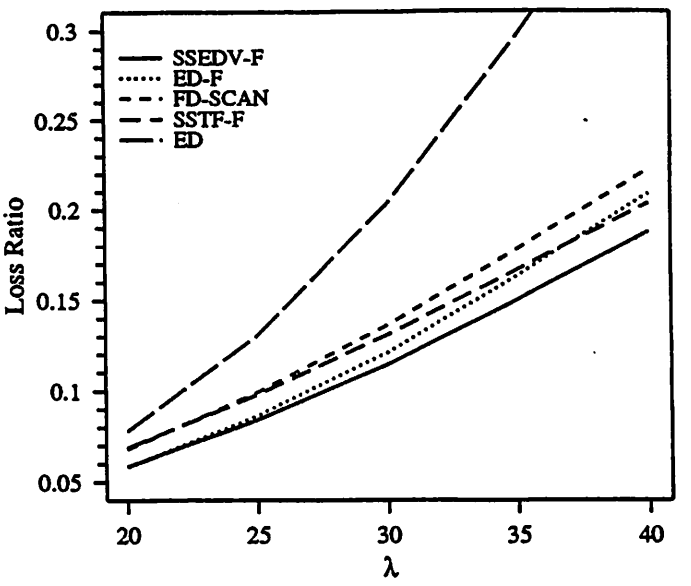
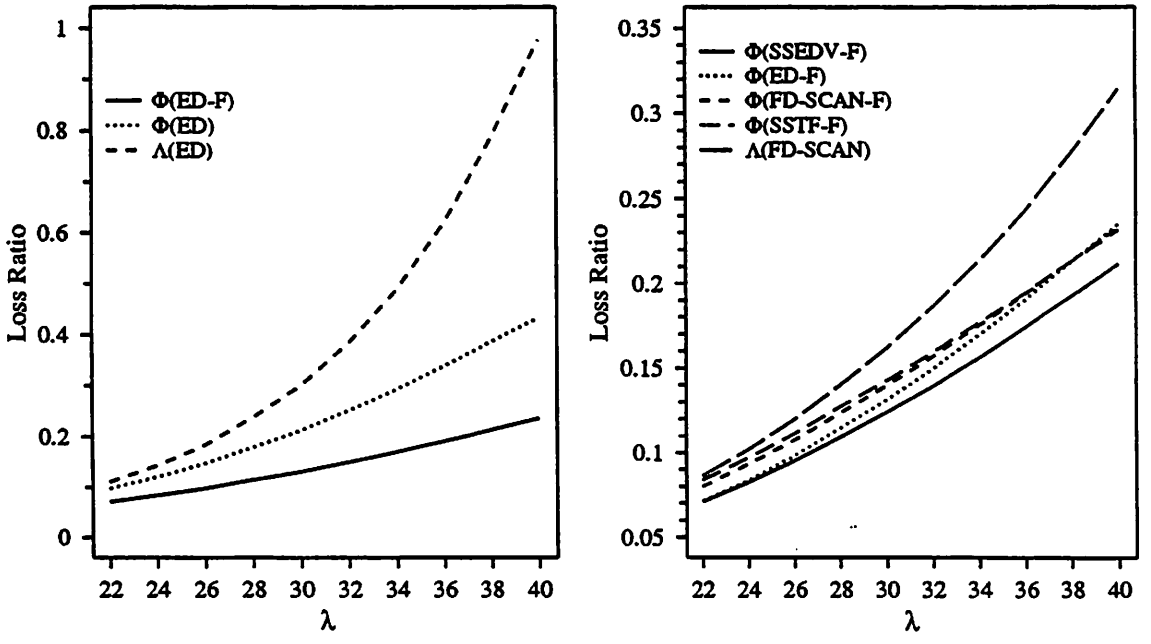


Figure 2: Results for Removal Disk System.

When preemption is allowed, we observe that the strategy which allows a new arrival to preempt the request in service only when it has been lost can lead to some performance improvement.

Removal real-time disk I/O subsystems

First, we consider a real-time disk where missed requests are removed without being scheduled. Although some of these policies have been reported in a previous study [7], the purpose here is to verify the first property of the paradigm Φ given in section 2.2. ¿From Figure 2 and Figure 3(b) (to be discussed later in this subsection), we observe that although the absolute differences in the performances of policies in removal disks may differ from those in non-removal disks, the relative orderings of these policies are the same for both systems. In addition, the ED-F and SSTF-F policies, which are simple and exhibit very good performance, were not included in our previous studies. Clearly, the addition of feasibility checks can provide significant performance benefits.



(a). Variations of the ED Policy.

(b). Comparison of Different Policies.

Figure 3: Performance of Different Policies for Non-Removal Disk System.

Variations of ED

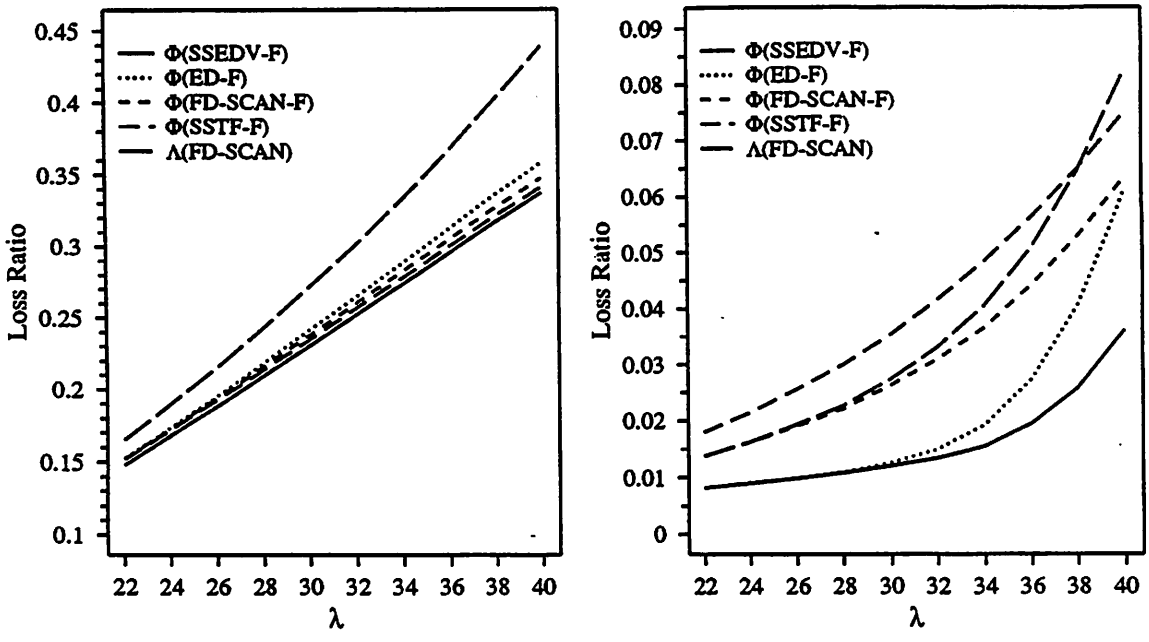
The purpose of this experiment is to study how different variations of ED behave in a non-removal disk I/O context (Figure 3(a)). As expected, Λ(ED) performs poorly as workload increases. This is because Λ(ED) always selects the request with the smallest deadline no matter whether it has been lost or not. Moreover, serving a missed request may further block other requests. The policy Φ(ED) outperforms Λ(ED) by avoiding scheduling missed requests whenever there is a request

with valid deadline waiting to be served. While requests scheduled by Φ(ED) may be lost during service, Φ(ED-F) tries to overcome this defect by using service time information to predict such a possibility, i.e., to perform a feasibility check. Consequently, Φ(ED-F) achieves up to a 45% improvement over Φ(ED), and Φ(ED), in turn, outperforms Λ(ED) by up to 55%. This illustrates the effectiveness of the paradigm. A common feature of these variations of the ED is that no effort is made to reduce the disk access time.

Loss ratio under different polices

Figure 3(b) shows the performance of various scheduling policies in our real-time disk I/O model. We observe that Φ(SSEDV-F) is the best and outperforms the next best policy, Φ(ED-F), by 10% at high loads. This is because Φ(SSEDV-F) combines the deadlines and the seek distances in making its decisions while Φ(ED-F) only uses the former. It is interesting to observe that Φ(FD-SCAN-F), which avoids serving infeasible requests as it scans, achieves up to a 25% improvement over Λ(FD-SCAN).

Both Φ(FD-SCAN-F) and Φ(SSTF-F) are observed to perform slightly worse than Φ(ED-F), since the the former two are characterized by placing more weight on reducing service times than on handling time constraints. In the following, we will study the performance of these policies under different deadline settings.



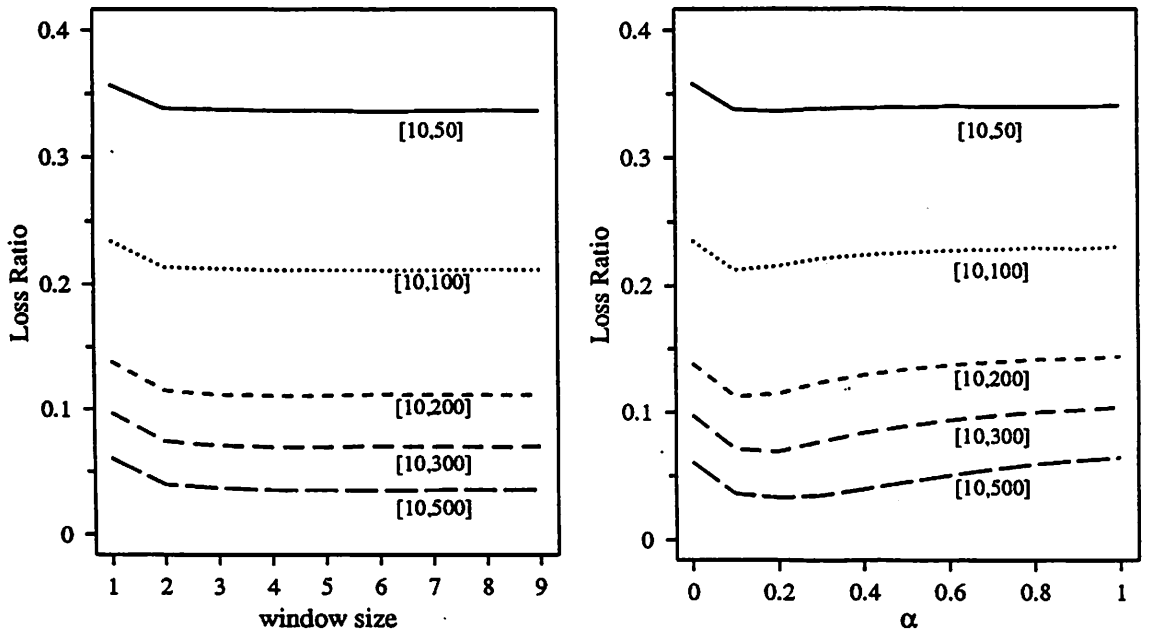
(a). Tight Deadline Setting. [10,50]

(b). Loose Deadline Setting. [10,500]

Figure 4: Varying Deadline Settings for Non-Removal Disk System.

Varying deadline settings

In this experiment, we study the effect of different deadline settings on performance. In particular, we maintain *Min_Slack* at 10 ms, and vary *Max_Slack* from 50 ms to 500 ms. Figure 4 shows the results for the two extreme cases. When *Max_Slack* equals 200 ms and 300 ms, the quality of these policies is the same as shown in Figure 4(b). When deadlines are extremely tight, all of these policies perform nearly the same, with Φ(SSEDV-F) outperforming Φ(SSTF-F) and Φ(FD-SCAN-F) only by 1% and 3%, respectively (Figure 4(a)). However, when deadlines become loose, Φ(SSEDV-F) may achieve up to a 40% performance improvement over all other policies (Figure 4(b)). One lesson we learn from this experiment is that in an environment where deadlines are extremely tight, reducing disk access times is more important than accounting for I/O requests' time constraints. But the reverse is true when deadlines are not very tight. Since rarely is a system designed to work in an environment with extremely tight deadlines, we conclude that the design of good policies should place more weight on time constraints.



(a). Varying Window Size in Φ(SSEDV-F) (α=0.1, λ=40)

(b). Varying Parameter α in Φ(SSEDV-F) (window=3, λ=40)

Figure 5: Sensitivity of Parameters for Φ(SSEDV-F)

Sensitivity of parameters for Φ(SSEDV-F)

In Figure 5, we illustrate the sensitivity of loss ratio to variations in the window size and the scheduling parameter α for the Φ(SSEDV-F) policy. Since both of these two parameters are more

sensitive to the performance when the systems is highly loaded, we show the case of high workload ($\lambda = 40$) here. From Figure 5(a), we observe that a window size of 3 is adequate for wide variety of deadline settings. Further increasing window size achieves little improvement. In Figure 5(b), we plot how system performance is affected when we change the scheduling parameter α under wide range of deadline settings. We see that a setting of $\alpha = 0.1$ is good for a variety of operational environments.

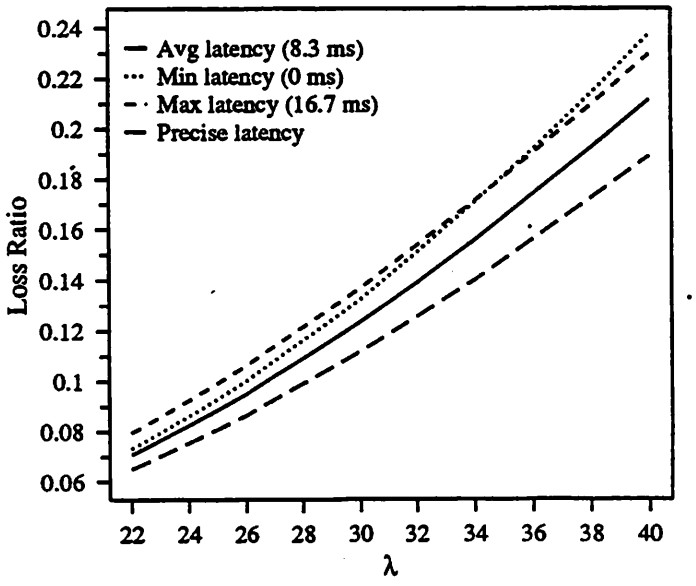
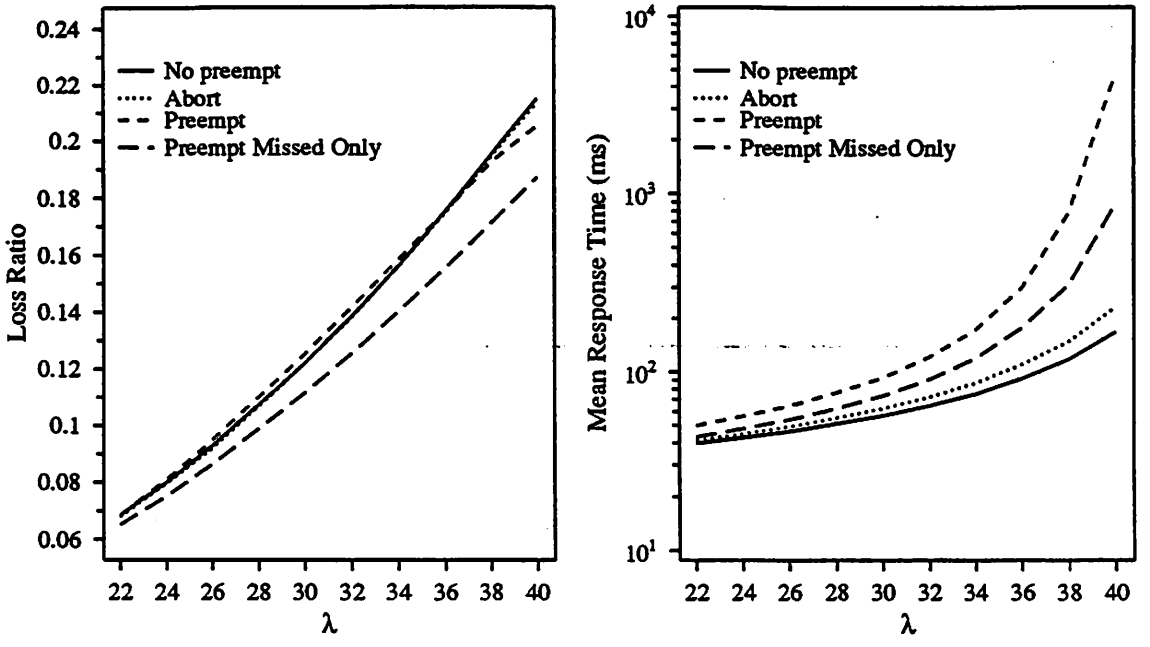


Figure 6: Alternate Strategies for Checking Feasibility under Φ(SSEDV-F).

Alternate ways to check feasibility

From the above discussion, we see that using the disk service time information to perform a feasibility check when scheduling may result in considerable performance improvement. While our previous discussion uses the average latency (8.3 ms) for feasibility check, here we consider other alternatives——maximum latency (16.7 ms), minimum latency (0 ms), as well as the exact latency. The results are illustrated in Figure 6. The *maximum strategy* is not good because it over-estimates the potential disk access time. This may result in some requests being rejected for service which in fact can make their deadlines. On the other hand, the *minimum strategy* under-estimates the potential access time and results in more request losses during service. As much as a 10% improvement over the performance of the *average strategy* can be achieved by using the exact value of the latency. Since the precise latency calculation requires more complicated hardware support,

there is a performance/cost trade-off.



(a). Request Loss Ratio.

(b). Mean I/O Response Time.

Figure 7: Preemption vs. Non-Preemption, Φ(SSEDV-F).

Preemptions vs. Non-Preemptions

While our discussions above assume disk service to be non-interruptable, here we relax the restriction and allow preemptions and/or abortions to occur during service. Notice that in a non-removal model, if a request is preempted while in service, it must be rescheduled later on, and there is no way for it to resume from the point of preemption. Hence, in addition to requiring a more sophisticated device controller and driver, preemptions also create more workload for the system. Therefore, there is a trade-off between the cost paid and the benefit gained (if any) by allowing preemptions. In the following, we identify three strategies for allowing preemptions (abortions), and compare their performance against non-preemptive policies. In doing so, we ignore the extra overhead incurred by the device controller and driver.

- The first strategy, called A, allows the controller is intelligent enough to preempt a request in service if its deadline expires and at the same time there are some requests with valid deadlines waiting to be served.

- The second strategy, in addition to allowing preemptions as in A, allows a new arrival to preempt the one in service if its deadline has expired. Under this strategy, when a preemption

occurs, all the requests waiting to be served must have missed their deadlines. We refer to this strategy as *preempt missed only* (PMO).

- The third strategy, called P, differs from strategy PMO by allowing a new arrival not only to preempt the one in service if it has been lost but also if the new arrival has a smaller deadline.

The effects of these strategies are shown in Figure 7 (a) for Φ(SSEDV-F). Similar behavior has been observed for other policies. Strategy A achieves no gains because Φ(SSEDV-F) effectively avoids the occurrence of requests missing deadlines during service. Strategy P performs basically the same as the non-preemptive strategy, but produces appreciably higher mean response times (Figure 7 (b)). This is because strategy P creates too much additional work for the system. It is, therefore, not a good strategy to pursue. The only strategy which noticeably improves performance is PMO, which may lead up to a 13% improvement in loss ratio at the cost of higher mean response time (especially for those missed requests) and of more complicated hardware and software support.

5 Summary

The objective of this paper was to develop scheduling policies suitable for non-removal real-time systems. This was done by studying the relationship between such systems and removal real-time systems. A paradigm was presented for mapping policies suitable for removal systems to policies suitable for non-removal systems. Both theoretical and empirical justifications were given on the utility of this paradigm. First, we showed that several policies known to be optimal in removal real-time systems map into policies that are optimal in non-removal real-time systems. Second, we applied the paradigm in the context of real-time disk subsystems. Using this paradigm, we were able to obtain high performance disk scheduling policies for a non-removal system. In the course of doing so, we also developed several new scheduling policies for removal disk systems that outperform previously known policies.

References

[1] Abbott, R. and Garcia-Molina, H., "Scheduling I/O Requests with Deadlines: A Performance Evaluation," *Proc. Real-Time System Symp.'90*, Dec. 1990.

[2] Baccelli, F., Boyer, P., and Hebuterne, G., "Single Server Queue with Impatient Customers," *Adv. Appl. Prob.*, **16**, pp.887-905, 1984.

[3] Baccelli, F. and Trivedi, K. S., "A Single Server Queue in a Hard Real-Time Environment," *Oper. Res. Lett.*, 4, No.4, pp.161-168, 1985.

[4] F. Baccelli, Z. Liu, D. Towsley, "Extremal Scheduling of Parallel Processing with and without Real-Time Constraints," to appear in *Journal of the ACM.*

[5] Bhattacharya, P. P. and Ephremides, A., "Optimal Scheduling with Strict Deadlines," *IEEE Trans. on Automatic Control,* **34**, No.7, July 1989.

[6] J.P.C. Blanc, P.R. de Waal, P. Nain, D. Towsley, "Optimal Control of Admission to a Multiserver Queue with Two Arrival Streams," *IEEE Transactions on Automatic Control*, **37**, 6, 785-797, June 1992.

[7] Chen S., Stankovic, J. A., Kurose, J. F., and Towsley, D., "Performance Evaluation of Two New Disk Scheduling Algorithms for Real-Time Systems," *COINS Tech. Report 90-77, UMass*, Aug. 1990.

[8] Dertouzos, M., "Control Robotics: The Procedural Control of Physical Processes," *Proc. of the IFIP Congress*, pp.807-813, 1974.

[9] B. T. Doshi, E. H. Lipper, "Comparisons of Service Disciplines in a Queueing System with Delay Dependent Customer Behavior," *Applied Probability - Computer Science: The Interface*, Vol. II (Eds. R. L. Disney, T.J. Ott) Cambridge, MA:Birkhauser, pp. 269-301, 1982.

[10] B.T. Doshi, "An M/G/1 Queue with a Hybrid Discipline," *Bell Syst. Tech. J.*, **62**, 5, pp. 1251-1271, 1983.

[11] Hofri, M., "Disk Scheduling: FCFS vs. SSTF Revisited," *ACM Communications*, **23**, No.11, Nov. 1980.

[12] M.H. Kallmes, D. Towsley, C.G. Cassandras, "Optimality of the Last-in-First-out (LIFO) Service Discipline in Queueing systems with Real-Time Constraints," *Proceedings of the 28-th IEEE Conference on Decision and Control*, pp. 1073-1074, December 1989.

[13] Z. Liu, D. Towsley, "Effects of Service Disciplines in $G/GI/s$ Queueing Systems," COINS Technical Report 92-26, March 1992.

[14] Ng, S. W., "Improving Disk Performance via Latency Reduction," *IEEE Trans. on Computers,* 40, No.1, pp.22-30, Jan. 1991.

[15] Panwar, S. S., Towsley, D., and Wolf, J. K., "Optimal Scheduling Policies for a Class of Queues with Customer Deadlines to the Beginning of Service," *J. ACM,* **35**, No.4, pp.832-844, Oct. 1988.

[16] Panwar, S. S. and Towsley, D., "On the Optimality of the STE Rule for Multiple Server Queues that Serve Customers with Deadlines," *COINS Tech. Rep. 88-81,* July 1988.

[17] Pinedo, M., "Stochastic Scheduling with Release Dates and Due Dates," *Operations Research*, **31**, pp.559-572, 1983.

[18] Ross, S. M., *Stochastic Processes*, John Wiley and Sons, 1983.

[19] Su, Z-S.,and Sevcik, K. C., "A Combinatorial Approach to Scheduling Problems," *Operations Research*, **26**, pp.836-844, 1978.

[20] D. Towsley, F. Baccelli, "Comparisons of Service Disciplines in a Tandem Queueing Network with Real-Time Constraints," *Operations Research Letters*, **10**, 49-55, February 1991.

[21] Towsley, D. and Panwar, S. S., "On the Optimality of Minimum Laxity and Earliest Deadline Scheduling for Real-Time Multiprocessors," *Proc. Euromicro'90 Workshop on Real-Time*, June 1990, pp.17-24.

[22] Wu, Z. J., Luh, P. B., Chang, S. C., and Castanon, D. A., "Optimal Control of a Queueing System with Two Interacting Service Stations and Three Classes of Impatient Tasks," *IEEE Trans. on Automat. Contr.*, **33**, pp.42-49, 1988.