Online Real-Time Preemptive Scheduling of Jobs with Deadlines on Multiple Machines[*]

Bhaskar DasGupta[†] and Michael A. Palis
Department of Computer Science
Rutgers University
Camden, NJ 08102, USA
Email: {bhaskar,palis}@crab.rutgers.edu

June 13, 2000

Abstract

In this paper, we derive bounds on performance guarantees of online algorithms for real-time preemptive scheduling of jobs with deadlines on K machines when jobs are characterized in terms of their minimum stretch factor α (or, equivalently, their maximum execution rate $r = 1/\alpha$). We consider two well known preemptive models that are of interest from practical applications: the *hard real-time scheduling* model in which a job must be completed if it was admitted for execution by the online scheduler, and the *firm real-time scheduling* model in which the scheduler is allowed not to complete a job even if it was admitted for execution by the online scheduler. In both models, the objective is to maximize the sum of execution times of the jobs that were executed to completion, preemption is allowed, and the online scheduler must immediately decide, whenever a job arrives, whether to admit it for execution or reject it. We measure the competitive ratio of any online algorithm as the ratio of the value of the objective function obtained by this algorithm to that of the best possible offline algorithm. We show that no online algorithm can have a competitive ratio greater than $1 - (1/\alpha) + \varepsilon$ for hard real-time scheduling with $K \geq 1$ machines and greater than $1 - (3/(4\lceil\alpha\rceil)) + \varepsilon$ for firm real-time scheduling on a single machine, where $\varepsilon > 0$ may be arbitrarily small, even if the algorithm is allowed to know the value of α in advance. On the other hand, we exhibit a simple online scheduler that achieves a competitive ratio of at least $1 - (1/\alpha)$ in either of these models with K machines. The performance guarantee of our simple scheduler shows that it is in fact an *optimal scheduler* for hard real-time scheduling with K machines. We also describe an alternative scheduler for firm real-time scheduling on a single machine in which the competitive ratio does not go to zero as α approaches 1. Both of our schedulers do not know the value of α in advance.

1 Introduction

The need to support applications with real-time characteristics, such as speech understanding and synthesis, animation, and multimedia, has spurred research in operating system frameworks that

[*]A preliminary version of this paper will appear in 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX), September 2000.

[†]Research supported by NSF Grant CCR-9800086.

provide quality of service (QoS) guarantees for real-time applications that run concurrently with traditional non-real-time workloads [8, 9, 11, 12, 13, 14, 19, 22, 23, 24, 27]). In such a framework, a real-time application negotiates with the resource manager a range of "operating levels" at which the application can run depending on the availability of resources. Based on the current state of the system, the resource manager may increase or decrease the application's operating level within this pre-negotiated range. As an example, consider an application that displays video frames over a network link. Such an application may require a nominal rate of 30 frames/second, but if it is not possible to achieve this rate, the rate may be reduced to 15 frames/second by skipping every other frame and still achieve reasonable quality. If there are still inadequate system resources, the rate may be further reduced to say 7.5 frames/second (by skipping every fourth frame). However, below this minimum rate the application would produce completely inadequate performance and hence should not be run.

In this paper, we consider the problem of scheduling real-time jobs for the case when the job's "operating level" is characterized by its *stretch factor*, which is defined as ratio of its response time (i.e., total time in the system) to its execution time. Specifically, we consider the problem of scheduling a set of n independent jobs, $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ on K machines $M_1, M_2, \ldots, M_K$. Each job J_i is characterized by the following parameters: its *arrival time* $a(J_i)$, its *execution time* $e(J_i, j)$ on machine M_j, and its *stretch factor* $\alpha(J_i)$ ($\alpha(J_i) \geq 1$), or equivalently its *rate* $r(J_i) = 1/\alpha(J_i)$ ($0 \leq r(J_i) \leq 1$). The parameter $\alpha(J_i)$ determines how late J_i may be executed on machine M_j; specifically, J_i must be completed no later than its *deadline* $d(J_i) = a(J_i) + e(J_i, j)\alpha(J_i)$ on machine M_j. A valid schedule for executing these jobs is one in which each job J_i is scheduled on at most one machine, each machine M_j executes only one job at any time and a job is executed only between its arrival time and its deadline. Preemption is allowed during job execution, i.e., a job may be interrupted during its execution and its processor may be allocated to another job. However, migration of jobs is not allowed, i.e., once a job is scheduled to run on a specific machine, it can only be executed on that machine.

We are interested in *online* preemptive scheduling algorithms based on two real-time models. In the *hard real-time model*, every job that is admitted by the system *must* be completed by its deadline or the system will be considered to have failed. This in contrast to a *soft real-time system* [20] that allows jobs to complete past their deadlines with no catastrophic effect except possibly some degradation in performance. In [1] Baruah *et. al* considered a special case of a soft real-time system, called a *firm real-time system*, in which no "value" is gained for a job that completes past its deadline. We consider both the hard and firm real-time models in this paper. In both cases, we are interested in online scheduling algorithms that maximizes the *utilization*, which is defined as the sum of the execution times of all jobs that are completed by their deadlines. Notice that in the hard real-time model, the scheduler must only admit jobs that are guaranteed to complete by their deadlines. In contrast, in the firm real-time model, the scheduler may admit some jobs that do not complete by their deadlines, but such jobs do not contribute to the utilization of the resulting schedule. We measure the performance of the online algorithm in terms of its competitive ratio, which is the ratio of the utilization obtained by this algorithm to that of the best possible offline algorithm.

For a given set of jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, an important parameter of interest in adaptive rate-controlled scheduling is their *maximum* execution rate $r(\mathcal{J}) = \max\{r(J_i) \mid 1 \leq i \leq n\}$ (or, equivalently, their *minimum* stretch $\alpha(\mathcal{J}) = \min\{\alpha(J_i) \mid 1 \leq i \leq n\}$). In fact, it is not difficult to see that without any bound on the execution rates of jobs no online algorithm for the hard real-time scheduling model has a competitive ratio greater than zero in the worst case (unless additional assumptions are made about the relative execution time of the jobs [18]). However, natural applications in rate-controlled scheduling leads to investigation of improved bounds on the competitive ratios of online algorithms with a given a priori upper bound on the execution rates of jobs. Because the stretch factor metric has been widely used in the literature (e.g., see [6, 7, 21]), we choose to continue with the stretch factor characterization of jobs (rather than the rate characterization) in the rest of the paper and present all our bounds in terms of the stretch factors of jobs.

The scheduling problem for jobs with deadlines (both preemptive and non-preemptive versions) has a rather rich history. Below we just provide a synopsis of the history, the reader is referred to a recent paper such as [4] for more detailed discussions. The offline non-preemptive version of the problem for a single machine is NP-hard even when all the jobs are released at the same time [25]; however this special case has a fully polynomial-time approximation scheme. The offline preemptive version of the scheduling problem was studied by Lawler [17], who found a pseudo-polynomial time algorithm, as well as polynomial time algorithms for two important special cases. Kise, Ibaraki and Mine [15] presented solutions for the special case of the offline non-preemptive version of the problem when the release times and deadlines are similarly ordered. Two recent papers [3, 5] considered offline non-preemptive versions of the scheduling problems on many related and/or unrelated machines and improved some of the bounds in a previous paper on the same topic [4, 26]. On-line versions of the problem for preemptive and nonpreemptive cases were considered, among others, in [2, 16, 18]. Baruah et. al. [2] provide a lower and upper bound of 1/4 for one machine and an upper bound of 1/2 for two machines on the competitive ratio for online algorithms for firm real-time scheduling. Lipton and Tomkins [18] provide an online algorithm with a competitive ratio of $O(1/(\log\Delta)^{1+\varepsilon})$ for scheduling intervals in the hard real-time model, where Δ is the ratio of the largest to the smallest interval in the collection and $\varepsilon > 0$ is arbitrary, and show that no online algorithm with a competitive ratio better than $O(1/\log\Delta)$ can exist for this problem. Some other recent papers that considered various problems related to stretch factors of jobs (such as minimizing the average stretch factor during scheduling) are [6, 7, 21].

The following notations and terminologies are used in the rest of paper. A_K denotes an online preemptive scheduling algorithm for K machines and OPT_K is an optimal offline preemptive scheduling algorithm for K machines. When $K = 1$, we will simply denote them by A and OPT, respectively. Unless otherwise stated, n denotes the number of jobs. For a given set of jobs $\mathcal{J}$, $U_{A_K}(\mathcal{J})$ (respectively, $U_{OPT_K}(\mathcal{J})$) denotes the processor utilization of A_K (respectively, of OPT_K). We say that an online scheduler A_K has *competitive ratio* $\rho(A_K, \alpha), 0 \leq \rho(A_K, \alpha) \leq 1$, if and only if

$$\frac{U_{A_K}(\mathcal{J})}{U_{OPT_K}(\mathcal{J})} \geq \rho(A_K, \alpha)$$

for every job set $\mathcal{J}$ with $\alpha(\mathcal{J}) \geq \alpha$. Finally, A_K has a time complexity of $O(m)$ if and only if each decision of the scheduler can be implemented in time $O(m)$. Our main results are as follows:

- In Section 2, we show that $\rho(A_K, \alpha) \leq 1 - (1/\alpha) + \varepsilon$ for hard real-time scheduling, and $\rho(A, \alpha) \leq 1 - 3/(4\lceil\alpha\rceil) + \varepsilon$ for firm real-time scheduling, where $\varepsilon > 0$ may be arbitrarily small.

- In Section 3, we design a simple $O(n)$ time online scheduler A_K with $\rho(A_K, \alpha) \geq 1 - (1/\alpha)$ for either the hard real-time or the firm real-time scheduling model.
- In Section 4, we describe an online scheduler for firm real-time scheduling on a single machine in which the competitive ratio does not go to zero as α approaches one.

Notice that the upper and (almost matching) lower bounds for the competitive ratio for hard real-time scheduling for K machines is independent of K. This should be contrasted with the offline case where the competitive ratio may decrease with increase in K if the machines are related [3, 4, 5]. Also notice that the performance guarantee of our simple scheduler shows that it is in fact an *optimal scheduler* for hard real-time scheduling with K machines. Finally, both the schedulers described in Sections 3 and 4 do not need to know the value of α in advance.

2 Upper Bounds of the Competitive Ratio

In this section, we prove upper bounds of the competitive ratio for hard real-time scheduling with K machines and firm real-time scheduling for a single machine. We need the following simple technical lemma which applies to both hard and firm real-time scheduling.

Lemma 1 *Assume that $\alpha \geq 1$ is an integer and we have a set of α jobs, each with stretch factor α, execution time $x > 0$, and the same arrival time t. Then, it is not possible to execute all these α jobs, together with another job of positive execution time, during the time interval $[t, t + \alpha x]$, in either the hard or the firm real-time scheduling model.*

Proof. The common deadline of all these α jobs is $t + \alpha x$. Since each job has execution time x, the lemma follows. □

2.1 Hard Real-Time Scheduling for K Machines

Theorem 2 *For every $\alpha \geq 1$, every $K \geq 1$, any arbitrarily small $\varepsilon > 0$, and for any online preemptive scheduling algorithm A_K, $\rho(A_K, \alpha) \leq 1 - (1/\alpha) + \varepsilon$.*

Proof. For later usage in the proof, we need the following inequality: for any $a > 1$, $\frac{1}{1+\frac{a^2}{a-1}} \leq \frac{a-1}{a}$. This is true since $\frac{1}{1+\frac{a^2}{a-1}} \leq \frac{a-1}{a} \Leftrightarrow a^2 + a - 1 \geq a \Leftrightarrow a^2 \geq 1$.

It is sufficient to prove the theorem for all $\varepsilon < 1/\alpha$. The proof proceeds by exhibiting explicitly a set of jobs $\mathcal{J}$ with stretch factor at least α for which $\rho(A_K, \alpha) \leq 1 - (1/\alpha) + \varepsilon$ for any scheduling algorithm A_K. All jobs in the example have stretch factor α. Furthermore, any job J in our example has the same execution time on all machines, hence we will simply use the notation $e(J)$ instead of $e(J, j)$. We break the proof into two cases depending on whether α is an integer or not.

Case 1: α is an integer. Let $a = \frac{\alpha}{1-(\alpha\varepsilon)}$ and $0 < \delta < \frac{1}{K^3}$ be any value. Note that $a > \alpha \geq 1$. The strategy of the adversary is as follows:

(a) At time 0, the adversary generates a job J_0 with $e(J_0) = 1$ and $\alpha(J_0) = \alpha$. A_K must accept this job because otherwise the adversary stops generating any more jobs and the competitive ratio is zero.

(b) At time δi, for $1 \le i \le K-1$, the adversary stops (does not generate any more jobs) if A_K did not accept J_{i-1}; otherwise it generates a job J_i with $e(J_i) = \frac{a^2}{a-1}\sum_{k=0}^{i-1} e(J_k)$ and $\alpha(J_i) = \alpha$. Notice that $e(J_i) \ge 1$ for all $0 \le i < K$.

First, we note that A_K is forced to accept all the jobs. Otherwise, if A_K did not accept job J_i (for some $1 \le i \le K-1$), then since OPT_K can execute all of the jobs $J_0, J_1, J_2, \ldots, J_i$ (each on a different machine), we have

$$\rho(A_K, \alpha) = \frac{\sum_{j=0}^{i-1} e(J_j)}{\sum_{j=0}^{i} e(J_j)} = \frac{1}{1 + \frac{e(J_i)}{\sum_{j=0}^{i-1} e(J_j)}} = \frac{1}{1 + \frac{a^2}{a-1}} \le \frac{a-1}{a} = 1 - (1/\alpha) + \varepsilon$$

as promised. Next, we show that A_K cannot execute two of these jobs on the same machine. Suppose that job J_i $(i > 0)$ is the first job that is executed on the same machine executing another job J_k for some $k < i$. We know that J_i arrives after J_k. Hence, if J_i has to execute to completion on the same machine together with J_k, then we must have $e(J_i) \le (\alpha - 1)e(J_k)$. However,

$$e(J_i) = \frac{a^2}{a-1}\sum_{j=0}^{i-1} e(J_j) > \frac{a^2}{a-1} e(J_k) > (a-1)e(J_k) > (\alpha - 1)e(J_k)$$

Hence, if A_k still did not give up, it must be executing all the K jobs, each on a different machine.

(c) Now, the adversary generates a new set of αK jobs. Each job arrives at the same time δK, has the same stretch factor α, and the same execution time $e = x(\sum_{i=0}^{K-1} e(J_i))$ where $x > 2a$. Notice that none of the previous K jobs finished on or before the arrival of these new set of jobs since $\delta K < 1$ and $e(J_i) \ge 1$ for all i. Hence, by Lemma 1, at most $(\alpha - 1)$ new jobs can be executed by A_K on a single machine. Hence, $U(A_K) \le \sum_{i=0}^{K-1} e(J_i) + K(\alpha - 1)e$. The optimal scheduling algorithm OPT_K will on the other hand reject all previous K jobs and accept all the new αK jobs. Thus, $U(OPT_K) \ge K\alpha e$ and hence

$$\frac{U_A}{U_{OPT}} \le \frac{\sum_{i=0}^{K-1} e(J_i) + K(\alpha - 1)e}{K\alpha e} = \frac{1 + K(\alpha - 1)x}{K\alpha x} \le 1 - (1/\alpha) + \varepsilon$$

where the last inequality follows from the fact that $x > 2a > \varepsilon$.

Case 2: α is not an integer. Let $p = \lfloor \alpha \rfloor \ge 1$; thus $\alpha > p \ge 1$. The strategy of the adversary is as follows.

(a) First, as in Case 1, the adversary first generates the following jobs. At time 0, the adversary generates a job J_0 with $e(J_0) = 1$ and $\alpha(J_0) = \alpha$ and at time δi, for $1 \le i \le K-1$, the adversary stops (does not generate any more jobs) if A_K did not accept J_{i-1}; otherwise it generates a job J_i with $e(J_i) = \frac{a^2}{a-1}\sum_{k=0}^{i-1} e(J_j)$ and $\alpha(J_i) = \alpha$. The same argument as in Case 1 indicates that A_K must schedule all these jobs, each on a separate machine.

(b) Let $\beta > \max\left\{\frac{1}{(\alpha-1)(\alpha-p)}, \left(\frac{1}{\alpha-p}\right)\max_{0 \le i \le K-1}\left\{\frac{e(J_i)}{e(J_i) - ((1/\alpha)+\varepsilon)\sum_{j=0}^{K-1} e(J_j)}\right\}, \frac{2}{\alpha-p}\right\}$ be a positive integer. Now, at time δK, the adversary generates a job J_K with $e(J_K) = \beta(\alpha - p)e(J_0)$ and

$\alpha(J_K) = \alpha$ and at time δi, for $K+1 \leq i \leq 2K-1$, the adversary stops (does not generate any more jobs) if A_K did not accept J_{i-1}; otherwise it generates a job J_i with $e(J_i) = \beta(\alpha - p)e(J_{i-K})$ and $\alpha(J_i) = \alpha$.

First, note that it is possible to execute all the $2K$ jobs by executing jobs J_i and J_{K+i}, for $0 \leq i \leq K-1$, on the same machine. Since the jobs J_i and J_{K+i} are released δK time apart, after J_i finishes execution, the time left for J_{K+i} to complete its execution is

$$\alpha e(J_{K+i}) - e(J_i) + \delta K > \left(\alpha - \frac{1}{\beta(\alpha-p)}\right) e(J_{K+i}) > e(J_{K+i})$$

which is sufficient for its execution. Next, note that A_K must accept *all* the new K jobs. If A_K did not accept J_{K+i}, for some $0 \leq i \leq K-1$, then since the adversary can execute all the jobs $J_1, J_2, \ldots, J_{K+i}$, we have

$$\begin{array}{rcll}
\frac{U_A}{U_{OPT}} & \leq & \frac{\sum_{j=0}^{K+i-1} e(J_j)}{\sum_{j=0}^{K+i} e(J_j)} & \\
& = & \frac{\beta(\alpha-p)\sum_{j=0}^{i-1} e(J_j) + \sum_{j=i}^{K-1} e(J_j)}{\beta(\alpha-p)\sum_{j=0}^{i} e(J_j) + \sum_{j=i+1}^{K-1} e(J_j)} & \\
& = & 1 - \frac{(\beta(\alpha-p)-1)e(J_i)}{\beta(\alpha-p)\sum_{j=0}^{i} e(J_j) + \sum_{j=i+1}^{K-1} e(J_j)} & \\
& < & 1 - \frac{(\beta(\alpha-p)-1)e(J_i)}{\beta(\alpha-p)\sum_{j=0}^{K-1} e(J_j)} & \text{since } \beta(\alpha-p) > 2 \\
& = & 1 - \left(\frac{(\beta(\alpha-p)-1}{\beta(\alpha-p)}\right)\left(\frac{e(J_i)}{\sum_{j=0}^{K-1} e(J_j)}\right) & \\
& < & 1 - (1/\alpha) + \varepsilon & \text{since } \beta(\alpha-p) > e(J_i)/(e(J_i) - ((1/\alpha)+\varepsilon)\sum_{j=0}^{K-1} e(J_j))
\end{array}$$

Hence, after this step, A_K must be executing all the $2K$ jobs presented to it.

(c) Now, similar to as in **(c)** in Case 1, the adversary generates a new set of αK jobs. Each job arrives at the same time $2\delta K$, has the same stretch factor α, and the same execution time $e = x(\sum_{i=0}^{2K-1} e(J_i))$ where $x > 2a$. In a manner similar to Case 1, it can be shown that at most $(\alpha - 1)$ new jobs can be executed by A_K on a single machine. Hence, $U(A_K) = \sum_{i=0}^{2K-1} e(J_i) + K(\alpha-1)e$. The optimal scheduling algorithm OPT_K will on the other hand reject all previous $2K$ jobs and accept all the new αK jobs. Thus, $U(OPT_K) = K\alpha e$ and hence

$$\frac{U_A}{U_{OPT}} \leq \frac{\sum_{i=0}^{2K-1} e(J_i) + K(\alpha-1)e}{K\alpha e} = \frac{1 + K(\alpha-1)x}{K\alpha x} \leq 1 - (1/\alpha) + \varepsilon$$

where the last inequality follows from the fact that $x > 2a > \varepsilon$. □

2.2 Firm Real-Time Scheduling For a Single Machine

Theorem 3 *For every $\alpha \geq 1$, any arbitrarily small $\varepsilon > 0$, and for any online preemptive scheduling algorithm A, $\rho(A, \alpha) \leq 1 - 3/(4\lceil\alpha\rceil) + \varepsilon$.*

Proof. Our proof follows an approach similar to the one in [2] in which they prove an upper bound of 1/4 on the competitive ratio for a single machine. First, we need the following technical lemma.

Lemma 4 *Let $\{x_i\}_{i\geq 1}$ be the sequence defined by the recurrence relation*

$$x_i = \begin{cases} 1 & \text{if } i = 1 \\ c-1 & \text{if } i = 2 \\ c(x_{i-1} - x_{i-2}) & \text{if } i > 2 \end{cases}$$

where $c > 1$ is a real number. Furthermore, let $S_0 = 0$ and $S_i = \sum_{j=1}^{i} x_j$, for $i \geq 1$. The following properties hold for the sequence $\{x_i\}$:

(a) *For every $i > 1$, $S_i = cx_{i-1}$.*

(b) *For every $i \geq 1$, $\frac{S_{i-1}+x_{i+1}}{S_{i+1}} = \frac{c-1}{c}$.*

(c) *For every $c < 4$, there exists a finite integer $m > 0$ such that $x_{m-1} \geq x_m$.*

Proof. Property (a) is proved by straightforward induction on i, and property (b) follows from property (a). Property (c) was proved by Baruah *et. al* in [2]. □

We now continue with the proof of Theorem 3. Let $\beta = \lceil \alpha \rceil$, and for a given value of c, let m be an integer such that property (c) of Lemma 4 is satisfied. The adversary generates a job sequence consisting of two types of jobs:

(1) Major jobs. These jobs have stretch β and are released by the adversary at times $t_1, t_2, \ldots, t_m$. At time t_i, the adversary releases exactly β major jobs each of length x_i, where $\{x_i\}$ is the sequence defined in Lemma 1 with $c = 4/(1+4\beta\varepsilon)$. Note that these jobs have a common deadline $d_i = t_i + \beta x_i$. The adversary chooses the release times such that $t_1 = 0$ and $t_{i+1} = d_i - \delta$ for $1 < i \leq m$, where δ is arbitrarily small.

(2) Baits. The adversary also generates a sequence of small jobs (called *baits*) each of length δ and stretch β. The baits are released one at a time every δ time units beginning at time 0.

Henceforth we will refer to the time interval $[t_i, t_{i+1})$ as *epoch i* and to the major jobs released at time t_i as the *major jobs of epoch i*.

At time 0, the adversary starts off by releasing the major jobs of epoch 1 and starting the bait sequence. In general, during the current epoch i, the adversary does the following in response to the actions of online algorithm A:

(1) We say that *A takes the bait in epoch i* if A chooses to execute a bait anytime during epoch i. In this case, the adversary immediately stops generating new jobs (major jobs and baits). On the other hand, if A does not take the bait, the adversary continues the bait sequence and releases the major jobs of epoch $i+1$ at time t_{i+1}.

(2) We say that *A completes epoch i* if throughout the time interval $[t_i, d_i]$ A executes only the major jobs of epoch i. In particular, it does not switch to a major job of epoch $i+1$ (released at time t_{i+1}) during the time interval $[t_{i+1}, d_i)$. In this case, the adversary stops generating new jobs after time d_i.

(3) We say that A *abandons epoch* i if during the time interval $[t_i, t_{i+1})$ A executes only the major jobs of epoch i, then switches to a major job of epoch $i+1$ sometime during the interval $[t_{i+1}, d_i)$ (possibly by aborting a currently executing major job of epoch i). In this case, the adversary continues the bait sequence and releases the next set of major jobs at the start of the next epoch.

The adversary continues generating new jobs, if necessary, but eventually ends the job sequence at time t_m with the release of the major jobs of epoch m.

We now prove that the online algorithm A achieves a competitive ratio of atmost $1-3/(4\beta)+\varepsilon$. The proof breaks down into the following three cases:

Case 1. A abandons epochs $1, 2, \ldots, k-1$ then takes the bait in epoch $k \le m$. Then A must have executed at most $\beta - 1$ major jobs of each epoch i, $1 \le 1 \le k$. On the other hand, an optimal offline scheduler attains a utilization of βS_k by executing the bait sequence from time 0 till time t_k then executing all β major jobs of epoch k. Thus,

$$\frac{U_A}{U_{OPT}} \le \frac{(\beta-1)S_k}{\beta S_k} = 1 - \frac{1}{\beta} \le 1 - \frac{3}{4\beta}$$

Case 2. A abandons epochs $1, 2, \ldots, k-1$ then completes epoch $k < m$. Then A must have executed at most $\beta - 1$ major jobs of epoch i, $1 \le i < k$, at most β jobs of epoch k, and at most $\beta - 1$ major jobs of epoch $k+1$. On the other hand, an optimal offline scheduler attains a utilization of βS_{k+1} by executing the bait sequence from time 0 till time t_{k+1} then executing all β jobs of epoch $k+1$. Thus,

$$\begin{aligned}
\frac{U_A}{U_{OPT}} &\le \frac{(\beta-1)S_{k-1} + \beta x_k + (\beta-1)x_{k+1}}{\beta S_{k+1}} \\
&= \frac{\beta S_{k+1} - S_{k-1} - x_{k+1}}{\beta S_{k+1}} \\
&= 1 - \frac{S_{k-1} + x_{k+1}}{\beta S_{k+1}} \\
&= 1 - \frac{c-1}{\beta c} && \text{by Lemma 4(b)} \\
&= 1 - \frac{3}{4\beta} + \varepsilon && \text{for } c = \tfrac{4}{1+4\varepsilon\beta}
\end{aligned}$$

Case 3. A abandons epochs $1, 2, \ldots, m-1$ and completes epoch m. Then A must have executed at most $\beta - 1$ major jobs of each epoch i, $1 \le i < m$, and at most β major jobs of epoch m. On the other hand, an optimal offline scheduler attains a utilization of βS_m by executing the bait sequence from time 0 till time t_m then executing all β major jobs of epoch m. Thus,

$$\frac{U_A}{U_{OPT}} \le \frac{(\beta-1)S_{m-1} + \beta x_m}{\beta S_m} = \frac{\beta S_m - S_{m-1}}{\beta S_m} = 1 - \frac{S_{m-1}}{\beta S_m}$$

Since $c = 4/(1+4\beta\varepsilon) < 4$, then by Lemma 4(c) and by choice of m, we have $x_{m-1} \ge x_m$. Now,

$$\begin{aligned}
x_{m-1} \geq x_m \quad &\Leftrightarrow \quad S_{m-1} + x_{m-1} \geq S_{m-1} + x_m \\
&\Leftrightarrow \quad cS_{m-1} + cx_{m-1} \geq cS_m \\
&\Leftrightarrow \quad cS_{m-1} + S_m \geq cS_m \qquad \text{by Lemma 4(a)} \\
&\Leftrightarrow \quad cS_{m-1} \geq (c-1)S_m \\
&\Leftrightarrow \quad \frac{S_{m-1}}{S_m} \geq \frac{c-1}{c}.
\end{aligned}$$

Therefore,

$$\frac{U_A}{U_{OPT}} \leq 1 - \frac{c-1}{\beta c} = 1 - \frac{3}{4\beta} + \varepsilon$$

since $c = \frac{4}{1+4\varepsilon\beta}$. □

3 An Online Scheduler for Both Models

In this section, we exhibit a simple $O(n)$ time online scheduler of K machines, where n is the total number of jobs, that achieves a competitive ratio of at least $1 - (1/\alpha)$ for either hard or firm real-time scheduling. The online scheduler A_K is based on the EDF (Earliest Deadline First) scheduling algorithm [10]. When a job J arrives, A_K first runs an admission test on the machines $M_1, M_2, \ldots, M_K$, in any predetermined order, to check if all previously admitted jobs that have not yet completed, plus job J, can be completed by their respective deadlines on some machine M_j. If so A_K admits J on machine M_j, otherwise it rejects J. Admitted jobs in each machine are executed by A_K in nondecreasing order of their deadlines. Thus, preemptions may occur: a currently executing job will be preempted in favor of a newly admitted job with an earlier deadline. The preempted job resumes execution when there are no more admitted jobs with earlier deadlines.

The details of the scheduling algorithm of A_K are as follows. A_K maintains a queue Q_j of jobs that have been admitted but have not yet completed on machine M_j. Each job in the queue Q_j contains three information: (1) its job number, (2) its deadline, and (3) its remaining execution time (i.e., its execution time minus the processor time that it has consumed so far). The jobs in the queue are ordered by nondecreasing deadlines. Thus, if the machine M_j is busy, then it is executing the job at the head of the queue Q_j (which has the earliest deadline) and the remaining execution time of this job decreases as it continues to execute. The job is deleted from Q_j when its remaining execution time becomes zero, and the job (if any) that becomes the new head of the queue Q_j is executed next on M_j.

Here are more implementation details of the scheduling algorithm A_K for the machine M_j. When a new job J arrives, if the job is already not scheduled in one of the machines $M_1, M_2, \ldots, M_{j-1}$ (for $j > 1$) then A_K inserts J in its proper position in Q_j (such that the deadlines of jobs in Q_j are ordered by nondecreasing deadlines), with its remaining execution time initially set to its (total) execution time (if there are jobs in Q_j with the same deadline as J then J is inserted after these jobs). Then A_K performs the following admission test to determine whether to admit or reject J on machine M_j:

```
reject = FALSE;
J' = J;
repeat
   s = sum of the remaining execution times of all jobs
       in the queue up to and including job J';
   if s > d(J') then reject = TRUE
     else J' = next job in the queue after J';
until (J' = NULL or reject = TRUE);
```

That is, $reject$ is set to $TRUE$ if admission of J will cause J or some job with a deadline later than J to miss its deadline (note that jobs with deadlines earlier than J will not be delayed by J and hence need not be checked in the admission test). If $reject = TRUE$, then J is deleted from Q_j and is rejected. Otherwise, J is admitted and retains its position in the queue Q_j. If the position of J is not at the head of the queue, then A_K continues execution of the current job at the head of the queue Q_j on machine M_j. However, if J is positioned at the head of the queue Q_j, then the current job is preempted and J begins execution on M_j. Clearly, the total time taken by the online scheduler over all machines when a new job arrives is $O(n)$.

As a final note, it may happen that several jobs may arrive simultaneously; in this case, A_K processes these jobs in any arbitrary order.

Before proceeding further, we first introduce some notations to simplify the equations that will be presented shortly. Let X be a set of jobs. We shall use $X\mid_{\leq d}$ ($X\mid_{>d}$) to denote the subset of jobs in X whose deadlines are $\leq d$ (respectively, $> d$). Additionally, we slightly abuse notation by using $e(X, j)$ to denote the sum of the execution times of all jobs in X on machine M_j.

Finally, the following inequality will be used later:

Fact 1 $\frac{a+u}{b+v} \geq \frac{u}{v}$ *whenever* $u \leq v$ *and* $a \geq b$.

Proof. $u \leq v$ and $a \geq b \Rightarrow ub \leq va \Rightarrow ub + uv \leq va + uv \Rightarrow u(b+v) \leq v(a+u) \Rightarrow \frac{u}{v} \leq \frac{a+u}{b+v}$. □

Theorem 5 *For every set of jobs* $\mathcal{J}$ *with* $\alpha(\mathcal{J}) = \alpha$ *and for every integer* $K \geq 1$, $\frac{U_{A_K}(\mathcal{J})}{U_{OPT_K}(\mathcal{J})} \geq 1 - (1/\alpha)$

Proof. Let $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ be the set of n jobs. We wish to compare the schedule of A_K with that of an optimal offline scheduler OPT_K. For a given machine M_j, the schedule of A_K for this machine produces an execution profile that consists of an alternating sequence of *busy* and *idle* time intervals of M_j. A busy time interval of M_j corresponds to the case when the machine M_j is busy executing some job. Each busy interval (except the last) of M_j is separated from its next busy interval by an *idle* interval, during which the machine M_j is not executing any job. Define a busy time interval for the scheduler A_K to be a time interval during which *all* of its K machines are busy; otherwise call it a *non-busy* time interval of A_K. Observe that any job that arrived during a non-busy time interval of A_K must have been scheduled by A_K immediately, since at least one its machines was not executing any job. In other words, any job that was rejected by A_K must have arrived during a busy time interval of A_K.

Let $B_1, B_2, \ldots, B_m$ be the busy intervals of A_K. The jobs in $\mathcal{J}$ can then be partitioned into the following disjoint subsets:

- $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_m$, where $\mathcal{J}_i$ is the set of jobs that arrive during busy interval B_i.
- The remaining set $\mathcal{J}'' = \mathcal{J} - \cup_{i=1}^{m} \mathcal{J}_i$ of jobs whose arrival time is during a non-busy time interval of A_K. All jobs in $\mathcal{J}''$ were executed by A_K.

Let $\mathcal{J}' = \cup_{i=1}^{m} \mathcal{J}_i$. Let $\mathcal{J}'_{A_K}$ (respectively, $\mathcal{J}'_{OPT_K}$) be the set of jobs that were executed by A_K (respectively, OPT_K) from $\mathcal{J}'$. We first claim that, to prove Theorem 5, it is sufficient to show that

$$\frac{e(\mathcal{J}'_{A_K})}{e(\mathcal{J}'_{OPT_K})} \geq 1 - (1/\alpha) \tag{1}$$

Why is this so? Assume that OPT_K executes a subset $\mathcal{J}''_{OPT_K}$ of the jobs in $\mathcal{J}''$. Hence, $U_{A_K}(\mathcal{J}) = e(\mathcal{J}'_{A_K}) + e(J'')$, $U_{OPT_K}(\mathcal{J}) = e(\mathcal{J}'_{OPT_K}) + e(\mathcal{J}''_{OPT_K})$ and $e(\mathcal{J}'') \geq e(\mathcal{J}''_{OPT_K})$. Now, if $e(\mathcal{J}'_{A_K}) \geq e(\mathcal{J}'_{OPT_K})$, then clearly $\frac{U_{A_K}(\mathcal{J})}{U_{OPT_K}(\mathcal{J})} \geq 1 > 1 - (1/\alpha)$. Otherwise, if $e(\mathcal{J}'_{A_K}) < e(\mathcal{J}'_{OPT_K})$, then by Fact 1, $\frac{U_{A_K}(\mathcal{J})}{U_{OPT_K}(\mathcal{J})} \geq \frac{e(\mathcal{J}'_{A_K})}{e(\mathcal{J}'_{OPT_K})}$.

Now we prove that Equation 1. Let $\mathcal{J}_i^{A_K}$ and $\mathcal{J}_i^{OPT_K}$ be the subsets of jobs in $\mathcal{J}_i$ admitted by A_K and OPT_K, respectively. Obviously, since $e(\mathcal{J}'_{A_K}) = \sum_{i=1}^{m} e(\mathcal{J}_i^{A_K})$ and $e(\mathcal{J}'_{OPT_K}) = \sum_{i=1}^{m} e(\mathcal{J}_i^{OPT_K})$, it suffices to prove that, for each busy interval B_i of A_K, $1 \leq i \leq m$:

$$\frac{e(\mathcal{J}_i^{A_K})}{e(\mathcal{J}_i^{OPT_K})} \geq 1 - (1/\alpha) \tag{2}$$

We now prove Equation 2. For notational convenience, we drop the subscript K from A_K and OPT_K. Let $\mathcal{J}_{i,j}^{A}$ (respectively, $\mathcal{J}_{i,j}^{OPT}$) be the subset of jobs in $\mathcal{J}_i^{A}$ (respectively, $\mathcal{J}_i^{OPT}$) that were scheduled on machine M_j. Since $e(\mathcal{J}_i^{A}) = \sum_{j=1}^{K} e(\mathcal{J}_{i,j}^{A})$ and $e(\mathcal{J}_i^{OPT}) = \sum_{j=1}^{K} e(\mathcal{J}_{i,j}^{OPT})$, to prove Equation 2 it is sufficient to show that

$$\frac{e(\mathcal{J}_{i,j}^{A})}{e(\mathcal{J}_{i,j}^{OPT})} \geq 1 - (1/\alpha) \tag{3}$$

We now prove Equation 3 for an arbitrary machine M_j. For notational convenience, for a set of jobs X we refer to $e(X, j)$ simply by $e(X)$. Let t be the time at which busy interval B_i begins; thus, all jobs in $\mathcal{J}_{i,j}$ have arrival times no earlier than t. Let $X \in \mathcal{J}_{i,j}^{OPT}$ be a job with the latest deadline among all jobs in $\mathcal{J}'_{OPT} - \mathcal{J}'_{A}$ that was executed in machine M_j. If there is no such job X, then $\mathcal{J}_{i,j}^{OPT} \subseteq \mathcal{J}_{i,j}^{A}$ and Equation 3 trivially holds, hence we assume such an X exists. Also, assume that $e(\mathcal{J}_{i,j}^{A}) < e(\mathcal{J}_{i,j}^{OPT})$, otherwise again Equation 3 trivially holds.

By the admission test, A rejected X for M_j because its admission would have caused some job Y (possibly X) on M_j to miss its deadline $d(Y)$; i.e.,

$$t + e(\mathcal{J}_{i,j}^{A} \mid_{\leq d(Y)}) + e(X) > d(Y)$$

The term t on the left hand side of the above equation is due to the fact that all jobs in $\mathcal{J}_{i,j}^{A}$ have arrival times no earlier than t and hence cannot be executed before time t. Since $\mathcal{J}_{i,j}^{A} = \mathcal{J}_{i,j}^{A} \mid_{\leq d(Y)} \cup \mathcal{J}_{i,j}^{A} \mid_{> d(Y)}$, we get:

$$e(\mathcal{J}_{i,j}^{A}) > d(Y) - e(X) - t + e(\mathcal{J}_{i,j}^{A}\mid_{>d(Y)}). \tag{4}$$

Now, since OPT must complete all jobs in $\mathcal{J}_{i,j}^{OPT}$ on M_j no later than their deadlines, it should be the case that $t + e(\mathcal{J}_{i,j}^{OPT}\mid_{\le d}) \le d$ for every d. In particular, when $d = d(Y)$, we have:

$$t + e(\mathcal{J}_{i,j}^{OPT}\mid_{\le d(Y)}) \le d(Y)$$

Since $\mathcal{J}_{i,j}^{OPT} = \mathcal{J}_{i,j}^{OPT}\mid_{\le d(Y)} \cup \mathcal{J}_{i,j}^{OPT}\mid_{>d(Y)}$, we get:

$$e(\mathcal{J}_{i,j}^{OPT}) \le d(Y) - t + e(\mathcal{J}_{i,j}^{OPT}\mid_{>d(Y)})$$

By definition, X has the latest deadline among all jobs admitted by OPT but rejected by A on machine M_j. Also, by the admission test, Y is either X or some other job admitted by A with deadline $d(Y) > d(X)$. It follows that $\mathcal{J}_{i,j}^{OPT}\mid_{>d(Y)} \subseteq \mathcal{J}_{i,j}^{A}\mid_{>d(Y)}$. The above equation then becomes:

$$e(\mathcal{J}_{i,j}^{OPT}) \le d(Y) - t + e(\mathcal{J}_{i,j}^{A}\mid_{>d(Y)}). \tag{5}$$

From Equations 4 and 5 we get:

$$\begin{aligned}
\frac{e(\mathcal{J}_{i,j}^{A})}{e(\mathcal{J}_{i,j}^{OPT})} &> \frac{d(Y) - e(X) - t + e(\mathcal{J}_{i,j}^{A}\mid_{>d(Y)})}{d(Y) - t + e(\mathcal{J}_{i,j}^{A}\mid_{>d(Y)})} \\
&= \frac{a(Y) + e(Y)\alpha(Y) - e(X) - t + e(\mathcal{J}_{i,j}^{A}\mid_{>d(Y)})}{a(Y) + e(Y)\alpha(Y) - t + e(\mathcal{J}_{i,j}^{A}\mid_{>d(Y)})} \\
&= \frac{\frac{a(Y)-t+e(\mathcal{J}_{i,j}^{A}\mid_{>d(Y)})}{\alpha(Y)} + e(Y) - \frac{e(X)}{\alpha(Y)}}{\frac{a(Y)-t+e(\mathcal{J}_{i,j}^{A}\mid_{>d(Y)})}{\alpha(Y)} + e(Y)} \\
&\ge \frac{e(Y) - \frac{e(X)}{\alpha(Y)}}{e(Y)}, \text{ by Fact 1} \\
&= 1 - \frac{e(X)}{\alpha(Y)e(Y)}.
\end{aligned}$$

We now show that $e(X)/\alpha(Y) \le e(Y)/\alpha$. If $X = Y$, then $e(X)/\alpha(Y) = e(Y)/\alpha(Y) \le e(Y)/\alpha$, since $\alpha(Y) \ge \alpha$. If $X \ne Y$, then Y is some job previously admitted by A on machine M_j such that $a(Y) \le a(X)$ and $d(Y) > d(X)$. Thus:

$$\begin{aligned}
& a(Y) + e(Y)\alpha(Y) &>& \ a(X) + e(X)\alpha(X) \\
\iff & e(Y)\alpha(Y) &>& \ (a(X) - a(Y)) + e(X)\alpha(J) \\
\iff & e(Y)\alpha(Y) &\ge& \ e(X)\alpha(X), \text{since } a(X) - a(Y) \ge 0 \\
\iff & e(Y)\alpha(Y) &\ge& \ e(X)\alpha, \text{since } \alpha(X) \ge \alpha.
\end{aligned}$$

Hence, it follows that $e(X)/\alpha(Y) \le e(Y)/\alpha$. We therefore conclude that $\frac{e(\mathcal{J}_{i,j}^{A})}{e(\mathcal{J}_{i,j}^{OPT})} > 1 - \frac{e(X)}{\alpha(Y)e(Y)} \ge 1 - \frac{1}{\alpha}$ □

4 An Combined Scheduler for Firm Real-Time Scheduling on a Single Machine

The competitive ratio $1 - (1/\alpha)$ for the scheduler described in Section 3 approaches zero as α approaches one. For the case of firm real-time scheduling on a single machine, this can be avoided by combining our scheduler with the version 2 of the TD_1 scheduler (Figure 2) of [2].

Of course, if α is known to the online algorithm in advance, then combining the two algorithms is easy: if $\alpha > 4/3$, the scheduling algorithm of Section 3 is chosen; otherwise the TD_1 scheduler of [2] is chosen. This would ensure that the competitive ratio of the combined scheduler is at least $\max\{1/4, 1-(1/\alpha)\}$. The above discussion leads to the following simple corollary.

Corollary 6 *If the value of α is known to the online scheduler in advance, then there is a scheduler A with $\rho(A,\alpha) \geq \max\{1/4, 1-(1/\alpha)\}$.*

On the other hand, if the value of α is not be known to the online scheduler in advance, then a different strategy needs to be used.

Theorem 7 *Even if the value of α is not known in advance to the scheduler, it is possible to design a scheduler* **CS** *whose performance guarantee is given by* $\rho(\mathbf{CS},\alpha) \geq \begin{cases} 1-(1/\alpha) & \text{if } \alpha \geq 4/3 \\ 7/64 & \text{otherwise} \end{cases}$

Proof. We describe our combined scheduler **CS** below. First, we need to modify slightly the constants of the TD_1 scheduler. The modified version 2 of the TD_1 scheduler is shown below in Figure 1 (we use the same notations as in [2] in this code).

```
(1)    whenever (idle && not-empty(Q)) {
(2)        $T_{run}$ = dequeue($Q$); $p_{loss} = v_{run}$;
(3)    }
(4)    whenever (running && alarm($Q$)) {
(5)        $T_{next}$ = dequeue($Q$);
(6)        update($\Delta_{run}$);
(7)        if ($v_{run} < \frac{7}{64}(\Delta_{run} + p_{loss})$)
(8)            { $T_{run} = T_{next}$; }
```

Figure 1: Modified version 2 of the TD_1 scheduler of [2]

The combined scheduler **CS** maintains α_{min}, the minimum stretch factor of all the jobs seen so far. **CS** executes the scheduler of Section 3 unless α_{min} drops below a threshold, in which case it starts executing the TD_1 scheduler permanently. In particular, when a new job J arrives, **CS** takes the following actions:

Case 1. If α_{min} is already less than 4/3, then **CS** continues to execute TD_1 and admits of rejects J based on the TD_1 strategy.

Case 2. Otherwise, if $\min\{\alpha_{min}, \alpha(J)\} \geq 4/3$, then **CS** continues to execute the scheduler of Section 3 and admits of rejects J based on the strategy described in Section 3.

Case 3. Otherwise, $\alpha_{min} \geq 4/3$ and $\alpha(J) < 4/3$. In that case, **CS** switches from the scheduler in Section 3 to the TD_1 scheduler. In particular, in the notations of [2] as used in Figure 1, v_{run} is set to the sum of execution times of all jobs in the queue Q (since each of them is scheduled for execution), p_{loss} is set to v_{run} (since in the worst case each job in Q may be executed outside the current interval in favor of future jobs), Δ_{run} is set to $\left(\frac{\alpha_{min}}{\alpha_{min}-1}\right) v_{run}$ (since, by Theorem 3, this is the maximum possible length of the current interval of an optimal schedule executing all jobs that arrived during the current busy cycle), and all the jobs currently in the queue Q together constitutes the set of running jobs T_{run}. Finally, lines (6)-(8) of the code in figure 1 are executed to check if the newly arrived job J will preempt all the jobs in T_{run}.

Now, we prove the performance guarantee of **CS** , as promised in Theorem 7. If $\alpha \geq 4/3$, then **CS** always executes the scheduler in Section 3 and hence, by Theorem 5, $\rho(\mathbf{CS}, \alpha) \geq 1 - (1/\alpha)$.

Otherwise, $1 \leq \alpha < 4/3$. As a result, on some input job J, **CS** switches to the TD_1 scheduler. Moreover, all jobs before J had stretch factor no less than 4/3 and $\alpha(J) < 4/3$. Let $\gamma \geq 4/3$ be the stretch factor of the set of jobs considered before J. Hence, before lines (6)-(8) of figure 1 in Case 3 are executed, using Theorem 5 we have

$$v_{run} = \left(1 - \frac{1}{\gamma}\right) \Delta_{run} \geq \left(\frac{1}{2} - \frac{1}{2\gamma}\right) (\Delta_{run} + p_{loss}) \geq (\Delta_{run} + p_{loss})/8 \tag{6}$$

An inspection of the proof of Lemma 6 of [2] shows that, since $1/8 > 7/64$, it suffices to show that the invariant $v_{run} \geq (\Delta_{run} + p_{loss})/8$ is maintained whenever a preemption occurs during a time interval (when time intervals are as defined in Definition 1 in [2]). This can be proved by induction in a manner similar to as in [2] as shown below. We use the same notations as in [2]. The basis of induction follows from the inequality 6 shown above. For general i,

$$\begin{array}{rcll} v_{i+1} & > & \Delta_{i+1} - \Delta_i & \\ & \geq & \Delta_{i+1} - 8v_i + p_{loss} & \text{since } v_i \geq (\Delta_i + p_{loss})/8 \\ & > & \Delta_{i+1} - \frac{7}{8}(\Delta_i + p_{loss}) + p_{loss} & \text{since } v_i < \frac{7}{64}(\Delta_i + p_{loss}) \\ & = & \Delta_{i+1} - \frac{7}{8}\Delta_i + (p_{loss}/8) & \\ & > & (\Delta_{i+1} + p_{loss})/8 & \text{since } \Delta_{i+1} - \Delta_i > \frac{8}{7}v_i > 0 \end{array}$$

□

5 Acknowledgments

The first author would like to thank Piotr Berman and Marek Karpinski for useful discussions.

References

[1] Baruah, S., G. Koren, B. Mishra, A. Ragunathan, L. Rosier, and D. Sasha, *On-line Scheduling in the Presence of Overload*, Proc. 32nd IEEE Symposium on Foundations of Computer Science, 100-110, October 1991.

[2] Baruah S., G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha and F. Wang, *On the competitiveness of on-line real-time scheduling*, Real-Time Systems **4**, 125-144, 1992.

[3] Bar-Noy, A., R. Bar-Yehuda, A. Freund, J. (S.) Naor and B. Schieber, *A Unified Approach to Approximating Resource Allocation and Scheduling*, Proc. 32nd Annual ACM Symposium on Theory of Computing, 735-744, May 2000.

[4] Bar-Noy, A., S. Guha, J. (S.) Naor and B. Schieber, *Approximating the throughput of multiple machines in real-time scheduling*, Proc. 31st Annual ACM Symposium on Theory of Computing, 622-631, 1999.

[5] Berman P. and B. DasGupta, *Improvements in Throughput Maximization for Real-Time Scheduling*, Proc. 32nd Annual ACM Symposium on Theory of Computing, 680-687, May 2000.

[6] Becchetti, L., S. Leonardi and S. Muthukrishnan, *Scheduling to Minimize Average Stretch without Migration*, Proc. 11th Annual ACM-SIAM Symp. on Discrete Algorithms, 548-557, 2000.

[7] Bender, M., S. Chakrabarti and S. Muthukrishnan, *Flow and Stretch Metrics for Scheduling Continuous Job Streams*, Proc. 10th Annual ACM-SIAM Symp. on Discrete Algorithms, 1999.

[8] Brandt, S., G. Nutt, T. Berk, and M. Humphrey, *Soft Real-Time Application Execution with Dynamic Quality of Service Assurance*, 1998 International Workshop on Quality of Service, 154-163, May 1998.

[9] Compton, C. and D. Tennenhouse, *Collaborative Load Shedding*, Proc. Workshop on the Role of Real-Time in Multimedia/Interactive Computing Systems, Dec. 1993.

[10] Dertouzos, M., *Control Robotics: the Procedural Control of Physical Processors*, Proc. IFIP Congress, 807-813, 1974.

[11] Fan, C., *Realizing a Soft Real-Tim Framework for Supporting Distributed Multimedia Applications*, Proc. 5th IEEE Workshop on the Future Trends of Distributed Computing Systems, 128-134, August 1995.

[12] Humphrey, M., T. Berk, S. Brandt, and G. Nutt, *Dynamic Quality of Service Resource Management for Multimedia Applications on General Purpose Operating Systems*, IEEE Workshop in Middleware for Distributed Real-Time Systems and Services, 97-104, Dec. 1997.

[13] Jones, M., J. Barbera III, and A. Forin, *An Overview of the Rialto Real-Time Architecture*, Proc. 7th ACM SIGOPS European Workshop, 249-256, Sept. 1996.

[14] Jones, M., D. Rosu, and M.-C. Rosu, *CPU Reservations and Time Constraints: Efficient, Predictable Scheduling of Independent Activities*, Proc. 16th ACM Symposium on Operating Systems Principles, Oct. 1997.

[15] Kise H., T. Ibaraki and H. Mine, *A solvable case of one machine scheduling problems with ready and due dates*, Operations Research **26**, 121-126, 1978.

[16] Koren G. and D. Shasha, *An optimal on-line scheduling algorithm for overloaded real-time systems*, SIAM J. on Computing **24**, 318-339, 1995.

[17] Lawler, E. L., *A dynamic programming approach for preemptive scheduling of a single machine to minimize the number of late jobs*, Annals of Operations Research **26**, 125-133, 1990.

[18] Lipton, R. J. and A. Tomkins, *Online interval scheduling*, Proc. 5th Annual ACM-SIAM Symp. on Discrete Algorithms, 302-311, 1994.

[19] Liu, H. and M. E. Zarki, *Adaptive source rate control for real-time wireless video transmission*, Mobile Networks and Applications **3**, 49-60, 1998.

[20] Mok, A., *Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment*, Doctoral Dissertation, M.I.T., 1983.

[21] Muthukrishnan, S., R. Rajaraman, A. Shaheen abd J. E. Gehrke, *Online Scheduling to Minimize Average Stretch*, Proc. 40th Annual IEEE Symp. on Foundations of Computer Science, 433-443, 1999.

[22] Nieh, J. and M. Lam, *The Design, Implementation and Evaluation of SMART: A Scheduler for Multimedia Applications*, Proc. 16th ACM Symposium on Operating Systems Principles, Oct. 1997.

[23] Nieh, J. and M. Lam, *Integrated Processor Scheduling for Multimedia*, Proc. 5th International Workshop on Network and Operating System Support for Digital Audio and Video, April 1995.

[24] Rajugopal, G. R. and R. H. M. Hafez, *Adaptive rate controlled, robust video communication over packet wireless networks*, Mobile Networks and Applications **3**, 33-47, 1998.

[25] Sahni, S, *Algorithms for scheduling independent tasks*, JACM **23**, 116-127, 1976.

[26] Spieksma, F. C. R., *On the approximability of an interval scheduling problem*, Journal of Scheduling **2**, 215-227, 1999 (preliminary version in the Proceedings of the APPROX'98 Conference, Lecture Notes in Computer Science, 1444, 169-180, 1998).

[27] Yau, D. K. Y. and S. S. Lam, *Adaptive rate-controlled scheduling for multimedia applications*, Proc. IS&T/SPIE Multimedia Computing and Networking Conf., San Jose, CA, January 1996.