Fair Real-Time Resource Allocation for Internet End System's QoS Support*

Jungkeun Park, Minsoo Ryu, and Seongsoo Hong†

Abstract

Delivered end-to-end QoS is often limited by the ineffective resource management at Internet end systems. To overcome this problem, we present a resource allocation framework that can handle both the bandwidth and deadline requirements in allocating time shared resources at an Internet end system. The proposed framework has a two-level hierarchy. At the top-level, a proportional share scheduler, called Earliest Finish Time Credit/Debit (EFT-C/D) scheduler, allocates a time shared resource like CPU to the bottom-level schedulers in proportion to the specified rate of each bottom-level scheduler. The bottom-level schedulers each employ different scheduling disciplines to handle different timing requirements. Specifically, tasks with deadline requirements are scheduled by an EDF (earliest deadline first) scheduler, and tasks with bandwidth requirements are scheduled by a proportional share scheduler such as the proposed EFT-C/D scheduler or an EEVDF (Earliest Eligible Virtual Deadline First) scheduler. Our major contributions are two-fold. First, we present the EFT-C/D algorithm that can achieve nearly perfect fairness when compared to the ideal GPS server. We also show that the EFT-C/D algorithm is considerably simple to implement and incurs low run-time overhead since its implementation does not involve establishing virtual time. Second, we present a utilization-based schedulability analysis for the EDF scheduler at the bottom-level. Our analysis technique enables on-line admission control by allowing us to easily test the schedulability of a given task set by simply computing required resource utilization.

*The work reported in this paper was supported in part by MOST under the National Research Laboratory (NRL) grant 2000-N-NL-01-C-136, by Automatic Control Research Center (ACRC), and by the Automation and Systems Research Institute (ASRI).

†School of Electrical Engineering and Computer Science, Seoul National University, San 56-1, Shinlim-Dong, Kwanak-Gu, Seoul 151-742, Korea. Phone: +82-2-880-8370. Fax: +82-2-880-4656. Email: msryu@redwood.snu.ac.kr.

1 Introduction

Recent advances in processor and network technologies have enabled computers to run combinations of QoS-demanding applications with different types of timing requirements. For instance, multimedia applications such as video streaming possess stringent bandwidth requirements in processor allocation to provide smooth presentation. On the other hand, some applications like online stock trading and networked games impose hard deadlines on data processing since untimely processing can lead to unacceptable financial losses or an inconsistent game experience. Thus, it is important for Internet end systems to meet various types of timing requirements via effective resource management.

In this paper we explore the resource allocation problem to support both the bandwidth and deadline requirements in Internet end systems ranging from desk-top computers to mobile terminals. Currently the dominant approach in such systems is just to provide best-effort services based on a simple time-sharing policy. However, due to the lack of bandwidth and deadline guarantees, this approach often fails to support the real-time applications mentioned above. As a result, delivered end-to-end QoS through Internet is often limited by the ineffective resource allocation at Internet end systems.

To overcome this problem, we propose and analyze a resource allocation framework that has a two-level hierarchy as shown in Figure 1. At the top-level, a proportional share scheduler allocates a time shared resource like CPU to each lower-level scheduler in proportion to its specified rate share. In doing so, we use an algorithm, called *Earliest Finish Time Credit/Debit (EFT-C/D)*, which is an extension of the credit/debit algorithm in [6]. The bottom-level schedulers each employ different scheduling disciplines to handle different timing requirements. Specifically, tasks with deadline requirements are scheduled by an EDF (earliest deadline first) scheduler [8], and tasks with bandwidth requirements are scheduled by a proportional share scheduler such as the proposed EFT-C/D scheduler or an EEVDF (Earliest Eligible Virtual Deadline First) scheduler [11]. Note that our scheduling hierarchy permits more schedulers to support other types of scheduling disciplines such as RM (rate monotonic), but our discussions will focus only on the top-level proportional share scheduler and the bottom-level EDF scheduler.

Our major contributions are two-fold. First, we present the Earliest Finish Time C/D (EFT-C/D) algorithm that can achieve nearly perfect fairness when compared to the ideal GPS (Generalized Processor Sharing) server [9]. Specifically, the service time difference between EFT-C/D and GPS schedulers is bounded by the size of time quantum. The complexity of the algorithm is $O(n)$. Nevertheless, the EFT-C/D algorithm is considerably simple to implement and incurs low run-time overhead since its implementation does not involve establishing virtual time [9, 5]. Second, we present a utilization-based schedulability analysis for the EDF scheduler at the low-level. Our analysis technique enables on-line admission control by allowing us to easily test the schedulability of a given task set by simply computing

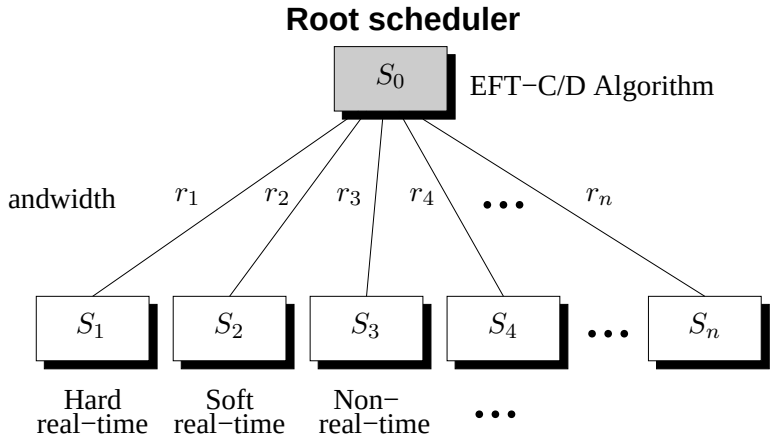

Figure 1: Two-level scheduling framework.

required resource utilization.

There are a few results on the hierarchical scheduling structure. In [3], Liu and Deng proposed a hierarchical scheduler that uses EDF scheduler as a top-level scheduler. Each application is handled by a dedicated bottom-level server based on the CBS (Constant Bandwidth Server) algorithm. The hierarchical scheduler by Liu and Deng can support both real-time applications with deadlines and non real-time applications, but cannot handle bandwidth requirements in processor allocation. On the other hand, Goyal et al. adopt a proportional share scheme at the top-level, called start time fair queueing, and propose to use various types of scheduling policies at lower levels [5]. While our scheduling structure is very similar to the hierarchical scheduler in [5], ours improves upon it by using the EFT-C/D algorithm that outperforms the start-time fair queueing algorithm in terms of fairness.

The remainder of this paper is organized as follows. Section 2 discusses our models and assumptions. Section 3 describes the basic credit/debit algorithm and the Earliest Finish Time C/D (EFT-C/D) for the top-level scheduler. Section 4 describes a schedulability analysis for the low-level EDF scheduler. Section 5 concludes this paper by describing future research directions.

2 Models and Assumptions

We consider two types of tasks: hard and soft real-time tasks. A hard real-time task τ_i is characterized by a deadline requirement d_i and a worst-case execution time (WCET) e_i. The deadline is defined to be a relative value from the request time of τ_i. If the task is periodic, it is denoted by $\tilde{\tau}_i$ and its period is denoted by T_i. A soft real-time task is characterized by a bandwidth requirement b_i which is the relative share of the time-shared resource.

There are two important bottom-level schedulers in our scheduling framework. The hard real-time scheduler S_1 is associated with a relative share r_1 of CPU resource and is scheduled by the root scheduler S_0. The hard real-time scheduler S_1 then schedules hard real-time tasks based on EDF algorithm. The soft real-time scheduler S_2 is also associated with a relative share r_2 and schedules soft real-time tasks using a proportional share algorithm such as the Stride scheduler [12] or the EEVDF scheduler [11]. The notation used throughout the paper is summarized in Table 1.

notation	description
τ_i	hard real-time task
e_i	worst-case execution time of τ_i
d_i	relative deadline of τ_i
$\tilde{\tau}_i$	periodic task
T_i	period of $\tilde{\tau}_i$
S_0	root scheduler
S_i	bottom-level scheduler
C_i	credit value for S_i
r_i	relative CPU share for S_i
Δ	time quantum
W_i^{CD}	service time received by S_i in the interval $[0,t]$ under EFT-C/D
W_i^{GPS}	service time received by S_i in the interval $[0,t]$ under GPS
$E_i(t)$	service lag of S_i
$Q(t)$	set of aperiodic tasks at time t
$Q(t, hp(\tau_i))$	set of taks that have higher priorities than τ_i
$e_{i,t}^{res}$	maximum residual execution time of τ_i at time t
$d_{i,t}^{res}$	lead time of τ_i at time t
$u_{Q(t)}(\tau_i)$	utilization demand of τ_i
$U_{Q(t)}$	maximum utilization demand of $Q(t)$
P	set of periodic tasks
Q_P	equivalent aperiodic task set of P
U_P	utilization of P
$G(t)$	mixed set of periodic and aperiodic tasks at time t

Table 1: Notations for the scheduling framework.

3 Proportional Share Allocation of Processor Bandwidth

In this section we first describe the basic credit/debit algorithm [6], and then present the Earliest Finish Time C/D (EFT-C/D) algorithm for the top-level scheduler.

3.1 Basic Credit/Debit Algorithm

The credit/debit algorithm is a proportional share allocation mechanism for time shared resources such as CPU time. It allocates the resource to bottom-level schedulers in discrete time slices, and the duration of a time slice is referred to as time quantum Δ. The credit/debit algorithm maintains for each bottom-level scheduler S_i a credit value C_i which can be viewed as a virtual time index for the top-level scheduler's quantum allocation. Initially, the top-level scheduler sets each credit value C_i to zero. At the beginning of each time quantum, it increases each C_i by $\frac{r_i}{\sum_j r_j} \cdot \Delta$ and selects the bottom-level scheduler with the maximum credit and ties are broken arbitrarily. It then subtracts the allocated quantum Δ from the credit C_i of the selected scheduler.

Figure 2 illustrates an example of the credit/debit algorithm. Three bottom-level schedulers S_1, S_2, and S_3 are scheduled by the credit/debit algorithm. Their relative resource shares are 7, 2, and 1 respectively. Initially, each credit value is set to 0 at $t = 0$. Since ties are broken arbitrarily, let us assume that the credit/debit scheduler selects S_1 at $t = 0$. Here we use rectangles to represent selected credit values. When the bottom-level scheduler S_1 finished receiving the first quantum at time $t = \Delta$, the top-level scheduler S_0 updates each credit value as below.

$$C_1(\Delta) = C_1(0) + \frac{7}{10} \cdot 10 - 10 = -3, \quad C_2(\Delta) = C_2(0) + \frac{2}{10} \cdot 10 = 2, \quad C_3(\Delta) = C_3(0) + \frac{1}{10} \cdot 10 = 1$$

At the beginning of the second quantum, S_2 is selected since its credit is the highest. Figure 2 shows the credit values during the interval $[0, 9\Delta]$.

Bottom-level scheduler (S_i)	Relative share (r_i)	Credit (C_i)									
S_1	7	[0]	-3	[4]	1	[8]	[5]	2	[9]	[6]	[3]
S_2	2	0	[2]	-6	-4	-2	0	[2]	-6	-4	-2
S_3	1	0	1	2	[3]	-6	-5	-4	-3	-2	-1
	time (t)	0	1Δ	2Δ	3Δ	4Δ	5Δ	6Δ	7Δ	8Δ	9Δ

Figure 2: An example of the credit/debit algorithm with $\Delta = 10$.

The credit $C_i(t)$ represents the service time that S_i received by time t under the ideal GPS discipline. If a bottom-level scheduler with $C_i = 0$ is chosen, then the service starts at the same time as it would under the GPS server. If a bottom-level scheduler with $C_i > 0$ is chosen, then the service starts later than it would under the GPS server.

3.2 Earliest Finish Time C/D Algorithm

We now modify the basic credit/debit algorithm to emulate WF^{2}Q, which is the most popular packet fair queueing discipline. Consider a network switch that multiplexes a set of incoming sessions on a packet-by-packet basis. If we assume that every packet has the same size and that it takes Δ to transmit single packet, then the packet queueing problem can be reduced to our processor scheduling problem. Thus, this gives a possibility that we can apply the resource allocation mechanism of WF^{2}Q to our proportional share algorithm.

The WF^{2}Q discipline serves packets in increasing order of packet departure time among packets that have started receiving service under GPS. To emulate WF^{2}Q, we consider the corresponding GPS server. Let $q_{i,j}$ be the j^{th} quantum received by S_i under GPS, and let $f_{i,j}$ be the time at which S_i finishes to receive $q_{i,j}$ under GPS. Then we modify the basic credit/debit scheduler so that it selects the bottom-level scheduler S_i with minimum $f_{i,j}$ among schedulers with positive credit value. The computation of $f_{i,j}$ is as follows. Since C_i represents the service time that S_i received by time t under the GPS discipline, $\Delta - C_i$ represents the remaining service time for $q_{i,j}$. For $q_{i,j}$ to complete, it needs $\frac{\Delta - C_i}{r_i / \sum_j r_j}$. Since the term $\sum_j r_j$ is common for all S_i, the modified algorithm selects the scheduler with

$$min\{\frac{\Delta - C_i}{r_i}\} \quad \text{and} \quad C_i \geq 0. \tag{1}$$

We call the modified algorithm *Earliest Finish Time C/D (EFT-C/D)*. Note that EFT-C/D algorithm works in the same way as the WF^{2}Q which selects the packet with the earliest departure time among packets that have started receiving service under GPS.

Figure 3 illustrates an example of the EFT-C/D algorithm. The set of bottom-level schedulers and bandwidth requirements are the same as in Figure 2. As an illustration, we show the selection of a bottom-level scheduler by EFT-C/D algorithm at $t = 3\Delta$. Since $C_2 = -4 < 0$, we calculate $\frac{\Delta - C_i}{r_i}$ only for S_1 and S_3 as below.

$$\frac{10-1}{7} = \frac{9}{7}, \quad \frac{10-3}{1} = 7$$

The minimum of these two values is $\min(\frac{9}{7}, 7) = \frac{9}{7}$. Thus, S_1 is selected by the EFT-C/D scheduler, whereas S_3 would be selected by the basic credit/debit scheduler.

Now we show the analysis for the EFT-C/D algorithm. Consider two scheduling systems that differ only by the service discipline, one using the EFT-C/D scheduler and one using the GPS server at the top-level. Note that the two scheduling systems are identical with the exception of the top-level service discipline. Thus, they have the same speed, same bottom-level schedulers with same service share, same set of tasks, and same task arrival patterns.

Bottom-level scheduler (S_i)	Relative share (r_i)	Credit (C_i)									
S_1	7	[0]	-3	[4]	[1]	-2	[5]	[2]	-1	[6]	[3]
S_2	2	0	[2]	-6	-4	-2	0	2	[4]	-4	-2
S_3	1	0	1	2	3	[4]	-5	-4	-3	-2	-1
	time (t)	0	1Δ	2Δ	3Δ	4Δ	5Δ	6Δ	7Δ	8Δ	9Δ

Figure 3: An example of the EFT-C/D algorithm with $\Delta = 10$.

Let $W_i^{CD}(0,t)$ be the service time that S_i received in the interval $[0,t]$ under EFT-C/D, and let $W_i^{GPS}(0,t)$ be the service time that S_i received in the interval $[0,t]$ under GPS. Then, the service lag of S_i is

$$E_i(t) = W_i^{GPS}(0,t) - W_i^{CD}(0,t). \tag{2}$$

Since the service lag represents the allocation accuracy of CD algorithm with respect to the GPS, we use $E_i(t)$ in evaluating the fairness provided by the EFT-C/D algorithm. The following theorem shows that the service lag $E_i(t)$ of any S_i under CD is bounded by the quantum size Δ.

Theorem 1 *The service lag of the EFT-C/D algorithm satisfies the following*

$$|E_i(t)| = |W_i^{GPS}(0,t) - W_i^{CD}(0,t)| \leq \Delta \tag{3}$$

for any $t > 0$.

Proof. See Theorem 1 in [1]. □

4 Deadline Guarantees at Bottom-Level EDF Scheduler

We now present a schedulability analysis for the bottom-level EDF scheduler. First, we describe our analysis techniques for normal EDF schedulers from our earlier work in [10], and then use those techniques to develop a schedulability condition for the bottom-level EDF scheduler in our hierarchical structure.

4.1 Overview of Utilization Demand Analysis

Consider a set of aperiodic tasks $Q = \{\tau_1, \tau_2, \ldots, \tau_i, \ldots\}$ under priority driven scheduling policy. We associate two dynamic variables with each task $\tau_i \in Q(t)$, maximum residual execution time $e_{i,t}^{res}$ and lead time $d_{i,t}^{res}$. At time t, the maximum residual execution time $e_{i,t}^{res}$ of τ_i is the maximum of remaining processor time to complete τ_i. The lead time $d_{i,t}^{res}$ of τ_i is the difference between its absolute deadline D_i and the current time t [7], i.e., $D_i - t$. Then, a utilization demand of $\tau_i \in Q(t)$ is defined as the processor time required to meet its deadline divided by its lead time. Let $Q(t, hp(\tau_i)) \subset Q(t)$ be the set of tasks that have higher priorities than τ_i. The utilization demand of τ_i is defined by

$$u_{Q(t)}(\tau_i) \stackrel{\text{def}}{=} \frac{\sum_{\tau_j \in Q(t, hp(\tau_i))} e_{j,t}^{res} + e_{i,t}^{res}}{d_{i,t}^{res}}. \tag{4}$$

The maximum utilization demand $U_Q(t)$ is defined for the set $Q(t)$ as below.

$$U_{Q(t)} \stackrel{\text{def}}{=} \max_i [u_{Q(t)}(\tau_i)] \tag{5}$$

The following theorem shows a necessary and sufficient schedulability condition for an aperiodic task set.

Theorem 2. *Aperiodic task set $Q(t) = \{\tau_m, \tau_{m+1}, \ldots, \tau_n\}$ is schedulable if and only if*

$$U_{Q(t)} \leq 1. \tag{6}$$

Proof. See [10]. □

The utilization demand can also be defined for periodic tasks since all instances of periodic tasks can be considered aperiodic tasks. Suppose that $P = \{\tilde{\tau}_1, \tilde{\tau}_2, \ldots, \tilde{\tau}_N\}$ is a set of periodic tasks. This periodic task set can be associated with an equivalent aperiodic task set Q_P which consists of all task instances generated by P. Thus, P is schedulable if and only if all tasks in Q_P are schedulable. The following theorem states that the utilization of P is equal to or greater than the maximum utilization demand of Q_P.

Theorem 3. *Let $U_P = \sum_{i=1}^{N} \frac{\tilde{e}_i^{max}}{\tilde{T}_i}$ be the utilization of periodic task set $P\{\tilde{\tau}_1, \tilde{\tau}_2, \ldots, \tilde{\tau}_N\}$. If P is*

schedulable by EDF, then

$$U_{Q_P(t)} \leq U_P \tag{7}$$

for all $t \geq 0$.

Proof. See [10]. □

Let P be the set of periodic tasks and its utilization be U_P. The following theorem gives a sufficient condition for a mixed set of periodic and aperiodic tasks.

Theorem 4. *Given periodic task set* P *and aperiodic task set* $Q(t)$*, if* $U_P + U_{Q(t)} \leq 1$*, then* $P \cup Q(t)$ *is schedulable by an EDF scheduler.*

Proof. See [10]. □

4.2 Schedulability Analysis

Consider a bottom-level scheduler S_i and the quantum allocation in Figure 4. By using Theorem 1, we can see that the maximum length of the interval between two consecutive time quanta allocated to S_i is given by $\frac{\sum_{j=1}^{n} r_j}{r_i} \cdot 2\Delta - \Delta$. Thus, it follows that the bottom-level scheduler S_i can receive at least Δ time in the interval $[t, t + \frac{\sum_{j=1}^{n} r_j}{r_i} \cdot 2\Delta]$ for any $t \geq 0$. In general, if we consider m consecutive quanta, the scheduler can receive $m-1$ quanta in the interval $[t, t + \frac{\sum_{j=1}^{n} r_j}{r_i} \cdot m\Delta]$ for any $t \geq 0$. Thus, it is always guaranteed that the bottom-level scheduler is provided a utilization

$$U_i^{lower} = \frac{(m-1)\Delta}{\frac{\sum_{j=1}^{n} r_j}{r_i} \cdot m\Delta} = \frac{m-1}{m} \frac{r_i}{\sum_{j=1}^{n} r_j}. \tag{8}$$

in the interval $[t, t + \frac{\sum_{j=1}^{n} r_j}{r_i} \cdot m\Delta]$ for any $t \geq 0$. Note that if we take any interval shorter than $\frac{\sum_{j=1}^{n} r_j}{r_i} \cdot 2\Delta$, then we cannot guarantee that the scheduler is serviced. This also means that the scheduler cannot guarantee any deadlines that are shorter than $\frac{\sum_{j=1}^{n} r_j}{r_i} \cdot 2\Delta$.

Let $G(t) = P \cup Q(t)$ be the mixed set of periodic and aperiodic tasks scheduled by the bottom-level EDF scheduler, and let D be the minimum deadline such that $D = \min\{d_k | \tau_k \in G\}$. We only consider the case of $D \geq \frac{\sum_{j=1}^{n} r_j}{r_i} \cdot 2\Delta$. For simplicity, assume that the speed of underlying resource is 1. Then we can derive a schedulability test based on the worst-case utilization.

Theorem 5. *Suppose that* $G = P \cup Q(t)$ *be the mixed set of periodic and aperiodic tasks. Let* M *be the*

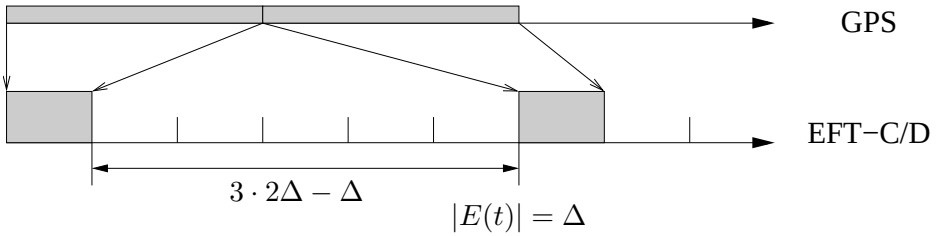

Figure 4: Maximum length of the interval between two consecutive time quanta when $\frac{r_i}{\sum_{f=0}^{n} r_f} = \frac{1}{3}$.

maximum of m *such that* $\frac{\sum_{j=1}^{n} r_j}{r_i} \cdot m\Delta \leq D$. *If* $U_{G(t)}$ *is always less than* $\frac{M-1}{M}\frac{r_i}{\sum_{j=1}^{n} r_j}$ *according to EDF algorithm on a slow processor with speed* $\frac{M-1}{M}\frac{r_i}{\sum_{j=1}^{n} r_j}$, *then they are also schedulable by the bottom-level EDF scheduler under EFT-C/D discipline.*

Proof. Let τ_k be any aperiodic task or an instance of periodic task of $G(t)$. It suffices to show that the bottom-level EDF scheduler receives sufficient resource time to complete τ_k during the interval $[a_k, a_k + d_k]$. By definition, the amount of resource time needed for τ_k to complete by its deadline is $d_k \cdot u_{G(t)}(\tau_k)$. From Eq.(8), the bottom-level EDF scheduler S_i can receive no less than $d_k \cdot \frac{M-1}{M}\frac{r_i}{\sum_{j=1}^{n} r_j}$. Since

$$u_{G(t)}(\tau_k) \leq U_{G(t)} \leq \frac{M-1}{M}\frac{r_i}{\sum_{j=1}^{n} r_j},$$

it immediately follows that

$$d_k \cdot u_{G(t)}(\tau_k) \leq d_k \cdot \frac{M-1}{M}\frac{r_i}{\sum_{j=1}^{n} r_j}.$$

This completes the proof. □

5 Conclusion

We have presented a hierarchical scheduler to handle both bandwidth and deadline requirements in processor time allocation within an end system. The hierarchical scheduler is based on the Earliest Finish Time C/D (EFT-C/D) at the top-level to allocate the time shared resource to each bottom-level scheduler in proportion to specified rate share. We have shown that EFT-C/D emulates the WF^2Q discipline and that it achieves nearly perfect fairness without establishing virtual time. We have also provided a utilization-based schedulability analysis for the EDF scheduler at the bottom-level. Our analysis allows us to easily test the schedulability of a given task set by simply computing required CPU utilization, thus enabling on-line admission control.

There are several future directions. First, we may allow adaptive rate modification for each bottom-level scheduler in response to run-time workload variation. When the workload of a bottom-level scheduler requires more resource share while others require less, then it would be desirable to adjust the rate share accordingly. Second, we can also increase the flexibility our hierarchical scheduler by allowing dynamic attachment and detachment of schedulers at bottom-level depending on the workload characteristics. Finally, we are currently investigating if the original credit/debit algorithm can be proven to achieve the same degree of fairness as EFT-C/D.

References

[1] J. Bennett and H. Zhang. Wf2q: Worst-case fair weighted fair queueing. In *Proceedings of IEEE INFOCOM*, 1996.

[2] A. Demers, S. Keshav, , and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Proceedings of ACM SIGCOMM*, 1989.

[3] Z. Deng and J. W. S. Liu. Scheduling real-time applications in an open environment. In *Proceedings of the Real-Time Systems Symposium*, 1997.

[4] S. J. Golestani. A self-clocked fair queueing scheme for broadband applications. In *Proceedings of IEEE INFOCOM*, 1994.

[5] Pawan Goyal, Xingang Guo, and Harrick M. Vin. A. A hierarchical cpu scheduler for multimedia operating systems. In *Proceedings of the Second Symposium on Operating Systems Design and Implementation*, 1996.

[6] Hao hua Chu and Klara Nahrstedt. Cpu service classes for multimedia applications. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, 1999.

[7] J. P. Lehoczky. Real-time queueing theory. In *Proceedings of IEEE Real-Time Systems Symposium*, pages 186–195, December 1996.

[8] C. Liu and J. Layland. Scheduling algorithm for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, January 1973.

[9] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, 1993.

[10] Minsoo Ryu and Seung-Jean Kim. Dynamic and statistical deadline guarantees for a mixed set of periodic and aperiodic tasks. In *Proceedings of International Conference on Real-Time and Embedded Computing Systems and Applications*, 2003.

[11] I. Stoica, H. Abdel-Wahab, K. Jeffay, S. K. Baruah, J. E. Gehrke, and C. G. Plaxton. A proportional share resource allocation algorithm for real-time, time-shared systems. In *Proceedings of IEEE Real-Time Systems Symposium*, December 1996.

[12] Carl A. Waldspurger. *Lottery and Stride Scheduling: Flexible Proportional-Share Resource Management.* PhD thesis, Massachusetts Institute of Technology, September 1995.