Deterministic and Statistical Deadline Guarantees for a Mixed Set of Periodic and Aperiodic Tasks *

Minsoo Ryu and Seung-Jean Kim

School of Electrical Engineering and Computer Science,
Seoul National University, San 56-1,
Shillim-Dong, Gwanak-Ku, Seoul 151-742, Korea
msryu@redwood.snu.ac.kr and sjkim@stanford.edu

Abstract. Current hard real-time technologies are unable to support a new class of applications that have real-time constraints but with dynamic request arrivals and unpredictable resource requirements. We propose two new admission control approaches to address this problem. First, we present an efficient schedulability test, called utilization demand analysis, to handle periodic and aperiodic tasks with deterministic execution times. The utilization demand is defined as the processor utilization required for a mixed task set to meet deadlines with certainty, thus for deterministic deadline guarantees. We show that the utilization demand analysis eliminates the need for complicated schedulability analysis and enables on-line admission control. Second, we present a statistical admission control scheme using effective execution times to handle stochastic execution times. Effective execution times are determined from the deadline miss probability demanded by the application and stochastic properties of task execution times. Every task is associated with an effective execution time and is restricted to using processor time not exceeding its effective execution time. This scheme allows every task to meet its deadline with a specified probability without being interfered with, and greatly simplifies the admission control when combined with the utilization demand analysis.

1 Introduction

The emergence of distributed multimedia applications with demanding QoS requirements is setting forth new challenges for real-time systems. Such new applications including video conferencing and interactive distance learning require real-time performance guarantees for the delivery and processing of continuous media data. However, despite recent developments in real-time computing, current hard real-time solutions cannot be directly applied to these applications.

* 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).

While most real-time research has put an emphasis on the periodic task model [15, 2, 12, 3, 14] in which task arrivals and execution times are deterministic, multimedia applications have two distinguishing characteristics. First, processor usage patterns include both periodic and aperiodic tasks. For example, a query for continuous media requires periodic tasks for delivery and processing of continuous data, and a query on a database of static data types requires aperiodic tasks. Second, task execution times are either deterministic or stochastic, such as CBR (constant bit rate) video data versus VBR (variable bit rate) data.

In this paper, we attempt to provide deadline guarantees via admission control for real-time tasks while allowing randomness in arrivals and execution times. Such deadline guarantees can be either deterministic or statistical depending on the characteristics of task execution times. When task execution times are upper bounded and their bounds are known, deterministic deadline guarantees can be provided so that all tasks meet deadlines at run-time. The deterministic guarantee provides the highest level of deadline guarantees, however, it may be an overly conservative approach for some multimedia applications which are not greatly impacted by infrequent deadline misses. This necessitates statistical deadline guarantees. When task execution times are not bounded or exhibit great variability, a statistical approach provides probabilistic deadline guarantees with a specified probability.

We present new admission control approaches for both types of deadline guarantees. First, we propose an efficient schedulability test, called *utilization demand analysis*, to handle periodic and aperiodic tasks with deterministic execution times. The utilization demand is defined as the processor utilization required for a mixed task set to meet all deadlines. We use the utilization demand to develop a schedulability test for deterministic deadline guarantees under EDF. We show that the utilization demand analysis eliminates the need for complicated schedulability analysis and enables on-line admission control. Also, as we will see later, the utilization demand provides a useful means for statistical deadline guarantees.

Second, we present two admission control schemes to provide statistical deadline guarantees by bounding the probability that tasks miss deadlines. In general, priority driven scheduling algorithms like EDF, unlike WFQ (weighted fair queueing), inherently lack the "isolation" mechanism to protect tasks from one another. If a task runs arbitrarily long, bounding deadline miss probabilities of its subsequent tasks is significantly problematic. To overcome this problem, we propose to discard tasks that match specific criteria. Our first approach is to discard tasks missing deadlines, and this allows us to compute deadline miss probabilities under the worst case. The shortcoming of this approach, however, is that it leads to computationally complex algorithms since computing probabilities generally requires expensive convolution operations. Our second approach improves upon the first one by aggressively discarding tasks. We use *effective execution times* which are determined from the deadline miss probability demanded by the application and stochastic properties of execution times. Every task is associated with an effective execution time and is restricted to using processor

time not exceeding its effective execution time. If a task consumes processor time more than its effective execution time, it is immediately discarded. This scheme allows every task to meet its deadline with a specified probability without being interfered with, and greatly simplifies the admission control when combined with the utilization demand analysis.

1.1 Related Work

A number of techniques have been proposed to handle mixes of periodic and aperiodic tasks [13, 16, 6, 17, 7, 8]. The algorithms in [13, 16, 6, 17] assume that aperiodic tasks are soft real-time and give preferential treatment to periodic tasks. In these aproaches, aperiodic tasks are handled at a lower priority level in the background, or at a some fixed priority level by a special periodic task which serves aperiodic requests with its limited capacity. The algorithms proposed in [11, 5] handle aperiodic tasks with explicit deadlines. Also, they are known to be optimal with regard to specific criteria, for example, of the response time or processor utilization. However, they not only require complete knowledge of the periodic tasks, but also have high computational complexities when used on-line. In our model, all aperiodic tasks have explicit deadlines and are scheduled by the same scheduling policy as periodic tasks. Moreover, our utilization demand method eliminates the need for complicated schedulability analysis, requiring low run-time overhead.

In the meantime, several researchers have worked on non-deterministic solutions to real-time scheduling problems with stochastic execution times. The statistical rate monotonic scheduling (SRMS) in [1] is a non-deterministic version of the classical rate monotonic scheduling. Under the assumption that the accurate execution time of a task is known when the task arrives, SRMS allows one to compute the percentage of deadline misses. Tia *et al.* [18] proposed two methods to handle stochastic task execution times, *probabilistic time-demand analysis* and *transform-task method*. The probabilistic time-demand analysis attempts to provide a lower bound on the probability that a periodic task meets its deadline under fixed priority scheduling. The probabilistic time-demand analysis is based on the notion of critical instant at which the first instances in all periodic tasks are released simultaneously. The critical instant leads to the worst case when all tasks complete before their deadlines, i.e., when no backlog exists. However, it has not been proven for unbounded execution times that the critical instant is the worst case. Another method, called transform-task method, divides each task into a periodic task and a sporadic task. The periodic task has the same period as the original task and has a fixed execution time that should be chosen such that all the periodic tasks in the system are schedulable. If the actual execution time of a periodic task is larger than the fixed execution time at run-time, the excessive portion of the task is modeled as a sporadic task that can be scheduled by either a sporadic server or a slack stealing algorithm.

The key idea of our effective execution time method is similar to that of the transform-task method in that each task is associated with a fixed amount of execution time and its processor usage is enforced accordingly. Our contribution

is to give a formal definition of effective execution times based on the notion of statistical schedulability and to combine effective execution times with the utilization demand analysis in order to provide an efficient, statistical version of admission control scheme. In fact, the use of effective execution times allows us to easily extend existing deterministic scheduling algorithms and analysis techniques to handle stochastic execution times.

The remainder of this paper is organized as follows. In Section 2, we discuss our models and assumptions. Section 3 describes the utilization demand method for schedulability analysis of aperiodic tasks with known worst case execution times. This method is then applied to a mixed set of periodic and aperiodic tasks. Section 4 introduces two techniques for statistical deadline guarantees. The first technique bounds deadline miss probabilities by discarding tasks missing deadlines. The second technique uses effective execution times as its discard criterion. We will combine effective execution times with utilization demands to provide an efficient admission test. We then conclude in Section 5.

2 Models and Assumptions

Consider a set of aperiodic tasks $Q = \{\tau_1, \tau_2, \ldots, \tau_i, \ldots\}$ where tasks are in arrival order, i.e., τ_i arrives earlier than τ_{i+1}. We use $Q(t) \subset Q$ to denote the set of tasks that have arrived before t and have not completed by t. Every aperiodic task $\tau_i \in Q$ has an arrival time A_i, an execution time requirement e_i, and a relative deadline d_i from its arrival time. The absolute deadline D_i of τ_i is computed by $D_i = A_i + d_i$. If the execution time e_i is bounded from above, then its least upper bound is denoted by e_i^{max}. Otherwise, we assume that e_i is an independent random variable and is distributed according to probability density function (pdf) $g_{e_i}(e)$.

We use similar notation for periodic tasks. Periodic task $\tilde{\tau}_i$ with period $\tilde{T}_i$ can be considered as a finite or infinite sequence of aperiodic requests. Such aperiodic requests are referred to as *periodic task instances* which are denoted by $\tilde{\tau}_{i,j}$. Each periodic task instance $\tilde{\tau}_{i,j}$ has an execution time requirement $\tilde{e}_{i,j}$ and a common relative deadline $\tilde{d}_i$. Note that we use the periodic task model [15] where the relative deadline of a task is equal to its period, i.e., $\tilde{d}_i = \tilde{T}_i$. If $\tilde{e}_{i,j}$ is upper bounded for all j, then the least upper bound is denoted by $\tilde{e}_i^{max}$. Otherwise, we assume that all $\tilde{e}_{i,j}$ are independent random variables that are identically distributed according to the same probability density function $g_{\tilde{e}_i}(e)$. Unlike aperiodic tasks, we use $\tilde{A}_i$ to denote the release time of the first instance $\tilde{\tau}_{i,1}$. Using this, the absolute deadline $\tilde{D}_{i,j}$ of $\tilde{\tau}_{i,j}$ is computed by $\tilde{D}_{i,j} = \tilde{A}_i + (j-1)\tilde{T}_i + \tilde{d}_i$.

In our discussions, we assume a simple system architecture consisting of two components, an admission controller and a processor scheduler, as in Figure 1. The admission controller, through admit or reject, is responsible for ensuring that the system can provide promised deadline guarantees for all tasks accepted. The processor scheduler in turn allocates processor time to tasks according a particular scheduling algorithm. This simple architecture allows us to consider

a wide variety of models for end system operation and configuration. Note that in the case of deterministic deadline guarantees, a periodic task is said to be schedulable if all instances meet their deadlines. To do so, the admission controller is responsible for admission of all future instances of accepted periodic tasks.

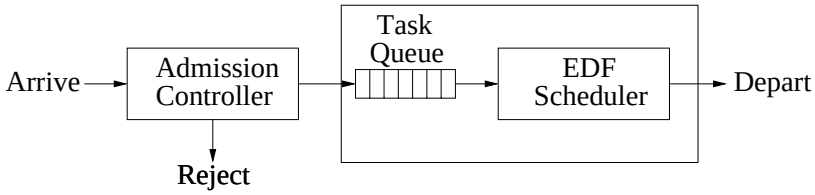

Fig. 1. End system architecture

The scheduling algorithm considered here is earliest deadline first (EDF)[15]. EDF was selected for two reasons. First, EDF is known to be optimal for deterministic deadline guarantees in the sense that it can schedule any task set which is schedulable by any other algorithm. Even though optimality of EDF has not been proven in a statistical environment, it still serves as a benchmark for other scheduling algorithms. Second, EDF algorithm allows for utilization-based schedulability tests which incur little run-time overhead. Under EDF, if the utilization of a task set does not exceed one, then the set is schedulable. We will show that, in the next section, the utilization-based test and our utilization demand analysis can be combined successfully into an integrated schedulability test. Note that though we choose EDF for task scheduling, most of our techniques are applicable to a variety of priority driven scheduling algorithms.

3 Utilization Demand Analysis and Deterministic Deadline Guarantees

In this section we introduce the utilization demand analysis which provides a schedulability test for a mixed task set. We first define utilization demands for aperiodic tasks, and derive a necessary and sufficient schedulability condition. We then develop an integrated schedulability test for a mixed set. The schedulability tests developed in this section are used for deterministic deadline guarantees.

3.1 Utilization Demands for Aperiodic Tasks

Consider a set of aperiodic tasks $Q = \{\tau_1, \tau_2, \ldots, \tau_i, \ldots\}$ under priority driven scheduling policy. In order to determine $Q(t)$ is schedulable at t, we need to consider two dynamic variables for 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

Table 1. Summary of notation

Notation	Meaning
Q	Set of aperiodic tasks
$Q(t)$	Set of aperiodic tasks that have arrived before t and have not completed by t
$Q(t, hp(\tau))$	Set of aperiodic tasks that have higher priorities than τ_i in $Q(t)$
τ_i, $\tilde{\tau}_i$, $\tilde{\tau}_{i,j}$	Aperiodic task, periodic task, periodic task instance
A_i, e_i, d_i, D_i	Arrival time, execution time, relative deadline, and absolute deadline of τ_i
f_i	Finish time of τ_i
e_i^{max}	Worst case execution time of τ_i
$e_{i,t}^{past}$	Allocated processor time for τ_i by t
$e_{i,t}^{res}$	Maximum residual execution time of τ_i at t ($e_{i,t}^{res} = e_i^{max} - e_{i,t}^{past}$)
$d_{i,t}^{res}$	Lead time of τ_i at t ($D_i - t$)
$\tilde{A}_i$, $\tilde{e}_{i,j}$, $\tilde{d}_i$, $\tilde{D}_{i,j}$	Release time, execution time, relative deadline, and absolute deadline of $\tilde{\tau}_{i,j}$
$\tilde{T}_i$	Period of $\tilde{\tau}_i$
$\tilde{e}_i^{max}$	Worst case execution time of $\tilde{\tau}_i$
$g_{e_i}(e)$, $g_{e_{i,t}^{res}}(e)$	pdf of e_i, pdf of $e_{i,t}^{res}$
$u_{Q(t)}(\tau_i)$	Utilization demand of $\tau_i \in Q(t)$
$U_{Q(t)}$	Maximum utilization demand of $Q(t)$

$e_{i,t}^{res}$ 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 [10], i.e., $D_i - t$. Keeping these two dynamic variables provides sufficient information for the schedulability test of $Q(t)$. Table 1 summarizes the notation used throughout this paper.

We are now ready to define utilization demands for aperiodic tasks. Roughly, 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. Since τ_i can start only after its higher-priority tasks complete, we need to consider the sum of residual execution times of itself and its higher-priority tasks. 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

$$u_{Q(t)}(\tau_i) \stackrel{\text{def}}{=} \frac{\sum_{\tau_j \in Q(t, hp(\tau_i))} e_{j,t}^{res} + e_{i,t}^{res}}{d_{i,t}^{res}}. \tag{1}$$

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

$$U_{Q(t)} \stackrel{\text{def}}{=} \max_i [u_{Q(t)}(\tau_i)]. \tag{2}$$

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

Theorem 3.1 *Aperiodic task set* $Q(t) = \{\tau_m, \tau_{m+1}, \ldots, \tau_n\}$ *is schedulable if and only if*

$$U_{Q(t)} \leq 1. \tag{3}$$

Proof. We consider the "if" part first. Let f_i be the worst case finish time of $\tau_i \in Q(t)$. The finish time f_i will be current time plus the sum of residual execution times of higher priority tasks including τ_i's execution time. By the definition of utilization demand in Eq.(1), we have

$$\begin{aligned} f_i &= t + \sum_{\tau_j \in Q(t, hp(\tau_i))} e_{j,t}^{res} + e_{i,t}^{res} \\ &= t + d_{i,t}^{res} \cdot u_{Q(t)}(\tau_i) \\ &= t + (D_i - t) \cdot u_{Q(t)}(\tau_i). \end{aligned}$$

Since $u_{Q(\tau_i, t_i)} \leq U_{Q(t)} \leq 1$,

$$\begin{aligned} t + (D_i - t) \cdot u_{Q(t)}(\tau_i) &\leq t + (D_i - t) \\ &\leq D_i. \end{aligned}$$

Next, we consider the "only if" part. The proof is by contradiction. If we assume that $Q(t) = \{\tau_1, \tau_2, \ldots, \tau_n\}$ is schedulable and $U_Q(t_i) > 1$, then there exists τ_i such that $u_Q(\tau_i, t) > 1$. Hence,

$$\begin{aligned} f_i &= t + \sum_{\tau_j \in Q(t, hp(\tau_i))} e_{j,t}^{res} + e_{i,t}^{res} \\ &= t + (D_i - t) \cdot u_{Q(t)}(\tau_i) \\ &> t + (D_i - t) = D_i. \end{aligned}$$

This contradicts the assumption that $Q(t)$ is schedulable. □

Obviously, a new task arrival affects the schedulability of $Q(t)$ while task departures do not. Therefore, the above schedulability test is valid only until the next arrival time of a new task. This necessitates testing of schedulability at every task arrival. Figure 2 illustrates the maximum utilization demand $U_{Q(t)}$ with several task arrivals and departures. At t_3, the utilization demand jumps to above one. It is easy to show that if $U_{Q(t)}$ is less than one at t, $U_{Q(t)}$ is a decreasing function of time until the next arrival time.

Our second theorem shows the *subadditivity* property of the utilization demand function. This property is essential in devising an integrated shedulability condition for a mixed set of periodic and aperiodic tasks.

Theorem 3.2 *For any two aperiodic task sets,*

$$U_{Q_1(t) \cup Q_2(t)} \leq U_{Q_1(t)} + U_{Q_2(t)}. \tag{4}$$

Proof. See Appendix A.

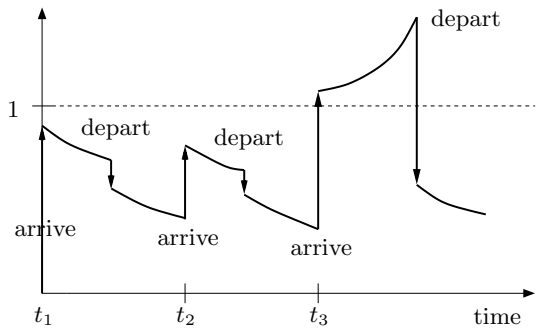

Fig. 2. Utilization demand for a dynamic task set with arrivals and departures

3.2 Schedulabiltiy Condition for a Mixed Task Set

We now generalize the utilization demand analysis for a mixed set of periodic and aperiodic tasks. Basically, all instances of periodic tasks can be considered as aperiodic tasks. This gives a possibility to apply the utilization demand method to periodic 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.

In the following theorem, we show an important relationship between the utilization demand and the utilization of a periodic task set. The following theorem states that the utilization of P is equal to or greater than the maximum utilization demand of Q_P.

Theorem 3.3 *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*

$$U_{Q_P(t)} \leq U_P \tag{5}$$

for all $t \geq 0$.

Proof. For an arbitrary t, suppose that $Q_P(t) = \{\tau_m, \ldots, \tau_i, \ldots, \tau_n\}$. Without loss of generality, assume that the maximum utilization demand is $U_{Q_P(t)} = u_{Q_P(t)}(\tau_i)$. At this moment t, we inject a new periodic task into P such that P is still schedulable. Consider a new periodic task $\tilde{\tau}_*$ whose period is $\tilde{T}_* = D_i - A_i$. We set $\tilde{e}_* = \tilde{T}_* \cdot (1 - U_P)$ so that $U_P + \frac{\tilde{e}_*}{\tilde{T}_*} = 1$, then $P \cup \{\tilde{\tau}_*\}$ will be schedulable by EDF. If we release the first instance $\tilde{\tau}_{*,1}$ immediately before A_i, then $\tilde{\tau}_{*,1}$ has an absolute deadline earlier than τ_i. According to EDF policy, the priority of $\tilde{\tau}_{*,1}$ is higher than that of τ_i. Hence, τ_i would be preempted and delayed by the amount of $\tilde{e}_*$, but τ_i still meets its deadline D_i. Let f_i^{new} be the finish time of delayed τ_i, then we have

$$\begin{aligned} f_i^{new} &= f_i + \tilde{e}_* \\ &= t + (D_i - t) \cdot u_{Q_P(t)}(\tau_i) + \tilde{e}_* \leq D_i. \end{aligned} \tag{6}$$

By subtracting $(t_i + \tilde{e}_*)$ from both sides of Ineq.(6) and deviding both sides by $(D_i - t)$, we have

$$u_{Q_P(t)}(\tau_i) \leq \frac{D_i - \tilde{e}_* - t}{D_i - t} \tag{7}$$

$$= 1 - \frac{\tilde{e}_*}{D_i - t} \tag{8}$$

$$= 1 - (1 - U_P) = U_P. \tag{9}$$

Eq.(9) follows from $\tilde{e}_* = \tilde{T}_*(1 - U_P) = (D_i - t)(1 - U_P)$. This completes the proof. □

We are now able to derive a schedulability condition for a mixed task set. Let P be the set of periodic tasks and its utilization be U_P. The following theorem gives a sufficient condition.

Theorem 3.4 *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. Let Q_P be the equivalent aperiodic task set of P. It suffices to show that $Q_P(t) \cup Q(t)$ is schedulable for any t. We show that $U_{Q_P(t) \cup Q(t)} \leq 1$.

$$U_{Q_P(t) \cup Q(t)} \leq U_{Q_P(t)} + U_{Q(t)} \tag{10}$$

$$\leq U_P + U_{Q(t)} \leq 1. \tag{11}$$

Ineq.(11) follows from Theorem 3.2 and Ineq.(11) follows from Theorem 3.3. This completes the proof. □

Using Theorem 3.4 one can easily determine the schedulability for a mixed task set in a similar fashion as with the utilization-based test for periodic task sets. Note that all periodic tasks can meet deadlines under EDF algorithm if the sum of their utilization factors does not exceed one. It is easy to see that the algorithm for the utilization demand analysis has a run time of $O(n)$ where n is the number of aperiodic tasks in the system. Computing utilization demands requires maintaining small data structure for residual execution times and lead times. Also, this requires low run-time overhead, since these variables need to be computed only when new tasks arrive.

4 Effective Execution Times and Statistical Deadline Guarantees

In this section, we present two statistical approaches to handling stochastic execution times. We use two *task discard* policies to bound deadline miss probabilities. The first approach is based on deadline miss handling. It discards tasks missing deadlines, and this allows us to bound deadline miss probabilities of tasks. The second approach associates each task with a fixed amount of processor time, *effective execution time*, that is allocated to the task. It discards any task whose processor usage exceeds its allocated processor time. Combined with the utilization demand analysis, effective execution times enable an efficient admission control with a surprising simplicity.

4.1 Statistical Deadline Guarantees with Deadline Miss Handling

A statistical approach allows for a small deadline miss probability. Specifically, the probabilistic deadline guarantee is provided in the form of

$$Pr(f_i > D_i) \leq \epsilon \tag{12}$$

where ϵ is generally small, e.g., $\epsilon = 0.01$. Using this condition, we can formally define the statistical version of schedulability.

Definition 1. If the probability that a task τ_i misses its deadline is equal to or less than ϵ, τ_i is said to be *statistically schedulable with probability* $1 - \epsilon$

Consider a task τ_i and a task set Q. We will use the execution time e_i and residual execution $e_{i,t}^{res}$ as random variables throughout this section. The deadline miss probability of τ_i can be stated as

$$Pr(f_i > D_i) = Pr(\sum_{\tau_j \in Q(A_i, hp(\tau_i))} e_{j,A_i}^{res} + \sum_{\tau_k \in Q((A_i, f_i], hp(\tau_i))} e_k + e_i > d_i) \tag{13}$$

where $Q((A_i, f_i], hp(\tau_i))$ contains τ_i's higher priority tasks that will be admitted between the arrival and completion of τ_i. Thus, to provide the statistical guarantee for τ_i, an admission policy must always ensure $Pr(f_i > D_i) \leq \epsilon$ by appropriately maintaining the future task set $Q((A_i, f_i], hp(\tau_i))$. Whenever a new task τ_k arrives, the system needs to ensure $Pr(f_i > D_i) \leq \epsilon$ for every τ_i as well as $Pr(f_k > D_k) \leq \epsilon$ for τ_k.

We now apply Eq.(13) to periodic tasks. As mentioned above, we assume that tasks missing deadlines are immediately discarded. Without this assumption, a periodic task instance $\tilde{\tau}_{i,j}$ may not complete by the release of a subsequent instance $\tilde{\tau}_{i,j+1}$. Since such a backlog $\tilde{\tau}_{i,j}$ can be arbitrarily long, all the subsequent task instances may miss deadlines. This is called the *domino effect* [4]. Discarding tasks that miss deadlines avoids such domino effects and keeps the system predictable. The following theorem provides a statistical schedulability condition for a periodic task set. The intuition that motivates the theorem is that we can find the worst case since future arrivals are known due to the periodicity.

Theorem 4.1 *Suppose tasks missing deadlines are immediately discarded for a given periodic task set* $P = \{\tilde{\tau}_1, \tilde{\tau}_2, \ldots, \tilde{\tau}_N\}$. *Task* $\tau_i \in P$ *is statistically schedulable with probability* $1 - \epsilon$ *if the following holds.*

$$Pr(\tilde{f}_{i,j} > \tilde{D}_{i,j}) \leq Pr(\sum_{k=1}^{N} \tilde{e}_k \cdot (\lfloor \frac{\tilde{T}_i}{\tilde{T}_k} \rfloor + 1) \geq \tilde{T}_i). \tag{14}$$

Proof. Consider the equivalent aperiodic task set Q_P of P. At time $\tilde{A}_{i,j}$, we have $Q_P(\tilde{A}_{i,j}) = \{\tau_m, \ldots, \tau_n\}$ where τ_n is $\tilde{\tau}_{i,j}$. Since $d_n = \tilde{d}_i = \tilde{T}_i$ for τ_n, we can

write Ineq.(13)

$$Pr(f_n > D_n) \\ = Pr(\sum_{\tau_k \in Q_P(A_n, hp(\tau_n))} e^{res}_{k,A_n} + \sum_{\tau_k \in Q_P((A_n, f_n], hp(\tau_n))} e_k + e_n > \tilde{T}_i).$$

We can see that $Q_P(A_n, hp(\tau_n))$ can include no more than one instance per each periodic task $\tilde{\tau}_k \in P$, since all the previous instances are finished or discarded before their deadlines. Thus, we have

$$\sum_{\tau_k \in Q_P(A_n, hp(\tau_n))} e^{res}_{k,A_n} \leq \sum_{\tilde{\tau}_k \in P} \tilde{e}_k. \tag{15}$$

We then find the worst-case workload of $\sum_{\tau_k \in Q_P((A_n, f_n], hp(\tau_n))} e_k + e_n$. For each periodic task $\tilde{\tau}_k \in P$, there are at most $\lfloor \frac{\tilde{T}_i}{\tilde{T}_k} \rfloor$ new arrivals at Q_P in the interval $(A_n, f_n]$ where $\tilde{T}_i = \tilde{d}_i \geq f_n - A_n$. Thus,

$$\sum_{\tau_k \in Q_P(A_n, hp(\tau_n))} e_k + e_n \leq \sum_{\tilde{\tau}_k \in P} \lfloor \frac{\tilde{T}_i}{\tilde{T}_k} \rfloor \tilde{e}_k. \tag{16}$$

It immediately follows from Eq.(15) and Eq.(16)

$$\sum_{\tau_k \in Q_P(A_n, hp(\tau_n))} e^{res}_{k,A_n} + \sum_{\tau_k \in Q_P(A_n, hp(\tau_n))} e_k + e_n \\ \leq \sum_{\tilde{\tau}_k \in P} \tilde{e}_k + \sum_{\tilde{\tau}_k \in P} \lfloor \frac{\tilde{T}_i}{\tilde{T}_k} \rfloor \tilde{e}_k. \tag{17}$$

This leads to Ineq.(14). □

By combining Eq.(13) and Eq.(14), we can obtain the following admission condition for a mixture of a periodic task set $P = \{\tilde{\tau}_1, \tilde{\tau}_2, \ldots, \tilde{\tau}_i, \ldots, \tilde{\tau}_N\}$ and an aperiodic task set $Q(t) = \{\tau_m, \ldots, \tau_j, \ldots, \tau_n\}$. Aperiodic task $\tau_i \in Q(t)$ can be admitted if the following can be satisfied.

$$Pr(f_i > D_i) = Pr(\sum_{\tau_j \in Q(A_i, hp(\tau_i)) \cup Q_P(A_i, hp(\tau_i))} e^{res}_{j,A_i} + \sum_{\tilde{\tau}_j \in P} \lfloor \frac{d_i}{\tilde{T}_j} \rfloor \tilde{e}_j \\ + \sum_{\tau_k \in Q(A_i, f_i], hp(\tau_i))} e_k + e_i > d_i) \leq \epsilon \tag{18}$$

where $\sum_{\tilde{\tau}_i \in P} \lfloor \frac{d_j}{\tilde{T}_i} \rfloor \tilde{e}_i$ represents the sum of execution times of periodic task instances that arrive with higher priorities than τ_i during the execution of τ_i.

Applying the above condition to admission control requires computing deadline miss probabilities at run-time. If task execution times are statistically independent, we can compute deadline miss probabilities by convolving sum of

random variables. For instance, the probability given in Eq.(13) can be written as below.

$$Pr(f_i > D_i)$$
$$= Pr(\sum_{\tau_j \in Q(A_i, hp(\tau_i))} e^{res}_{j,A_i} + \sum_{\tau_k \in Q((A_i, f_i], hp(\tau_i))} e_k + e_i > d_i) \quad (19)$$
$$= 1 - \int_{Q(A_i, hp(\tau_i)) \cup Q_{[A_i, f_i]} \cup \{\tau_i\}} g^{res}_{e_{j,t}}(e) * \ldots g_{e_k}(e) * \ldots g_{e_{i,t}}(e) de \quad (20)$$

where $g_{e^{res}_{j,t}}(e)$ is the pdf of $e^{res}_{j,t}$ for $\tau_j \in Q(A_i, hp(\tau_i))$, $g_{e_k}(e)$ is the pdf of e_k for $\tau_k \in Q((A_i, f_i], hp(\tau_i))$, and $g_{e_{i,t}}(e)$ is the pdf of e_i. Let $e^{past}_{j,t}$ be the processor time consumed by τ_j from its arrival time to current time t. Given the probability density function $g_{e_j}(e)$, we have

$$g_{e_{j,t}}(e) = \begin{cases} 0 & \text{if } e < 0 \\ \frac{g_{e_j}(e + e^{past}_{j,t})}{1 - G_{e_j}(e^{past}_{j,t})} & \text{otherwise} \end{cases} \quad (21)$$

where $G_{e_j}(e^{past}_{j,t}) = \int_0^{e^{past}_{j,t}} g_{e_j}(e) de$.

In fact, the admission control using Eq.(18) leads to computationally complex algorithms since it involves expensive convolution operations. Note that convolution operations are very expensive. For instance, the computational complexity of convolution $g * h$ is known to be $O(n^2)$ where n is the number of points in discretized functions of g and h. Although the run-time overhead can be reduced if we use FFT (Fast Fourier Transform) [9], the algorithm still requires $O(nlog_2 n)$ for $g * h$. Our next approach eliminates the need for convolutions by taking advantage of effective execution times, thus enabling efficient on-line admission control.

4.2 Effective Execution Times and Overrun Handling

The approach in the previous section is based on the assumption that tasks missing deadlines are discarded. This allows us to bound deadline miss probabilities but leads to computationally complex algorithms. Our second approach improves upon this by aggressively discarding tasks. Every task is associated with a particular amount of processor time, called effective execution time, and the admission control is performed using effective execution times. If any task overruns its effective execution time, it is immediately discarded. By overrun, we mean that a task consumes processor time more than its effective execution time.

The objective of preventing task overruns is to isolate tasks from one another. Under this scheme, every task can independently receive processor time up to the amount of its effective execution time. Thus, the deadline miss probability of a task is not adversely affected by other tasks. If we choose appropriate values for effective execution times for a given bound ϵ, tasks can be statistically schedulable with probability $1 - \epsilon$. To choose the minimal processor time required for a

given bound, we can define the effective execution time e_i^ϵ of τ_i as a function of the required deadline miss probability ϵ and probability density function $g_{e_i}(e)$.

$$\int_0^{e^\epsilon} g_{e_i}(x)dx = 1 - \epsilon. \tag{22}$$

Clearly, discarding overrun tasks has the implication that execution times are bounded. The great benefit of this is that it allows us to integrate effective execution times and the deterministic techniques we developed in section 3. Using effective execution times, we can define statistical versions of utilization demand and maximum utilization demand as below.

$$u^\epsilon_{Q(t)}(\tau_i) \stackrel{\text{def}}{=} \frac{\sum_{\tau_j \in Q(t,hp(\tau_i))} e^{res,\epsilon}_{j,t} + e^\epsilon_i}{d^{res}_{i,t}} \quad and \quad U^\epsilon_{Q(t)} \stackrel{\text{def}}{=} \max_i[u^\epsilon_{Q(t)}(\tau_i)] \tag{23}$$

where $e^{res,\epsilon}_{j,t} = e^\epsilon_{j,t} - e^{past}_{j,t}$.

Using the above definitions, the following theorem provides a statistical version of schedulability condition for a mixed set.

Theorem 4.2 *Given a periodic task set P and aperiodic task set $Q(t)$, $Q_P \cup Q(t)$ is statistically schedulable with probability $1-\epsilon$ if the following holds.*

$$U^\epsilon_{Q(t)} + U^\epsilon_P \leq 1 \tag{24}$$

where $U^\epsilon_P = \sum_{\tau_i \in P} \frac{\tilde{e}^\epsilon_i}{T_i}$.

Proof. Let $S(t)$ be $Q(t) \cup Q_P$. Thus, it suffices to show that any aperiodic task τ_i in $S = \{\tau_1, \ldots, \tau_i, \ldots\}$ is statistically schedulable if $U^\epsilon_{S(t)} \leq 1$. Consider the deadline miss probability of $\tau_i \in S$.

$$Pr(f_i > D_i) = Pr(\sum_{\tau_j \in S(A_i, hp(\tau_i))} e^{res}_{j,A_i} + \sum_{\tau_j \in S((A_i,f_i],hp(\tau_i))} e_j + e_i > D_i). \tag{25}$$

Since $e^{res}_j \leq e^{res,\epsilon}_j$ and $e_j \leq e^\epsilon_j$ for any τ_j, we have

$$Pr(f_i > D_i) = Pr(\sum_{\tau_j \in S(A_i, hp(\tau_i))} e^{res}_{j,A_i} + \sum_{\tau_j \in S((A_i,f_i],hp(\tau_i))} e_j + e_i > D_i) \tag{26}$$

$$\geq Pr(\sum_{\tau_j \in S(A_i, hp(\tau_i))} e^{res,\epsilon}_{j,A_i} + \sum_{\tau_j \in S((A_i,f_i],hp(\tau_i))} e^\epsilon_j + e_i > D_i). \tag{27}$$

$U^\epsilon_{S(t)} \leq 1$ implies $\sum_{\tau_j \in S(A_i, hp(\tau_i))} e^{res,\epsilon}_{j,A_i} + \sum_{\tau_j \in S((A_i,f_i],hp(\tau_i))} e^\epsilon_j + e^\epsilon_i \leq D_i$, thus we have

$$\begin{aligned} Pr(f_i > D_i) \geq Pr(&\sum_{\tau_j \in S(A_i, hp(\tau_i))} e^{res,\epsilon}_{j,A_i} + \sum_{\tau_j \in S((A_i,f_i],hp(\tau_i))} e^\epsilon_j + e_i > \\ &\sum_{\tau_j \in S(A_i, hp(\tau_i))} e^{res,\epsilon}_{j,A_i} + \sum_{\tau_j \in S((A_i,f_i],hp(\tau_i))} e^\epsilon_j + e^\epsilon_i) \\ = &Pr(e_i > e^\epsilon_i). \end{aligned} \tag{28}$$

This completes the proof. □

In many applications, it may be unnecessarily stringent to discard overrun tasks. If the system is not overloaded, it is often advantageous to allow overruns as long as its further execution does not interfere with other admitted tasks. There are two other possibilities for handling overruns without affecting statistical guarantees for other admitted tasks. The first one is to give second chances to overrun tasks. Under this, the overrun task, whether it is periodic or aperiodic, is treated as a new aperiodic task. This task can receive processor time if it passes new admission test. The other one is to provide utilization slack. The use of utilization slack is similar to the idea of slack stealing [6, 11]. By Theorem 3.4, we can determine utilization slack and estimate available processor time for an overrun task. The following theorem shows how to estimate available processor time.

Theorem 4.3 *Suppose that $\tau_i \in Q(t) \cup Q_P$ under EDF overruns at time t, where Q_P is an equivalent aperiodic task set of P. Let e_i^{slack} be the available processor time for τ_i such that every task τ_j in $Q(t) \cup Q_P$ is statistically schedulable with probability $1-\epsilon$. The available processor time $e_i^{slack}(t)$ satisfies the following*

$$e_i^{slack}(t) \le d_{i,t}^{res} \cdot (1 - U_P^{\epsilon} - U_{Q(t)}^{\epsilon}) \tag{29}$$

where U_P is the utilization of P.

Proof. Let $S(t)$ be $Q(t) \cup Q_P$. Clearly, τ_i has the highest priority in $S(t)$ at t, since τ_i is executing at t. Thus, if we increase the execution time of τ_i to $e_i + e_i^{slack}$, this affects utilization demands of all the remaining tasks in $S(t)$. Let $\hat{u}_{S(t)}(\tau_j)$ be a new utilization demand for any $\tau_j \in S(t)$, then we can write

$$\hat{u}_{S(t)}(\tau_j) = \frac{\sum_{\tau_k \in S(t,hp(\tau_j))} e_{k,t}^{res,\epsilon} + e_{j,t}^{res,\epsilon} + e_i^{slack}}{d_{j,t}^{res}} \tag{30}$$

$$= \frac{\sum_{\tau_k \in S(t,hp(\tau_j))} e_{k,t}^{res,\epsilon} + e_{j,t}^{res,\epsilon}}{d_{j,t}^{res}} + \frac{d_{i,t}^{res} \cdot (1 - U_P^{\epsilon} - U_{Q(t)}^{\epsilon})}{d_{j,t}^{res}}. \tag{31}$$

Since $d_{i,j}^{res} \le d_{j,t}^{res}$,

$$\hat{u}_{S(t)}(\tau_j) \le \frac{\sum_{\tau_k \in S(t,hp(\tau_j))} e_{k,t}^{res,\epsilon} + e_{j,t}^{res,\epsilon}}{d_{j,t}^{res}} + (1 - U_P^{\epsilon} - U_{Q(t)}^{\epsilon}) \tag{32}$$

$$\le U_P + U_{Q(t)}^{\epsilon} + (1 - U_P^{\epsilon} - U_{Q(t)}^{\epsilon}) = 1. \tag{33}$$

5 Conclusion

We have proposed three approaches to deadline guarantees for a mixed set of periodic and aperiodic tasks. First, we have presented a new schedulability analysis, called utilization demand analysis, which can be applied to periodic and aperiodic tasks with deterministic execution times. We have shown that the

algorithm for this analysis has a run time of $O(n)$, and thus it enables an efficient on-line admission control. Second, we have presented a statistical admission control scheme based on deadline miss handling. By discarding tasks missing deadlines, this scheme allows us to bound deadline miss probabilities of tasks. Third, we have presented an improved statistical scheme using effective execution times. By handling overruns, effective execution times allow tasks to meet deadlines with a specified probability without being interfered with. Combined with the utilization demand analysis, effective execution times greatly simplify the admission control.

There are several future research directions. First, we could extend the untilization demand analysis for fixed priority scheduling algorithms such as rate monotonic (RM) algorithm. Second, we could evaluate a tradeoff between deadline miss probability and throughput of the system. Although we have not considered this problem in this paper, the results presented here will be useful in such evaluation.

References

1. Atlas, A. K., Bestavros, A.: Statistical Rate Monotonic Scheduling. IEEE Real-Time Systems Symposium, IEEE Computer Society Press (1998), 123–132
2. Audsley, N., Burns, A., Richardson, M., Wellings, A.: Hard Real-Time Scheduling: The Deadline-Monotonic Approach. IEEE Workshop on Real-Time Operating Systems and Software (1991), 133–137
3. Baker, T. and Shaw, A.: The Cyclic Executive Model and Ada. The Journal of Real-Time Systems (1989), 1(1):7–25
4. Buttazzo, G.: Value vs. Deadline Scheduling in Overload Conditions. IEEE Real-Time Systems Symposium, IEEE Computer Society Press (1995), 90–99
5. Chetto, H., Chetto, M.: Some Results of the Earliest Deadline First Scheduling Algorithm. IEEE Transactions on Software Engineering, IEEE Computer Society Press (1989), 15(10):1261–1268
6. Davis, R., Tindell, K., Burns, A.: Scheduling Slack Time in Fixed Priority Preemptive Systems. IEEE Real-Time Systems Symposium, IEEE Computer Society Press (1993), 222–231
7. Fohler, G.: Joint Scheduling of Distributed Complex Periodic and Hard Aperiodic Tasks in Statically Scheduled Systems. IEEE Real-Time Systems Symposium, IEEE Computer Society Press (1995), 22–33
8. Isovic, D., Fohler, G.: Online Handling of Hard Aperiodic Tasks in Time Triggered Systems. The 11th Euromicro Conference on Real-Time Systems (1999)
9. Johnson, J. R., Johnson, R. W.: Challenges of Computing the Fast Fourier Transform. Optimized Portable Application Libraries Workshop (1997)
10. Lehoczky, J. P.: Real-Time Queueing Theory. IEEE Real-Time Systems Symposium, IEEE Computer Society Press (1996), 186–195
11. Lehoczky, J. P., Ramos-Thuel, S.: An Optimal Algorithm for Scheduling Soft-Aperiodic Tasks in Fixed-Priority Preemptive Systems. IEEE Real-Time Systems Symposium, IEEE Computer Society Press (1992), 110–123
12. Lehoczky, J. P., Sha, L., Ding, Y.: The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior. IEEE Real-Time Systems Symposium, IEEE Computer Society Press (1989), 166–171

13. Lehoczky, J. P., Sha, L., Strosnider, J.: Enhanced Aperiodic Responsiveness in Hard Real-Time Environments. IEEE Real-Time Systems Symposium, IEEE Computer Society Press (1987), 261–270
14. Leung, J., Merill, M.: A Note on the Preemptive Scheduling of Periodic, Real-Time Tasks. Information Processing Letters (1980), 11(3):115–118
15. Liu, C., Layland, J.: Scheduling Algorithm for Multiprogramming in a Hard Real-Time Environment. Journal of the ACM (1973), 20(1):46–61
16. Sprunt, B., Sha, L., Lehoczky, J. P.: Aperiodic Task Scheduling for Hard-Real-Time Systems. The Journal of Real-Time Systems (1989), 1(1):27–60
17. Spuri, M., Buttazzo, G.: Scheduling Aperiodic Tasks in Dynamic Priority Systems. Journal of Real-Time Systems (1996), 10(2):1979–2012
18. Tia, T.-S., Deng, Z., Shankar, M., Storch, M., Sun, J., Liu, L.-C.: Probabilistic Performance Guarantee for Real-Time Tasks with Varying Computation Times. IEEE Real-Time Technology and Applications Symposium (1995) 164–173

Appendix: Proof of Theorem 3.2

Let $Q_1(t) \cup Q_2(t) = \{\tau_m, \ldots, \tau_p, \ldots, \tau_n\}$. Using Eq.(1) and Eq.(2), we have

$$U_{Q_1(t)\cup Q_2(t)} = max\{\frac{e^{res}_{m,t}}{d^{res}_{m,t}}, \ldots, \frac{e^{res}_{m,t}+, \ldots, +e^{res}_{p,t}}{d^{res}_{p,t}}, \frac{e^{res}_{m,t}+, \ldots, +e^{res}_{n,t}}{d^{res}_{n,t}}\} \quad (34)$$

Suppose that the maximum utilization demand is $U_{Q_1(t)\cup Q_2(t)} = \frac{e^{res}_{1,t}+,\ldots,+e^{res}_{p,t}}{d^{res}_{p,t}}$. Without loss of generality, suppose $\tau_p \in Q_1(t)$. Let $Q^*_1(t) \subset Q_1(t)$ be the set of tasks whose residual execution times $e^{res}_{i,t}$ appear in $\frac{e^{res}_{m,t}+,\ldots,+e^{res}_{p,t}}{d^{res}_{p,t}}$, and let $Q^*_2(t) \subset Q_2(t)$ be the set of tasks whose residual execution times $e^{res}_{j,t}$ appear in $\frac{e^{res}_{m,t}+,\ldots,+e^{res}_{p,t}}{d^{res}_{p,t}}$. Then, we can write

$$\frac{e^{res}_{m,t} + \ldots + e^{res}_{p,t}}{d^{res}_{p,t}} = \frac{\sum_{\tau_i \in Q^*_1(t)} e^{res}_{i,t} + \sum_{\tau_j \in Q^*_2(t)} e^{res}_{j,t}}{d^{res}_{p,t}} \quad (35)$$

Since priorities are assigned according to EDF, $d^{res}_{p,t}$ is the maximum of $\{d^{res}_{i,t} : \tau_i \in Q^*_1(t) \cup Q^*_2(t)\}$. Let $d^{res}_{q,t}$ be the maximum of $\{d^{res}_{i,t} : \tau_i \in Q^*_2(t)\}$, then we have $d^{res}_{q,t} \leq d^{res}_{p,t}$. Hence,

$$\frac{\sum_{\tau_i \in Q^*_1(t)} e^{res}_{i,t} + \sum_{\tau_j \in Q^*_2(t)} e^{res}_{j,t}}{d^{res}_{p,t}} \leq \frac{\sum_{\tau_i \in Q^*_1(t)} e^{res}_{i,t}}{d^{res}_{p,t}} + \frac{\sum_{\tau_j \in Q^*_2(t)} e^{res}_{j,t}}{d^{res}_{q,t}} \quad (36)$$

$$\leq U_{Q_1(t)} + U_{Q_2(t)} \quad (37)$$

This completes the proof.