Global DM schedulability analysis: upper and lower bounds*

Sanjoy Baruah
The University of North Carolina

Abstract

A new technique was recently introduced [7] for the analysis of real-time systems scheduled on multiprocessor platforms by the global Earliest Deadline First (EDF) scheduling algorithm. In this paper, this technique is adapted and applied to the schedulability analysis of real-time systems scheduled on multiprocessor platforms by the global Deadline-Monotonic (DM) scheduling algorithm. It is shown that the resulting analysis technique is quantitatively superior to pre-existing DM schedulability analysis tests; in addition, the degree of its deviation from any hypothetical optimal scheduler (that may be clairvoyant) is also quantitatively bounded.

Keywords: *Global multiprocessor scheduling; sporadic tasks; Deadline Monotonic; schedulability analysis; processor speedup factor*

1 Introduction

Real-time systems comprised of recurrent processes or tasks are often represented using the *sporadic* task model [11, 5]. The *Deadline-Monotonic* (DM) [10] scheduling algorithm is widely used for scheduling such task systems on uniprocessor platforms. Recently, work has been done on extending DM to apply to multiprocessor platforms as well; such work addresses both implementation and analysis issues.

In this paper, we study the DM scheduling of sporadic task systems on preemptive multiprocessor platforms, under the assumption that inter-processor migration is permitted and incurs no penalty. We derive a new sufficient *schedulability test*, and quantify the "goodness" of this test by deriving its *processor speedup factor*. We also prove a lower bound on the processor speedup factor, thereby bounding the degree by which our test deviates from an optimal one.

We describe the problem in greater detail during the remainder of this section.

Task Model. A real-time system is often modeled as a finite collection of independent recurring tasks, each of which generates a potentially infinite sequence of *jobs*. Each job is characterized by an arrival time, an execution requirement, and a deadline, and it is required that a job complete execution between its arrival time and its deadline. Different formal models for recurring tasks place different restrictions on the values of the parameters of jobs generated by each task. One of the more commonly used formal models is the *sporadic task model* [11, 5]. A *sporadic task* $\tau_i = (C_i, D_i, T_i)$ is characterized by a *worst-case execution requirement* C_i, a *(relative) deadline* D_i, and a *minimum inter-arrival separation* parameter T_i, also referred to as the *period* of the task. Such a sporadic task generates a potentially infinite sequence of jobs, with successive job-arrivals separated by at least T_i time units. Each job has a worst-case execution requirement equal to C_i and a deadline that occurs D_i time units after its arrival time. We refer to the interval, of size D_i, between such a job's arrival instant and deadline as its *scheduling window.* A *sporadic task system* is comprised of several such sporadic tasks. Let τ denote a system of such sporadic tasks: $\tau = \{\tau_1, \tau_2, \ldots \tau_n\}$, with $\tau_i = (C_i, D_i, T_i)$ for all i, $1 \leq i \leq n$. Task system τ is said to be a *constrained* sporadic task system if it is guaranteed that each task $\tau_i \in \tau$ has its relative deadline parameter no larger than its period: $D_i \leq T_i$ for all $\tau_i \in \tau$. *We restrict our attention here to constrained task systems.* For notational convenience, we assume tasks are indexed in order of non-decreasing relative deadline parameters: $D_i \leq D_{i+1}$ for all i, $1 \leq i < n$.

*Supported in part by NSF Grant Nos. CNS-0834270, CNS-0834132, and CNS-0615197, ARO Grant No. W911NF-06-1-0425, and funding from IBM, Sun Microsystems, and the Intel Corporation.

Processor Model. We consider a multiprocessor platform that is comprised of several identical processors. We assume that the platform is fully *preemptive* — an executing job may be interrupted at any instant in time and have its execution resumed later with no cost or penalty. We also assume that the platform allows for *global* inter-processor migration – a job may begin execution on any processor and a preempted job may resume execution on the same processor as, or a different processor from, the one it had been executing on prior to preemption. (However, each job may execute on at most one processor at each instant in time.)

Deadline Monotonic scheduling. Priority-driven scheduling algorithms operate as follows: at each instant in time they assign a priority to each job that is awaiting execution, and choose for execution the jobs with the greatest priority. The *Deadline Monotonic* (DM) scheduling algorithm [10] is a priority-driven scheduling algorithm that assigns priority to jobs according to the relative-deadline parameter of the task that generates them — the smaller the relative deadline, the greater the priority — with ties broken arbitrarily but consistently. (Since we assume here that tasks are indexed in order of non-decreasing deadlines, this means that DM assigns jobs of τ_i priority over jobs of τ_{i+1}, for all i.)

With respect to a given platform, a given sporadic task system is said to be *feasible* if there exists a schedule meeting all deadlines, for every collection of jobs that may be generated by the task system. A given sporadic task system is said to be *(global) DM schedulable* if DM meets all deadlines for every collection of jobs that may be generated by the task system. While every DM-schedulable task system is trivially feasible, it is known that not all feasible task systems are DM-schedulable. A *schedulability test* for global DM accepts as input the specifications of a sporadic task system and an identical multiprocessor platform, and determines whether the system is global-DM schedulable upon the platform. Such a test is *exact* if is correctly identifies all schedulable systems, and *sufficient* if it identifies some, but not necessarily all, schedulable systems (however, it must not incorrectly declare some non schedulable system to be global-DM schedulable).

Processor speedup factor Processor speedup factors represent a quantitative approach towards comparing different sufficient schedulability tests. A schedulability test is defined to have a processor speedup factor f, $f \geq 1$, if any task system not deemed to be schedulable by this test upon a particular platform is guaranteed to not be *feasible* – schedulable by an optimal clairvoyant scheduler – upon a platform in which each processor is at most $1/f$ times as fast. More formally,

Definition 1 (Processor speedup factor) *A schedulability test has a processor speedup factor f, $f \geq 1$, if it is guaranteed that any task system that is feasible upon a specified platform is deemed to be schedulable by the test upon a platform in which each processor is at least f times as fast.*

Intuitively, the processor speedup factor of a schedulability test quantifies both the pessimism of the test and the non-optimality of the scheduling algorithm. According to this metric, schedulability tests with smaller processor speedup factors are superior to ones with larger processor speedup factors, with a processor speedup factor equal to one implying both that the scheduling algorithm is optimal and that the test is exact.

It is reasonable to ask whether it makes sense to bundle both the non-optimality of the scheduling algorithm and the pessimism of the test into a single metric. From a pragmatic perspective, we believe that the answer is "yes" for our domain of interest, which is hard-real-time systems. Since it must be a priori guaranteed in such hard-real-time systems that all deadlines will be met during run-time, a scheduling algorithm is only as good as its associated schedulability test. In other words, a scheduling algorithm, no matter how close to optimal, cannot be used in the absence of an associated schedulability test able to guarantee that all deadlines will be met; what matters is that the *combination* of scheduling algorithm and schedulability test together have desirable properties.

Our contributions. The main contribution contained in this paper is a new sufficient schedulability test (Theorem 4) for the global DM scheduling of constrained-deadline sporadic task systems. We prove (Section 5) that our test has a superior processor speedup factor when compared to previously-proposed tests (some of these prior tests are described in Section 3). We also bound the amount by which our schedulability test may deviate from a hypothetical optimal test, by proving a lower bound on the processor speedup factor of any sufficient schedulability test for global DM.

Organization. The remainder of this paper is organized as follows. In Section 2 we present an abstraction for quantifying the computational demand of a sporadic task system. In Section 3, we present some related work that is particu-

larly relevant to the work we are doing here. In Section 4 we derive our new sufficient schedulability test, and in Section 5 we derive its processor speedup factor: these sections are the technical heart of the paper. In Section 6 we obtain a lower bound on the processor speedup factor of any global-DM schedulability test, by constructing a feasible task system that is not schedulable using global DM unless all processors are sped up by a certain minimum amount.

2 The forced-forward demand bound function

In order to devise schedulability tests, it is necessary to quantify the cumulative execution requirement that sequences of jobs may place on a computing platform. Given a sequence of jobs J and a specified time-interval $[t_a, t_f)$, the *demand* of J over $[t_a, t_f)$ is defined to be the sum of the execution requirements of all jobs in J with arrival times $\geq t_a$ and deadlines $\leq t_f$. The *demand bound function* $\text{DBF}(\tau, t)$ of sporadic task system τ for an interval length t is then defined to be the largest demand by any legal sequence of jobs that may be generated by τ over any interval of length t.

However, Baker and Cirinei [1] observed that even jobs whose scheduling windows only intersect partially with an interval can require computation within the interval and hence contribute to the computational demand within it. They introduced a notion that they call *minimum demand*, which refines the demand concept as defined above. The minimum demand of a given collection of jobs in any specific time interval is the minimum amount of execution that the sequence of jobs could require within that interval in order to meet all its deadlines. An essentially identical concept was independently proposed in [7], and called the *necessary demand*; a related concept called *forced-forward demand* was also introduced. This metric was further generalized in [3], to the form discussed below — this metric is general enough to allow for the specification that the execution may be occurring on speed-σ processors, for arbitrary $\sigma > 0$.

Let τ_i denote a sporadic task, t any positive real number, and σ any positive real number ≤ 1. The *forced forward demand bound function* $\text{FF-DBF}(\tau_i, t, \sigma)$ is defined as follows:

$$\text{FF-DBF}(\tau_i, t, \sigma) \stackrel{\text{def}}{=} q_i C_i + \begin{cases} C_i & \text{if } r_i \geq D_i \\ C_i - (D_i - r_i)\sigma & \text{if } D_i > r_i \geq D_i - \frac{C_i}{\sigma} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where

$$q_i \stackrel{\text{def}}{=} \left\lfloor \frac{t}{T_i} \right\rfloor \text{ and } r_i \stackrel{\text{def}}{=} t \bmod T_i \ ,$$

Informally speaking, $\text{FF-DBF}(\tau_i, t, \sigma)$ can be thought of as a bound on the demand of τ_i for interval-length t, when execution *outside* the interval occurs on a speed-σ processor. This function is illustrated for an example task in Figure 1.

As with DBF, the FF-DBF concept extends from individual tasks to task systems as well; for a task system τ

$$\text{FF-DBF}(\tau, t, \sigma) \stackrel{\text{def}}{=} \sum_{\tau_\ell \in \tau} \text{FF-DBF}(\tau_\ell, t, \sigma)$$

$$\text{FF-LOAD}(\tau, \sigma) \stackrel{\text{def}}{=} \max_{t>0} \left(\frac{\text{FF-DBF}(\tau, t, \sigma)}{t} \right) \quad (2)$$

Some additional notation that we will use in this paper. Let τ denote a task system, and τ_k any task in τ:

$$\textit{density } \text{DENS}_k \stackrel{\text{def}}{=} C_k/D_k \ .$$

For the entire task system τ,

$$\textit{Utilization } U(\tau) \stackrel{\text{def}}{=} \sum_{\tau_i \in \tau} C_i/T_i \ .$$

$$\textit{Maximum density } \text{DENS}_{\max}(\tau) \stackrel{\text{def}}{=} \max_{\tau_i \in \tau} \text{DENS}_i.$$

$$\textit{Hyperperiod } P(\tau) \stackrel{\text{def}}{=} \mathsf{lcm}_{\tau_i \in \tau}\{T_i\} \ .$$

3 Prior global-DM schedulability tests

In this section, we briefly review some previously-derived sufficient DM-schedulability tests. We will focus on those tests for which quantitative performance bounds, in the form of processor speedup factors, have been determined. While some other tests have been proposed that have only been evaluated by extensive experimental simulations, such tests are not directly comparable to the new one we will derive here, and hence orthogonal to this paper. The interested reader is referred to Bertogna's dissertation [6] for a comprehensive discussion of such schedulability tests.

In [9], a sufficient test was derived for determining whether a given sporadic task system is global-DM schedulable upon a preemptive multiprocessor platform comprised comprised of m unit-capacity processors. Assume that the tasks in τ are indexed in deadline-monotonic order ($D_i \leq D_{i+1}$ for all i), and let

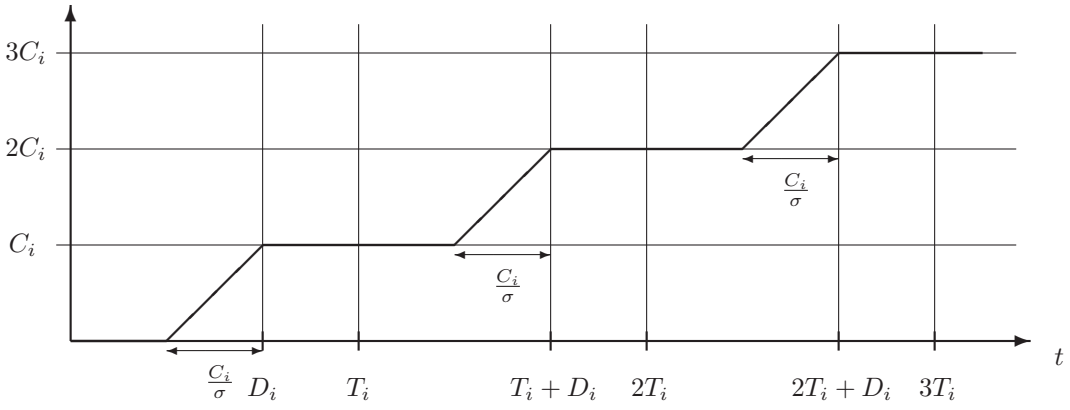

Figure 1. Illustrating FF-DBF(τ_i, t, σ)**.**

Theorem 1 (from [9]) *Sporadic task system τ is global-DM schedulable upon a platform comprised of m unit-capacity processors, provided*

$$\text{LOAD}(k) \leq \frac{m - (m-1)\text{DENS}_k}{3} \tag{3}$$

for all k, $1 \leq k \leq n$.■

It was shown that the processor speedup factor for this global-DM schedulability test is at most $(4 - \frac{1}{m})$, when implemented upon m-processor platforms. This result was subsequently improved [2], and extended to sporadic task systems [4] which are not constrained (i.e., in which individual tasks' relative deadlines may exceed their periods):

Theorem 2 (from [2, 4]) *Sporadic task system τ is global-DM schedulable upon a platform comprised of m unit-capacity processors, provided*

$$\text{LOAD}(k) \leq \max\left(\frac{\mu_k}{3}, \frac{\mu_k - (\lceil \mu_k \rceil - 1)\text{DENS}_{\max}(k)}{2}\right) \tag{4}$$

for all k, $1 \leq k \leq n$, where μ_k is as defined as follows:

$$\mu_k \stackrel{def}{=} m - (m-1)\text{DENS}_k$$

and $\text{DENS}_{\max}(k)$ *denotes the quantity* $\max_{i=1}^{k}\{\text{DENS}_i\}$ *(recall that we're assuming that the tasks in τ are indexed in deadline-monotonic order).*■

The processor speedup factor for this improved schedulability test was shown to equal

$$\frac{2(m-1)}{(4m-1) - \sqrt{12m^2 - 8m + 1}} \tag{5}$$

This approaches $2\sqrt{3}$ (which is ≈ 3.73) as $m \to \infty$.

4 An improved sufficient schedulability condition

In this section, we derive a new sufficient schedulability condition for global DM scheduling of constrained-deadline sporadic task systems, that characterizes DM-schedulability in terms of the forced forward demand bound function. Later, we will show that this new test offers superior performance guarantees — i.e., a smaller processor speedup factor — when compared to the test in Theorem 2.

Theorem 3 *Suppose that constrained-deadline sporadic task system τ is not DM-schedulable upon a platform comprised of m unit-capacity processors. For each s,* $\text{DENS}_{\max}(\tau) < s \leq 1$*,*

$$\text{FF-LOAD}(\tau, s) > \frac{m - (m-1)s}{2} \tag{6}$$

Proof: Suppose that sporadic task system τ is not DM-schedulable. Let $s \geq \text{DENS}_{\max}(\tau)$ denote some value for σ. We will derive a corresponding L such that FF-DBF$(\tau, L, s) > (m - (m-1)s) \times L$.

Consider a minimal sequence of jobs of τ upon which DM misses deadlines. Let t_o denote the (first) instant at which a deadline miss occurs. Let j_1 denote a job that misses its deadline at t_o, and let t_1 denote j_1's arrival-time.

Since lower-priority jobs have no effect on the scheduling of greater-priority jobs, the minimal sequence of jobs on which DM misses a deadline does not include any job with relative deadline $> (t_o - t_1)$. Hence, we note that no jobs with deadline$> t_o + (t_o - t_1)$ are executed in this minimal schedule.

We define a sequence of jobs j_i, time-instants t_i, and an index k, according to the pseudo-code in Figure 2 (also see Figure 3).

for $i \leftarrow 2, 3, \ldots$ **do**
  let j_i denote a job that
    – arrives at some time-instant $t_i < t_{i-1}$;
    – has a deadline after t_{i-1};
    – has not completed execution by t_{i-1}; and
    – has executed for strictly less than $(t_{i-1} - t_i) \cdot s$ units over the interval $[t_i, t_{i-1})$.
  if there is no such job **then**
    $k \leftarrow (i-1)$
    break (out of the for loop)
  end if
end for

Figure 2. Proof of Theorem 3: defining the j_i's, the t_i's and k.

Let L denote the length of the interval $[t_k, t_o + (t_o - t_1))$:

$$\begin{aligned} L &\stackrel{\text{def}}{=} (t_0 - t_k) + (t_o - t_1) \\ &= 2t_o - t_1 - t_k \end{aligned}$$

Since $t_k \leq t_1$, we observe that $(t_o - t_k)$ is $\geq (t_o - t_1)$ and must

$$(t_o - t_k) \geq L/2 \tag{7}$$

For each i, $1 \leq i \leq k$, let W_i denote the total amount of execution that occurs over the interval $[t_i, t_{i-1})$.

Lemma 3.1 $\text{FF-DBF}(\tau, L, s) \geq \sum_{i=1}^{k} W_i$.

Proof: All jobs that execute in $[t_k, t_o)$ (and hence contribute to $\sum_{i=1}^{k} W_i$) have their deadlines within the interval $[t_k, 2t_o - t_1)$. Some of them will also have arrived within this interval, while others may not.

Now it may be verified that the amount of execution that jobs of any task τ_ℓ contribute to $\sum_{i=1}^{k} W_i$ is bounded from above by the scenario in which a job of τ_ℓ has its deadline coincident with the end of the interval, and prior jobs have arrived exactly T_ℓ time-units apart. Under this scenario, the jobs of τ_ℓ that may contribute to $\sum_{i=1}^{k} W_i$ include

- at least $q_\ell \stackrel{\text{def}}{=} \lfloor L/T_\ell \rfloor$ jobs of τ_ℓ that lie entirely within the interval $[t_k, 2t_o - t_1)$; and
- (perhaps) an additional job that has its deadline at time-instant $t_k + r_\ell$, where $r_\ell \stackrel{\text{def}}{=} L \bmod T_\ell$.

We now consider two separate cases:

1. $r_\ell \geq D_\ell$; i.e., the additional job with deadline at $t_k + r_\ell$ arrives at or after t_k. In this case, its contribution is C_ℓ.
2. $r_\ell < D_\ell$; i.e., the additional job with deadline at $t_k + r_\ell$ arrives prior to t_k. From the exit condition of the for-loop, it must be the case that this job has completed at least $(D_\ell - r_\ell) \times s$ units of execution prior to time-instant t_k; hence, its remaining execution is at most $\max(0, C_\ell - (D_\ell - r_\ell) \times s)$.

In either case, it may be seen that the upper bound on the total contribution of τ_ℓ to $\sum_{i=1}^{k} W_i$ is equal to $\text{FF-DBF}(\tau_\ell, L, s)$ (see Equation 1). The lemma follows, by summing over all tasks $\tau_\ell \in \tau$. ■

Lemma 3.2 *For each i, $1 \leq i \leq k$,*

$$W_i > ((m - (m-1)s) \times (t_{i-1} - t_i)$$

Proof: Let x denote the total length of the time-intervals over $[t_i, t_{i-1})$ during which job j_i executes. By choice of job j_i, it is the case that

$$x < (t_{i-1} - t_i) \cdot s$$

By choice of job j_i, it has not completed execution by time-instant t_{i-1}. Hence over $[t_i, t_{i-1})$, all m processors must be executing whenever j_i is not; it follows that

$$\begin{aligned} W_i &\geq m(t_{i-1} - t_i - x) + x \\ &= m(t_{i-1} - t_i) - (m-1)x \\ &> m(t_{i-1} - t_i) - (m-1)(t_{i-1} - t_i)s \\ &= (m - (m-1)s) \times (t_{i-1} - t_i) \end{aligned}$$

and the lemma is proved. ■

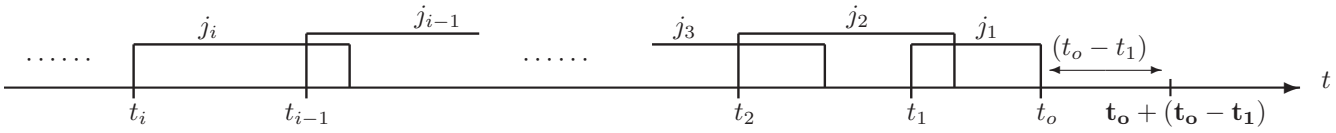

Figure 3. Proof of Theorem 3: notation. Suppose that job j_1 misses its deadline at t_o under DM scheduling. No job with a deadline after $t_o + (t_o - t_1)$ need be considered.

We now observe that

$$\begin{aligned}
&\sum_{i=1}^{k} W_i \\
&> \sum_{i=1}^{k} ((m-(m-1)s) \times (t_{i-1} - t_i)) \quad \text{(By Lemma 3.2)} \\
&= (m-(m-1)s) \times \sum_{i=1}^{k} (t_{i-1} - t_i) \\
&= (m-(m-1)s) \times (t_o - t_k) \\
&\geq (m-(m-1)s) \times \frac{L}{2} \quad \text{(By Equation 7)}
\end{aligned}$$

By Lemma 3.1, we therefore have

$$\begin{aligned}
&\text{FF-DBF}(\tau, L, s) > (m-(m-1)s) \times \frac{L}{2} \\
&\equiv \frac{\text{FF-DBF}(\tau, L, s)}{L} > \frac{1}{2}(m-(m-1)s) \\
&\Rightarrow \text{FF-LOAD}(\tau, s) > \frac{m-(m-1)s}{2}
\end{aligned}$$

and Theorem 3 is proved. ■

Theorem 3 above asserts that, for any DM-schedulable τ, there is some $s \geq \text{DENS}_{\max}(\tau)$ for which Condition 6 is violated. To determine whether a given τ is global DM-schedulable, therefore, our approach would be to methodically search through the range $[\text{DENS}_{\max}(\tau), 1)$ of candidate values for s, seeking to determine whether there is any such s that causes Condition 6 to be violated and thereby validates the schedulability of τ. (A similar search in the context of global-EDF schedulability is detailed in [3] — as can be seen from perusing that paper, the search is somewhat non-trivial.)

However, our objective in this paper is to obtain global-DM schedulability test with the best possible processor speedup factor. For this purpose, it turns out that the following specialized form of Theorem 3 suffices. (This is a "specialization" of Theorem 3 in the sense that it is essentially only checking whether Condition 6 is violated at the single value $s \leftarrow \text{DENS}_{\max}(\tau)$; therefore, while all systems deemed schedulable by Theorem 4 will also be deemed schedulable by Theorem 3, the converse is not true: there are task systems determined to be schedulable according to Theorem 3 that Theorem 4 will not deem schedulable — those for which Condition 6 is only violated for values of s other than $\text{DENS}_{\max}(\tau)$.)

Theorem 4 *Any constrained-deadline sporadic task system τ satisfying*

$$\text{FF-LOAD}(\tau, \text{DENS}_{\max}(\tau)) \leq \frac{1}{2}\big(m-(m-1)\text{DENS}_{\max}(\tau)\big) \tag{8}$$

is guaranteed to be DM-schedulable upon a platform comprised of m unit-capacity processors.

Proof: Immediately follows from Theorem 3, by instantiating the variable s in Inequality 6 to $\text{DENS}_{\max}(\tau)$. ■

5 A speedup result

First, we observe that a *necessary* multiprocessor feasibility condition can be derived in terms of FF-LOAD:

Lemma 1 *If $\text{FF-LOAD}(\tau, s) > ms$ then τ is not feasible on m speed-s processors.*

Proof: Suppose that $\text{FF-LOAD}(\tau, s) > ms$. Let t_o denote a value for t that maximizes the RHS of Equation 2:

$$\text{FF-LOAD}(\tau, s) = \frac{\sum_{\tau_\ell \in \tau} \text{FF-DBF}(\tau_\ell, t_o, s)}{t_o}$$

By definition of FF-DBF, each task τ_ℓ can generate a sequence of jobs that together require $\geq \text{FF-DBF}(\tau_\ell, t_o, s)$ units of execution over some interval of length t_o, when executing upon speed-s processors. Since the different tasks of a sporadic task system are assumed to be independent of each other, such intervals for the different tasks can be aligned; the total execution requirement by all the tasks over

the aligned interval is

$$\begin{aligned} &\geq \sum_{\tau_\ell \in \tau} \text{FF-DBF}(\tau_\ell, t_o, s) \\ &= \text{FF-LOAD}(\tau, s) \times t_o \text{ (By definition of } t_o) \\ &> mst_o \text{ (Since FF-LOAD}(\tau, s) \text{is assumed to be} > ms) \end{aligned}$$

But mst_o denotes the total computing capacity over an interval of size t_o upon m speed-s processors. We therefore conclude that the total execution requirement by all the tasks in τ over the interval cannot be met, and some deadline must necessarily be missed. ■

Lemma 2 *If task system τ fails the DM schedulability test of Theorem 4, then it is not feasible upon a platform comprised of m speed-$\frac{m}{3m-1}$ processors*

Proof: Suppose that τ fails the EDF schedulability test of Theorem 4.

If $\text{DENS}_{\max}(\tau) > (m/(3m-1)$ then τ is trivially not feasible on a platform comprised of (any number of) speed-$(m/(3m-1)$ processors, and we are done.

Assume now that $\text{DENS}_{\max}(\tau) \leq (m/(3m-1)$. Since τ fails the DM schedulability test of Theorem 4, it must be the case that

$$\begin{aligned} \text{FF-LOAD}(\tau, \text{DENS}_{\max}(\tau)) &> \frac{1}{2}\Big(m - (m-1)\frac{m}{3m-1}\Big) \\ &= \frac{3m^2 - m - m^2 + m}{2(3m-1)} \\ &= m\frac{m}{3m-1} \\ &\geq m\,\text{DENS}_{\max}(\tau) \end{aligned}$$

It therefore follows from Lemma 1 above that $\tau$is not feasible upon a platform comprised of m speed-$\text{DENS}_{\max}(\tau)$ processors. Since $\text{DENS}_{\max}(\tau)$ is assumed to be $\leq (m/(3m-1))$, it is therefore not feasible upon a platform comprised of m speed-$\frac{m}{3m-1}$ processors. ■

By taking the contrapositive of Lemma 2 above and observing that $1/(\frac{m}{3m-1})$ is equal to $(3 - \frac{1}{m})$, we have

Theorem 5 *The global-DM sufficient schedulability test of Theorem 4 has a processor speedup factor equal to* $[3 - (1/m)]$.

■

The bound of Theorem 5 approaches 3 as $m \to \infty$. Since the best previously-known speedup factor (Equation 5) approaches ≈ 3.73, we see that this asymptotically represents a significant improvement.

6 A lower bound on processor speedup factor

We have seen, in Theorem 5 above, that our sufficient schedulability test has a processor speedup factor of $(3 - (1/m))$. In this section, we will obtain a lower bound on the processor speedup factor of any global-DM schedulability test. To do so, we will now construct a sporadic task system and show that it is both

1. feasible on a platform comprised of m speed-$(x + \epsilon)$ processors where ϵ is an arbitrarily small positive number while the precise value of x will be specified later[1]; and

2. not schedulable using global-DM on a platform comprised of m speed-1 processors.

Taken together, these facts imply a lower bound of $\frac{1}{x}$ (more precisely, $\lim_{\epsilon \to 0} \frac{1}{x+\epsilon}$) on the processor speedup factor of any global-DM schedulability test.

§**1. The task system.** Our technique for constructing the desired task system is inspired by a similar construction in [8]. Let x denote a constant, $0 < x < \frac{1}{2}$, whose value we will derive later. Let n denote a very large positive integer $(n \to \infty)$.

Let k be defined as follows (since n is assumed to $\to \infty$, observe that $k \to 0$):

$$k \stackrel{\text{def}}{=} \frac{2n-2}{1-x}$$

For each j, $1 \leq j < n$, let

$$T_j \stackrel{\text{def}}{=} x + \frac{n+j-2}{k}$$

Let τ denote the sporadic task system comprised of the following $m(n-1)+1$ tasks:

- For each j, $1 \leq j < n$, there are m tasks each with execution requirement $1/k$, and relative deadline and period both equal to T_j.

- There is one task with relative deadline and period both equal to one, with execution requirement greater than x by an arbitrarily small amount.

[1] This value will turn out to depend on the number of processors m; for $m \to \infty$, $x = (W(2e) - 1) \approx 0.3748$, where $W(t)$ denotes the well-known Omega function). This value of x is derived in the remainder of this section.

Observe that by the definition of the T_j's,

$$\begin{aligned} T_1 &= \frac{1}{2} + \frac{x}{2} \\ T_{j+1} &= T_j + \frac{1}{k} \text{ for all } j, \text{ and} \\ T_{n-1} &= 1 - \frac{1}{k} \end{aligned}$$

§**2. Demonstrating DM-unschedulability.** Suppose that each task has its first job arrive at time-instant 0, and subsequent jobs as soon as legally permitted to do so. According to DM priorities, the $m(n-1)$ jobs each with execution-requirement $1/k$ will be assigned greater priority; together, their execution requirement is

$$\Big(m(n-1) \times \frac{1}{k}\Big) = \Big(m(n-1) \times \frac{1-x}{2n-2}\Big) = \Big(m \times \frac{1-x}{2}\Big)$$

Hence, these jobs execute on all m processors over the interval $[0, \frac{1-x}{2})$.The earliest that the 2nd job of any of these high-priority tasks arrives is $T_1 = \frac{1}{2} + \frac{x}{2}$; hence over the interval $[\frac{1-x}{2}, \frac{1+x}{2})$, which is of length x, the only executing job is the job of the lowest-priority task while the remaining $(m-1)$ processors are idled. Since its execution requirement exceeds x, this lowest-priority job has not completed execution at time-instant $\big(\frac{1+x}{2}\big)$ when more higher-priority jobs arrive.

Since exactly m jobs with period T_j and execution-requirement $\frac{1}{k}$ arrive at each $T_j, 1 \leq j \leq n-1$, and successive T_j's are exactly $\frac{1}{k}$ time units apart, these higher-priority jobs completely consume the processor over the interval $[\frac{1+x}{2}, 1)$. Therefore the lowest-priority job receives no further execution prior to its deadline at time-instant 1, and consequently misses its deadline (and thus bears witness to the non-schedulability of this task system τ by global-DM).

§**3. Computing τ's utilization.** In order to determine the utilization $U(\tau)$ of the task system τ specified above, let us first compute the total utilization of all the tasks other than the lowest-priority one:

$$\begin{aligned} &\sum_{j=1}^{n-1} \Big(m\frac{C_j}{T_j}\Big) \\ &= \sum_{j=1}^{n-1} m\frac{1/k}{x + (n+j-2)/k} \\ &= \sum_{j=1}^{n-1} \frac{m}{kx + n + j - 2} \\ &= \sum_{j=1}^{n-1} \frac{m}{\frac{2n-2}{1-x}x + n + j - 2} \\ &\leq m \int_{\left(\frac{2n-2}{1-x}x + n - 1\right)}^{\left(\frac{2n-2}{1-x}x + 2n - 3\right)} \frac{1}{x-1}\,dx \quad \text{(See Figure 4)} \\ &= m\ln\Big(\frac{2n-2}{1-x}x + 2n - 4\Big) - m\ln\Big(\frac{2n-2}{1-x}x + n - 2\Big) \\ &= m\ln\Big(\frac{(2n-2)x + (2n-4)(1-x)}{(2n-2)x + (n-2)(1-x)}\Big) \\ &= m\ln\Big(\frac{2n + 2x - 4}{n(1+x) - 2}\Big) \end{aligned}$$

As n is made arbitrarily large ($n \to \infty$), this utilization bound approaches $m\ln\Big(\frac{2}{1+x}\Big)$.

The utilization of the entire task system is obtained by adding the lowest-priority task's utilization (which exceeds $x/1$ by an arbitrarily small amount) to this; i.e., $U(\tau)$ exceeds $\Big(x + m\ln\big(\frac{2}{1+x}\big)\Big)$ by an arbitrarily small amount.

§**4. Obtaining the processor speedup factor.** First, we determine the platform comprised of m processors of the slowest speeds, on which τ is feasible.

Lemma 3 *Task system τ is feasible upon a platform comprised of m speed-$(x+\epsilon)$ processors, where x satisfies*

$$x\big(1 - \frac{1}{m}\big) = \ln\big(\frac{2}{1+x}\big) \tag{9}$$

and ϵ is an arbitrary small positive number.

Proof: Observe that each task in the task system τ that we have constructed has its relative deadline and period parameters equal to each other. It is well known that such "implicit-deadline" systems are feasible on multiprocessor platforms provided the processor speed is $\geq \text{DENS}_{\max}(\tau)$ and the total capacity of the platform is $\geq U(\tau)$. Now, $\text{DENS}_{\max}(\tau)$ is equal to the density of the lowest-priority task (since the densities of all other tasks is arbitrarily small as $n \to \infty$), and hence exceeds x by an arbitrarily small amount. Therefore τ is feasible on a platform comprised of m processors of speed exceeding x by an arbitrarily small

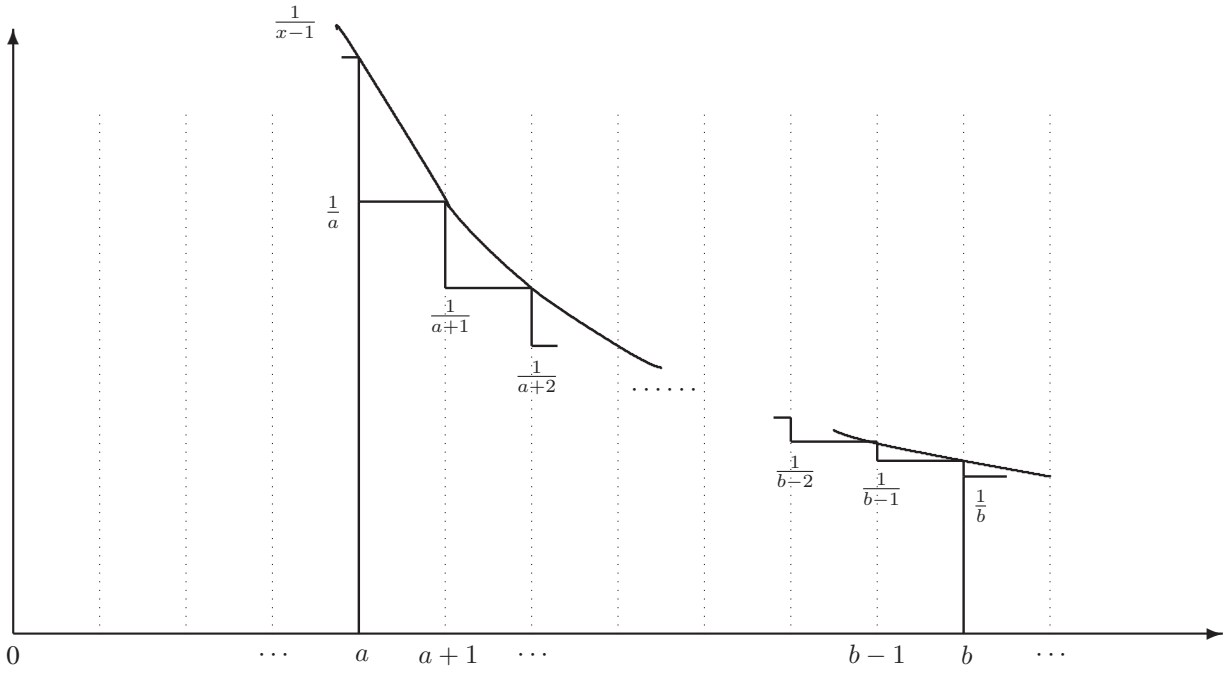

Figure 4. Illustrating that the left Riemann sum $\left(\frac{1}{a}+\frac{1}{a+1}+\cdots+\frac{1}{b-1}+\frac{1}{b}\right)$ is upper-bounded by $\int_a^b \frac{1}{x-1}\,dx$. The area beneath the jagged line between a and b on the x-axis equals the summation; the area beneath the curved line between a and b on the x-axis equals $\int_a^b \frac{1}{x-1}\,dx$.

$\mathbf{m}$:	2	3	4	5	10
$\mathbf{x}$:	0.532	0.466	0.439	0.424	0.398

Table 1. Values of x that satisfy Equation 9 for selected values of m.

amount, where x satisfies

$$\begin{aligned} & mx = x + m\ln(2/(1+x)) \\ \equiv \quad & x(1-(1/m)) = \ln(2/(1+x)) \end{aligned}$$

as claimed in the statement of the lemma. ■

We have therefore constructed a task system that is not schedulable using global-DM on m unit-speed processors but is feasible on m processors each of speed just a bit more than x, where x is as defined by Equation 9. It hence follows that

Theorem 6 *No global-DM schedulability test can have a processor speedup factor smaller than $1/x$, where x is as defined by Equation 9.*

Equation 9 can be solved numerically for different fixed values of m; some of these solutions are presented in Table 1. For very large m, $1/m$ becomes very small; hence as $m \to \infty$ the value of x is asymptotically equal to the solution of the equation

$$\begin{aligned} & x = \ln\Big(\frac{2}{1+x}\Big) \\ \equiv\ & e^x = \frac{2}{1+x} \\ \equiv\ & (1+x)e^{1+x} = 2e \end{aligned}$$

from which we conclude that

$$x = W(2e) - 1\,,$$

where $W(t)$ denotes the *Lambert W-function* (also known as the *Omega function*). Mathematical tables tell us that $W(2e) \approx 1.3748$; i.e., x asymptotically approaches 0.3748 as the number of processors increases. That is, for $m \to \infty$, $(1/0.3748)$ — ≈ 2.668 — is a lower bound on the processor speedup factor of any global-DM schedulability test. By contrast, as $m \to \infty$ the processor speedup factor of the test of Theorem 4 is 3.

7 Conclusions

We have derived a new sufficient schedulability test (Theorem 4) for determining whether a given constrained-deadline sporadic task system is DM-schedulable upon

a preemptive multiprocessor platform, when global inter-processor migration is permitted. We have shown that the processor speedup factor for our test is equal to $(3-(1/m))$, thereby demonstrating that any constrained-deadline sporadic task system that is feasible upon an m-processor multiprocessor platform is correctly identified by our test as being DM-schedulable upon a platform in which each processor is $(3 - (1/m))$ times as fast. We have also computed a lower bound on the processor speedup factor of any sufficient schedulability test for global DM, thereby bounding the degree by which our test deviates from a hypothetical optimal one.

References

[1] BAKER, T., AND CIRINEI, M. A necessary and sometimes sufficient condition for the feasibility of sets of sporadic hard-deadline tasks. In *Proceedings of the IEEE Real-time Systems Symposium* (Rio de Janeiro, December 2006), IEEE Computer Society Press, pp. 178–187.

[2] BARUAH, S. Schedulability analysis of global deadline-monotonic scheduling. Available at `http://www.cs.unc.edu/~ baruah`, 2007.

[3] BARUAH, S., BONIFACI, V., MARCHETTI-SPACCAMELA, A., AND STILLER, S. Implementation of a speedup-optimal global EDF schedulability test. In *Proceedings of the EuroMicro Conference on Real-Time Systems* (Dublin, July 2008), IEEE Computer Society Press.

[4] BARUAH, S., AND FISHER, N. Global deadline-monotonic scheduling of arbitrary-deadline sporadic task systems. In *Proceedings of the 11th International Conference on Principles of Distributed Systems* (Guadeloupe, French West Indies, December 2007), Springer-Verlag.

[5] BARUAH, S., MOK, A., AND ROSIER, L. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the 11th Real-Time Systems Symposium* (Orlando, Florida, 1990), IEEE Computer Society Press, pp. 182–190.

[6] BERTOGNA, M. *Real-Time Scheduling Analysis for Multiprocessor Platforms*. PhD thesis, Scuola Superiore Santa Anna, Pisa, Italy, 2008.

[7] BONIFACI, V., MARCHETTI-SPACCAMELA, A., AND STILLER, S. A constant-approximate feasibility test for multiprocessor real-time scheduling. In *Proceedings of the 16th Annual European Symposium on Algorithms* (Karlsruhe, Germany, 2008), pp. 210–221.

[8] DAVIS, R. L., ROTHVOSS, T., BARUAH, S., AND BURNS, A. Exact quantification of the sub-optimality of uniprocessor fixed-priority pre-emptive scheduling. Under submission; currently available off the author's home-page, 2009.

[9] FISHER, N., AND BARUAH, S. Global static-priority scheduling of sporadic task systems on multiprocessor platforms. In *Proceeding of the IASTED International Conference on Parallel and Distributed Computing and Systems* (Dallas, TX, November 2006), IASTED.

[10] LEUNG, J., AND WHITEHEAD, J. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation 2* (1982), 237–250.

[11] MOK, A. K. *Fundamental Design Problems of Distributed Systems for The Hard-Real-Time Environment*. PhD thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, 1983. Available as Technical Report No. MIT/LCS/TR-297.