Schedulability analysis of global Deadline-Monotonic scheduling

Sanjoy Baruah

Abstract

The multiprocessor Deadline-Monotonic (DM) scheduling of sporadic task systems is studied. A new sufficient schedulability test is presented and proved correct. It is shown that this test offers non-trivial quantitative guarantees, including a resource augmentation bound.

Keywords: Multiprocessor scheduling; real-time systems; sporadic tasks; global scheduling; Deadline Monotonic; schedulability analysis

1 Introduction

In this paper, we study the *Deadline-Monotonic* (DM) scheduling of *sporadic task systems* upon *preemptive* multiprocessor platforms that allow *global* inter-processor migration. We derive a new sufficient *schedulability test*, and obtain *resource-augmentation bounds* that quantify the "goodness" of this test.

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* [6, 1].

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.* We assume tasks are indexed in order or non-decreasing relative deadline parameters: $D_i \leq D_{i+1}$ for all i, $1 \leq i < n$.

Processor Model. In this paper, we study the scheduling of sporadic task systems upon a platform 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 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 [5] 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. (Since we assume 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 DM scheduling accepts as input the specifications of a sporadic task system and an identical multiprocessor platform, and determines whether the system is DM-schedulable upon the platform. Such a test is *exact* if is correctly identifies all DM-schedulable systems, and *sufficient* if it identifies some, but not necessarily all, DM-schedulable systems (however, it must not incorrectly declare some non DM-schedulable system to be DM-schedulable).

Resource augmentation bounds. Resource augmentation bounds are used for quantifying the quality of sufficient schedulability tests. A sufficient schedulability test is said to have a resource-augmentation bound of c if

- Any task system deemed schedulable by the test is guaranteed to actually be so; and
- For any task system that is not deemed schedulable by the test, it is the case that the task system is actually not schedulable upon a platform in which each processor is $\frac{1}{c}$ times as fast.

Intuitively speaking, a resource augmentation bound of c for a sufficient schedulability test implies that the inexactness of the test penalizes its used by at most a speedup factor of c when compared to an exact test. The smaller the resource augmentation bound, the better the sufficient schedulability test: a resource augmentation bound of 1 would mean that the test is in fact an exact one.

Additional definitions. We conclude this section with some definitions and notation that are used in the remainder of this paper.

- The **density** δ_i of a task τ_i is the ratio C_i/D_i of its execution requirement to its relative deadline.
- For each k, $1 \leq k \leq n$, $\delta_{\max}(k)$ denotes the largest density from among the tasks $\tau_1, \tau_2, \ldots, \tau_k$:

$$\delta_{\max}(k) \stackrel{\text{def}}{=} \max_{i=1}^{k}(\delta_i)$$

- For any interval length t, the **demand bound function** $\text{DBF}(\tau_i, t)$ of a sporadic task τ_i bounds the maximum cumulative execution requirement by jobs of τ_i that both arrive in, and have deadlines within, any interval of length t. It has been shown [1] that

$$\text{DBF}(\tau_i, t) = \max\left(0, \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1\right) C_i\right)$$

- For each k, $1 \leq k \leq n$, a **load** parameter, based upon the DBF function, may be defined as follows:

$$\text{LOAD}(k) \stackrel{\text{def}}{=} \max_{t>0}\left(\frac{\sum_{i=1}^{k} \text{DBF}(\tau_i, t)}{t}\right)$$

2 Related work

In [2], 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. We will now briefly review this result from [2].

Consider any legal sequence of job requests of task system τ, on which DM misses a deadline. Suppose that a job J_k of task τ_k is the one to first miss a deadline, and that this deadline miss occurs at time-instant $[t_a + D_k)$ (i.e., this job arrives at time-instant t_a).

It must be the case that all m processors were busy executing jobs of priority greater than J_k's priority, for strictly more than $(D_k - C_k)$ time units over J_k's scheduling window $[t_a, t_a + D_k)$; hence, the total amount of execution that DM tries to fit within J_k's scheduling window is $> m(D_k - C_k) + C_k$, with the added C_k denoting J_k's execution requirement.

In DM scheduling, all jobs of priority greater than J_k's priority must have relative deadline parameter (and hence, scheduling window size) no larger than D_k. In order for these scheduling windows to overlap with $[t_a, t_a + D_k)$, it must be the case that all such jobs arrive no earlier than D_k time units prior to t_a (i.e., after $t_a - D_k$), and have their deadlines no later than D_k time units after $t_a + D_k$ (i.e., before $t_a + 2D_k$). In other words, all these jobs have their arrival times and deadlines within the $3D_k$-sized interval $[a_k - D_k, a_k + 2D_k)$. By definition of $\text{LOAD}(k)$, the total execution requirement of all such jobs is bounded from above by $3D_k\text{LOAD}(k)$. Consequently, in order for job J_k of task τ_k to miss its deadline under DM scheduling it is necessary that

$$\begin{aligned} 3D_k\text{LOAD}(k) &> m(D_k - C_k) + C_k \\ \equiv \quad \text{LOAD}(k) &> \frac{1}{3}(m - (m-1)\delta_k) \end{aligned} \quad (1)$$

We can obtain a sufficient condition for DM-schedulability by negating Inequality 1:

Theorem 1 (from [2]) *Sporadic task system τ is global-*DM *schedulable upon a platform comprised of m unit-capacity processors, provided*

$$\text{LOAD}(k) \leq \frac{m - (m-1)\delta_k}{3} \quad (2)$$

for all k, $1 \leq k \leq n$.■

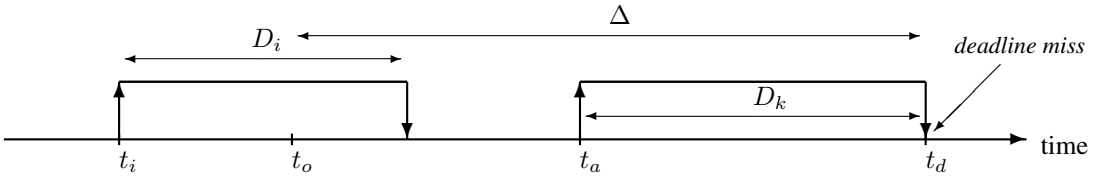

Figure 1. Notation. A job of task τ_k arrives at t_a and misses its deadline at time-instant t_d.

3 An improved test

In this section, we derive (Theorem 2) a new DM-schedulability test that out-performs the test from [2], described in Section 2 above, for many task systems. By combining this new test with the prior one, a superior test (Theorem 3) is obtained.

As in Section 2 above, consider any legal sequence of job requests of task system τ, on which DM misses a deadline. Suppose that a job J_k of task τ_k is the one to first miss a deadline, and that this deadline miss occurs at time-instant t_d (see Figure 1). Let t_a denote this job's arrival time: $t_a = t_d - D_k$.

Discard from the legal sequence of job requests all jobs of tasks with priority lower than τ_k's, and consider the DM schedule of the remaining (legal) sequence of job requests. Since lower-priority jobs have no effect on the scheduling of greater-priority ones under preemptive DM, it follows that a deadline miss of τ_k occurs at time-instant t_d (and this is the earliest deadline miss), in this new DM schedule.

Since all jobs with priority lower than J_k's have been removed and since we are restricting our attention to *constrained* systems, observe that the presence of J_k has no effect whatsoever on whether and when the other jobs in this job arrival sequence are scheduled by DM. We now introduce some notation: for any time-instant $t \leq t_d$,

- let $W(t)$ denote the total amount that all jobs other than J_k execute over the interval $[t, t_d)$ in this DM schedule, plus C_k. (Informally, $W(t)$ denotes the amount of work that DM needs –but fails– to execute over $[t, t_d)$.)

- Let $\Omega(t)$ denote $W(t)$ normalized by the interval-length: $\Omega(t) \stackrel{\text{def}}{=} W(t)/(t_d - t)$.

Since J_k receives strictly less than C_k units of execution over $[t_a, t_d)$, all m processors must be executing tasks other than J_k for a total duration greater than $(D_k - C_k)$ over this interval. Hence it must be the case that

$$W(t_a) > (D_k - C_k)m + C_k$$

i.e.,

$$\begin{aligned} \Omega(t_a) &> \frac{(D_k - C_k)m + C_k}{D_k} \\ &= m - (m-1)\frac{C_k}{D_k} \\ &= m - (m-1)\delta_k \end{aligned}$$

Let

$$\mu_k \stackrel{\text{def}}{=} m - (m-1)\delta_k \tag{3}$$

Let t_o denote the smallest value of $t \leq t_a$ such that $\Omega(t) \geq \mu_k$. Let $\Delta \stackrel{\text{def}}{=} t_d - t_o$ (see Figure 1).

Now the work done by DM over $[t_o, t_d)$ (which we denote as $W(t_o)$) arises from two sources: those jobs that arrived at or after t_o, and those that arrived prior to t_o but have not completed execution in the DM schedule by time-instant t_o. We will refer to jobs arriving prior to t_o that need execution over $[t_o, t_d)$ as ***carry-in jobs.*** (The job of τ_i arriving at time-instant t_i in Figure 1 is an example of a carry-in job, provided it is still active at time-instant t_o.)

How many carry-in jobs can there be? This question is addressed in the following lemma.

Lemma 1 *There are at most $\lceil \mu_k \rceil - 1$ carry-in jobs.*

Proof: Let ϵ denote an arbitrarily small positive number. By definition of the instant t_o, $\Omega(t_o - \epsilon) < \mu_k$ while $\Omega(t_o) \geq \mu_k$; consequently, strictly fewer than μ_k jobs must have been executing at time-instant t_o. And since $\mu_k \leq m$ (as can be seen from Equation 3 above), it follows that some processor was idled over $[t_o - \epsilon, t_o)$, implying that all jobs active at this time would have been executing. This allows us to conclude that there are strictly fewer than μ_k – equivalently, at most $(\lceil \mu_k \rceil - 1)$ – carry-in jobs. ■

Since there are at most $(\lceil \mu_k \rceil - 1)$ carry-in jobs, the total amount of carry-in work is bounded from above by the sum of the $(\lceil \mu_k \rceil - 1)$ largest C_i's from among the execution requirements $C_1, C_2, \ldots, C_k$:

Definition 1 *Let $C_\Sigma(k)$ denote the sum of the $(\lceil \mu_k \rceil - 1)$ largest values from among $[C_1, C_2, \ldots, C_k]$.*

We are now ready to derive a global-DM schedulability test.

Theorem 2 *Sporadic task system τ is global-*DM *schedulable upon a platform comprised of m unit-capacity processors, provided that for all k, $1 \leq k \leq n$,*

$$\frac{C_\Sigma(k)}{D_k} + 2\ \text{LOAD}(k) < \mu_k\ , \tag{4}$$

where μ_k and $C_\Sigma(k)$ are as defined in Equation 3 and Definition 1 above.

Proof: Let us bound the total amount of execution that contributes to $W(t_o)$.

- First, there are the carry-in jobs: as seen above, their total contribution to $W(t_o)$ is bounded from above by $C_\Sigma(k)$.
- All other jobs that contribute to $W(t_o)$ arrive within the Δ-sized interval $[t_o, t_d)$, and hence have their deadlines within $[t_o, t_d + D_k)$, since their relative deadlines are all $\leq D_k$. Their total execution requirement is therefore bounded from above by $(\Delta + D_k) \times \text{LOAD}(k)$.

Considering both sources together, we obtain the following bound on $W(t_o)$:

$$W(t_o) \leq (\Delta + D_k)\text{LOAD}(k) + C_\Sigma(k) \tag{5}$$

Since, by the definition of t_o, it is required that $\Omega(t_o)$ be at least as large as μ, we must have

$$\left(1 + \frac{D_k}{\Delta}\right) \text{LOAD}(k) + \frac{C_\Sigma(k)}{\Delta} \geq \mu_k$$

as a necessary condition for DM to miss a deadline; equivalently, the negation of this condition is sufficient to ensure DM-schedulability:

$$\begin{aligned} &\left(1 + \frac{D_k}{\Delta}\right) \text{LOAD}(k) + \frac{C_\Sigma(k)}{\Delta} < \mu_k \\ \Leftarrow \quad &\text{(since } D_k < \Delta\text{)} \\ &2\ \text{LOAD}(k) + \frac{C_\Sigma(k)}{D_k} \leq \mu_k \end{aligned}$$

which is as claimed in the theorem. ■

We can combine the results from Theorem 1 and Theorem 2 to come up with the following result:

Theorem 3 *Sporadic task system τ is global-*DM *schedulable upon a platform comprised of m unit-capacity processors, provided for all k, $1 \leq k \leq n$,*

$$\text{LOAD}(k) \leq \max\left(\frac{\mu_k}{3}, \frac{\mu_k - C_\Sigma(k)/D_k}{2}\right) \tag{6}$$

where μ_k is as defined in Equation 3 above. ■

3.1 Analysis

The test in Theorem 3 above is the most accurate one we derive in this paper, and should be used for actual schedulability-testing. However, the simpler albeit less accurate form of this test derived in Corollary 1 below is useful for analyzing our test, and comparing it to prior tests.

First, a technical lemma:

Lemma 2

$$\frac{C_\Sigma(k)}{D_k} \leq (\lceil \mu_k \rceil - 1)\delta_{\max}(k)\ .$$

Proof: Since $D_i \leq D_k$ for all $i \leq k$, it follows that each of $C_1, C_2, \ldots, C_k$ is $\leq (D_k\ \delta_{\max}(k))$. Therefore $C_\Sigma(k)$, the sum of $(\lceil \mu_k \rceil - 1)$ C_i's from among $C_1, C_2, \ldots, C_k$, is $\leq (\lceil \mu_k \rceil - 1)(D_k\ \delta_{\max}(k))$. The lemma follows, by dividing both sides by D_k. ■

Lemma 2 immediately yields the following corollary to Theorem 3:

Corollary 1 *Sporadic task system τ is global-*DM *schedulable upon a platform comprised of m unit-capacity processors, provided for all k, $1 \leq k \leq n$,*

$$\text{LOAD}(k) \leq \max\left(\frac{\mu_k}{3}, \frac{\mu_k - (\lceil \mu_k \rceil - 1)\delta_{\max}(k)}{2}\right) \tag{7}$$

where μ_k is as defined in Equation 3 above. ■

The bound of Corollary 1 is depicted visually in Figure 2, for platforms comprised of a large number of processors ($m \to \infty$). The largest density ($\delta_{\max}(k)$) is plotted on the x-axis, and the load normalized by the number of processors ($\text{LOAD}(k)/m$) on the y-axis. The straight line corresponds to the previously-known bound (the one in Theorem 1), and the curve denotes the bound in Theorem 2. Any task system for which $(\delta_{\max}(k), \text{LOAD}(k)/m)$ lies beneath one or both of these curves for all k, is guaranteed to be (correctly) identified as being DM-schedulable by our test. The two curves intersect at a maximum density of $\frac{1}{3}$; we thus see that on such platforms the new test is superior to the old one for sporadic task systems in which the maximum density $\delta_{\max}(n)$ is $< \frac{1}{3}$, while the previous test is superior for systems with $\delta_{\max}(n) > \frac{1}{3}$.

4 A resource augmentation bound

In this section, we obtain a resource augmentation bound for the new DM-schedulability test of Theorem 2. Our approach is as follows. We will identify (Lemma 4) the smallest value of $x \leq 1$ for which we can prove that *any sporadic task system not deemed* DM*-schedulable upon m speed-1 processors by the test of Theorem 2 is infeasible upon m*

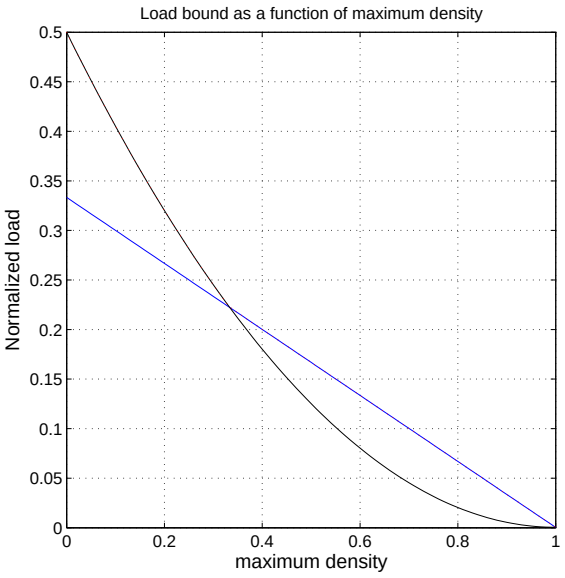

Figure 2. Graphical depiction of Corollary 1. The bound on $\text{LOAD}(k)/m$ is represented as a function of $\delta_{\max}(k)$, for $m \to \infty$. The straight line represents the first term in the $\max$, and the curved line the second.

speed-x processors. It then follows that $\frac{1}{x}$ is a resource augmentation bound for the test of Theorem 2.

We first use Lemma 2 to obtain the following corollary to Theorem 2:

Corollary 2 *Sporadic task system τ is global-*DM *schedulable upon a platform comprised of m unit-capacity processors, provided that for all k, $1 \leq k \leq n$,*

$$\text{LOAD}(k) \leq \frac{1}{2}(m - (m-1)\delta_k)(1 - \delta_{\max}(k)) \quad (8)$$

Proof: From Theorem 2, we know that τ_k meets its deadline provided

$$\begin{aligned} & 2\,\text{LOAD}(k) + \frac{C_\Sigma(k)}{D_k} \leq \mu_k \\ \Leftarrow\ & \text{By Lemma 2} \\ & 2\,\text{LOAD}(k) + (\lceil \mu_k \rceil - 1)\delta_{\max}(k) \leq \mu_k \\ \equiv\ & \text{LOAD}(k) \leq \frac{\mu_k - (\lceil \mu_k \rceil - 1)\delta_{\max}(k)}{2} \\ \Leftarrow\ & \text{Since } \lceil \mu_k \rceil - 1 < \mu_k \\ & \text{LOAD}(k) \leq \frac{1}{2}\,\mu_k\,(1 - \delta_{\max}(k)) \\ \equiv\ & \text{LOAD}(k) \leq \frac{1}{2}(m - (m-1)\delta_k)(1 - \delta_{\max}(k)) \end{aligned}$$

which is as claimed in the corollary. ■

We are almost ready to obtain a resource-augmentation bound. But first, we need another technical lemma.

Lemma 3 *Any sporadic task system τ that is feasible upon a multiprocessor platform comprised of m speed-x processors must satisfy*

$$\delta_{\max}(k) \leq x \ \text{ and } \ \text{LOAD}(k) \leq mx \quad (9)$$

for all k, $1 \leq k \leq n$.

Proof Sketch: Suppose that task system τ is feasible upon m speed-x processors. To prove that $\delta_{\max}(k) \leq x$, consider each task τ_i separately. Since any individual job of τ_i can receive at most $D_i \times x$ units of execution by its deadline, we must have $C_i \leq D_i \times x$; i.e., $C_i/D_i \leq x$. Taken over all tasks in τ, this observation yields the first condition.

To prove that $\text{LOAD}(k) \leq mx$, recall the definition of $\text{LOAD}(k)$ from Section 1. Let t' denote some value of t which defines $\text{LOAD}(k)$:

$$t' \stackrel{\text{def}}{=} \text{argmax}\left(\frac{\sum_{i=1}^{k} \text{DBF}(\tau_i, t)}{t}\right).$$

Suppose that all tasks in $\{\tau_1, \tau_2, \ldots, \tau_k\}$ generate a job at time-instant zero, and each task τ_i generates subsequent jobs exactly T_i time-units apart. The total amount of execution that is available over the interval $[0, t')$ on this platform is equal to mxt'; hence, it is necessary that $\text{LOAD}(k) \leq mx$ if all deadlines are to be met. ■

Using Corollary 2 and Lemma 3, we obtain below a bound on the processor speedup that is sufficient in order for the test of Theorem 2 to identify DM-schedulability:

Lemma 4 *Any sporadic task system that is feasible upon a multiprocessor platform comprised of m speed-x processors platform is determined to be global-*DM *schedulable on m unit-capacity processors by the* DM*-schedulability test of Corollary 2, provided*

$$x \leq \frac{(4m-1) - \sqrt{12m^2 - 8m + 1}}{2(m-1)} \quad (10)$$

Proof: Suppose that τ is feasible upon a platform comprised of m speed-x processors. From Lemma 3, it must be the case that $\text{LOAD}(k) \leq mx$ and $\delta_{\max}(k) \leq x$. For τ to be determined to be DM-schedulable upon m unit-capacity processors by the test of Corollary 2, it is sufficient that:

$$\begin{aligned} & \text{LOAD}(k) \leq \frac{1}{2}(m - (m-1)\delta_k)(1 - \delta_{\max}(k)) \\ \Leftarrow\ & mx \leq \frac{1}{2}(m - (m-1)x)(1 - x) \\ \equiv\ & (m-1)x^2 - (4m-1)x + m \geq 0 \end{aligned}$$

Solving for x using standard techniques for the solution of quadratic inequalities yields Equation 10. ■

m	**res. aug. bound**
3	1.438613184
4	1.81191016
5	2.080891765
10	2.761302893
20	3.199969097
30	3.365582213
50	3.506187357
100	3.616767217
1000	3.720302285

Table 1. The resource augmentation bound as a function of m.

Lemma 4 above bounds from above the values of x such that task systems feasible on speed-x processors are correctly identified as being DM-schedulable by the test of Theorem 2. The resource augmentation bound of Theorem 2 is obtained by taking the multiplicative inverse of this x:

Corollary 3 *The* DM*-schedulability test of Theorem 2 has a resource augmentation bound of*

$$\frac{2(m-1)}{(4m-1)-\sqrt{12m^2-8m+1}} \tag{11}$$

■

Equation 11 expresses the resource augmentation bound as a function of the number of processors m in the platform. The computed values for selected example values of m are listed in Table 1 above. As seen from this table, the bound seems to increase with increasing m. Standard techniques may be used to show that it is indeed the case that the bound of Equation 11 increases with increasing m, approaching $(2+\sqrt{3})$ as $m \to \infty$.

We already know that global DM is not an optimal scheduling algorithm. It is also easy to show that the schedulability test of Theorem 3 is not optimal. The significance of this resource-augmentation result lies in what it tells us about the "goodness" of both global DM and of our schedulability test: in essence, it is asserting that a processor speedup of $(2+\sqrt{3}) \approx 3.73$ compensates for *both* the non-optimality of global DM *and* the inexactness of our schedulability test.

5 Conclusions

We have derived a new sufficient schedulability test 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 this test is superior to previously-known tests for sporadic task systems in which all tasks have density no greater than one-third.

We have also obtained a resource-augmentation bound for our test. This resource bound of $(2+\sqrt{3}) \approx 3.73$ tells us that any constrained-deadline sporadic task system that is feasible upon a multiprocessor platform is correctly identified by our test as being DM-schedulable upon a platform in which each processor is 3.73 times as fast.

References

[1] 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.

[2] 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.

[3] HA, R. *Validating timing constraints in multiprocessor and distributed systems*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1995. Available as Technical Report No. UIUCDCS-R-95-1907.

[4] HA, R., AND LIU, J. W. S. Validating timing constraints in multiprocessor and distributed real-time systems. In *Proceedings of the 14th IEEE International Conference on Distributed Computing Systems* (Los Alamitos, June 1994), IEEE Computer Society Press.

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

[6] 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.