Constant-Time Admission Control for Deadline Monotonic Tasks

Alejandro Masrur Samarjit Chakraborty Georg Färber
Institute for Real-Time Computer Systems, TU Munich, Germany
{Alejandro.Masrur, Samarjit.Chakraborty, Georg.Faerber}@rcs.ei.tum.de

***Abstract*—The admission control problem is concerned with determining whether a new task may be accepted by a system consisting of a set of running tasks, such that the already admitted and the new task are all schedulable. Clearly, admission control decisions are to be taken on-line, and hence, this constitutes a general problem that arises in many real-time and embedded systems. As a result, there has always been a strong interest in developing efficient admission control algorithms for various setups. In this paper, we propose a novel constant-time admission control test for the Deadline Monotonic (DM) policy, i.e., the time taken by the test does not depend on the number of admitted tasks currently in the system. While it is possible to adapt known utilization bounds from the literature to derive constant-time admission control tests (e.g., the Liu and Layland bound, or the more recent hyperbolic bound), the test we propose is less pessimistic. We illustrate this analytically where possible and through a set of detailed experiments. Apart from the practical relevance of the proposed test in the specific context of DM tasks, the underlying technique is general enough and can possibly be extended to other scheduling policies as well.**

I. Introduction

In this paper, we present an efficient constant-time test for admission control of real-time tasks under the Deadline Monotonic (DM) policy. Such an admission control test has to decide whether a new task can be feasibly scheduled (i.e., without causing any deadline misses) together with a set of already accepted tasks currently running in a system. Since admission control decisions are taken on-line, it is important to develop efficient algorithms/tests with predictable execution time which does not depend on the number of tasks in the system.

The DM policy consists in assigning fixed priorities to tasks according to their deadlines: the shorter the deadline, the higher the priority assigned to the task. This scheduling case is of particular practical relevance because fixed priorities are supported by most real-time operating systems and, in addition, DM is the optimal priority assignment for the case that deadlines (d_i) are less than periods (p_i) [1].

Exact schedulability tests are already known for fixed priorities, e.g., [2], [3] and [4], however, they are generally not eligible for admission control. This is because they all have pseudo-polynomial complexity and thus it is difficult to predict their running time (which depends not only on the number of tasks, but also on task parameters, e.g., periods, etc.).

Although the complexity of the admission control problem is the one of testing schedulability for only one new task in the system, an exact schedulability test still results in a pseudo-polynomial-time admission control test. Additionally, all exact schedulability tests require tasks to be sorted according to decreasing (non-increasing) priority, i.e., increasing (non-decreasing) deadlines under DM. Of course, it is possible to sort tasks on-line as they arrive. This way, when a new task arrives, we only need to add it to a sorted list. However, if a new task is added to the system, using an exact schedulability test also implies retesting all already accepted lower-priority tasks. This is time-consuming and can be impracticable particularly for a large number of tasks.

On the other hand, we can adapt the utilization bounds obtained for Rate Monotonic (RM) to the case of DM where $d_i \leq p_i$ holds for all tasks. As discussed later, we can use, for example, the Liu and Layland bound [5] with very little modification. An admission control test based on this bound presents constant complexity $\mathcal{O}(1)$, however, it becomes rather pessimistic as the number of accepted tasks grows.

In order to increase accuracy while retaining constant complexity $\mathcal{O}(1)$, we propose a novel admission control test, called *load test*, for DM tasks with deadlines less than or equal to periods. This test calculates an upper bound on the worst-case response time of tasks considering all already accepted and the arriving task. If this upper bound is always less than or equal to the respective deadlines, the accepted and the new task are going to be schedulable under DM. As shown later, the load test outperforms the constant-time tests that can be designed for DM based on approaches from the literature.

This paper is structured as follows: The next two sections provide a survey of related work and a description of the used task model and notation. Afterwards, the proposed admission control test is presented and analyzed in Section IV. Section V shows the results of an extensive comparison against approaches with the same complexity based on well-known techniques from the literature. Finally, some concluding remarks are discussed in Section VI.

II. Related Work

Numerous utilization bounds have been proposed for RM, which we can modify to design admission control tests for the DM policy. In this section, the most relevant related work is discussed briefly, whereas we analyze how to adapt RM utilization bounds to DM in Section IV.

For a set of preemptive, independent, periodic, real-time tasks with deadlines equal to periods, Liu and Layland proved in [5] that the worst-case scheduling situation on one processor

called *critical instance* occurs when tasks are released simultaneously. Further, they presented in [5] a utilization bound for the case where tasks are scheduled under RM and $d_i = p_i$ holds for all tasks.

A better utilization upper bound for this case was proposed independently by Liu in [6] and by Bini et al. in [7], [8]. This latter utilization bound does not depend on the number of tasks as the Liu and Layland bound does. Bini et al. called it hyperbolic bound and proved that it improves the acceptance ratio over the utilization bound of Liu and Layland by a factor of $\sqrt{2}$ for a large number of tasks. A similar utilization bound was proposed by Oh et al. in [9] to be used in the task allocation problem.

There are some other utilization bounds for the case that $d_i = p_i$ holds for all tasks under RM. For example, Kuo and Mok presented in [10] a bound that exploits the fact that 100% utilization is possible under RM when tasks have harmonic periods. Further, Burchard et al. presented in [11] another utilization bound that varies not only with the number of tasks but also with a factor quantifying how close tasks are to having harmonic periods. For arbitrary deadlines, Lehoczky proposed in [12] an RM utilization bound that depends on the number of tasks and on the ratio $\frac{d_i}{p_i}$ that is assumed to be the same for all tasks.

An exact schedulability test with pseudo-polynomial complexity was presented by Lehoczky et al. in [2] for the case of RM and deadlines equal to periods. In [3], Audsley et al. improved Lehoczky's exact schedulability test by observing that tasks' worst-case response times can be found in an iterative manner. Further, Audsley et al. considered deadlines less than or equal to periods and other priority assignments. More recently, Bini and Buttazzo presented in [4] a tunable schedulability test for the case that $d_i \leq p_i$ holds for all tasks under fixed priorities. Bini and Buttazzo's test allows configuring complexity versus acceptance ratio for the testing.

Finally, in [13], Fisher and Baruah proposed a polynomial-time approximation scheme (PTAS) for the known exact schedulability test [2], [3]. However, as this PTAS also requires tasks to be sorted according to priorities, if a new task is added to the system, all already accepted lower-priority tasks will have to be retested.

III. Task Model and Notation

In this section, we specify the task model and some related notation used in this paper. We denote by $\mathbf{T}$ a set of n periodic, independent, fully preemptive, real-time tasks. Further, we assume that $\mathbf{T}$ is scheduled on one processor under the DM policy. $\mathbf{T}_i \subset \mathbf{T}$ is used to denote the subset of tasks with higher priority than or equal priority to a task $T_i \in \mathbf{T}$.

In principle, each task T_i is an infinite succession of jobs and is characterized by a period of repetition p_i, a relative deadline d_i and a worst-case execution time e_i. Further, the ratio $u_i = \frac{e_i}{p_i}$ is called task utilization and w_i represents T_i's worst-case response time. For this paper, it is assumed that all tasks are released simultaneously at the beginning of the schedule (at time $t = 0$), i.e., $\mathbf{T}$ is a synchronous task set. As already mentioned, all relative deadlines d_i are assumed to be less than or equal to the respective periods p_i for $1 \leq i \leq n$.

IV. Constant-Time Tests for DM

As stated, a number of approaches originally developed for RM may be adapted so as to come up with constant-time tests for admission control under DM. For example, the utilization bound of Liu and Layland [5] can be adapted to:

$$\sum_{i=1}^{n} \frac{e_i}{d_i} \leq n(2^{1/n} - 1). \tag{1}$$

Note that here periods p_i have been replaced by the respective deadlines d_i to reflect the DM rather than the RM policy. The validity of Equation (1) for DM follows from the validity of the Liu and Layland utilization bound for RM and from the fact that $d_i \leq p_i$ holds for all tasks.

Similarly, the hyperbolic bound [6], [7], [8] may be modified to (with p_i being replaced by d_i):

$$\prod_{i=1}^{n} \left(1 + \frac{e_i}{d_i}\right) \leq 2. \tag{2}$$

As we are considering general task sets, i.e., we are not assuming harmonic periods, the utilization bound of Kuo and Mok [10] reduces to the hyperbolic bound.

Further, although the utilization bounds of Burchard et al. [11] and of Lehoczky [12] can also be used in the above manner, our experiments with a large number of synthetic task sets show that the test given by Equation (2) is the least pessimistic. That is, while all the mentioned tests are *safe*, Equation (2) accepts the largest number of schedulable task sets.

The previously discussed methods can be used to perform a constant-time admission control under DM, i.e., the computational complexity for accepting (or rejecting) a new task does not depend on the number of tasks already accepted. The main drawback of these methods is, however, that they are rather pessimistic particularly as the number of accepted tasks grows. In what follows, we introduce a novel admission control test for DM that has constant complexity while it is less pessimistic than the methods already discussed.

A. The Load Test

The schedulability of $\mathbf{T}$ under DM can be tested in pseudo-polynomial time by calculating the worst-case response time w_l for each task T_l in $\mathbf{T}$. If $w_l \leq d_l$ for every T_l, then the task set $\mathbf{T}$ is schedulable under DM [3].

Since tasks are periodic and their deadlines are less than or equal to periods, the worst-case response time of a T_l is the one of its first job in the critical instance, i.e., when this job is released together with jobs of all higher priority tasks [5]. As a consequence, the worst-case response time of a task T_l can be computed in the following manner [3]:

$$t^{(k+1)} = e_l + \sum_{\forall T_i \in \mathbf{T}_l} \left\lceil \frac{t^{(k)}}{p_i} \right\rceil e_i, \tag{3}$$

where $\mathbf{T}_l$ denotes the subset of tasks with higher priority than or equal priority to T_l (i.e., for every T_i in $\mathbf{T}_l$, $d_i \leq d_l$ must hold under the DM policy). Thus, $\sum_{\forall T_i \in \mathbf{T}_l} \lceil \frac{t^{(k)}}{p_i} \rceil e_i$ results in the execution demand of higher (or equal) priority jobs. Equation (3) can be solved iteratively starting from $t^{(1)} = e_l$ and until $t^{(k+1)} = t^{(k)}$ is satisfied for some $k \geq 1$. This resulting value of $t^{(k+1)}$ is T_l's worst-case response time denoted by w_l.

The following two lemmas are used later to prove Theorem 1 and, this way, the validity of the proposed test.

LEMMA 1 *Let the task set* $\mathbf{T}$ *be scheduled under the Deadline Monotonic policy. If* $w_l \leq d_l$ *holds for a task* T_l *in* $\mathbf{T}$ *where* w_l *is* T_l*'s worst-case response time, then* $\frac{\lceil \frac{w_l}{p_i} \rceil e_i}{d_l} \leq \max\left(\frac{e_i}{d_i}, \frac{2e_i}{p_i+e_i}\right)$ *holds for every* T_i *with higher priority than or the same priority as* T_l *for which* $e_i \leq p_i$ *also holds.*

Proof: Because w_l denotes T_l's worst-case response time, the following equality holds:

$$w_l = e_l + \sum_{\forall T_i \in \mathbf{T}_l} \left\lceil \frac{w_l}{p_i} \right\rceil e_i, \tag{4}$$

which results by replacing $t^{(k+1)}$ and $t^{(k)}$ by w_l in Equation (3). Each term $\lceil \frac{w_l}{p_i} \rceil e_i$ stands for the execution demand within w_l of an individual task T_i with higher priority than or the same priority as T_l.

For $w_l < p_i$, $\frac{\lceil \frac{w_l}{p_i} \rceil e_i}{d_l} = \frac{e_i}{d_l}$ holds, $\frac{e_i}{d_l} \leq \frac{e_i}{d_i}$ also holds because $d_i \leq d_l$. In this case, the lemma holds true.

On the other hand, if $\frac{\lceil \frac{w_l}{p_i} \rceil e_i}{d_l} = \frac{k_i \cdot e_i}{d_l}$ holds for any $w_l \geq p_i$ where $k_i \geq 1$ is an integer number, then $w_l \geq k_i \cdot p_i + e_i$ must hold because Equation (4) holds. As $w_l \leq d_l$ is assumed to be true, we have:

$$\frac{\lceil \frac{w_l}{p_i} \rceil e_i}{d_l} = \frac{k_i \cdot e_i}{d_l} \leq \frac{k_i \cdot e_i}{k_i \cdot p_i + e_i} = \frac{e_i}{p_i + \frac{e_i}{k_i}}.$$

As $p_i \geq \frac{p_i+e_i}{2}$ for $p_i \geq e_i$, $\frac{\lceil \frac{w_l}{p_i} \rceil e_i}{d_l} \leq \frac{e_i}{p_i + \frac{e_i}{k_i}} \leq \frac{2e_i}{p_i+e_i}$ holds and the lemma follows. □

LEMMA 2 *Let the task set* $\mathbf{T}$ *be scheduled under the Deadline Monotonic policy. Given a task* T_l *in* $\mathbf{T}$, $\frac{\lceil \frac{d_l}{p_i} \rceil e_i}{d_l} \leq \max\left(\frac{e_i}{d_i}, \frac{2e_i}{p_i+e_i}\right)$ *holds for every* T_i *with higher priority than or the same priority as* T_l *for which* $e_i \leq p_i$ *also holds.*

Proof: T_i has higher priority than or the same priority as T_l. As a consequence, under DM, d_i is less than or equal to d_l. For $d_l < p_i$, $\frac{\lceil \frac{d_l}{p_i} \rceil e_i}{d_l} = \frac{e_i}{d_l}$ holds and $\frac{e_i}{d_l} \leq \frac{e_i}{d_i}$ also holds. In this case, the lemma is correct.

Further, if $\frac{\lceil \frac{d_l}{p_i} \rceil e_i}{d_l} = \frac{k_i \cdot e_i}{d_l}$ holds for an integer number $k_i \geq 1$, then $d_l \geq k_i \cdot p_i$ also holds and we have:

$$\frac{\lceil \frac{d_l}{p_i} \rceil e_i}{d_l} = \frac{k_i \cdot e_i}{d_l} \leq \frac{k_i \cdot e_i}{k_i \cdot p_i} = \frac{e_i}{p_i}.$$

As $p_i \geq \frac{p_i+e_i}{2}$ for $p_i \geq e_i$, $\frac{\lceil \frac{d_l}{p_i} \rceil e_i}{d_l} \leq \frac{e_i}{p_i} \leq \frac{2e_i}{p_i+e_i}$ holds and the lemma follows. □

The following theorem constitutes the main contribution of this paper and forms the basis of the proposed test called load test.

THEOREM 1 *Let* $\mathbf{T}$ *be a set of* n *fully preemptive, independent, synchronous, periodic, real-time tasks with deadlines less than or equal to periods.* $\mathbf{T}$ *can be feasibly scheduled on one processor under the Deadline Monotonic policy, if the following inequality holds:*

$$\sum_{i=1}^{n} \max\left(\frac{e_i}{d_i}, \frac{2e_i}{p_i+e_i}\right) \leq 1. \tag{5}$$

Proof: Let us initially suppose that $\mathbf{T}$ is schedulable under DM. Then, for every T_l in $\mathbf{T}$ where $1 \leq l \leq n$, T_l's worst-case response time w_l is less than or equal to d_l [3], i.e., $\frac{w_l}{d_l} \leq 1$ holds for every T_l. Replacing $t^{(k+1)}$ and $t^{(k)}$ by w_l in Equation (3) and dividing by d_l, we obtain:

$$\frac{w_l}{d_l} = \frac{e_l}{d_l} + \sum_{\forall T_i \in \mathbf{T}_l} \frac{\lceil \frac{w_l}{p_i} \rceil e_i}{d_l}.$$

As $w_l \leq d_l$ is assumed to hold, we can apply Lemma 1 to reach: $\frac{w_l}{d_l} \leq \frac{e_l}{d_l} + \sum_{\forall T_i \in \mathbf{T}_l} \max\left(\frac{e_i}{d_i}, \frac{2e_i}{p_i+e_i}\right)$.

Considering that $\frac{e_l}{d_l} \leq \max\left(\frac{e_l}{d_l}, \frac{2e_l}{p_l+e_l}\right)$ holds and summing the n tasks in $\mathbf{T}$ (and not only the tasks in the subset $\mathbf{T}_i$), we obtain: $\frac{e_l}{d_l} + \sum_{\forall T_i \in \mathbf{T}_l} \max\left(\frac{e_i}{d_i}, \frac{2e_i}{p_i+e_i}\right) \leq \sum_{i=1}^{n} \max\left(\frac{e_i}{d_i}, \frac{2e_i}{p_i+e_i}\right)$.

As a consequence, if $\sum_{i=1}^{n} \max\left(\frac{e_i}{d_i}, \frac{2e_i}{p_i+e_i}\right) \leq 1$ holds, $\frac{w_l}{d_l} \leq 1$ also holds for $1 \leq l \leq n$ and the task set $\mathbf{T}$ is schedulable under DM. However, as it can be seen from the previous analysis, this is a sufficient but not necessary condition.

Let us assume now that a job of task T_l misses its deadline. As a consequence, the first job of T_l in the critical instance misses its deadline as well. Hence, the execution demand in $(0, d_l]$ (due to all tasks with higher priority than or equal priority to T_l) is greater than d_l: $e_l + \sum_{\forall T_i \in \mathbf{T}_l} \lceil \frac{d_l}{p_i} \rceil e_i > d_l$. Dividing by d_l, we reach:

$$\frac{e_l}{d_l} + \sum_{\forall T_i \in \mathbf{T}_l} \frac{\lceil \frac{d_l}{p_i} \rceil e_i}{d_l} > 1.$$

Applying Lemma 2, this inequality leads to: $\frac{e_l}{d_l} + \sum_{\forall T_i \in \mathbf{T}_l} \max\left(\frac{e_i}{d_i}, \frac{2e_i}{p_i+e_i}\right) \geq \frac{e_l}{d_l} + \sum_{\forall T_i \in \mathbf{T}_l} \frac{\lceil \frac{d_l}{p_i} \rceil e_i}{d_l} > 1$.

Considering again that $\frac{e_l}{d_l} \leq \max\left(\frac{e_l}{d_l}, \frac{2e_l}{p_l+e_l}\right)$ holds and summing the n tasks in $\mathbf{T}$, we obtain: $\sum_{i=1}^{n} \max\left(\frac{e_i}{d_i}, \frac{2e_i}{p_i+e_i}\right) > 1$.

This means that if the task set $\mathbf{T}$ is not schedulable under DM, the first job of a task T_l misses its deadline in the critical

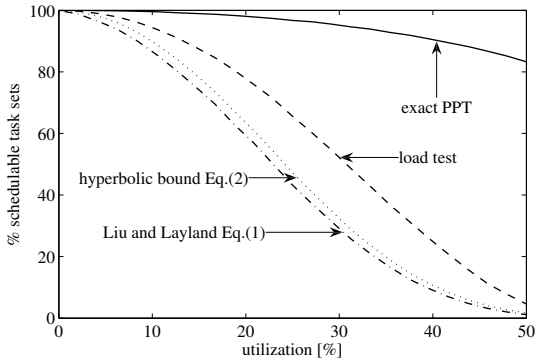

Figure 1. Schedulability versus utilization for 10 tasks

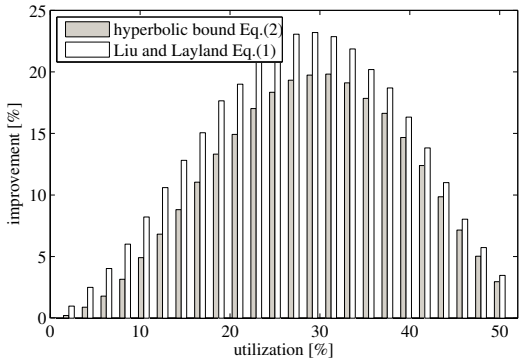

Figure 2. Improvement of the load test over the hyperbolic and the Liu and Layland bound for 10 tasks

instance for any $1 \leq l \leq n$. Hence, Equation (5) does not hold and the theorem follows. □

Notice that Theorem 1 does not depend on the order of tasks in $\mathbf{T}$. Consequently, the load test can be used to perform a constant-time admission control of DM tasks.

The following lemma compares the load test with the hyperbolic bound of Equation (2) which was adapted for DM.

LEMMA 3 *The load test of Theorem 1 is less pessimistic than the hyperbolic bound of Equation (2), if* $d_i \leq \frac{p_i+e_i}{2}$ *holds for* $1 \leq i \leq n$ *and* $n > 1$ *where* n *is the number of tasks in* $\mathbf{T}$.

Proof: If $d_i \leq \frac{p_i+e_i}{2}$ holds for all tasks in $\mathbf{T}$, $\max\left(\frac{e_i}{d_i}, \frac{2e_i}{p_i+e_i}\right) = \frac{e_i}{d_i}$ holds for all possible i where $1 \leq i \leq n$. Consequently, Equation (5) can be reshaped to:

$$\sum_{i=1}^{n} \frac{e_i}{d_i} \leq 1. \tag{6}$$

On the other hand, decomposing $\prod_{i=1}^{n}\left(1+\frac{e_i}{d_i}\right)$ into its terms—only the first terms are written to simplify, the hyperbolic bound of Equation (2) can be rewritten as follows:

$$1 + \sum_{i=1}^{n} \frac{e_i}{d_i} + \sum_{i=1}^{n-1}\left(\frac{e_i}{d_i}\sum_{j=i+1}^{n}\frac{e_j}{d_j}\right) + \cdots \leq 2. \tag{7}$$

For $n > 1$, i.e., for more than one task, the third term of the left-hand member in Equation (7) is not zero. In this case, the hyperbolic bound implies summing additional non-zero terms to $\sum_{i=1}^{n}\frac{e_i}{d_i}$—recall that not all terms are shown in Equation (7). As a consequence, it is more pessimistic than the load test and the lemma follows. □

If $d_i > \frac{p_i+e_i}{2}$ holds for some T_i in $\mathbf{T}$, Lemma 3 is not valid anymore. Whether the load test performs well in this case is going to be analyzed in the next section through an extensive comparison.

V. EXPERIMENTAL RESULTS

In this section, we present some experimental results comparing the proposed test against the hyperbolic bound of Equation (2) and the Liu and Layland bound of Equation (1), which were adapted to DM. As already mentioned, our experimental results have shown that the hyperbolic bound of Equation (2) is the least pessimistic test—the one that accepts the largest number of task sets—that can be obtained from the literature. For ease of exposition, we do not include the comparison results for all the other possible methods discussed in Section IV. On the other hand, we include the exact pseudo-polynomial-time schedulability test from [2], [3] denoted by *exact PPT* in this comparison.

A. Synthetic Task Sets

The mentioned algorithms are compared with respect to their accuracy versus utilization for 10 and 100 tasks respectively. The accuracy of methods is measured at the percentage of schedulable task sets that they are able to accept.

Although we are concerned with the admission control problem, i.e., a new task should be admitted on a running system, the experimental results with synthetic tasks are presented in the form of whether an entire task set is schedulable or not. This way, if a given task set of n tasks is schedulable, it means that adding an n-th task to the set of $n-1$ tasks was possible.

For the presented curves, we first generated a random set of task utilizations u_i with *UUniFast* introduced in [14], [15]. Then, we created periods p_i also in a random way with uniform distribution and obtained $e_i = u_i \cdot p_i$. The relative deadlines d_i were uniformly chosen from the range $[e_i, p_i]$. Additionally, we increased the utilization in uniform steps (a total of 24 steps) generating each time 100,000 different task sets.

Figure 1 shows the percentage of accepted task sets for the different algorithms and 10 tasks. In this case, the proposed load test can accept more task sets than the hyperbolic and the Liu and Layland bound between 5% and 50% utilization. For the utilization interval $(20\%, 40\%)$, the load test presents an improvement of around 15% more accepted task sets over these other tests. This can be better appreciated in Figure 2, where the length of bars illustrates the improvement of the load test over the other constant-time admission control tests.

For 100 tasks, the performance of all constant-time tests for admission control, the proposed and the known ones, decays. Where for 10 tasks, the constant-time tests detect

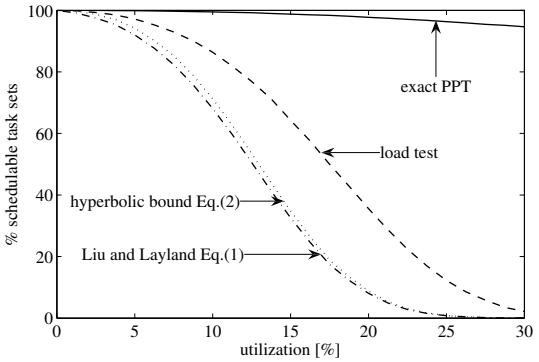

Figure 3. Schedulability versus utilization for 100 tasks

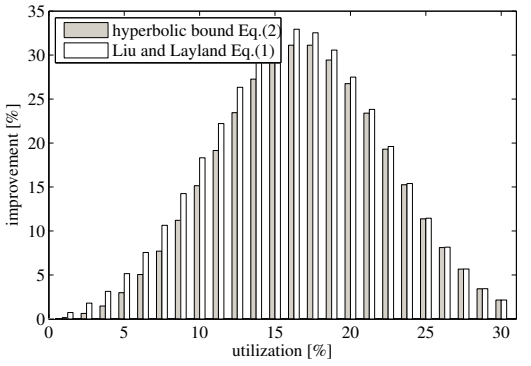

Figure 4. Improvement of the load test over the hyperbolic and the Liu and Layland bound for 100 tasks

schedulable task sets up to 60% utilization, they are unable to find schedulable task sets for a utilization over 40% and 100 tasks. However, the performance improvement of the load test over the hyperbolic bound and the test of Liu and Layland increases—see Figure 3. Figure 4 shows that the load test is able to detect up to 30% more schedulable task sets in the utilization range $(10\%, 20\%)$.

B. Case Study

Although a comparison based on synthetic tasks gives a notion of how algorithms behave, it is always meaningful to compare them on the basis of a real application. For this purpose, we present a case study consisting of a real-time multimedia server where requests from clients (tasks) are constantly arriving and have to be accommodated or rejected on-line.

A number of modern multimedia applications such as high-definition video processing and interactive applications such as games are associated with tight timing constraints. In particular, the QoS (quality of service) requirements of these

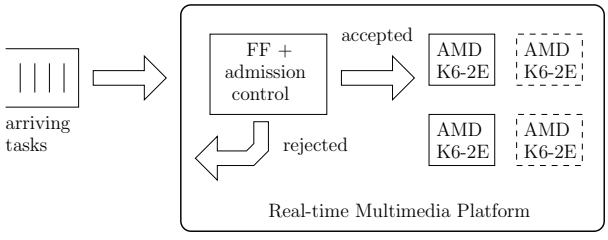

Figure 5. Setup for the real-time multimedia server

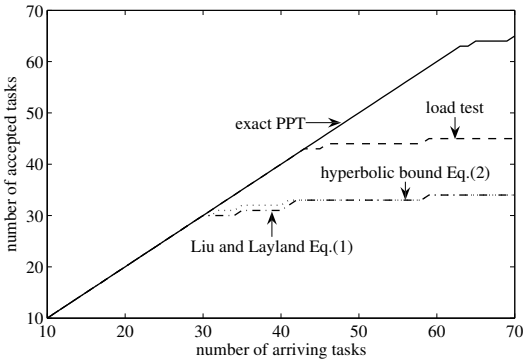

Figure 6. Number of accepted vs. number of arriving tasks considering two processors

applications often necessitate that deadlines be less than the corresponding periods (which is typical in the sensor data processing or control domain). As a consequence, many of the arriving tasks in our multimedia example are assumed to have deadlines less than periods.

The considered multimedia server consists of two to four identical processors of type AMD K6-2E operating at 400 Mhz. We considered a partitioned DM scheduling without task migration, i.e., once a task is assigned to a processor, it remains on that processor. Furthermore, the task assignment is also performed on-line using the well-known First Fit (FF) heuristic in combination with the discussed constant-time admission control tests. Figure 5 schematizes the described setup for the multimedia server in this example.

In order to obtain realistic task parameters, we make use of the Embedded Systems Synthesis Benchmarks Suite (E3S) [16], which is based on embedded processor and task information from the Embedded Microprocessor Benchmark Consortium (EEMBC). We first created a *task pool* with the tasks typically encountered in high-end multimedia applications such as autocorrelation, Fast Fourier Transform, compressing/decompressing JPEG, high-pass gray-scale filter, etc. Further, we consider that an arbitrary task from this pool can arrive at any time to the multimedia server (triggered by a client request). When a new task arrives, the combination of FF and the admission control tests leads to rejection or acceptance of the task. Apart from the previously discussed constant-time tests, we include again the exact pseudo-polynomial-time test in this analysis [2], [3]. Of course, an admission control based on this exact test has pseudo-polynomial complexity and would normally not be used for on-line computations, however, it acts as reference in this comparison.

Figure 6 shows how many tasks the different admission control tests are able to accept when two processors are used. The hyperbolic as well as the Liu and Layland bound can accept up to around 30 real-time multimedia tasks, after which they become pessimistic and start rejecting new tasks. On the other hand, the load test admits approximately 10 more additional tasks before it starts denying acceptance to further requests. The exact pseudo-polynomial-time test (shortly exact PPT) has a better performance than all constant-time algorithms and can

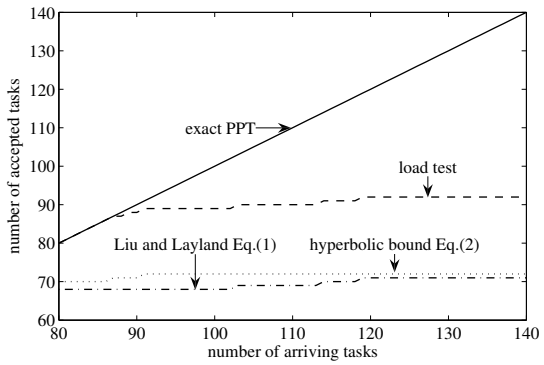

Figure 7. Number of accepted vs. number of arriving tasks considering four processors

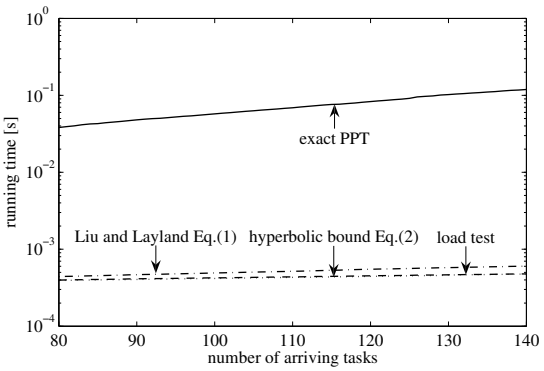

Figure 8. Running time vs. number of arriving tasks considering four processors

accommodate up to 63 tasks without rejections.

When considering four processors, the number of accepted tasks increases because of the additional computation capacity. Figure 7 illustrates the performance of the different algorithms in this case. Both, the hyperbolic and the Liu and Layland bound cannot accept much more than 70 tasks before they start refusing additional computation demand on the server. On the contrary, the load test is able to allocate around 90 of the arriving tasks onto the processors. As expected, the load test is not as efficient as the exact pseudo-polynomial-time test, but it allows accepting some extra tasks (20 tasks) compared to the other constant-time methods.

With respect to the running time of algorithms, Figure 8 shows a comparison for the case of four processors. Although the exact PPT is more efficient than the other tests, its running time depends not only on the number of accepted tasks already in the system but also on task parameters. On the other hand, constant-time algorithms requires always almost the same running time to test a new arriving task, which is desirable for on-line computations in real-time systems. (Clearly, the time for computing the right-hand member of Equation (1) depends on the value of n, i.e., the number of the accepted plus the new task, however, this requires only one and not n computations.) In general, when considering four processors, the exact PPT needs more than two orders of magnitude longer running time than the other tests to accept/reject a new task in the system—see Figure 8.

VI. Conclusions

In this paper, a more accurate constant-time admission control test, called load test, was proposed for periodic real-time tasks with deadlines less than or equal to periods, which are preemptively scheduled under Deadline Monotonic (DM) and on one processor.

We proved that the load test is always less pessimistic than the hyperbolic bound of Equation (2), if $d_i \leq \frac{p_i+e_i}{2}$ holds for all tasks in the task set $\mathbf{T}$. Additionally, some experimental results were presented using synthetic and real-world tasks. Our results show that the load test allows a clear accuracy improvement over other possible constant-time tests from the literature, like the mentioned hyperbolic bound and the Liu and Layland bound (both adapted to DM). This means that the load test is able to detect more schedulable task sets than these other tests under DM (i.e., considering the already running and the new arriving task as a set). In general, this improvement tends to increase as the number of tasks grows. For 10 tasks, the load test accepts around 15% more schedulable task sets, whereas it detects approximately 25% more task sets when considering 100 tasks per set.

References

[1] J.-T. Leung and J. Whitehead, "On the complexity of fixed-priority scheduling of periodic, real-time tasks," in *Performance Evaluation*, vol. 2, 1982, pp. 237–250.

[2] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: exact characterization and average case behavior," in *Proceedings of the Real-Time Systems Symposium*, December 1989, pp. 166–171.

[3] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. Wellings, "Applying new scheduling theory to static priority pre-emptive scheduling," *Software Engineering Journal*, vol. 8, no. 5, pp. 284–292, September 1993.

[4] E. Bini and G. Buttazzo, "Schedulability analysis of periodic fixed priority systems," *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1462–1473, November 2004.

[5] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in hard real-time environments," *Journal of the Association for Computing Machinery*, vol. 20, no. 1, pp. 40–61, 1973.

[6] J. Liu, *Real-Time Systems*. New Jersey, USA: Prentice Hall, 2000.

[7] E. Bini, G. Buttazzo, and G. Buttazzo, "A hyperbolic bound for the rate monotonic algorithm," in *Proceedings of the 13th Euromicro Conference on Real-Time Systems*, June 2001.

[8] E. Bini and G. Buttazzo, "Rate monotonic analysis: the hyperbolic bound," *IEEE Transactions on Computers*, vol. 52, no. 7, pp. 933–942, July 2003.

[9] Y. Oh and S. Son, "Allocating fixed-priority periodic tasks on multiprocessor systems," *Real-Time Systems*, vol. 9, no. 3, pp. 207–239, 1995.

[10] T.-W. Kuo and A. Mok, "Load adjustment in adaptive real-time systems," in *Proceedings of the 12th IEEE Real-Time Systems Symposium*, December 1991, pp. 160–170.

[11] A. Burchard, J. Liebeherr, Y. Oh, and S. Son, "New strategies for assigning real-time tasks to multiprocessor systems," *IEEE Transactions on Computers*, vol. 44, no. 12, pp. 1429–1442, 1996.

[12] J. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," in *Proceedings of the 11th IEEE Real-Time Systems Symposium*, December 1990, pp. 201–209.

[13] N. Fisher and S. Baruah, "A fully polynomial-time approximation scheme for feasibility analysis in static-priority systems with arbitrary relative deadlines," in *Proceedings of the 17th Euromicro Conference on Real-Time Systems*, July 2005, pp. 117–126.

[14] E. Bini and G. Buttazzo, "Biasing effects in schedulability measures," in *Proceedings of the 16th Euromicro Conference on Real-Time Systems*, June-July 2004.

[15] E. Bini and G. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1-2, pp. 129–154, 2005.

[16] http://ziyang.eecs.umich.edu/~dickrp/e3s/.