Response Time Bounds for Sporadic Arbitrary-Deadline Tasks under Global Fixed-Priority Scheduling on Multiprocessors

Wen-Hung Huang and Jian-Jia Chen
Department of Computer Science
TU Dortmund University, Germany

ABSTRACT

In this paper, we study the problem of scheduling arbitrary-deadline real-time sporadic task sets on a multiprocessor system under global fixed-priority scheduling. Two contributions are made in this paper. First, it has been shown that the existing response time analysis in arbitrary-deadline systems is flawed: the response time may be larger than the derived bound. This paper provides a revised analysis resolving the problems with the original approach, and then propose a corresponding schedulability test. Secondly, we derive a linear-time upper bound on the response time of arbitrary-deadline tasks in multiprocessor systems. To the best of our knowledge, this is the first work presenting a linear-time response time upper bound for arbitrary-deadline sporadic tasks in multiprocessor systems. Empirically, this linear-time response time bound is shown to be highly effective in terms of the number of task sets that are deemed schedulable.

1. INTRODUCTION

Analyzing the worst-case response time is one of the most important issues for designing hard real-time systems to ensure the timeliness of tasks, especially for safety-critical embedded systems. There have been extensive results in such a general direction since the seminal paper by Liu and Layland [24]. A well-adopted real-time task model is the sporadic task model, in which a sporadic task τ_i defines an infinite sequence of task instances, also called *jobs*: (1) Any two consecutive jobs of task τ_i should be temporally separated by at least T_i. (2) The execution time of a job of task τ_i is at most C_i (also known as the worst-case execution time). The response time of a job of task τ_i is its completion time minus its arrival time. Therefore, the worst-case response time of task τ_i is (an upper bound of) the maximum response time among all the jobs of task τ_i.

Typically, for real-time systems, a sporadic task τ_i also has an associated relative deadline D_i. To ensure that the relative deadline of task τ_i is met, we essentially have to verify whether the worst-case response time of task τ_i is at most D_i. Verifying whether a set of sporadic tasks can meet all the relative deadlines by a scheduling algorithm is called a *schedulability test*. The difficulty of the schedulability tests also depends on the relationship of the relative deadlines and the periods of the tasks, which are typically classified with the following cases: (1) *implicit deadlines* if the relative deadlines of the sporadic tasks are equal to their minimum inter-arrival times, i.e., $T_i = D_i, \forall \tau_i$, (2) *constrained deadlines* if the minimum inter-arrival times are no less than their relative deadlines, i.e., $D_i \leq T_i, \forall \tau_i$, and (3) *arbitrary deadlines*, otherwise. It is evident that the case with arbitrary deadlines is the most general one.

For uniprocessor fixed-priority scheduling in arbitrary-deadline task systems, the exact schedulability tests and the (tight) worst-case response time using *time-demand analysis* (TDA) are proposed by Lehoczky [23, 30]. Such results may suffer from their high time-complexity because of the explosion of *busy interval* in arbitrary-deadline systems. From the designers' perspective, a fast test is more applicable during design space exploration, even at a price of the accuracy. Several approaches have been proposed for reducing the computational complexity of the TDA [13, 27]. Lehoczky proposes a utilization upper bound for a set of periodic arbitrary-deadline tasks under fixed-priority scheduling [12]. Bini proposes a quadratic bound by considering some information about the task periods [12]. The linear-time response-time bound for fixed-priority systems has been first proposed by Davis and Burns [17]. This bound has been later improved by Bini et al. [15].

To schedule real-time tasks on multiprocessor platforms, there are three widely adopted paradigms: partitioned, global, and semi-partitioned scheduling [19]. In this paper, we consider *global scheduling*, in which a job can migrate from one processor to another processor during its executions. But, a job cannot be simultaneously executed on more than one processor.

Unfortunately, deriving exact schedulability tests under multiprocessor global scheduling is extremely harder than in uniprocessor due to the non-existence of *critical instant*. The first brute force approach regarding the exact response time is proposed by Baker and Cirinei [5]. The analysis consists in solving *reachability* problem in a finite state machine that represents all possible combinations of arrival times and execution sequences for a task set. In addition, some results are recently proposed to reduce the number of states to be analyzed in the reachability analysis [21, 28]. Gen-

RTNS '15 November 04-06, 2015, Lille, France

ACM ISBN 978-1-4503-3591-1/15/11.

DOI: http://dx.doi.org/10.1145/2834848.2834849

erally speaking, these approaches are so complex that only task sets with very small integer periods can be tractably solved. Alternatively, most of the research in the literature focus on finding approximate upper bounds to the response time. The response time analysis for implicit-deadline sporadic task systems has been studied by Andersson et al. [1]. The fixed-priority scheduling of constrained-deadline sporadic task systems has been studied in the literature [3,6,11].

Regarding arbitrary-deadline task systems, several results have been proposed in the literature [4, 7, 22, 29]. Baker [4] designs a test based on some deep insights, in that if a task τ_k misses its deadline, then the load in the analyzed interval, called *problem window*, must satisfy some necessary conditions on the parameters of all the tasks. Baruah and Fisher [7] use another annotation to extend the analysis window and derive a corresponding pseudo-polynomial-time schedulability test. The first worst-case response-time analysis for arbitrary-deadline task systems is proposed by Guan et al. [22], in which the authors use the insight proposed by Baruah [8] to limit the number of carry-in tasks, and then apply the workload function proposed by Bertogna et al. [11] to quantify the requested demand of higher-priority tasks. Unfortunately, it has recently been shown in [29] that the analysis by Guan et al. [22] is optimistic for arbitrary-deadline tasks. The assumption that each carry-in task has only one carry-in job in [22] is in fact problematic, as a carry-in task can be shown to carry more than one carry-in job [29]. Sun et al. [29] derive a complex carry-in workload function for the response time analysis where all possible combinations of carry-in and non-carry-in functions have to be explicitly enumerated. However, such a method is computationally intractable since the time complexity is exponential.

Contributions. We focus in this paper on providing worst-case response time analyses by using the concept that has been adopted by Guan et al. [22] with proper annotations. Even though the assumption that each carry-in task has only one carry-in job is problematic in [22], we can safely bound the interference from the carry-in task by means of proper definitions of the analysis window and annotations, to be shown in Section 4. We summarize the significance of this work as follows:

- The proper annotation of the number of carry-in jobs of a carry-in task results in a revisited response time analysis, in Section 4. Our analysis is distinct from other analyses [4, 7] in that our analysis quantifies the carry-in function more precisely.

- In Section 6, we derive the first known linear-time response time bound on arbitrary-deadline multiprocessor systems. *To the best of our knowledge, this is the first work presenting a linear-time response time bound on arbitrary-deadline multiprocessors.*

- We evaluate our results in Section 7 by comparing to the state-of-the-art results for both constrained-deadline systems [3, 6, 11, 22] and arbitrary-deadline systems [4, 7]. The experimental results show that our time-demand analysis is comparable to the state of the art [22] and superior to the others [3, 6, 11] for constrained-deadline systems. Regarding arbitrary-deadline systems, our TDA is much better than the existing results [4, 7]. Moreover, compared to the existing analyses [3, 4, 6, 7, 11], the proposed linear-time upper bound on the worst-case response time is shown to be highly effective, especially for large ranges of periods, for which the time-demand analysis suffers from its high computational complexity.

2. SYSTEM MODEL AND PROBLEM SETTING

This paper considers a set of n independent arbitrary-deadline sporadic real-time tasks $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$ upon a system comprised of m identical processors, where $m \geq 1$ is an integer. Each task can release an infinite number of jobs (also called task instances) under the given minimum inter-arrival time (temporal) constraint. We consider the sporadic task model [25], where C_i denotes the *worst-case execution time* (WCET) of task τ_i, the minimum inter-arrival time or period T_i represents the minimum temporal distance between consecutive arrivals of task instances, and the relative deadline D_i represents the response time constraint for executing the job. If a task instance arrives at time θ_a, the execution of this instance must be finished no later than its *absolute deadline* $\theta_a + D_i$ and the next instance of the task must arrive no earlier than θ_a plus the minimum inter-arrival time, i.e., $\theta_a + T_i$. We assume that all task parameters are positive integers. Since all the m processors are identical, the execution time of a job of a task does not depend on the processors that execute the job.

Task system τ is an *implicit-deadline* system if each task $\tau_i \in \tau$ has its relative deadline D_i equal to its period T_i, and a *constrained-deadline* system if each task $\tau_i \in \tau$ has relative deadline D_i no larger than its period T_i. Otherwise, task system τ is an *arbitrary-deadline* system. In this paper we consider the most general *arbitrary-deadline* system.

We assume that the system is fully *preemptive*, and allows for *global* inter-processor migration. We assume that the cost of preemption and migration has been subsumed into the worst-case execution time of each task. In this paper each task is assumed to be preemptively scheduled on m identical multiprocessor systems according to a global fixed-priority scheduling, in which each task is associated with a unique priority level. We assume that the jobs of the same task are served by the first-come first-serve (FCFS) policy; hence, a job becomes eligible to be executed only if all previous jobs from the same task have been completed. At any time point, the jobs of the (at most) m-highest priority tasks in the ready queue are executed on the processors. Moreover, *intra-task parallelism* is forbidden; hence, each task may have at most one job executing on at most one processor at any time. Due to the prohibition of intra-task parallelism, we may have more than m jobs available to be executed, but less than m jobs are executed.

Throughout this paper, we refer to the *utilization* factor of task τ_i as $U_i = C_i/T_i$. We further assume the total utilization $U_\Sigma = \sum_{i=1}^{n} U_i \leq m$ and $U_i \leq 1$. We let $hp(i)$ denote the set of tasks with priority higher than that of task τ_i. For the simplicity of presentations, we define the following terms:

- A *legal sequence* of jobs of the task system τ: The arrival times of any two consecutive jobs of task $\tau_i \in \tau$ are separated by *at least* T_i.
- A *legal schedule* of a legal sequence of jobs: All the

temporal and scheduling constraints (*except the deadlines*) are met.

- Worst-case response time RT_k of task τ_k: An upper bound of the response times of all the jobs of task $\tau_k \in \tau$ under the legal execution for any legal sequence of jobs of τ.

Problem Definition: The objective of this paper is to find the worst-case response time of every task $\tau_k \in \tau$ under the given priority assignment. Throughout the paper, we will implicitly only analyze a specific task τ_k under the assumption that the worst-case response time $RT_i \leq D_i$ of every higher-priority task τ_i in $hp(k)$ has been already analyzed. We will implicitly focus on the cases when $|hp(k)| \geq m$, since the cases with $|hp(k)| < m$ can be trivially solved by $RT_k \leq C_k$ (under the assumption $C_k \leq T_k$).

3. PRELIMINARY RESULTS AND DEFINITIONS

This section reviews some results in the literature with respect to the worst-case response-time analysis. We will first explain the level-k busy interval (period) concept by Lehoczky [23] in uniprocessor systems and extend the concept for multiprocessor systems. As our analysis also exploits the workload function, as defined in [10, 15], we also provide its definition at the end of this section.

3.1 Level-k Busy Interval for Task τ_k

For the special case with uniprocessor systems when $m = 1$, the response-time analysis for sporadic arbitrary-deadline task systems under fixed-priority scheduling has been developed in [23]. The analysis utilizes a *level-k busy interval* concept to evaluate the worst-case response time. For such a case, we release all the higher-priority tasks in $hp(k)$ together with task τ_k at time 0 and all the subsequent jobs are released as early as possible by respecting to the minimum inter-arrival time. The level-k busy interval finishes when a job of task τ_k finishes before the next release of a job of task τ_k.

For the h-th job of task τ_k in the level-k busy interval, the finishing time $R_{k,h}$ is the minimum t such that

$$hC_k + \sum_{\tau_i \in hp(k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t,$$

and, hence, its response time is $R_{k,h} - (h-1)T_k$. The level-k busy interval of task τ_k finishes at the h-th job if $R_{k,h} \leq hT_k$.

The above analysis works on one processor. However, it requires quite some annotations for the general cases when $m \geq 2$. There are some difficulties for multiprocessor systems. In general, we are unaware of the worst-case release pattern of the higher-priority tasks in $hp(k)$ when $m \geq 2$. The level-k busy interval of task τ_k may have different arrival patterns to create the worst-case response time for different hs. As a result, all possible cases with some unfinished jobs before the arrival of a job have to be considered!

Towards this, we need more precise definitions to capture all the busy intervals for task τ_k and use them to safely bound the worst-case response time. The first definition defines whether a task is busy at time θ in a legal schedule.

Definition 1 (Task Busy). *A task is said busy at time instant θ in a legal schedule if at least one job of the task that arrives earlier than θ has not yet completed its execution at time instant θ.*

Note that, by the definition, even if a task is busy at time θ, it may not be executed at time θ. From Definition 2 to Definition 3, we will look at a specific time interval starting from time θ_a in a legal schedule of a legal sequence of jobs of τ.

Definition 2 (A Level-k Busy Interval). *A level-k busy interval of task τ_k in a legal schedule is an interval $[\theta_a, \theta_a + t)$ of time in which, in the legal schedule, (1) task τ_k is busy at all the time points in the interval, and (2) task τ_k is not busy right prior to θ_a.*

Definition 3 (An h-Incomplete Busy Interval). *A level-k busy interval $[\theta_a, \theta_a + t)$ in a legal schedule is an h-incomplete busy interval if the hth-job of task τ_k released in the busy interval $[\theta_a, \theta_a + t)$ has not completed yet at time $\theta_a + t$.*

The above definitions are general forms of the level-k busy interval in [23], i.e., θ_a is 0 for the case when m is 1. If θ_a is given, the maximum h-incomplete busy interval $[\theta_a, \theta_a + t^*)$ (under the assumption to start from θ_a) in a legal schedule happens when task τ_k is not busy at time $\theta_a + t^*$ or task τ_k is $(h+1)$-incomplete at time $\theta_a + t^*$. Moreover, according to the above definition, for a given θ_a as a fixed left point in the busy intervals, an $(h+1)$-incomplete busy interval (if it exists) completely covers an h-incomplete busy interval.

Definition 4 (Maximum h-Incomplete Length). *The maximum h-incomplete length $R_{k,h}$ is the maximum interval length among all the h-incomplete busy intervals in the legal schedules of all the legal sequences of the jobs of task system τ.*

According to the above definitions, we can reach the following lemma regarding the worst-case response time $RT_{k,h}$ of the hth-job released in a level-k busy interval.

Lemma 1. *Suppose that $R_{k,h}$ is known. The worst-case response time $RT_{k,h}$ of the hth-job released in an h-incomplete busy interval is at most $R_{k,h} - (h-1)T_k$, i.e.,*

$$RT_{k,h} \leq R_{k,h} - (h-1)T_k$$

Proof. According to Definition 4, we know that the length of any h-incomplete busy interval of task τ_k is at most $R_{k,h}$. For an h-incomplete busy interval, starting at time θ_a, this implies that the finishing time of the h-th job released no earlier than θ_a is at most $\theta_a + R_{k,h}$. In a legal sequence of jobs of task system τ, we also know that the arrival time of the h-th job released no earlier than θ_a is at least $\theta_a + (h-1)T_k$. Therefore, we reach the conclusion. □

Based on Lemma 1, we can now obtain the worst-case response time of task τ_k by evaluating all possible $RT_{k,h}$:

$$RT_k \leq max_{h=1,2,\ldots} \{RT_{k,h}\}$$

3.2 Workload Functions

The concept of *workload function* has been widely used in real-time schedulability analysis [10, 15]. Briefly, for a given interval length t, the *workload function* of task τ_i is defined to be the largest accumulative execution time of the legal sequence of the jobs of task τ_i over an interval of length t that could be *legally* executed within the interval. We provide the formal definition as follows:

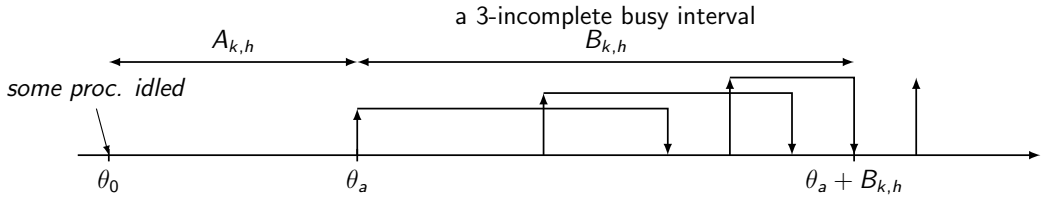


Figure 1: An example of an h-incomplete busy interval and the downward extension to θ_0.

DEFINITION 5 (WORKLOAD FUNCTION [10, 15]). *For any interval length t, the workload function $W_i(t)$ of a sporadic task τ_i bounds the maximum cumulative execution requirement by jobs of τ_i that are released and may execute within any interval of length t.*

$$W_i(t) = \left\lfloor \frac{t}{T_i} \right\rfloor C_i + min(t \bmod T_i, C_i) \quad (1)$$

It has been shown in [15] that the workload function is upper-bounded by the following linear function:

$$W_i(t) \leq U_i t + C_i(1 - U_i) \quad (2)$$

4. RESPONSE TIME ANALYSIS

Based on the observations in Section 3, we will first analyze a safe upper bound of $R_{k,h}$ for a given h. Then, the worst-case response time of task τ_k can be obtained by evaluating all possible values of h. Although the analysis by Guan et al. [22] is optimistic for arbitrary-deadline task systems, some observations can still be applied. Throughout the section, we will implicitly only analyze a specific task τ_k under the assumption that the worst-case response time $RT_i \leq D_i$ of every higher-priority task τ_i in $hp(k)$ has been already analyzed.

4.1 Safe Bound of $R_{k,h}$ for A Given h

We will first shortly review the analysis by Guan et al. [22] and then explain how we plan to conquer the problem to safely derive $R_{k,h}$. Suppose that an h-incomplete busy interval of task τ_k begins at time-instant θ_a. Suppose that this h-incomplete busy interval finishes at time $\theta_a + B_{k,h}$. That is, at time $\theta_a + B_{k,h}$, the hth-job in this h-incomplete busy interval finishes. Throughout this section, we only focus on this interval $[\theta_a, \theta_a + B_{k,h})$ under a legal schedule of a legal sequence of jobs of task system τ.

We first remove some unnecessary jobs in the legal sequence of jobs of task system τ. From this legal sequence of jobs, we first discard all those jobs that have priority lower than task τ_k. Since those jobs with priority lower than τ_k have no effect on the scheduling of the jobs generated by $hp(k)$ and task τ_k, this removal does not change the scheduling for tasks τ_k and $hp(k)$ in the interval $[\theta_a, \theta_a + B_{k,h})$. Moreover, we also remove the jobs of task τ_k that are released strictly before θ_a. Similarly, due to the definition of the h-incomplete busy interval, these jobs also have no influence on the scheduling of the jobs generated by $hp(k)$ and task τ_k in the interval $[\theta_a, \theta_a + B_{k,h})$. For the rest of our analysis in this subsection, we will consider only the above *reduced* legal sequence of jobs.

4.1.1 Extending the h-Incomplete Busy Interval

The technique for extending the h-incomplete busy interval is needed for precisely bounding the amount of work resulting from higher-priority tasks τ_i in $hp(k)$. Here, we will use the techniques proposed by Baruah [8] and also adopted by Guan et al. [22]. Let θ_0 denote the latest time-instant $\leq \theta_a$ at which at least one processor is idle in the legal schedule of the reduced legal sequence of jobs. For notational brevity, let $A_{k,h} = \theta_a - \theta_0$.

Example: Figure 1 illustrates a 3-incomplete busy interval $[\theta_a, \theta_a + B_{k,h})$ and a downward extension of the interval to time θ_0. Task τ_k is not *busy* prior to the arrival of the first of these 3 jobs, the first job completes its execution after the second job arrives, and the second job completes its execution after the third job arrives. Thus, the task is continuously busy after the arrival of the first job shown, and θ_a is hence set equal to the arrival time of this job. □

We then have the following observations:

- θ_a is the arrival time of some job of task τ_k.
- Over $[\theta_0, \theta_a)$, all the processor must be busily executing jobs having priority higher than τ_k.

Whenever we shift the release time of the job arriving at θ_a to θ_0, this job cannot be executed over $[\theta_0, \theta_a)$, and this task that is being h-incomplete at $\theta_a + B_{k,h}$ previously is still being h-incomplete at $\theta_a + B_{k,h}$ afterwards. In other words, we can consider the more pessimistic case where the release time of the job arriving at θ_a is shifted such that $\theta'_a = \theta_0$, without decreasing the response time of task τ_k. Meanwhile, the amount of work executing prior to θ_0 can still be more precisely quantified than that without extending the busy interval. Thus, we assume $\theta_a = \theta_0$ in the rest of this paper.

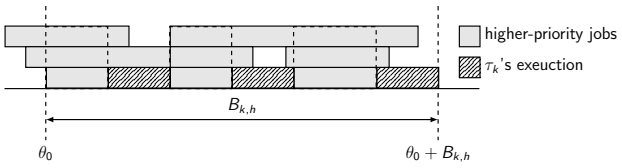


Figure 2: All processors are executing other higher-priority tasks whenever τ_k is not executing. Let $\theta_a = \theta_0$.

Another insight on multiprocessors is the lower bound on the workload of the higher-priority tasks in $hp(k)$ in the h-incomplete busy interval, as used in most of analysis in the literature [4,8,22,26]: Due to the h-incomplete busy interval of task τ_k in the interval $[\theta_0, \theta_0 + B_{k,h})$ and the definition of θ_0, for any $0 < t < B_{k,h}$, the sum of the length in interval $[\theta_0, \theta_0 + t)$ in which τ_k does not execute must be strictly larger than $t - hC_k$. The situation is illustrated for $m = 3$

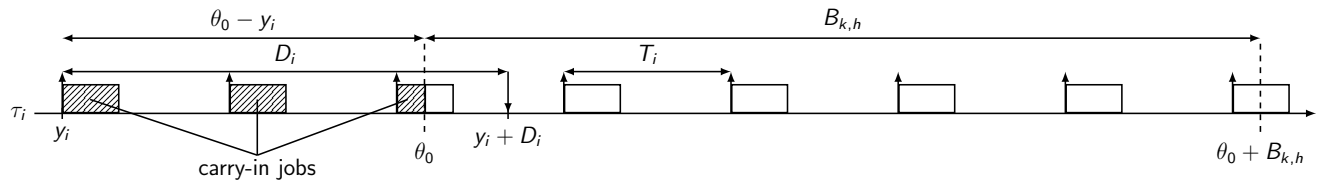


Figure 3: An illustration of releases of a higher-priority task τ_i over $[y_i, \theta_0 + B_{k,h})$. The release time of the first job of τ_i released before θ_0 and having an absolute deadline (or worst-case completion time) after θ_0 is denoted as y_i.

processors in Figure 2. The striped rectangles indicate the execution of task τ_k during the h-incomplete busy period. The dashed rectangle indicates the intervals in which all m processors execute higher-priority jobs.

Let us denote by Γ_k a collection of intervals, not necessarily contiguous, of cumulative length $B_{k,h} - hC_k$ over $[\theta_0, \theta_0 + B_{k,h})$, during which all m processors are executing jobs other than τ_k's jobs in the legal schedule.

DEFINITION 6 (HIGHER-PRIORITY INTERFERENCE). *We denote $\Omega(t)$ as the maximum total amount of execution times of the higher-priority interference from the tasks in $hp(k)$ in the time interval $[\theta_0, \theta_0 + t)$.*

From the above observations, an h-incomplete busy interval $[\theta_0, \theta_0 + B_{k,h})$ of task τ_k requires that

$$\forall 0 < t < B_{k,h}, \Omega(t) > m \times (t - hC_k) \quad (3)$$

Therefore, we can reach the following lemma.

LEMMA 2. *For a given h, the maximum h-incomplete length $R_{k,h}$ is upper bounded by the minimum value $B_{k,h}$ satisfying the following equation*

$$\Omega(B_{k,h}) \leq m \times (B_{k,h} - hC_k) \quad (4)$$

Note that Lemma 2 can be thought of as a generalized result of Theorem 2 in [8], where $h = 1$ and $B_{k,h} = D_k$.

4.1.2 Calculating Higher-Priority Interference

The treatment up to here is identical to the analysis by Guan et al. in [22]. As long as we can safely bound $\Omega(B_{k,h})$ for an h-incomplete busy interval of task τ_k defined above, we can use Lemma 2 to get $R_{k,h}$. However, calculating $\Omega(B_{k,h})$ is not trivial. The derivation in [22] was not safe enough.

To derive the contribution of task τ_i in $hp(k)$ to the interference $\Omega(B_{k,h})$, the contribution from higher-priority task τ_i can be considered by two parts in the interval $[\theta_0, \theta_0 + B_{k,h})$ (see Figure 3): (i) carry-in jobs: the jobs (portion) from task τ_i that arrive prior to θ_0 and have not yet completed their execution by θ_0, and (ii) body jobs: the jobs of τ_i that have the release time within the interval $[\theta_0, \theta_0 + B_{k,h})$.

By the definition of θ_0, at most $(m-1)$ tasks are with carry-in jobs at time instant θ_0. Consequently, Lemma 3 follows immediately:

LEMMA 3 (GUAN ET AL. [22]). *There are at most $(m-1)$ tasks having carry-in jobs in the interval $[\theta_0, \theta_0 + B_{k,h}]$.*

A higher-priority task τ_i in $hp(k)$ can be considered as either with or without carry-in jobs. We now introduce some notations by considering two cases for the amount of execution time executed for a higher-priority task τ_i in the time interval $[\theta_0, \theta_0 + B_{k,h}]$ in the legal schedule:

1. the maximum work done by task τ_i without carry-in jobs over $[\theta_0, \theta_0 + B_{k,h})$ is denoted by $I_i^1(B_{k,h})$, and
2. the maximum work done by task τ_i with carry-in jobs[1] over $[\theta_0, \theta_0 + B_{k,h})$ is denoted by $I_i^2(B_{k,h})$.

Then we explain how to compute a safe interference upper bound on these two functions $I_i^1(B_{k,h})$ and $I_i^2(B_{k,h})$, provided that h and $B_{k,h}$ are given.

Computing $I_i^1(B_{k,h})$. First, if a task τ_i contributes no carry-in jobs, then its contribution to the work over $[\theta_0, \theta_0 + B_{k,h})$ must arrive no earlier than θ_0 and no later than $\theta_0 + B_{k,h}$, and the total amount of work is bounded from above by the workload function with an interval of length $B_{k,h}$. Consequently, I_i^1 can be sufficiently computed by $W_i(B_{k,h})$, as defined in Section 3.2.

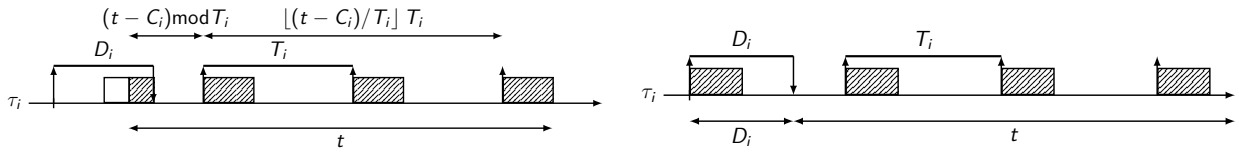
Secondly, the total amount of executed work cannot exceed the total length of the interval in Γ_k. The reason behind this is that any work that exceeds the total length of the interval in Γ_k cannot be *parallelly* executed with the other work from τ_i that have contributed to Γ_k, in which the total interference by all tasks busily executes on m processors.

Since both are safe upper bounds, we can safely choose the minimum one between them as the maximum interference. Notice that the total length of Γ_k is equal to $(B_{k,h} - hC_k)$ and "+1" models the minimum interference that prevents τ_k from executing in the integer time-domain [9, 22]. Consequently,

$$I_i^1(B_{k,h}) = min\left(W_i(B_{k,h}), max(0, B_{k,h} - hC_k + 1)\right) \quad (5)$$

Computing $I_i^2(B_{k,h})$. Now, consider the case that a higher-priority task τ_i contributes carry-in jobs over $[\theta_0, \theta_0 + B_{k,h})$. Let y_i denote the arrival time of the first job of τ_i released before θ_0 and having an absolute deadline (or worst-case completion time) after θ_0. Under the assumption that a higher-priority task τ_i has worst-case response time $RT_i \leq D_i$, the work of task τ_i executed in the interval $[\theta_0, \theta_0 + B_{k,h})$ must arrive no earlier than y_i and no later than $\theta_0 + B_{k,h}$. Consequently, the total amount of work $I_i^2(B_{k,h})$ is bounded from above by the workload function

[1]The assumption by [22] that at most one carry-in job of a task contributes to the interval $[\theta_0, \theta_0 + B_{k,h})$ is optimistic, as shown in [29].



(a) The workload function in [11, 22] **(b)** Our workload function

Figure 4: Comparison between the workload function in [11,22] and ours for a carry-in task, assumed $R_i = D_i$

with an interval length of $\theta_0 - y_i + B_{k,h}$ (see Figure 3). Since $\theta_0 - y_i \leq RT_i \leq D_i$, consequently, $I_i^2(B_{k,h})$ can be sufficiently computed by $W_i(D_i + B_{k,h})$. Together with the bounded length of the interval in Γ_k, we therefore have that

$$I_i^2(B_{k,h}) = min\left(W_i(D_i + B_{k,h}), max(0, B_{k,h} - hC_k + 1)\right) \tag{6}$$

Moreover, it is not difficult to see that given an upper bound on the worst-case response time RT_i, the workload function for carry-in task τ_i can be more precisely bounded from above by $W_i(RT_i + B_{k,h})$. Hence, a tighter $I_i^2(B_{k,h})$ can be computed as follows:

$$\hat{I}_i^2(B_{k,h}) = min\left(W_i(RT_i + B_{k,h}), max(0, B_{k,h} - hC_k + 1)\right) \tag{7}$$

For simplifying the analysis, we only use $I_i^2(B_{k,h})$ as the maximum work of a higher-priority task τ_i with carry-in jobs in the following presentation.

From the discussion above, we have seen how to sufficiently quantify the interference from these two types of higher-priority tasks. By Lemma 3, there are at most $(m-1)$ tasks with carry-in jobs at time instant θ_0. Hence, there are at most $(m-1)$ tasks contributing to $I_i^2(B_{k,h})$, and the remaining $(k-m)$ tasks in $hp(k)$ must contribute to $I_i^1(B_{k,h})$.

Computing $\Omega(B_{k,h})$. With the above settings of $I_i^1(B_{k,h})$ and $I_i^2(B_{k,h})$, we can now compute $\Omega(B_{k,h})$. Let us denote by $I_i^{DIFF}(B_{k,h})$ the difference between $I_i^2(B_{k,h})$ and $I_i^1(B_{k,h})$:

$$I_i^{DIFF}(B_{k,h}) = I_i^2(B_{k,h}) - I_i^1(B_{k,h})$$

Therefore, by Lemma 3, we know that

$$\Omega(B_{k,h}) = \sum_{\tau_i \in hp(k)} I_i^1(B_{k,h}) + \sum_{\substack{\text{the } (m-1) \text{ largest} \\ \tau_i \in hp(k)}} I_i^{DIFF}(B_{k,h}) \tag{8}$$

where $hp(k)$ denotes the *set* of tasks that are assigned higher priority than τ_k. Note that Eq. (8) can be computed in linear time by using linear-time selection [16] when $B_{k,h}$ is given.

4.2 Worst-Case Response Time of Task τ_k

Once having the upper bound on the interferences from higher-priority tasks $\Omega(B_{k,h})$, we can compute the maximum h-incomplete length $R_{k,h}$ by using Time Demand Analysis.

Compute $R_{k,h}$ using TDA. By Lemma 2 and the derived $\Omega(B_{k,h})$, $R_{k,h}$ is upper bounded by the smallest t satisfying the following inequality:

$$\Omega(t) \leq m \times (t - hC_k) \tag{9}$$

By the above analysis, we can now derive the worst-case response time of task τ_k by the following theorem.

THEOREM 1. *The worst-case response time RT_k of task τ_k is at most:*

$$RT_k \leq RT_k^* = max_{h \in \{1,\ldots,H\}}\{R_{k,h}^* - (h-1)T_k\}$$

where $H = min\{h \geq 1 | \frac{\Omega(hT_k)}{m} + hC_k \leq hT_k\}$, and $R_{k,h}^*$ is defined as the minimum t satisfying Eq. (9).

PROOF. Roughly speaking, H represents the maximum h to have h-incomplete busy intervals for task τ_k. Therefore, by Lemmas 1 and 2, we reach the conclusion. □

Even though H may become infinite, the TDA can be terminated whenever a deadline miss occurs:

$$\frac{\Omega((h-1)T_k + D_k)}{m} + hC_k > (h-1)T_k + D_k \tag{10}$$

COROLLARY 1 (TDA). *An arbitrary-deadline sporadic task system τ is schedulable on m identical processors under fixed-priority scheduling if for all tasks $\tau_k \in \tau$,*

$$RT_k^* \leq D_k \tag{11}$$

where RT_k^* is as defined in Theorem 1.

5. COMPARISON BETWEEN CARRY-IN WORKLOAD FUNCTIONS

The carry-in workload function derived in [11, 22] shows that the worst-case scenario in constrained-deadline systems occurs when some job of τ_i executing C_i units precedes the end of a contiguous interval of length t and the earliest job having absolute deadline within the interval finishes its execution at the end of its deadline (also see Figure 4a). This scenario may accord with the work executing in the schedule as long as the worst-case response time is equal to its relative deadline. On the other hand, our carry-in workload function overly counts the carry-in job that is assumed to arrive D_i time-units prior to t (also see Figure 4b), even if such a job cannot effectively execute over the interval. Hence, our carry-in workload function for constrained-deadline tasks is inferior to that in [11,22]: at most C_i may be overestimated. Nevertheless, empirical results show that the difference is insignificant in terms of schedulability, as will be seen later. On the other hand, the assumption that there is at most one carry-in job of a carry-in arbitrary-deadline task is problematic for the carry-in workload function derived in [11,22]. A corresponding counterexample can be found in [29]. Hence, the revised workload function is the first one proved correct for arbitrary-deadline tasks.

6. LINEAR-TIME UPPER BOUND UNDER FP

In this section we present a linear-time response time analysis. This test determines the upper bound on the response time under global fixed-priority scheduling upon the values of the tasks' parameters, i.e., the worst-case execution time, the relative deadline, and the period.

We first linearize the upper interference function $\Omega(t)$ by using Eq. (2) in the following lemma:

LEMMA 4. *For all $t > 0$*

$$\Omega(t) \leq \left(Z_\Sigma + \sum_{\tau_i \in hp(k)} (tU_i + C_i(1 - U_i)) \right) \quad (12)$$

where Z_Σ denotes the sum of the $(m-1)$ largest $U_i D_i$'s among the tasks in $hp(k)$.

PROOF. Let S denote the *set* of the $(m-1)$ largest $I_i^{DIFF}(t)$.

$$\begin{aligned}
\Omega(t) &= \sum_{\tau_i \in hp(k)} I_i^1(t) + \sum_{\text{the } (m-1) \text{ largest}} I_i^{DIFF}(t) \\
&= \sum_{\tau_i \in hp(k) \setminus S} I_i^1(t) + \sum_{\tau_i \in S} I_i^2(t) \\
&\overset{(5) \text{ and } (6)}{\leq} \sum_{\tau_i \in hp(k) \setminus S} W_i(t) + \sum_{\tau_i \in S} W_i(t + D_i) \\
&\overset{(2)}{\leq} Z_\Sigma + \sum_{\tau_i \in hp(k)} (tU_i + C_i(1 - U_i))
\end{aligned}$$

□

Subsequently, the following lemma provides an upper bound on the maximum length of the h-incomplete busy periods of task τ_k.

LEMMA 5.

$$R_{k,h} \leq R_{k,h}^\dagger \overset{def}{=} \frac{hmC_k + Z_\Sigma + \sum_{\tau_i \in hp(k)} C_i(1 - U_i)}{m - \sum_{\tau_i \in hp(k)} U_i} \quad (13)$$

where Z_Σ denotes the sum of the $(m-1)$ largest $D_i U_i$'s.

PROOF. Substituting $\Omega(t)$ in the recurrence Eq. (9) by the one in Lemma 4, we can find the point t of the intersection, which leads to the statement. □

By Lemmas 1 and 5, $RT_{k,h}$ is bounded from above by

$$RT_{k,h} \leq RT_{k,h}^\dagger \overset{\text{def}}{=} R_{k,h}^\dagger - (h-1)T_k \quad (14)$$

where $R_{k,h}^\dagger$ is defined in Lemma 5. Intuitively, one may examine all possible h in the above equation to ensure that the upper bound on the worst-case response time. Our observation is that $RT_{k,h}^\dagger$ defined by Eq. (14) is monotonically decreasing with respect to h, under certain condition. We state this with the following lemma:

LEMMA 6. *Suppose that*

$$mU_k + \sum_{\tau_i \in hp(k)} U_i < m \quad (15)$$

Then, for any integer $h \geq 1$,

$$RT_{k,1}^\dagger \geq RT_{k,h}^\dagger$$

where $RT_{k,h}^\dagger$ is as defined in Eq. (14).

PROOF. We now prove that $R_{k,h}^\dagger - (h-1)T_k$ is maximized when h is 1, under Condition (15). Differentiating $R_{k,h}^\dagger - (h-1)T_k$ with respect to h, we have that

$$\begin{aligned}
\frac{\partial}{\partial h}\left(R_{k,h}^\dagger - (h-1)T_k\right) &= \frac{mC_k}{m - \sum_{\tau_i \in hp(k)} U_i} - T_k \\
&= T_k \left(\frac{mU_k}{m - \sum_{\tau_i \in hp(k)} U_i} - 1 \right) \\
&= T_k \left(\frac{mU_k + \sum_{\tau_i \in hp(k)} U_i - m}{m - \sum_{\tau_i \in hp(k)} U_i} \right)
\end{aligned}$$

which is negative due to our assumption $mU_k + \sum_{\tau_i \in hp(k)} U_i < m$. Since $R_{k,h}^\dagger - (h-1)T_k$ is decreasing with respect to h, the maximum occurs when $h = 1$. □

Putting these pieces together, Theorem 2 provides the response time upper bound of arbitrary-deadline tasks on multiprocessor systems.

THEOREM 2. *The worst-case response time RT_k of task τ_k is at most*

$$RT_k \leq RT_k^\dagger = \begin{cases} R_k^{up} & \text{if } mU_k + \sum_{\tau_i \in hp(k)} U_i < m, \\ \infty & \text{otherwise.} \end{cases}$$

where $R_k^{up} = \frac{mC_k + Z_\Sigma + \sum_{\tau_i \in hp(k)} C_i(1-U_i)}{m - \sum_{\tau_i \in hp(k)} U_i}$ and Z_Σ denotes the sum of the $(m-1)$ largest $D_i U_i$'s among the tasks in $hp(k)$.

PROOF. By Lemmas 5 and 6, we can conclude this theorem by setting $h = 1$ in Eq. (14). □

Subsequently, we provide our linear-time upper bound (LTUB) on the worst-case response time in the follow corollary:

COROLLARY 2 (LTUB). *An arbitrary-deadline sporadic task system τ is schedulable on m identical processors under fixed-priority scheduling if for all tasks $\tau_k \in \tau$,*

$$RT_k^\dagger \leq D_k \quad (16)$$

where $RT_k^\dagger$ is as defined in Theorem 2.

7. EXPERIMENTS

In this section, we conduct extensive experiments using synthesized task sets for evaluating the proposed tests. The metric to compare results is to measure the *acceptance ratio* of the above tests with respect to a given goal of task set utilization. We generate 100 task sets for each utilization level, from 0.01 to 0.99, in steps of 0.01. For brevity, the tickers were set every three steps. The acceptance ratio of a level is said to be the number of task sets that are schedulable divided by the number of task sets for this level, i.e., 100.

7.1 Simulation Environment

We first generated a set of sporadic tasks. The cardinality of the task set was 5 times the number of processors. The UUniFast-Discard method [14] was adopted to generate a set

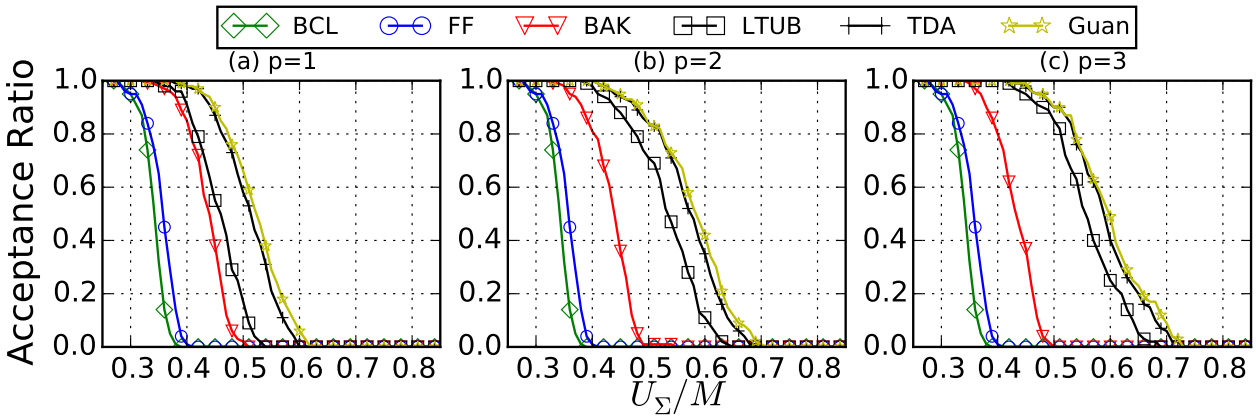


Figure 5: Acceptance ratio comparison with different p on 8-multiprocessor, constrained-deadline systems where $\frac{D_i}{T_i} \in [0.8, 1]$.

of utilization values with the given goal. We here used the approach suggested by Davis and Burns [20] to generate the task period according to the exponential distribution. The order of magnitude p to control the period values between largest and smallest periods is parameterized in evaluations. (E.g., $1 - 10ms$ for $p = 1$, $1 - 100ms$ for $p = 2$, etc.).

We evaluate these tests in 8 multiprocessor systems with $p \in [1, 2, 3]$. Similar results can be seen in different numbers of processors. The priority of task is assigned according to deadline-monotonic (DM) scheduling: the smaller the relative deadline, the higher the priority level.

We note that deadline-monotonic (DM) scheduling is not *optimal* on multiprocessor systems. Alternatively, one can use the *laxity-monotonic* (LM) scheduling: the smaller the laxity $D_i - C_i$, the higher the priority level [2]. Also, it is not difficult to see that the proposed TDA analysis and the linear-time response bound comply with the required conditions for the optimal priority assignment (OPA) compatibility provided in [18]. Hence, our proposed tests are compatible with OPA: if there exists feasible priority assignments by our test, OPA returns one of them. It is also worth noting that Davis and Burns have suggested that being OPA-compatible is as important as being effective since priority assignment is an important factor in determining the schedulability of tasksets under global fixed priority preemptive scheduling [18]. Consequently, applying the tighter test but not OPA-comptiable, e.g. maximum work function $\hat{I}_i^2(B_{k,h})$, by using DM, may not yield better performance than the less effective but OPA-comptiable test using OPA priority assignment. Nevertheless, we here focus on showing the effectiveness of the tests themselves, and similar results can be seen under different priority assignment policies.

7.2 Constrained-Deadline Systems & Results

The execution time was set accordingly, i.e., $C_i = T_i U_i$. Task relative deadlines were uniformly drawn from the interval $[0.8T_i, T_i]$.

The tests evaluated are shown as follows:

- *BCL*: the polynomial-time test in Theorem 4 in [11].
- *FF*: the force-forward (FF) analysis in Eq. (5) in [6].
- *BAK*: the $O(n^3)$ test in Theorem 11 in [4].
- *Guan*: the TDA analysis by Guan [22].
- *TDA*: Corollary 1 in this paper.
- *LTUB*: Corollary 2 in this paper.

Results. Figure 5 presents the evaluation result in constrained-deadline systems. We first notice that the response time analysis with limited carry-in tasks, namely *Guan* and *TDA*, can admit the most number of task sets. Overall, these two tests achieve similar performance, even though our *TDA* is inferior than *Guan* due to the over-approximate workload function, as mentioned earlier. As shown in Figure 5, *BCL*, *FF*, and *BAK* perform poorer than *LTUB*, even though their computational complexity is higher than *LTUB*. (pseudo-polynomial time for *FF* and $O(n^3)$ for *BAK*). It is noticeable that *TDA* is only slightly advantageous to *LTUB* in case of $p = 2$ and $p = 3$ (Figure 5(b) and (c)) where the TDA essentially suffers from its high computational complexity. Hence, in practice, one would expect that the TDA be adopted in the low order of magnitude of task periods and the linear-time response time bound in the high order of the magnitude.

7.3 Arbitrary-Deadline Systems & Results

The execution time was set accordingly, i.e., $C_i = T_i U_i$. Task relative deadlines were uniformly drawn from the interval $[0.8T_i, 2T_i]$. The tests evaluated are shown as follows:

- *LOAD*: the load-based analysis [7].
- *BAK*: the $O(n^3)$ test in Theorem 11 [4].
- *LTUB*: Corollary 2 in this paper.
- *TDA*: Corollary 1 in this paper.

Results. Figure 6 presents the result in arbitrary-deadline systems where $\frac{D_i}{T_i} \in [0.8, 2]$. Due to the overly optimistic workload function, *Guan*'s analysis becomes unsafe for arbitrary-deadline tasks. The proposed TDA is the best TDA with limited carry-in interferences in arbitrary-deadline systems, as can be seen in Figure 6. Also, in terms of schedulability and time complexity, the proposed linear-time upper bound *LTUB* is superior to *LOAD* and *BAK*, where the time complexity is pseudo-polynomial for *LOAD* and $O(n^3)$ for *BAK*.

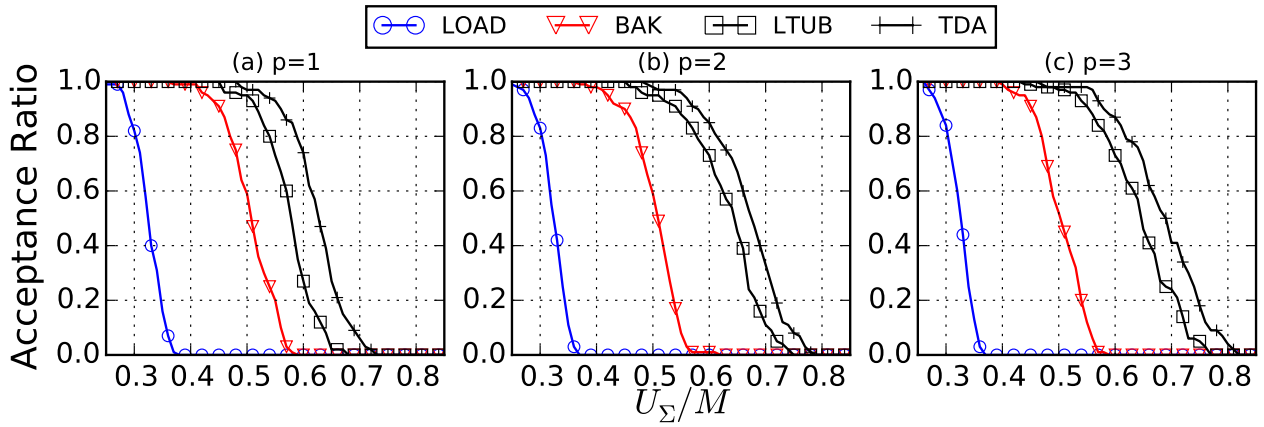


Figure 6: Acceptance ratio comparison with different p on 8-multiprocessor, arbitrary-deadline systems where $\frac{D_i}{T_i} \in [0.8, 2]$.

8. CONCLUSIONS

In this work, we propose the response time analysis for sporadic arbitrary-deadline tasks on multiprocessor systems under fixed-priority scheduling. Further, we derive a linear-time response time bound that provides an amenable solution for the the admission control and the interactive system and prototyping. To the best of our knowledge, this is the first work presenting a linear-time response time bound on multiprocessor systems. We further conclude that the linear-time response time bound yields high performance compared to the state-of-the-art TDA tests, especially in case of large task periods, where time-demand analysis tests suffer from their computational complexity.

9. ACKNOWLEDGMENTS

This paper is supported by DFG, as part of the Collaborative Research Center SFB876 (http://sfb876.tu-dortmund.de/).

10. REFERENCES

[1] B. Andersson, S. Baruah, and J. Jonsson. Static-priority scheduling on multiprocessors. In *Real-Time Systems Symposium (RTSS 2001)*, pages 193–202, 2001.

[2] B. Andersson and J. Jonsson. Fixed-priority preemptive multiprocessor scheduling: to partition or not to partition. In *Real-Time Computing Systems and Applications, 2000. Proceedings. Seventh International Conference on*, pages 337–346. IEEE, 2000.

[3] T. P. Baker. Multiprocessor edf and deadline monotonic schedulability analysis. In *IEEE Real-Time Systems Symposium*, pages 120–129, 2003.

[4] T. P. Baker. An analysis of fixed-priority schedulability on a multiprocessor. *Real-Time Systems*, 32(1-2):49–71, 2006.

[5] T. P. Baker and M. Cirinei. Brute-force determination of multiprocessor schedulability for sets of sporadic hard-deadline tasks. In *Principles of Distributed Systems, 11th International Conference, OPODIS 2007, Guadeloupe, French West Indies, December 17-20, 2007. Proceedings*, pages 62–75, 2007.

[6] S. Baruah, V. Bonifaci, A. Marchetti-Spaccamela, and S. Stiller. Improved multiprocessor global schedulability analysis. *Real-Time Systems*, 46(1):3–24, 2010.

[7] S. Baruah and N. Fisher. Global fixed-priority scheduling of arbitrary-deadline sporadic task systems. In *Distributed Computing and Networking*, pages 215–226. Springer, 2008.

[8] S. K. Baruah. Techniques for multiprocessor global schedulability analysis. In *Proceedings of the 28th IEEE Real-Time Systems Symposium*, pages 119–128, 2007.

[9] M. Bertogna and M. Cirinei. Response-time analysis for globally scheduled symmetric multiprocessor platforms. In *Real-Time Systems Symposium*, pages 149–160, 2007.

[10] M. Bertogna, M. Cirinei, and G. Lipari. Improved schedulability analysis of edf on multiprocessor platforms. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 209–218, 2005.

[11] M. Bertogna, M. Cirinei, and G. Lipari. New schedulability tests for real-time task sets scheduled by deadline monotonic on multiprocessors. In *Principles of Distributed Systems*, pages 306–321. Springer, 2006.

[12] E. Bini. The quadratic utilization upper bound for arbitrary deadline real-time tasks. *Computers, IEEE Transactions on*, 64(2):593–599, 2015.

[13] E. Bini and G. C. Buttazzo. Schedulability analysis of periodic fixed priority systems. *Computers, IEEE Transactions on*, 53(11):1462–1473, 2004.

[14] E. Bini and G. C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005.

[15] E. Bini, T. H. C. Nguyen, P. Richard, and S. K. Baruah. A response-time bound in fixed-priority scheduling with arbitrary deadlines. *IEEE Transactions on Computers*, 58(2):279, 2009.

[16] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of computer and system sciences*, 7(4):448–461, 1973.

[17] R. I. Davis and A. Burns. Response time upper

bounds for fixed priority real-time systems. In *Real-Time Systems Symposium, 2008*, pages 407–418. IEEE, 2008.

[18] R. I. Davis and A. Burns. Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. *Real-Time Systems*, 47(1):1–40, 2011.

[19] R. I. Davis and A. Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Computing Surveys (CSUR)*, 43(4):35, 2011.

[20] R. I. Davis, A. Zabos, and A. Burns. Efficient exact schedulability tests for fixed priority real-time systems. *Computers, IEEE Transactions on*, 57(9):1261–1276, 2008.

[21] G. Geeraerts, J. Goossens, and M. Lindström. Multiprocessor schedulability of arbitrary-deadline sporadic tasks: complexity and antichain algorithm. *Real-Time Systems*, 49(2):171–218, 2013.

[22] N. Guan, M. Stigge, W. Yi, and G. Yu. New response time bounds for fixed priority multiprocessor scheduling. In *Real-Time Systems Symposium, 2009, RTSS 2009. 30th IEEE*, pages 387–397. IEEE, 2009.

[23] J. P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Real Time Systems Symposium*, pages 201–209, 1990.

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

[25] A. K. Mok. Fundamental design problems of distributed systems for the hard-real-time environment. 1983.

[26] C. A. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 140–149. ACM, 1997.

[27] M. Sjodin and H. Hansson. Improved response-time analysis calculations. In *Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE*, pages 399–408. IEEE, 1998.

[28] Y. Sun and G. Lipari. A weak simulation relation for real-time schedulability analysis of global fixed priority scheduling using linear hybrid automata. In *22nd International Conference on Real-Time Networks and Systems, RTNS '14, Versaille, France, October 8-10, 2014*, page 35, 2014.

[29] Y. Sun, G. Lipari, N. AGuan, W. Yi, et al. Improving the response time analysis of global fixed-priority multiprocessor scheduling. In *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 1–9, 2014.

[30] K. Tindell, A. Burns, and A. J. Wellings. An extendible approach for analyzing fixed priority hard real-time tasks. *Real-Time Systems*, 6(2):133–151, 1994.