Efficient computation of response time bounds under fixed-priority scheduling

Enrico Bini
Scuola Superiore Sant'Anna
Pisa, Italy
`e.bini@sssup.it`

Sanjoy K. Baruah
University of North Carolina
Chapel Hill, NC
`baruah@cs.unc.edu`

Abstract

All algorithms currently known for computing the response time of tasks scheduled under fixed-priority scheduling have run-time pseudo-polynomial in the representation of the task system. We derive a formula that can be computed in polynomial time for determining an upper bound on response times; our upper bound on response time has the added benefit of being continuous in the task system parameters. We evaluate the effectiveness of our approximation by a series of simulations; these simulations reveal some interesting properties of (exact) response time, which give rise to an open question that we pose as a conjecture.

Finally, the proposed upper bound of the response time can be used to test effectively the schedulablity of task sets in time linear with the number of tasks.

1. Introduction

In many real-time systems specific jobs are expected to complete by specified deadlines. Basically, two main categories of algorithms have been proposed for determining the response times of tasks in DM-scheduled systems: Rate Monotonic Analysis (RMA) [13] and Response Time Analysis (RTA) [10, 2].

RTA computes, for each task, the *worst-case response times* — the maximum amount of time that may elapse between the instant that a job is released for execution and the instant it completes execution. If, for all tasks, the response time is shorter than the deadline, then the task set is feasible. Instead, RMA searches, for each task, any instant earlier than the deadline, large enough to accommodate the computational requirement of the task itself and all the higher priority tasks. If such an instant exists for all tasks then the task set is feasible.

Both approaches are known to have pseudo-polynomial worst-case time complexity, and it is currently unknown whether the task set feasibility can be computed in time polynomial in the representation of the task system.

Despite the pseudo-polynomial time complexity, both RMA and RTA have very efficient implementations in practice that render them suitable for feasibility analysis of Fixed Priority (FP) systems. However, these algorithms may not be particularly well-suited for use in interactive real-time system design environments. When using such design environments, the system designer typically makes a large number of calls to a feasibility-analysis algorithm during a process of interactive system design and rapid system prototyping, since proposed designs are modified according to the feedback offered by the feasibility-analysis algorithm (and other analysis techniques). In such scenarios, a pseudo-polynomial algorithm for computing the task set feasibility may be unacceptably slow; instead, it may be acceptable to use a faster algorithm that provides an approximate, rather than exact, analysis.

Moreover, there are some circumstances in the real-time system design, such as in control systems [8] and in holistic analysis [19], where it is required to know the response time of the tasks, and not only the system feasibility provided by RMA. For this reason in this paper, we propose an algorithm for computing efficiently an approximate upper bound of the response time. In addition to computation efficiency, our algorithm has the benefit of representing the (bound on) response time as a continuous function of the task system parameters, thereby facilitating optimisation of system design in applications, such as some control systems, where task parameters may be tweaked locally without causing catastrophic changes to application semantics. (Response time is not in general a continuous function of system parameters; hence, no exact algorithm for computing response times can possibly make a similar guarantee.)

There are many scenarios in which efficient computation of (exact or approximate) response times is desirable.

- In distributed systems, tasks may be activated after the completion of some other task [22, 19]. In such cases it is necessary to know the response time of the first task in order to analyse the scheduling of the second. This task model is called *transaction model* [19], and the analysis is performed by means of the *holistic analy-*

sis [22].

- In control systems, the response time of a task measure the delay between the instant where the input are read from the sensors and the output are written to the actuators. The performance of the control system depends upon this value [8] hence the response time has a direct impact on the system performance. Moreover, as our provided bound of the response time is a differentiable function, it is possible to estimate the effect of the variation of any system parameter.

- Finally, when the relative deadline parameters are permitted to be larger than the periods, current algorithms for the exact computation of response time require the evaluation of the response times of each and every job within the busy period [12, 23]. The resulting complexity may be unacceptably high, especially in all those design environments where the response time routine is largely invoked.

1.1. Related work

The problem of reducing the time complexity of feasibility tests has been largely addressed by the real-time research community. The Rate Monotonic Analysis, after the first formulation by Lehoczky et al. [13], has been improved by Manabe and Aoyagi [17] who reduced the number of points where the time demand needs to be checked. Bini and Buttazzo [4] proposed a method to trade complexity vs. accuracy of the RMA feasibility tests.

The efforts in the simplification of the Response Time Analysis has been even stronger, probably due to the greater popularity of RTA. Sjödin and Hansson [21] proposed several lower bounds to the response time so that the original response time algorithm [10] could start further and the time spent in computing the response time is reduced. Bril [7] proposed a similar technique to reduce the time complexity of the exact RTA. Starting from the idea of Albers and Slomka [1], who developed an estimate of the demand bound function for EDF scheduled tasks, Fisher and Baruah [9] have derived a fully polynomial time approximation scheme (FPTAS) of the RTA. Very recently, Richard and Goossens [20] have extended the task model of a previous FPTAS [9] to take into account release jitter. Finally, Lu et al. [16] proposed a method to reduce the number of iterations for finding the task response times.

The remainder of this paper is organised as follows. In Section 2 we formally state our task model, and reduce the problem of bounding the response time of each task in a task system to a problem of bounding the total workload generated by the task system. In Section 3 we derive a bound on the workload, which immediately yields the desired response time bound. We describe a series of simulation experiments in Section 4 for determining the "goodness" of our upper bound. We conclude in Section 5 with a brief summary of the main results presented in this paper.

2. The Response Time Bound

We assume that a real-time system is modelled as being comprised of a pre-specified number n of independent *sporadic* tasks [18, 3] $\tau_1, \tau_2, \ldots, \tau_n$, executing upon a single shared preemptive processor. Each sporadic task τ_i is characterised by a worst-case execution time (WCET) C_i; a relative deadline parameter D_i; and a period/ minimum inter-arrival separation parameter T_i. Notice that the deadlines are arbitrary, meaning that no particular relationship is assumed between D_i and T_i. Each such task generates an infinite sequence of jobs, each with execution requirement at most C_i and deadline D_i time-units after its arrival, with the first job arriving at any time and subsequent successive arrivals separated by at least T_i time units. We assume that the system is scheduled using a fixed-priority (FP) scheduling algorithm such as the Deadline-Monotonic (DM) scheduling algorithm [14], which is known to be an optimal fixed-priority algorithm when all the sporadic tasks have their relative deadline parameters no larger than their periods.

We will use the term *utilisation* of τ_i (denoted by U_i), to represent the ratio C_i/T_i, and let U denote the *system utilisation*: $U = \sum_{i=1}^{n} U_i$. We assume that *tasks are indexed according to priorities*: task τ_1 is the highest-priority task, and τ_{i+1} has lower priority than τ_i for all i, $1 \le i < n$. Notice that we do not assume any specific priority assignment.

We start with some notations and definitions. Let us define the *worst-case workload* as follows:

Definition 1 *Let $W_i(t)$ denote the* worst-case workload *of the i highest priority tasks over an interval of length t, which is the maximum amount of time that a task τ_j, with $1 \le j \le i$ can run over an interval of length t.*

As proved by Liu and Layland in their seminal paper [15], the worst-case workload $W_i(t)$ occurs when all the tasks $\tau_1, \ldots, \tau_i$ are simultaneously activated, and each task generates subsequent jobs as soon as legally permitted to do so (i.e., consecutive jobs of τ_i arrive exactly T_i time units apart, for all i) – this sequence of job arrivals is sometimes referred to as the *synchronous arrival sequence*. Thus, $W_i(t)$ equals the maximum amount of time for which the CPU may execute some task from among $\{\tau_1, \ldots, \tau_i\}$, over the time interval $[0, t)$, for the synchronous arrival sequence.

We highlight that our definition of worst-case workload is different than the *worst-case demand*, which is expressed by the "classical ceiling" expression $\sum_i \left\lceil \frac{t}{T_i} \right\rceil C_i$. The worst-case workload is the fraction of the demand which can be executed in $[0, t)$, under the synchronous arrival sequence hypothesis, whereas the demand is the maximum amount of work which can be *demanded* in $[0, t)$.

A closely-related concept is that of the *worst-case idle time*:

Definition 2 *Let $H_i(t)$ denote the* worst-case idle time *of the i highest priority tasks over an interval of length t.*

This is the minimum amount of time that the CPU is not executing some task in $\{\tau_1, \ldots, \tau_i\}$ over the time interval $[0, t)$. It is straightforward to observe that

$$H_i(t) = t - W_i(t) \tag{1}$$

Let us define the *(pseudo) inverse* of the idle time, as follows:

Definition 3 *The (pseudo) inverse function $X_i(c)$ of $H_i(t)$ is the smallest time instant such that there are at least c time units when the processor is not running any tasks in $\{\tau_1, \ldots, \tau_i\}$, over every interval of length $X_i(c)$. That is,*

$$X_i(c) = \min_t \{t : H_i(t) \geq c\}$$

We note that $H_i(t)$ is not an invertible function, since there may be several time-instants t for which $H_i(t)$ is constant — that is why we refer to $X_i(c)$ as a <u>pseudo</u> inverse. In the remainder of this paper we will abuse notation somewhat, and use the following notation:

$$X_i(c) = [H_i(t)]^{-1} \tag{2}$$

Based upon this definition of the inverse of the idle time, we obtain the following alternative representation of task response time. (Observe that this relationship holds regardless of whether task deadlines are lesser than, equal to, or greater than periods.)

Lemma 1 *The worst-case response time R_i of task τ_i is given by:*

$$R_i = \max_{k=1,2,\ldots} \{X_{i-1}(k\, C_i) - (k-1)\, T_i\} \tag{3}$$

Proof. $X_{i-1}(k\, C_i)$ is the instant when the first $i-1$ tasks have left $k\, C_i$ units of time available for the lower priority tasks. Hence it is also the finishing time of the k^{th} job of τ_i in the busy period. $(k-1)\, T_i$ is the activation of such a job. The proof hence follows directly as in [23]. $\square$

Notice that if $\sum_{j=1}^{i} U_i > 1$ then the we clearly have $R_i = +\infty$. For this reason in realistic cases we assume $\sum_{j=1}^{i} U_i \leq 1$.

Some further notation: for any function $f(x)$, $f^{\mathsf{ub}}(x)$ denotes an upper bound, and $f^{\mathsf{lb}}(x)$ denote a lower bound on the function $f(x)$, so that we have $f^{\mathsf{lb}}(x) \leq f(x) \leq f^{\mathsf{ub}}(x)$ for all x.

Theorem 1 *For any upper bound $W_i^{\mathsf{ub}}(t)$ on the workload $W_i(t)$, there is a corresponding upper bound R_i^{ub} on the worst-case response time R_i.*

Proof. Since $W_i^{\mathsf{ub}}(t)$ is an upper bound of $W_i(t)$ we have by definition

$$W_i^{\mathsf{ub}}(t) \geq W_i(t)$$

from which it follows the obvious relationship for the idle time

$$H_i^{\mathsf{lb}}(t) = t - W_i^{\mathsf{ub}}(t) \leq t - W_i(t) = H_i(t)$$

which gives us a lower bound of the idle time. From this relationship it follows that for any possible value c we have

$$\{t : H_i^{\mathsf{lb}}(t) \geq c\} \subseteq \{t : H_i(t) \geq c\}$$

Now it is possible to find a relationship between the pseudo-inverse functions. In fact we have

$$X_i^{\mathsf{ub}}(c) = \min_t \{t : H_i^{\mathsf{lb}}(t) \geq c\} \geq \min_y \{t : H_i(t) \geq c\} = X_i(c)$$

from which it follows that

$$R_i^{\mathsf{ub}} = \max_{k=1,2,\ldots} \{X_{i-1}^{\mathsf{ub}}(k\, C_i) - (k-1)\, T_i\} \geq R_i$$

as required. $\square$

3. The workload upper bound

As stated above, it was proved by Liu and Layland [15] that the worst-case workload $W_i(t)$ occurs for the synchronous arrival sequence of jobs — i.e., when all the tasks $\tau_1, \ldots, \tau_i$ are simultaneously activated, and consecutive jobs of τ_i arrive exactly T_i time units apart, for all i. Hence the function $W_i(t)$ may be expressed by the sum of the individual workload of each task τ_j. If we let $w_j(t)$ denote the maximum amount of time that the processor executes task τ_j over the interval $[0, t)$ in this worst-case scenario, we can write:

$$W_i(t) = \sum_{j=1}^{i} w_j(t)$$

This is shown in Figure 1.

Letting $w_j^o(t)$ denote the maximum amount of time that the processor executes task τ_j in any interval of length t, **when task τ_j is the only task in the system**, clearly we have:

$$\forall j \quad \forall t \qquad w_j^o(t) \geq w_j(t)$$

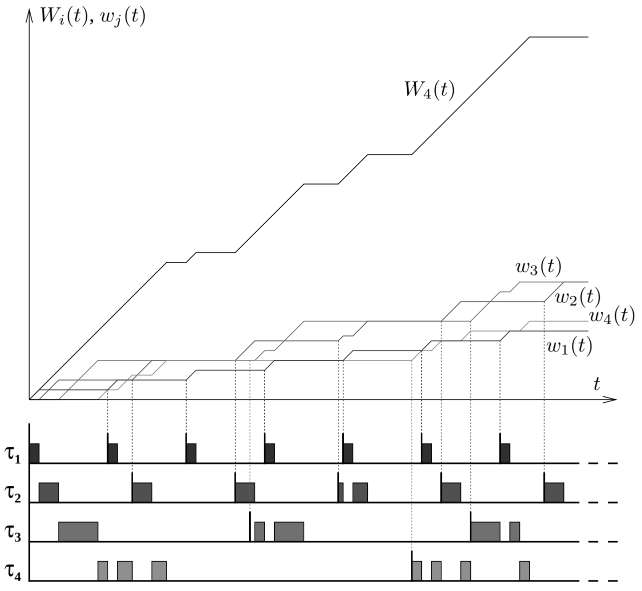

Figure 1. An example of the $W_i(t)$ and $w_j(t)$

since the presence of additional jobs may only delay the execution of τ_j's jobs.

The workload $w_j^o(t)$, which is equal to $\min\left\{t-(T_j-C_j)\left\lfloor\frac{t}{T_j}\right\rfloor, \left\lceil\frac{t}{T_j}\right\rceil C_j\right\}$, can be conveniently upper bounded by the linear function as shown in Figure 2. The equation of the linear bound is $U_j\, t + C_j(1-U_j)$.

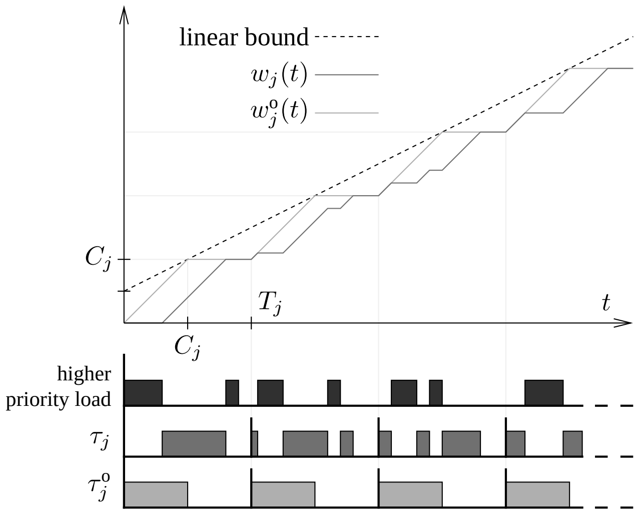

Figure 2. The upper linear bound of $w_j(t)$

Using these relationships found for the workload $w_j(t)$ of each task, if we sum over j from 1 to i we obtain an upper bound on the workload function $W_i(t)$:

$$W_i(t) = \sum_{j=1}^{i} w_j(t) \le \sum_{j=1}^{i} w_j^o(t) \le \le \sum_{j=1}^{i} (U_j\, t + C_j\,(1-U_j)) \quad (4)$$

We have so obtained the upper bound we were looking for. The property of this bound is that we can compute conveniently its inverse function and then apply the Theorem 1 to finally find the bound of the response time.

Theorem 2 *The worst-case response time R_i of task τ_i is bounded from above as follows:*

$$R_i \le \frac{C_i + \sum_{j=1}^{i-1} C_j(1-U_j)}{1-\sum_{j=1}^{i-1} U_j} = R_i^{\mathsf{ub}} \quad (5)$$

Proof. The proof of this theorem is obtained by applying Theorem 1 to the workload bound provided by the Eq. (4). So we have:

$$W_i^{\mathsf{ub}}(t) = \sum_{j=1}^{i}(U_j\, t + C_j(1-U_j))$$
$$H_i^{\mathsf{lb}}(t) = t\left(1-\sum_{j=1}^{i} U_j\right) - \sum_{j=1}^{i}(C_j(1-U_j))$$

Since $H_i^{\mathsf{lb}}(t)$ is invertible, it can be used to compute $X_i^{\mathsf{ub}}(h)$.

$$X_i^{\mathsf{ub}}(h) = \frac{h+\sum_{j=1}^{i} C_j(1-U_j)}{1-\sum_{j=1}^{i} U_j}$$

Then the response time is bounded by:

$$\max_{k=1,2,\ldots}\left(\frac{kC_i+\sum_{j=1}^{i-1} C_j(1-U_j)}{1-\sum_{j=1}^{i-1} U_j} - (k-1)T_i\right) \quad (6)$$

We will now prove that the maximum in the Eq. (6) occurs for $k = 1$. Let us consider this function on the real extension $[1,+\infty)$. On this interval we can differentiate with respect to k. Doing so we get:

$$\frac{\mathrm{d}}{\mathrm{d}k}\left(\frac{kC_i + \sum_{j=1}^{i-1} C_j(1-U_j)}{1-\sum_{j=1}^{i-1} U_j} - (k-1)T_i\right) =$$
$$\frac{C_i}{1-\sum_{j=1}^{i-1} U_j} - T_i =$$
$$T_i\left(\frac{U_i}{1-\sum_{j=1}^{i-1} U_j} - 1\right) =$$
$$T_i\left(\frac{\sum_{j=1}^{i} U_j - 1}{1-\sum_{j=1}^{i-1} U_j}\right)$$

which is always negative (or zero). In fact, if $\sum_{j=1}^{i} U_j > 1$ the response time is known to be arbitrarily long, and so unbounded. Then, since the function is decreasing (or constant), its maximum occurs in the left bound of the interval, which means $k = 1$. Finally, by substituting $k = 1$ in Eq. (6), we get:

$$R_i^{\mathsf{ub}} = \frac{C_i + \sum_{j=1}^{i-1} C_j(1-U_j)}{1-\sum_{j=1}^{i-1} U_j} \tag{7}$$

as required. □

Moreover we can divide by T_i to normalise the bound and we get:

$$r_i^{\mathsf{ub}} = \frac{R_i^{\mathsf{ub}}}{T_i} = \frac{U_i + \sum_{j=1}^{i-1} a_j\, U_j(1-U_j)}{1-\sum_{j=1}^{i-1} U_j} \tag{8}$$

where $a_j = T_j/T_i$.

The time complexity of computing the response time upper bound R_i^{ub} of task τ_i is $O(i)$. Hence the complexity of computing the bound for all the tasks seems to be $O(n^2)$. However, it can be noticed that the computation of R_{i+1}^{ub} can take advantage of the completed computation of R_i^{ub}. In fact the two sums involved in Equation (5) can be simply computed by adding only the values relative to the last index to the sum values of the previous computation. This observation allows us to say that the computation of the response time upper bound of all the tasks in $O(n)$.

There are other techniques to bound the response time. Similarly as suggested by Sjödin and Hansson [21], a different upper bound on the worst-case response times may be obtained from the recurrence used in response-time analysis [10, 2] by replacing the ceiling function $\lceil x \rceil$ with $x + 1$. We have:

$$R_i = C_i + \sum_{j=1}^{i-1}\left\lceil\frac{R_i}{T_j}\right\rceil C_j$$
$$R_i \le C_i + \sum_{j=1}^{i-1}\left(\frac{R_i}{T_j}+1\right) C_j$$
$$R_i - R_i\sum_{j=1}^{i-1} U_j \le C_i + \sum_{j=1}^{i-1} C_j$$
$$R_i \le \frac{\sum_{j=1}^{i} C_j}{1-\sum_{j=1}^{i-1} U_j}$$

Observe that this is a looser bound than the one we have obtained above, in Theorem 2.

We conclude by reiterating the benefits of using the response time upper bound presented in Theorem 2 above:

- it can be computed in $O(n)$ time;
- it is continuous and differentiable in all the variables;
- the bound holds even for deadlines greater than the period. In this case the exact algorithm for the response time calculation [23] requires to check all the jobs within the busy period;
- the bound has a closed formulation, instead that an iterative definition. Hence it is possible to adopt some feedback on task parameters (C_j or T_j) so that the response time is modified in some desired direction.

3.1. A sufficient schedulability test

In the same way as the exact values of the response times allow to formulate a necessary and sufficient schedulability test, the response time upper bound R_i^{ub} allows to express a sufficient schedulability condition for the fixed priority algorithm. It is then possible to enunciate the following $O(n)$ sufficient schedulability condition for tasks scheduled by fixed priority with arbitrary deadline.

Corollary 1 *A task set $\tau_1, \ldots, \tau_n$ is schedulable by fixed priorities **if**:*

$$\forall i \quad R_i^{\mathsf{ub}} = \frac{C_i + \sum_{j=1}^{i-1} C_j(1-U_j)}{1-\sum_{j=1}^{i-1} U_j} \le D_i \tag{9}$$

Proof. From Theorem 2 it follows that $R_i \le R_i^{\mathsf{ub}}$. From the hypothesis it follows that $R_i^{\mathsf{ub}} \le D_i$. Then it follows that $R_i \le D_i$, which means that all the tasks do not miss their deadlines. □

Corollary 1 provides a very efficient means for testing the feasibility of task sets. This condition can also be restated as a utilisation upper bound, and compared with many existing schedulability tests [15, 12, 11, 6]. Since some of these results are achieved assuming deadlines equal to periods, we also provide the following corollary in this hypothesis although this restriction doesn't apply to our response time upper bound.

Corollary 2 *A task set $\tau_1, \ldots, \tau_n$, with deadlines equal to periods ($D_i = T_i$) is schedulable by fixed priorities **if**:*

$$\forall i \quad \sum_{j=1}^{i} U_j \leq 1 - \sum_{j=1}^{i-1} a_j \, U_j (1 - U_j) \tag{10}$$

where $a_j = T_j / T_i$.

Proof. From Equation (9) it follows that the task τ_i is schedulable if

$$\frac{U_i + \sum_{j=1}^{i-1} a_j \, U_j (1 - U_j)}{1 - \sum_{j=1}^{i-1} U_j} \leq 1$$

where $a_j = \frac{T_j}{T_i}$. Also notice that if tasks are scheduled by RM then $a_j \leq 1$ always. From the last equation we have

$$U_i + \sum_{j=1}^{i-1} a_j \, U_j (1 - U_j) \leq 1 - \sum_{j=1}^{i-1} U_j$$

$$\sum_{j=1}^{i} U_j \leq 1 - \sum_{j=1}^{i-1} a_j \, U_j (1 - U_j)$$

which proves the corollary, when ensured for all tasks. □

It is quite interesting to observe that when the periods are quite large compared to the preceding one — meaning that $a_j \to 0$ — then the test is very effective. On the other hand, when all the periods are similar each other then the right hand side of Eq. (10) may also become negative, making the condition impossible. This intuition will be confirmed in the next section dedicated to the experiments.

4. Experiments

The major benefits of the response time upper bound that we have computed in Section 3 above lie in **(i)** the *time complexity* which, at $O(n)$ where n denotes the number of tasks, is linear in the representation of the task system; and **(ii)** the fact that the upper bound is *continuous* with respect to the task system parameters (and hence more useful in interactive system design). It is however, also important to evaluate the quality of the bound. Clearly, the tightness of the approximation depends upon the task set parameters. In order to estimate the distance between the exact value of the response time and of our derived upper bound (thereby determining the "goodness" of our upper bound), we performed a series of experiments that explored the impact of the different task characteristics.

4.1. Effect of task periods

In the first set experiments we evaluate the impact of task periods on the response time upper bound. For this purpose, we use a system comprised of only 2 tasks. The period of the higher-priority task is set $T_1 = 1$, whereas the period T_2 of the low priority task is calculated so that the ratio T_1/T_2 ranges in the interval $[0, 1]$. The task computation times C_1 and C_2 are chosen such that:

- the relative utilisations of the two tasks does not change in the experiments. This is achieved by setting $U_1/U_2 = 0.25$ always;
- the total utilisation $U = U_1 + U_2$ is equal to one of the four values $\{0.2, 0.4, 0.6, 0.8\}$ (we run four classes of experiments, one for each value).

We leave the values of D_1 and D_2 unspecified, since these parameters have no effect on either the exact response time, or our computed upper bound, under FP scheduling.

For each simulation, we computed the exact response time R_2 and our upper bound R_2^{ub} for the task τ_2. Notice that both the tasks will have response times smaller than or equal to their respective periods since the Liu and Layland utilisation bound for two tasks is $2(\sqrt{2} - 1) \approx 0.828$, which is greater than all the total utilisations assumed in this experiment. Hence the maximum response time occurs in the first job of τ_2. Both the response time and the upper bound are normalised with respect to the period T_2, so that the comparison between different values of the period T_2 is easier. The results are shown in Figure 3. Black lines are the normalised R_2^{ub} values, gray plots are the exact response times.

It may be noticed that the approximation is very good when $T_2 \gg T_1$ (i.e. when the ratio T_1/T_2 is close to zero). In fact, in this condition the workload estimate, upon which the response time bound is built, becomes very tight. The discontinuities in the response times occur when an additional job of τ_1 interferes with the response time of τ_2. Finally, it may be noticed that the approximation degrades as the total utilisation increases. This can be explained by reiterating that the upper estimate of the workload is tight for low utilisations, as can be observed from Figure 2.

Given this last observation, it becomes quite interesting to test the case when $U = 1$. In this condition of heavy load, the task system utilisation is no longer $\leq$ the Liu and

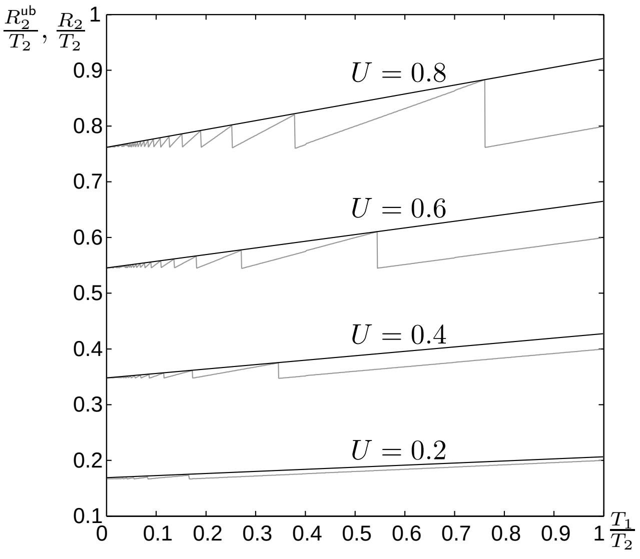

Figure 3. Effect of task periods

Layland utilisation bound, and hence it is not guaranteed that both tasks' response times will be $\leq$ their respective period parameters. Furthermore, the response time R_2 does not necessarily occur at the first job, and hence all the jobs within the first busy period must be checked. (In fact, under the condition $U = 1$ the processor is always busy and the busy period never ends, but the response time can still be computed by checking all the jobs up to hyperperiod — the least common multiple of all the periods.) A second — more serious — problem is related to the nature of the experiment: since we are running simulations as the period T_2 varies from T_1 to infinity, the hyperperiod can be extremely large! (Indeed, the hyperperiod does not even exist if T_1/T_2 is irrational, although this phenomenon is not encountered with machine representable numbers.) Hence, in our simulation setting the computation of the response time is stopped after 1000 jobs of τ_2. In the top part of Figure 4 we report the difference $R_2^{\text{ub}} - R_2$ normalised with respect to T_2 as usual. The result is quite surprising.

From the figure we see that the upper bound is a very tight approximation of the exact response time, unless *some harmonic relationship exists* between T_1 and T_2. Moreover, the stronger the harmonicity the greater the difference between the bound and the exact value (for example when $\frac{T_1}{T_2} \in \{1, \frac{1}{2}, \frac{1}{4}, \frac{2}{3}\}$.) When the periods are poorly harmonic the upper bound is extremely tight.

In these experiments we observed that in poorly harmonic periods, the response time routine needs to be conducted much further than in more harmonic conditions. The bottom part of Figure 4 reports, on a log scale, the index of the job of τ_2 that experiences the maximum response time (the *critical job*). When the periods are in some harmonic relationship the critical job occurs relatively early. However, when the harmonic relationship is poor we often stop our computation because of our job limit at 1000 jobs.

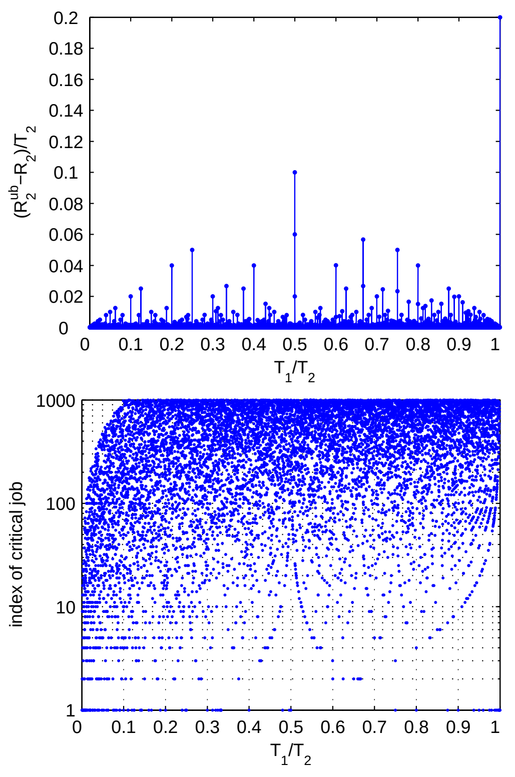

Figure 4. Response time bound, when $U = 1$

This observation motivated the third and last set of experiments exploring the influence of periods. We want to evaluate what the critical job is, when the periods are poorly harmonic. For this purpose, we set $\frac{T_2}{T_1} = \sqrt{2}$ so that the notion of hyperperiod doesn't exist (clearly on machine representable numbers, T_1 and T_2 are still rational.) We set the ratio $\frac{U_1}{U_2} = 0.25$ (meaning that the τ_1 has a significantly lower load than τ_2, although this setting did not seem to significantly affect the simulation results). The experiments are carried out varying the total utilisation in the proximity of $U = 1$. Again, we stopped the computation of response time after 10000 jobs. Figure 5 reports the index of the critical job in log scale.

It may be noticed clearly that as the total utilisation approaches 1 the index of the critical job progressively increases, until the computation is artificially interrupted at job 10000. Actually when the utilisation is exactly 1, we believe that *there always exists some future job with longer response time*. Observing this phenomenon has lead us to formulate the following conjecture.

Conjecture 1 *When $U = 1$ and the ratio $\frac{T_2}{T_1}$ is irrational then the index of the critical job is unbounded. Moreover*

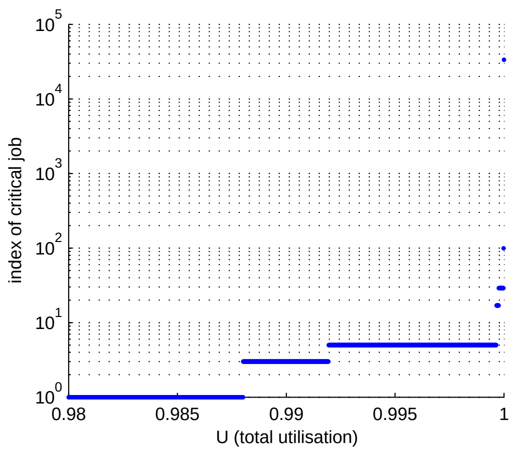

Figure 5. The index of the critical job.

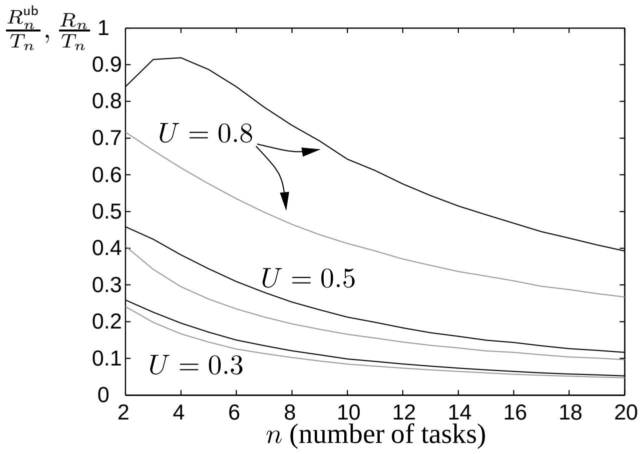

Figure 6. Response bound and tasks number

we have

$$\limsup_k R_{2,k} = R_2^{\mathsf{ub}} \tag{11}$$

where $R_{2,k}$ denotes the response time of the k^{th} job of task τ_2.

4.2. Effects of the number of tasks

In this set of experiments we focus on the influence of the number of tasks both on the actual response time and on the upper bound derived by us in Section 3. The number of tasks ranges from 2 to 20. The experiment is run under three different total load condition represented by $U = 0.3$ (light load), $U = 0.5$ (average load) and $U = 0.8$ (heavy load). The total load is uniformly distributed among the single tasks using the simulation routine suggested by Bini and Buttazzo [5]. Notice that as the number of tasks increases all the individual utilisations U_i tend to decrease because the total utilisation U is kept constant. The period T_1 of τ_1 is set equal to one, and the remaining periods are randomly selected such that T_{i+1}/T_i is uniformly distributed in $[1, 3]$. For each pair (number of tasks n, total utilisation U) we ran 10000 simulations and computed the normalised response time R_n/T_n — drawn in gray — and the normalised upper bound R_n^{ub}/T_n — in black. Figure 6 reports the average value of all the simulations. The figure shows three pairs of plots, relative to the three different values of utilisation simulated.

It may seem quite unexpected that the response times does not increase with the number of tasks. However, we must remember that we are plotting values normalised with the period T_n. To confirm the validity of the experiments we can compute the limit of the normalised response time, reported in Eq. (8), as n grows to infinity. In order to compute the limit we assume that all the tasks utilisations are the same (i.e., each is equal to U/n) and all the period ratios a_j are equal to a common value a. The asymptotic value of the response time then is

$$\begin{aligned}\lim_{n\to\infty} r_n^{\mathsf{ub}} &= \frac{\frac{U}{n} + (n-1)a\frac{U}{n}(1-\frac{U}{n})}{1-(n-1)\frac{U}{n}} \\ &= \frac{U + (n-1)a\,U(1-\frac{U}{n})}{n-(n-1)U} \\ &= \frac{U + (n-1)a\,U}{n-(n-1)U} \\ &= \frac{a\,U}{1-U}\end{aligned}$$

which is constant.

4.3. The sufficient test

In the final experiments we evaluated the number of tasks sets accepted by the sufficient test stated in Corollary 2. This test is compared with other simple sufficient tests: the Hyperbolic Bound [6] and the utilisation RBound [11]. We remind that the complexity of the test presented here and the Hyperbolic Bound in $O(n)$, whereas the complexity of the utilisation RBound is $O(n \log n)$, where n denotes the number of tasks.

First we investigated the effect of the period on the quality of the sufficient tests. We arbitrarily set the number of tasks equal to 5 and the total utilisation $U = 0.8$ so that the random task sets are not trivially schedulable. The periods are randomly extracted as follows: **(i)** T_1 is set equal to 1 and **(ii)** the other periods T_i are uniformly extracted in the interval $[T_{i-1}, r\,T_{i-1}]$. The parameter r, denoted by *period dispersion* in Figure 7, measures how close each other are the periods. For example if $r = 1$ then all the periods are the same, if r is large then the next random period tends to be large compared with the previous one. The experiments are conducted for r varying from 1 to 7, and for each setting we extracted 5000 task set. The quality of the tests is measured by the *acceptance ratio*, which is the percentage of schedulable task sets accepted by each of the three sets [5]. The results are shown in Figure 7.

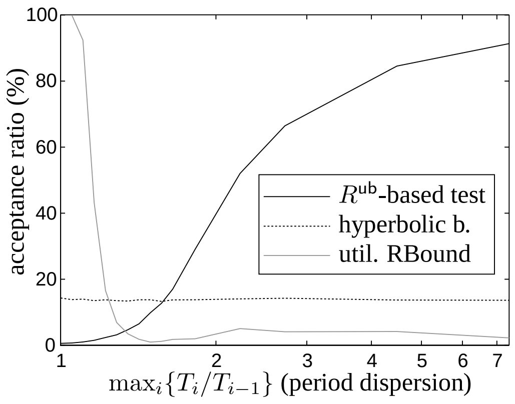

Figure 7. Acceptance ratio and periods

First, the figure confirms that the Hyperbolic Bound is not affected at all by the variation of the periods. In fact, this test is performed only on task utilisations which are left unchanged. Then we observe that when the periods are close each other (period dispersion close to 1) the RBound dominates, whereas for large periods the test based on the response time bound performs better than the others. The possible explanation is that the RBound is built starting from the Liu and Layland [15] worst-case periods which are all very close each other.

Finally, we evaluated the acceptance ratio as the number of tasks varies from 2 to 20. The total utilisation is equal to 0.75 so that a considerable number of task sets are schedulable also when the number of tasks is maximum. The period dispersion r, as defined previously, is set equal to 1.4 so that we work in an area where all the three tests seem comparable from Figure 7. The acceptance ratio is reported in Figure 8.

In this case the Hyperbolic Bound is always superior to the RBound, although this may happen because the period dispersion r is chosen too high. Anyhow, the most interesting aspect is that when the number of tasks grows beyond 8, the quality of the R^{ub}-based test starts increasing. This phenomenon is justified by observing that as the number of tasks grows, all the individual utilisations become smaller and smaller. Under this condition, as discussed previously, the workload upper bound — and the test based on it — is very tight.

5. Conclusions

Response time analysis (RTA) is an important approach to feasibility analysis of real-time systems that are scheduled using fixed-priority (FP) scheduling algorithms. Two drawbacks of RTA are: **(i)** computing response times takes time pseudo-polynomial in the representation of the task system; and **(ii)** response times are not in general continuous in task system parameters.

In this paper, we have derived an upper bound on the response times in sporadic task systems scheduled using FP algorithms. Our upper bound can be computed in polynomial time, and has the added benefit of being continuous and differentiable in the task system parameters. We have designed and conducted a series of simulation experiments to evaluate the goodness of our approach. These simulations have had the added benefit of giving rise to an interesting theoretical conjecture concerning response times for systems in which all parameters need not be rational numbers.

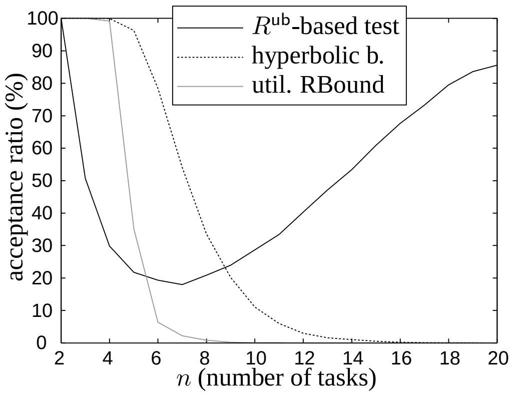

Figure 8. Acceptance ratio and tasks number

References

[1] Karsten Albers and Frank Slomka. An event stream driven approximation for the analysis of real-time systems. In *Proceedings of the* 16th *Euromicro Conference on Real-Time Systems*, pages 187–195, Catania, Italy, June 2004.

[2] Neil C. Audsley, Alan Burns, Mike Richardson, Ken W. Tindell, and Andy J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, September 1993.

[3] Sanjoy K. Baruah, Aloysius K. Mok, and Louis E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the* 11th *IEEE Real-Time Systems Symposium*, pages 182–190, Lake Buena Vista (FL), U.S.A., December 1990.

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

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

[6] Enrico Bini, Giorgio C. Buttazzo, and Giuseppe M. Buttazzo. Rate monotonic scheduling: The hyperbolic bound. *IEEE Transactions on Computers*, 52(7):933–942, July 2003.

[7] Reinder J. Bril, Wim F. J. Verhaegh, and Evert-Jan D. Pol. Initial values for on-line response time calculations. In *Proceedings of the* 15^{th} *Euromicro Conference on Real-Time Systems*, pages 13–22, Porto, Portugal, July 2003.

[8] Anton Cervin and Johan Eker. Control-scheduling codesign of real-time systems: The control server approach. *Journal of Embedded Computing*, 1(2):209–224, 2005.

[9] Nathan Fisher and Sanjoy Baruah. A fully polynomial-time approximation scheme for feasibility analysis in static-priority systems. In *Proceedings of the* 17^{th} *Euromicro Conference on Real-Time Systems*, pages 117–126, Palma de Mallorca, Spain, July 2005.

[10] Mathai Joseph and Paritosh K. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, October 1986.

[11] Sylvain Lauzac, Rami Melhem, and Daniel Mossé. An improved rate-monotonic admission control and its applcations. *IEEE Transactions on Computers*, 52(3):337–350, March 2003.

[12] John P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadline. In *Proceedings of the* 11^{th} *IEEE Real-Time Systems Symposium*, pages 201–209, Lake Buena Vista (FL), U.S.A., December 1990.

[13] John P. Lehoczky, Lui Sha, and Ye Ding. The rate-monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the* 10^{th} *IEEE Real-Time Systems Symposium*, pages 166–171, Santa Monica (CA), U.S.A., December 1989.

[14] Joseph Y.-T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic real-time tasks. *Performance Evaluation*, 2(4):237–250, December 1982.

[15] Chung Laung Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, January 1973.

[16] Wan-Chen Lu, Jen-Wei Hsieh, and Wei-Kuan Shih. A precise schedulability test algorithm for scheduling periodic tasks in real-time systems. In *Proceedings of the ACM Symposium on Applied Computing*, pages 1451–1455, Dijon, France, April 2006.

[17] Yoshifumi Manabe and Shigemi Aoyagi. A feasibility decision algorithm for rate monotonic and deadline monotonic scheduling. *Real-Time Systems*, 14(2):171–181, March 1998.

[18] Aloysius Ka-Lau Mok. *Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment*. PhD thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Boston (MA), U.S.A., May 1983.

[19] José Carlos Palencia and Michael González Harbour. Schedulability analysis for tasks with static and dynamic offsets. In *Proceedings of the* 19^{th} *IEEE Real-Time Systems Symposium*, pages 26–37, Madrid, Spain, December 1998.

[20] Pascal Richard and Joël Goossens. Approximating response times of static-priority tasks with release jitters. In 18^{th} *Euromicro Conference on Real-Time Systems, Work-in-Progress*, Dresden, Germany, July 2006.

[21] Mikael Sjödin and Hans Hansson. Improved response-time analysis calculations. In *Proceedings of the* 19^{th} *IEEE Real-Time Systems Symposium*, pages 399–408, Madrid, Spain, December 1998.

[22] Ken Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming*, 50:117–134, April 1994.

[23] Ken W. Tindell, Alan Burns, and Andy Wellings. An extendible approach for analysing fixed priority hard real-time tasks. *Journal of Real Time Systems*, 6(2):133–152, March 1994.