A Response Time Bound in Fixed-Priority Scheduling with Arbitrary Deadlines

Enrico Bini, Thi Huyen Châu Nguyen, Pascal Richard, Sanjoy K. Baruah

Abstract— **Since worst-case response times must be determined repeatedly during the interactive design of real-time application systems, repeated exact computation of such response times would slow down the design process considerably. In this research, we identify three desirable properties of estimates of the exact response times: *continuity* with respect to system parameters; *efficient computability*; and *approximability*. We derive a technique possessing these properties for estimating the worst-case response time of sporadic task systems that are scheduled using fixed priorities upon a preemptive uniprocessor.**

Index Terms— **fixed priority scheduler, response time, approximation schemes, resource augmentation.**

I. Introduction and Motivation

When a group of tasks share a common resource (such as a processor, a communication medium,...), a scheduling policy is necessary to arbitrate access to the shared resource. One of the most intuitive policies consists of assigning *Fixed Priorities* (FP) to the tasks, so that at each instant in time the resource is granted to the highest priority task requiring it at that instant. Depending on the assigned priority, a task can have longer or shorter *response time*, which is the time elapsed from request of the resource to the completion of the task.

Critical applications often require that the worst-case response times (i.e. the maximum possible response time) do not exceed a given deadline. Hence it is necessary to perform the *Response-Time Analysis* (RTA) [1], [2] to compute exactly the worst-case response time of each task in the task system.

Consider a system of n tasks $\tau_1, \tau_2, \ldots, \tau_n$, with the i^{th} task τ_i characterized by a worst-case execution requirement C_i, a relative deadline parameter D_i, and a minimum inter-arrival separation parameter T_i. Without loss of generality, assume that the tasks are indexed in decreasing order of priority. RTA allows for the exact computation of the worst-case response time of each task, by representing the response-time of each job of each task as the solution to a recurrence equation. E.g., if $D_i \leq T_i$ $(\forall i)$, it has been shown that the worst-case response time of τ_i is equal to the smallest t satisfying the following equality:

$$t = C_i + \sum_{j<i} \left\lceil \frac{t}{T_j} \right\rceil C_j \tag{1}$$

This equation is easily solved using standard techniques for the solution of recurrence equations, in time pseudo-polynomial in the representation of the task system.

Despite this pseudo-polynomial time complexity, RTA has very efficient implementations in practice that renders it quite suitable for feasibility analysis of Fixed Priority (FP) systems. However, there are certain real-time design scenarios during which there are drawbacks to using RTA:

E. Bini is with the Scuola Superiore Sant'Anna, Pisa, Italy.
T. H. C. Nguyen and P. Richard are with the University of Poitiers, France.
S. K. Baruah is with The University of North Carolina at Chapel Hill.

§*1:* The *computational complexity* of these algorithms may render them unsuitable for use in interactive real-time system design environments [3]. During a process of interactive system design and rapid system prototyping, the system designer typically makes a large number of calls to a feasibility-analysis algorithm, 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. (The computation of exact response times is especially expensive when the deadline D_i is allowed to be larger than the period T_i [4], [5] and/ or when resource-sharing may lead to limited priority-inversion and blocking; for such systems, several recurrences of the form Equation (1), one for each job of τ_i within the "first busy period," must be solved in order to compute the exact response time — see [6], [7] for details.)

§*2:* One of the features of the worst-case response time that is easily seen from Equation (1) above is that *it is not a continuous function of system parameters*. Informally speaking, this is a consequence of the ceiling function: for certain values of t an infinitesimally small increase in some T_j will cause $\lceil t/T_j \rceil$ to increase by one and the right-hand side of Equation (1) to consequently increase by an amount not smaller than C_j.

Discontinuities like these are a major hurdle to a process of incremental, interactive system design. Ideally, such a design process would allow for the interactive exploration of the state space of possible designs; this would be greatly facilitated if making minor changes to a design (equivalently, moving small distances in the state space of possible system designs) results in minor changes to system properties.

Now these discontinuities are unavoidable, since they are a feature of worst-case response time per se, and not just of the RTA technique for computing them. Nevertheless, we believe that there is some benefit to studying *upper bounds* on worst-case response times that do not suffer such discontinuities with system parameters. If we were to use these continuous upper bounds (rather than the discontinuous exact bounds) during the system design process, then we would know that neighboring points in the design state space would have similar response time bounds. Thus, for example, if we were at a point in the design state space where every task except one had its response-time bound well within acceptable limits, we could safely consider making small changes to the task parameters in order to search for neighboring points in the design state space in which the response time bound of the non-compliant task is decreased, without needing to worry that the response time of some currently compliant task would unexpectedly increase by a large amount.

§*3:* Any such continuous upper bound on response-time is *necessarily not tight* since, as stated above, worst-case response-time is itself not continuous in the system parameters. In other words, by choosing a continuous upper bound we would be trading off accuracy for continuity: it is desirable that this loss of accuracy be *quantified* in some manner in order that the system designer may determine whether the loss of accuracy is worth the benefits that continuous bounds provide to the system design process.

To summarize the points made above: we seek upper bounds on worst-case response time that are *continuous* functions of system parameters; are *efficiently computable*; and have *quantifiable deviation* from the exact bounds. In the remainder of this document, we derive such a bound and prove that it does indeed possess these properties.

A. 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. [8], has been improved by Manabe and Aoyagi [9] who reduced the number of points where the time demand needs to be checked. Bini and Buttazzo [10] 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 [11] proposed several lower bounds to the response time so that the original response time algorithm [1] could start further and the time spent in computing the response time is reduced. Mäki-Turja and Nolin instead refined the estimate of the amount of interference in the transactional task model [12].

Starting from the idea of Albers and Slomka [13], Fisher and Baruah [14] have derived a fully polynomial time approximation scheme (FPTAS) for RTA. Very recently, Richard et al. [15] have extended the task model of a previous FPTAS [14] to take into account release jitter. Bini and Baruah [16] proposed a continuous upper bound of the response time, although they did not show any property of the approximation scheme. Finally, Lu et al. [17] proposed a method to reduce the number of iterations for finding the task response times.

II. Task Model

In this paper, we assume that a real-time system may be modeled as a collection of n sporadic [18], [19] tasks $\tau_1, \tau_2, \ldots, \tau_n$. Each sporadic task τ_i is characterized by a worst-case execution requirement C_i, a relative deadline parameter D_i, and a minimum inter-arrival separation parameter T_i. We assume *arbitrary deadline*, meaning that we allow $D_i > T_i$. Task τ_i generates a potentially infinite sequence of jobs, with successive jobs arriving at least T_i time units apart, and each job needing to execute for at most C_i time units, by a deadline that occurs D_i time units after the job's arrival time. We will use the term *utilization* of τ_i (denoted by U_i), to represent the ratio C_i/T_i. U denotes the *system utilization*: $U = \sum_{i=1}^{n} U_i$. R_i denotes the worst-case response time of τ_i when this system is scheduled according to these fixed priorities. In the initial part of the paper we suppose that tasks are independent. In Section IV, we also account for the blocking times due to the mutual exclusive portions of tasks accessing shared resources.

We assume that priority-based scheduling is used: each job is (implicitly or explicitly) assigned a priority at each instant in time, and the processor is allocated to the highest-priority job that has not completed execution. We restrict our attention to *fixed-priority* (FP) scheduling [20] — there is a unique priority associated with each task, and all the jobs generated by a task are assigned this priority. An example FP scheduling algorithm is the Deadline-Monotonic (DM) scheduling algorithm [21], which assigns priorities to tasks in inverse order of their relative deadline parameters. Without loss of generality, we assume that the tasks are indexed in decreasing order of priority: task τ_i's jobs have priority over task τ_j's jobs for all i, j such that $1 \leq i < j \leq n$.

III. The Response Time Bound

In this section, we derive an upper bound (Theorem 1) on the worst-case response time under FP preemptive uniprocessor scheduling. We also demonstrate that this bound is efficiently computable in time linear in the number of tasks in the system.

We start with some notation and definitions.

Definition 1 (Worst-case workload): Let $W_i(t)$, the *worst-case workload* of the i highest priority tasks over an interval of length t, denote the maximum amount of time over any contiguous interval of length t during which some task in $\{\tau_1, \tau_2, \ldots, \tau_i\}$ is executing. As proved by Liu and Layland in their seminal paper [20], the worst-case workload $W_i(t)$ occurs when all the tasks $\tau_1, \ldots, \tau_i$ are simultaneously activated, and each τ_i generates subsequent jobs 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 "ceiling" expression $\sum_i \left\lceil \frac{t}{T_i} \right\rceil C_i$. The worst-case workload measures the amount 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 (Worst-case idle time): 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

$$H_i(t) = t - W_i(t) \tag{2}$$

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 $\underline{\text{pseudo}}$ inverse.

Based upon this definition of the inverse of the idle time, we obtain the following alternative representation of task response time.

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

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

Proof: Let R_{ik} be the response time of the k^{th} job of task τ_i. Let also k^* be the maximum index of τ_i jobs such that the processor always runs a task in the set $\{\tau_1, \dots, \tau_i\}$ during the interval $[0, R_{ik^*}]$ ($k^* = +\infty$ is also possible when $\sum_{j=1}^{i} U_i \geq 1$). Lehoczky [6] proved that the maximum task response time R_i is

$$R_i = \max_{k=1,\dots,k^*} \{X_{i-1}(k\,C_i) - (k-1)\,T_i\}. \tag{4}$$

If $k^* = +\infty$ then Eq. (4) is the same as Eq. (3) and the Lemma is proved. If $k^* < +\infty$, we study the response time of the k^{th} job of τ_i, with $k > k^*$. Let $I \geq 0$ be the amount of time in $[0, R_{ik}]$ during which the processor is not running a task in $\{\tau_1, \dots, \tau_i\}$. Then for all $k > k^*$ we have

$$R_{ik} = X_{i-1}(k\,C_i + I) - (k-1)\,T_i \geq X_{i-1}(k\,C_i) - (k-1)\,T_i$$

because $X_{i-1}(c)$ is a non-decreasing function. Finally we have that

$$R_i \geq \max_{k>k^*}\{R_{ik}\} \geq \max_{k>k^*}\{X_{i-1}(k\,C_i) - (k-1)\,T_i\}$$

from which the theorem follows. ■

Notice that if $\sum_{j=1}^{i} U_i > 1$ then the we clearly have $R_i = +\infty$; i.e., the worst-case response time R_i of task τ_i is unbounded. For this reason 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 function, and $f^{\mathsf{lb}}(x)$ denotes a lower bound function, on the function $f(x)$: we have

$$\forall\ x\ \ ::\ f^{\mathsf{lb}}(x) \leq f(x) \leq f^{\mathsf{ub}}(x)\ .$$

Lemma 2: 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

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

from which it follows that

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

as required. ■

Lemma 2 is quite powerful, because it allows to derive a response time upper bound starting from any upper bound of the workload $W_i(t)$. However the problem in applying the Lemma lies in the difficulty of finding a workload upper bound which can be easily inverted. In the rest of this section we will exploit Lemma 2 by defining a linear upper bound $W_i^{\mathsf{ub}}(t)$.

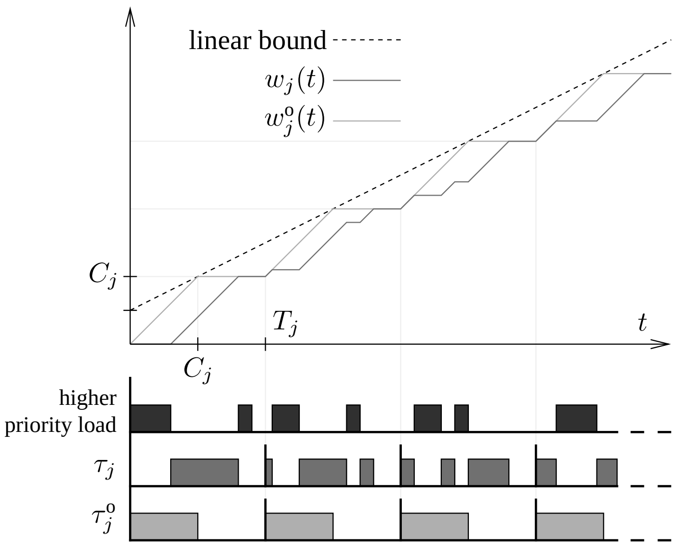

Fig. 1. The upper linear bound of $w_j(t)$

As stated above, it was proved by Liu and Layland [20] that the worst-case workload $W_i(t)$ occurs for the synchronous arrival sequence of jobs. Hence the function $W_i(t)$ may be expressed as a sum of the contributions of each individual task τ_j, $1 \leq j \leq i$, to this workload in the synchronous arrival sequence. I.e., 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 the synchronous arrival sequence, we can write: $W_i(t) = \sum_{j=1}^{i} w_j(t)$.

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:

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

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 1. The equation of the linear bound is as follows:

$$w_j^o(t) \leq U_j\,t + C_j(1 - U_j). \tag{5}$$

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 $W_i^{\mathsf{ub}}(t)$ of the workload function:

$$\begin{aligned} W_i(t) = \sum_{j=1}^{i} w_j(t) &\leq \sum_{j=1}^{i} w_j^o(t) \\ &\leq \sum_{j=1}^{i} \big(U_j\,t + C_j\,(1 - U_j)\big) = W_i^{\mathsf{ub}}(t) \end{aligned} \tag{6}$$

We have thus 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 Lemma 2 to finally find the bound of the response time.

Theorem 1: The worst-case response time R_i of task τ_i is bounded from above as follows:

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

Proof: The proof of this theorem is obtained by applying Lemma 2 to the workload bound provided by the Eq. (6). 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:

$$\max_{k=1,2,\ldots}\left(\frac{kC_i + \sum_{j=1}^{i-1} C_j(1-U_j)}{1-\sum_{j=1}^{i-1} U_j} - (k-1)T_i\right) \tag{8}$$

We will now prove that the maximum in the Eq. (8) 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. (8), we get:

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

as required. ■

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

$$r_i^{\mathsf{ub}} = \frac{R_i^{\mathsf{ub}}}{T_i} = \frac{U_i + \sum_{j<i} a_j\, U_j(1-U_j)}{1-\sum_{j<i} U_j} \tag{9}$$

where $a_j = T_j/T_i$.

Computational complexity.: It is evident from Equation (9) above that the computational complexity of determining the response time upper bound R_i^{ub} of task τ_i is $\mathcal{O}(i)$. Hence the complexity of computing the bound for all the tasks would appear to be $\mathcal{O}(n^2)$. However, one can do better by noticing that the computation of R_{i+1}^{ub} may take advantage of the completed computation of R_i^{ub}. In fact the two sums involved in Equation (7) 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 conclude the response time upper bound of all the tasks may be determined with a time complexity $\mathcal{O}(n)$.

Comparison to other bounds.: There are other techniques to bound the response time. For instance, an efficiently-computable continuous upper bound on response time is easily derived from the technique of [11] (this technique was originally proposed in [11] for computing a lower bound):

$$R_i = C_i + \sum_{j<i}\left\lceil \frac{R_i}{T_j}\right\rceil C_j$$

$$R_i \le C_i + \sum_{j<i}\left(\frac{R_i}{T_j} + 1\right) C_j$$

$$R_i - R_i \sum_{j<i} U_j \le C_i + \sum_{j<i} C_j$$

$$R_i \le \frac{\sum_{j\le i} C_j}{1-\sum_{j<i} U_j} \tag{10}$$

Observe that this is a looser bound than the one we have obtained above, in Theorem 1 — the numerator of the expression in Theorem 1 is smaller than the numerator above, by an amount $\sum_{j<i} U_j C_j$.

Schedulability testing.: 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:

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

$$\forall i \quad R_i^{\mathsf{ub}} = \frac{C_i + \sum_{j=1}^{i-1} C_j(1-U_j)}{1-\sum_{j=1}^{i-1} U_j} \le D_i \tag{11}$$

Corollary 1 provides a very efficient means for testing the feasibility of task sets. This condition can also be restated as a utilization upper bound, and compared with many existing schedulability tests [20], [6], [22], [23]. 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**:

$$\forall i \quad \sum_{j=1}^{i} U_j \le 1 - \sum_{j=1}^{i-1} a_j\, U_j(1-U_j) \tag{12}$$

where $a_j = T_j/T_i$.

Proof: From Equation (11) it follows that the task τ_i is schedulable if

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

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

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

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

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. (12) may also become negative, making the condition impossible.

IV. Accounting for blocking times

In the presence of shared resources, it may happen that a task τ_i is blocked by a semaphore which is locked by a lower priority task. In this circumstance τ_i suffers an extra amount of delay, called *blocking time* B_i, required to wait for the permission to use the shared resources. Several techniques have been proposed to handle the shared resources [24], [25].

The entire level-i busy interval may only be blocked once on each shared resource, by some lower-priority job that already had a lock at the critical instant. Once the busy interval starts, no lower-priority job can possibly acquire a lock.

Hence the response time of the task τ_i can be computed by increasing the computation time of the first job by B_i as follows

$$R_i = \max_{k=1,2,\ldots} \{X_{i-1}(B_i + k\,C_i) - (k-1)\,T_i\} \tag{13}$$

If we replace the linear upper bound of the pseudo inverse as done in Theorem 1 then we find that the maximum of Eq. (13) is assumed again when $k = 1$. It follows that the response time upper bound becomes

$$R_i^{\mathsf{ub}} = \frac{C_i + B_i + \sum_{j=1}^{i-1} C_j(1-U_j)}{1 - \sum_{j=1}^{i-1} U_j} \tag{14}$$

V. Approximation results

Exact response-time bounds are necessarily not continuous in task parameters; furthermore, the computational complexity of determining such bounds is currently unknown (and suspected to not be in polynomial time unless P = NP). Our upper bound R_i^{ub}, however, is continuous and is efficiently computable in time linear in the number of tasks. The question needs to be asked: *how much accuracy are we sacrificing in order to obtain a bound that possesses these desirable properties?* In this section, we attempt to answer this question in a quantitative manner.

A. Approximation Ratio analysis

The approximation quality can be measured with the classical *approximation ratio*. Let a be the value obtained by an algorithm A computing upper bounds of worst-case response times and opt be the exact worst-case response time; we say that the algorithm A has an approximation ratio c, where $c \geq 1$, when $opt \leq a \leq c \times opt$, for all inputs of A (if such a c does not exists, the algorithm A is said to have no approximation ratio).

The next result states that the upper bound R_i^{ub} does not, in fact, have an approximation ratio:

Theorem 2: The upper bound R_i^{ub} of Theorem 1 does not have an approximation ratio.

Proof: We prove this by demonstrating a task system and a task τ_i for which R_i^{ub}/R_i tends to ∞. All tasks in our system will have $D_i = T_i$; hence, let us represent the parameters of a task τ_i by an ordered pair (C_i, T_i). Consider the following task set: $\tau_1 = (K, 2K+\epsilon)$, $\tau_2 = (K, 2K+\epsilon)$ and $\tau_3 = (\epsilon, 2K+\epsilon)$, where ϵ is an arbitrarily small positive number and K is an arbitrary

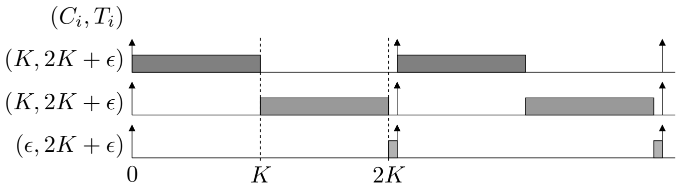

Fig. 2. Example with no approximation ratio.

number greater than ϵ. In Figure 2 we depict the synchronous arrival sequence of this task set.

The system utilization $U_1 + U_2 + U_3$ is 1. Since tasks have all periods equal to $2K+\epsilon$, then $R_3 = 2K+\epsilon$. The upper bound of Eq. (7) is:

$$\begin{aligned} R_i^{\mathsf{ub}} &= \frac{\epsilon + 2K(1 - \frac{K}{2K+\epsilon})}{1 - \frac{2K}{2K+\epsilon}} \\ &= \frac{\epsilon(2K+\epsilon) + 2K(2K+\epsilon-K)}{2K+\epsilon-2K} \\ &= \frac{(2K+\epsilon)\epsilon + 2K(K+\epsilon)}{\epsilon} \\ &= 4K + \epsilon + \frac{2K^2}{\epsilon} \end{aligned}$$

Thus we have,

$$\lim_{\epsilon \to 0} R_i^{\mathsf{ub}} = \lim_{\epsilon \to 0} \left(4K + \epsilon + \frac{2K^2}{\epsilon}\right) = \infty$$

However $R_i = 2K + \epsilon$ for all $\epsilon > 0$, then the ratio R_i^{ub}/R_i also tends to infinity and the theorem is proved. ■

B. Resource Augmentation analysis

Theorem 2 above reveals that the upper bound R_i^{ub} on response-time Theorem 1 does not offer any quantifiable performance guarantee, according to the conventional approximation ratio measure that is used in optimization theory. However, an alternative approach towards approximate analysis is the technique of *resource augmentation*, which is becoming increasingly popular in real-time scheduling theory. In this technique, the performance of the algorithm under analysis is compared with that of an optimal algorithm *that runs on a slower processor.* In this section, we apply this resource augmentation technique to quantify the deviation of R_i^{ub} from optimality.

In deriving the upper bound R_i^{ub} above, recall that we had upper-bounded the worst-case workload $W_i(t)$ (Definition 1) by the sum $\sum_{j=1}^{i} w_j^o(t)$, and furthermore bounded $w_j^o(t)$ as follows, in Equation (5):

$$w_j^o(t) \leq U_j\,t + C_j(1-U_j)\ .$$

Let us introduce the following notation:

$$\textsc{la}(\tau_j, t) \overset{\text{def}}{=} U_j\,t + C_j(1-U_j) \tag{15}$$

An upper bound on amount by which $\textsc{la}(\tau_j, t)$ deviates from $w_j(t)$ can be obtained:

Lemma 3:

$$\forall t \geq C_j \qquad \textsc{la}(\tau_j, t) \leq 2\left\lceil \frac{t}{T_j} \right\rceil C_j\ .$$

Proof Sketch: We first prove that $\forall t \geq C_j$ we have $\textsc{la}(\tau_j, t) \leq 2\,w_j^o(t)$. This is visually evident from Figure 1, in which the linear bound depicts $\textsc{la}(\tau_j, t)$. For $t \geq C_j$, it is clear that the ratio $\text{LA}(\tau_j,t)\ /\ w_j^o(t)$ is maximized at $t=T_j$; for this value of t,

$$\text{LA}(\tau_j,T_j) = \frac{C_j}{T_j}T_j + C_j(1-U_j)$$
$$\text{LA}(\tau_j,T_j) = 2C_j - C_jU_j$$

while $w_j^o(T_j) = C_j$. Lastly, $w_j^o(t) \le \left\lceil \frac{t}{T_j} \right\rceil C_j$ directly follows from $w_j^o(t)$ definition. The lemma is proved. ■

Lemma 4: Bound R_i^{ub} is equal to the smallest value of t satisfying the following recurrence[1]:

$$t = C_i + \sum_{j<i}\text{LA}(\tau_j,t) \tag{16}$$

Proof: First, we show that R_i^{ub} is a solution to this recurrence. By definition of R_i^{ub} of Eq. (7), we have

$$R_i^{\mathsf{ub}} = \frac{C_i + \sum_{j<i} C_j(1-U_j)}{1-\sum_{j<i}U_i}$$
$$R_i^{\mathsf{ub}}(1-\sum_{j<i}U_i) = C_i + \sum_{j<i}C_j(1-U_j)$$
$$R_i^{\mathsf{ub}} = C_i + \sum_{j<i}U_iR_i^{\mathsf{ub}} + \sum_{j<i}C_j(1-U_j)$$
$$R_i^{\mathsf{ub}} = C_i + \sum_{j<i}\text{LA}(\tau_j,R_i^{\mathsf{ub}})$$

thereby implying that R_i^{ub} satisfies recurrence of Eq. (16).

To see that R_i^{ub} is the *smallest* value of t satisfying Equation (16), observe that the rate of change of the right hand side (RHS) of the recurrence in Eq. (16) with respect to t is $(\sum_{j<i}U_j)$. By assumption, this is <1; consequently, the RHS of recurrence 16 grows no faster than the LHS. Since RHS=LHS at $t=R_i^{\mathsf{ub}}$, it must be the case that RHS>LHS for all $t<R_i^{\mathsf{ub}}$. Hence the lemma. ■

We can use Lemma 4 to prove a resource-augmentation bound on R_i^{ub}, as follows. Since R_i^{ub} is the smallest value of t to satisfy Equation (16) above, it must be the case that for all $t<R_i^{\mathsf{ub}}$

$$t < C_i + \sum_{j<i}\text{LA}(\tau_j,t)$$
$$t < C_i + 2\sum_{j<i}\left\lceil\frac{t}{T_j}\right\rceil C_j \quad \text{(using Lemma 3)}$$
$$t < 2\left(C_i + \sum_{j<i}\left\lceil\frac{t}{T_j}\right\rceil C_j\right)$$
$$\frac{1}{2}t < C_i + \sum_{j<i}\left\lceil\frac{t}{T_j}\right\rceil C_j$$

That is, the cumulative workload of jobs with priority greater than or equal to τ_i's that must be scheduled over the interval $[0,t)$ prior to the completion of task τ_i's first job in the critical instant exceeds the total capacity of a processor with computing capacity $1/2$ over the same interval. Hence no such $t<R_i^{\mathsf{ub}}$ can represent the worst-case response time of τ_i upon a processor of computing capacity one-half. Theorem 3 follows:

Theorem 3: The bound R_i^{ub} (Equation (7)) is

1) An upper bound on the worst-case response time of τ_i; and
2) A lower bound on the worst-case response time of τ_i *if the system is implemented upon a processor of speed one-half*.

We now show through a simple example that the resource augmentation bound is tight. Consider the set of K tasks where $\tau_i = (K+1, 2K^2)$, $\forall i = 1,\ldots,K-1$, and $\tau_K = (1,2K^2)$. In a processor running at half of the speed, the computation times are doubled and the system utilization is $U = \frac{2(K-1)(K+1)+2}{2K^2} = 1$. Since the periods are all equal to $2K^2$, the response time of τ_K on the half-speed processor (denoted by $\hat{R}_K$) is $2K^2$, because

$$2\,C_K + \sum_{j=1}^{K-1}\left\lceil\frac{\hat{R}_K}{T_j}\right\rceil 2\,C_j = 2 + (K-1)\left\lceil\frac{2K^2}{2K^2}\right\rceil 2(K+1) =$$
$$= 2(1+(K-1)(K+1)) = 2K^2 = \hat{R}_K$$

Let us now compute the ratio between the response time upper bound R_K^{ub} and the exact response time $\hat{R}_K$ of τ_K on the half-speed processor.

$$\lim_{K\to+\infty}\frac{R_K^{\mathsf{ub}}}{\hat{R}_K} = \frac{1+(K-1)(K+1)\left(1-\frac{K+1}{2K^2}\right)}{2K^2\left(1-(K-1)\frac{K+1}{2K^2}\right)}$$
$$= \frac{2K^2+(K^2-1)(2K^2-K-1)}{2K^2(K^2+1)} = 1$$

which means that as K grows we can build sets of tasks whose augmentation ratio is arbitrarily close to $1/2$.

How is the systems designer to interpret Theorem 3 above? First, it is guaranteed that R_i^{ub} is indeed an upper bound on R_i; hence, it is a safe estimate of the exact worst response time. And while Theorem 3 is unable to bound the amount by which R_i^{ub} exceeds the actual value of R_i (indeed, Section V has shown that there can, in general, be no such bound), it does assure the designer that [s]he could have obtained a worst-case response time no better than R_i^{ub} if the system had instead been implemented upon a processor half as fast. Stated differently, *a processor speedup of two is an upper bound on the price being paid for using an efficiently computable upper bound on response time that is a continuous function of the system parameters.*

VI. Experiments

In the previous section, we established the worst-case performance guarantee of R_i^{ub} from two different approximation models: the conventional approximation ratio and the resource augmentation metric. In this section, we describe numerical experimentations that we performed in order to capture its average performance guarantee.

Approximation ratio: The task set have been randomly generated as follows. Periods were randomly drawn from the interval $[1,2500]$ and deadlines from the interval $[1,2600]$. For each value of utilization factor of the task set, the task utilizations are generated uniformly using the UUniFast algorithm [26]. The worst-case execution requirements are computed accordingly. For a given problem size, the simulation has been replicated 400 times.

We compared the bound of Equation (7) and the bound of Equation (10) with the exact worst-case response time, which we computed in pseudo-polynomial time using the exact algorithm. In Figure 3 we report the error $\frac{R_i^{\mathsf{ub}}-R_i}{R_i}$ of the two bounds, when the number of tasks n is set equal to 20 (the amount of error of Eq. (10) with respect to Eq. (7) does not change for a

[1] In fact, R_i^{ub} happens to be the *only* value of t satisfying this recurrence; however, this fact is not germane to our discussion.

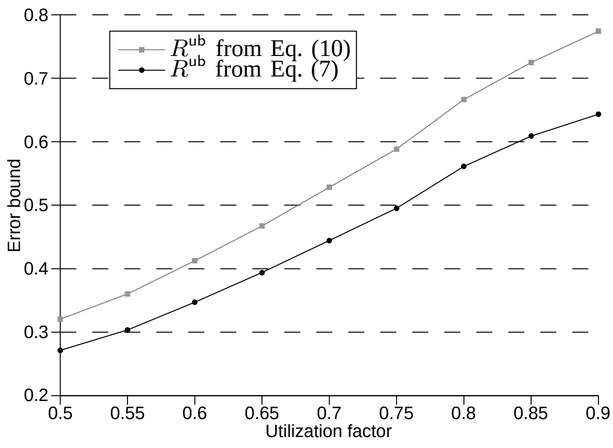

Fig. 3. Comparison of linear bounds.

different number of tasks). It can be noticed that the response time upper bound derived in this paper (Equation (7)) shows an approximately 5% improvement over the previous linear bound (Equation (10)).

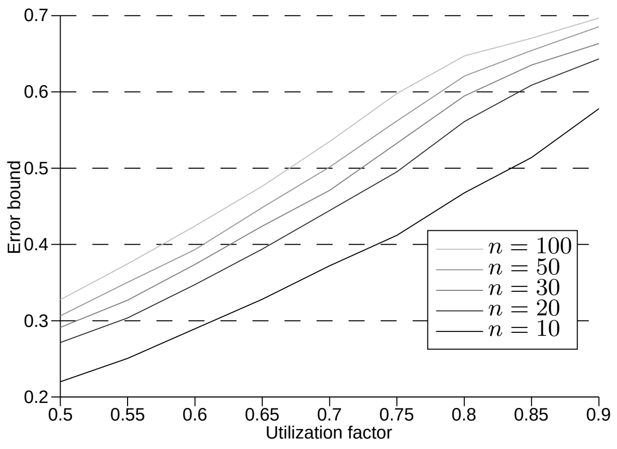

Fig. 4. Error of the response bound.

In Figure 4 we plot the error of the response time upper bound with respect to the task set utilization. The error is plotted for different number of tasks. It can be noticed that the error increases for high utilizations and for a large number of tasks.

Resource augmentation analysis: For every task set, we computed (using a binary search) the exact slowdown factor s so that the response time upper bound R_i^{ub} becomes the actual response time upon an s-speed processor. We monitored the average slowdown factor according to the utilization factor. The same experiments is done for several number of tasks n. Simulation results are presented in Figure 5. It can be noticed that the slowdown factor is mainly related to the utilization factor rather than the number of tasks in the system. As expected, the slowdown factors is always between $\frac{1}{2}$ and 1.

Feasible task sets: In the final experiments we evaluated the number of tasks sets accepted by the sufficient test stated in Corollary 2.

In Figure 6 we plot the percentage of accepted tasks for increasing values of utilization U. The same experiment is replicated for a different number of tasks. From the result we can notice that the proposed linear time test can indeed accept a considerable number of feasible task sets having utilization higher than the asymptotic utiization least upper bound [20] (which is $\log 2 = 0.693$).

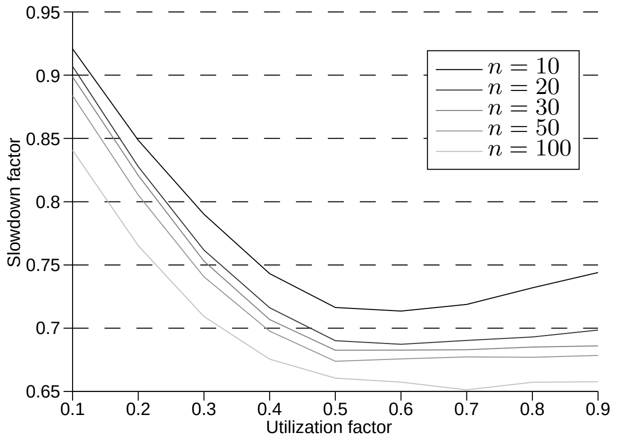

Fig. 5. Average slowdown factor s.

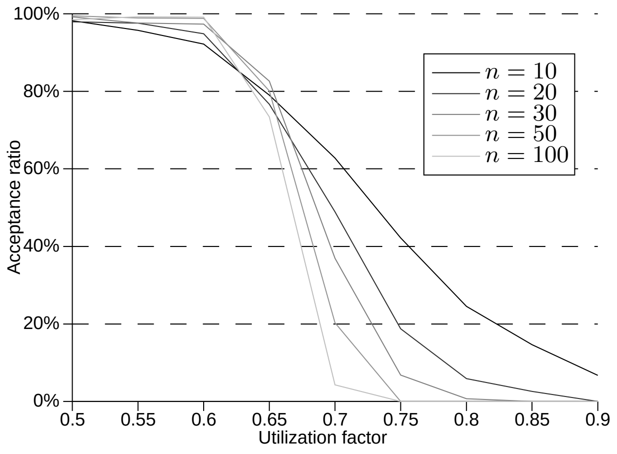

Fig. 6. Acceptance ratio for increasing U.

VII. Conclusions

We have argued that *continuity* with respect to system parameters is a very desirable property of response-time bounds, if such bounds are to be used as an integral part of an incremental, interactive, design process. Accordingly, we have derived a continuous upper bound on response time for task systems that are scheduled upon a preemptive uniprocessor using the Deadline Monotonic scheduling algorithm.

Such continuity necessarily comes with a loss of accuracy; in this work, we have attempted to quantify this loss of accuracy. We have shown that while our upper bound does not offer non-trivial performance guarantees according to conventional metrics, performance guarantees can nevertheless be obtained using the concept of resource augmentation. Specifically We demonstrated that our bound offers the following quantitative guarantee — it is indeed an upper bound on the exact response time, and the exact response time would necessarily be at least as large as this bound if the system were instead implemented upon a processor that is at most only one-half times as fast.

References

[1] M. Joseph and P. K. Pandya, "Finding response times in a real-time system," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, Oct. 1986.

[2] N. C. Audsley, A. Burns, M. Richardson, K. W. Tindell, and A. J. Wellings, “Applying new scheduling theory to static priority pre-emptive scheduling,” *Software Engineering Journal*, vol. 8, no. 5, pp. 284–292, Sept. 1993.

[3] R. Racu, A. Hamann, and R. Ernst, “A formal approach to multi-dimensional sensitivity analysis of embedded real-time systems,” in *Proceedings of the* 18th *Euromicro Conference on Real-Time Systems*, Dresden, Germany, July 2006, pp. 3–12.

[4] K. W. Tindell, A. Burns, and A. Wellings, “An extendible approach for analysing fixed priority hard real-time tasks,” *Real Time Systems*, vol. 6, no. 2, pp. 133–152, Mar. 1994.

[5] J. C. Palencia and M. González Harbour, “Schedulability analysis for tasks with static and dynamic offsets,” in *Proceedings of the* 19th *IEEE Real-Time Systems Symposium*, Madrid, Spain, Dec. 1998, pp. 26–37.

[6] J. P. Lehoczky, “Fixed priority scheduling of periodic task sets with arbitrary deadline,” in *Proceedings of the* 11th *IEEE Real-Time Systems Symposium*, Lake Buena Vista (FL), U.S.A., Dec. 1990, pp. 201–209.

[7] R. L. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, “Controller area network (CAN) schedulability analysis: Refuted, revisited, and revised,” *Real-Time Systems*, vol. 35, pp. 239–272, 2007.

[8] J. P. Lehoczky, L. Sha, and Y. Ding, “The rate-monotonic scheduling algorithm: Exact characterization and average case behavior,” in *Proceedings of the* 10th *IEEE Real-Time Systems Symposium*, Santa Monica (CA), U.S.A., Dec. 1989, pp. 166–171.

[9] Y. Manabe and S. Aoyagi, “A feasibility decision algorithm for rate monotonic and deadline monotonic scheduling,” *Real-Time Systems*, vol. 14, no. 2, pp. 171–181, Mar. 1998.

[10] E. Bini and G. C. Buttazzo, “Schedulability analysis of periodic fixed priority systems,” *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1462–1473, Nov. 2004.

[11] M. Sjödin and H. Hansson, “Improved response-time analysis calculations,” in *Proceedings of the* 19th *IEEE Real-Time Systems Symposium*, Madrid, Spain, Dec. 1998, pp. 399–408.

[12] J. Mäki-Turja and M. Nolin, “Efficient implementation of tight response-times for tasks with offsets,” *Real Time System Journal*, Feb. 2008.

[13] K. Albers and F. Slomka, “An event stream driven approximation for the analysis of real-time systems,” in *Proceedings of the* 16th *Euromicro Conference on Real-Time Systems*, Catania, Italy, June 2004, pp. 187–195.

[14] N. Fisher and S. Baruah, “A fully polynomial-time approximation scheme for feasibility analysis in static-priority systems,” in *Proceedings of the* 17th *Euromicro Conference on Real-Time Systems*, Palma de Mallorca, Spain, July 2005, pp. 117–126.

[15] P. Richard, J. Goossens, and N. Fisher, “Approximate feasibility analysis and response-time bounds of static-priority tasks with release jitters,” in 15th *International Conference on Real-Time and Network Systems*, Nancy, France, Mar. 2007.

[16] E. Bini and S. K. Baruah, “Efficient computation of response time bounds under fixed-priority scheduling,” in 15th *International Conference on Real-Time and Network Systems*, Nancy, France, Mar. 2007.

[17] W.-C. Lu, J.-W. Hsieh, and W.-K. Shih, “A precise schedulability test algorithm for scheduling periodic tasks in real-time systems,” in *Proceedings of the 2006 ACM symposium on Applied computing*, Dijon, France, Apr. 2006, pp. 1451–1455.

[18] A. K.-L. Mok, “Fundamental design problems of distributed systems for the hard-real-time environment,” Ph.D. dissertation, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Boston (MA), U.S.A., May 1983.

[19] S. K. Baruah, A. K. Mok, and L. E. Rosier, “Preemptively scheduling hard-real-time sporadic tasks on one processor,” in *Proceedings of the* 11th *IEEE Real-Time Systems Symposium*, Lake Buena Vista (FL), U.S.A., Dec. 1990, pp. 182–190.

[20] C. L. Liu and J. W. Layland, “Scheduling algorithms for multiprogramming in a hard real-time environment,” *Journal of the Association for Computing Machinery*, vol. 20, no. 1, pp. 46–61, Jan. 1973.

[21] J. Y.-T. Leung and J. Whitehead, “On the complexity of fixed-priority scheduling of periodic real-time tasks,” *Performance Evaluation*, vol. 2, no. 4, pp. 237–250, Dec. 1982.

[22] S. Lauzac, R. Melhem, and D. Mossé, “An improved rate-monotonic admission control and its applications,” *IEEE Transactions on Computers*, vol. 52, no. 3, pp. 337–350, Mar. 2003.

[23] E. Bini, G. C. Buttazzo, and G. M. Buttazzo, “Rate monotonic scheduling: The hyperbolic bound,” *IEEE Transactions on Computers*, vol. 52, no. 7, pp. 933–942, July 2003.

[24] L. Sha, R. Rajkumar, and J. P. Lehoczky, “Priority inheritance protocols: An approach to real-time synchronization,” *IEEE Transactions on Computers*, vol. 39, no. 9, pp. 1175–1185, Sept. 1990.

[25] T. P. Baker, “Stack-based scheduling of real-time processes,” *Real-Time Systems*, vol. 3, no. 1, pp. 67–100, 1991.

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

PLACE PHOTO HERE

Enrico Bini is assistant professor at the Scuola Superiore Sant’Anna in Pisa. He received the PhD in Computer Engineering from the same institution in October 2004. His research interests cover scheduling algorithms, real-time operating systems, and sensitivity analysis in embedded systems design.

PLACE PHOTO HERE

Thi Huyen Châu Nguyen received the BS degree in computer science from University of Technology, Hanoi National University. She currently works toward the PhD degree in the Laboratory of Applied Computer Science at the University of Poitiers, France. Her research interests include real-time scheduling and approximation algorithms.

PLACE PHOTO HERE

Pascal Richard is a professor in computer science at the University of Poitiers, France. He received the PhD degree in computer science from the University of Tours (France) in 1997. His research interests include real-time systems, scheduling theory, on-line algorithms and combinatorial optimization.

PLACE PHOTO HERE

Sanjoy Baruah is a professor in the Department of Computer Science at the University of North Carolina at Chapel Hill. He received his Ph.D. from the University of Texas at Austin in 1993. His research and teaching interests are in scheduling theory, real-time and safety-critical system design, and resource-allocation and sharing in distributed computing environments.