Pessimism in the Stochastic Analysis of Real-Time Systems: Concept and Applications

José Luis Díaz† José María López† Manuel García† Antonio Manuel Campos†
Kanghee Kim¶ Lucia Lo Bello§

Abstract

The exact stochastic analysis of most real-time systems is becoming unffordable in current practice. On one side, the exact calculation of the response time distribution of the tasks is not possible except for simple periodic and independent task sets. On the other side, in practice, tasks introduce complexities like release jitter, blocking in shared resources, stochastic dependencies, etc, which can not be handled by the periodic and independent task set model.

This paper introduces the concept of pessimism in the stochastic analysis of real-time systems in the following sense: the exact probability of missing any deadline is always lower than that derived from the pessimistic analysis. Therefore, if real-time constraints are expressed as probabilities of missing deadlines, the pessimistic stochastic analysis provides safe results.

Some applications of the pessimism concept are presented. Firstly, the practical problems that arise in the stochastic analysis of periodic and independent task sets are addressed. Secondly, we extend to the stochastic case some well known techniques of the deterministic analysis, such as the blocking in shared resources, and the task priority assignment.

1. Introduction

Traditional techniques of real-time analysis, such as the processor utilization analysis [12, 16] and response time analysis [20], assume single-valued execution times of the tasks. However, in practice execution times are not single-valued, so techniques of analysis based on single-valued execution times usually consider only the worst-case execution time. This introduces a great degree of pessimism in the analysis, giving rise to oversized real-time systems.

To overcome the problem, the execution times can be modelled as random variables, which may be obtained by measurement, or using hybrid techniques such as the ones described in [5] . Some approaches to this problem require a special scheduling model that provides isolation between tasks, so that each task can be analyzed independently of other tasks in the system [1, 2]. Other methods use common scheduling algorithms but introduce worst-case assumptions (e.g., the critical instant assumption [10, 11, 19]), restrictive load conditions (e.g., the heavy traffic condition in the Real-Time Queueing Theory [13, 14]), or restrictions on maximum system utilization and preemption [17] to simplify the analysis. A less restrictive approach was proposed in [7], in which we perform an analysis of periodic and independent tasks sets without assuming any worst-case or restrictive conditions. An interesting property of this analysis is the capacity to deal with systems with maximum system utilization higher than one, whenever the average system utilization remains lower than one. However, the analysis presented in [7] is not exempt from practical problems, as we will show in Section 5.

In order to overcome the problems and limitations of the analysis proposed in [7], the concept of pessimism in the stochastic analysis of real-time systems is introduced in this paper. Pessimism is not new in the real-time systems theory; it is everywhere in the deterministic analysis. For example, worst-case execution times are pessimistic execution times, blocking times and release jitter are also worst-case values, sufficient schedulability conditions are calculated from pessimistic task sets, etc. The concept of stochastic pessimism is theoretically developed in this paper, leaving the practical evaluation for future works.

The rest of the paper is organized as follows. Section 2 describes the system model. Section 3 summarizes the stochastic analysis presented in [7]. Section 4 for-

† José L. Díaz, José M. López, Manuel García and Antonio M. Campos ({jdiaz,chechu,manuel,campos}@atc.uniovi.es), *Depto. de Informática, Universidad de Oviedo* (33204, Gijón, Spain)

¶ Kanghee Kim (khkim@archi.snu.ac.kr) *School of Computer Science and Engineering, Seoul National University* (Shillim-dong San 56-1, Kwanak-Gu, Seoul, 151-742 Korea)

§ Lucia Lo Bello (llobello@diit.unict.it), *Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, Facoltà di Ingegneria, Università di Catania* (Viale A. Doria 6, 95125 Catania, Italy)

malizes the concept of pessimism as an ordering among random variables, and shows some general properties of this ordering. The implications of these properties for the stochastic analysis are also discussed. Section 5 presents the first application of pessimism in the stochastic analysis; the practical problems in the stochastic analysis of periodic and independent task sets are solved. Section 6.1 extends the stochastic analysis to deal with tasks that can be blocked in shared resources. The extension is based on calculating a pessimistic blocking time distribution for each task. Section 6.2 proves that the deterministic optimal algorithm for priority assignment [3] is also applicable to the stochastic case. Again, the proof is possible thanks to the concept of pessimism in the stochastic analysis. Finally, Section 7 presents our conclusions and future work.

2. System model

The system is composed of a set of N independent periodic tasks $S = \{\tau_1, \ldots, \tau_i, \ldots, \tau_N\}$, each task τ_i being defined by the tuple $(T_i, \Phi_i, \mathcal{C}_i, D_i, M_i)$, where T_i is the period of the task, Φ_i its initial phase, $\mathcal{C}_i$ its execution time and the pair (D_i, M_i) define the real-time constraint of the task.

The execution time is a discrete random variable[1] with a known probability function (PF), denoted by $f_{\mathcal{C}_i}(\cdot)$, where $f_{\mathcal{C}_i}(c) = \mathbb{P}\{\mathcal{C}_i{=}c\}$. Alternatively, the execution time distribution can also be specified using its cumulative distribution function (CDF), denoted by $F_{\mathcal{C}_i}(\cdot)$, where $F_{\mathcal{C}_i}(x) = \sum_{c=0}^{x} f_{\mathcal{C}_i}(c)$. In the stochastic analysis three system utilizations are defined, namely $U^{\min}$, $U^{\max}$ and $\bar{U}$, which are calculated using the minimum, maximum and average task execution times, respectively.

Each periodic task gives rise to an infinite sequence of jobs, Γ_j, with deterministic release times λ_j. Each job requires an execution time which is a random variable whose distribution is given by the probability function of the task it comes from, $f_{\mathcal{C}_i}(\cdot)$, and it is assumed to be independent of other jobs of the same task and those of other tasks. The response time of a job Γ_j is a random variable, $\mathcal{R}_j$, whose probability function has to be obtained by the analysis. D_i is the task relative deadline and M_i the maximum allowable probability of missing it. Task τ_i is said to be schedulable if $\mathbb{P}\{\mathcal{R}_i{>}D_i\} \leq M_i$, $\mathcal{R}_i$ being the response time of τ_i.

The scheduling policy we assume is a general, preemptive, priority-driven policy that assigns a static priority to each job and schedules jobs according to this priority. The scheduler guarantees that the running job is the one with the highest priority among the ready jobs. We are not concerned with the policy used to assign priorities to jobs, as long as they are assigned in a deterministic way. This model includes well known fixed priority policies such as *Deadline Monotonic* (DM), and non-fixed priority policies such as *Earliest Deadline First* (EDF).

1 Throughout this paper we use a calligraphic typeface to denote random variables, e.g. $\mathcal{C}$, $\mathcal{W}$, $\mathcal{R}$, etc.

3. Previous work

Next, we summarize the stochastic analysis of independent and periodic task sets, presented in [7]. We have simplified the original notation and introduced a new notation useful for the next sections.

The response time of a job Γ_j is given by $\mathcal{R}_j = \mathcal{W}_{P_j}(\lambda_j) + \mathcal{C}_j + \mathcal{I}_j$, where $\mathcal{W}_{P_j}(\lambda_j)$ is the backlog of priority P_j at time λ_j, which represents the workload of priorities P_j and higher that have not yet been processed just before the release time λ_j of Γ_j. $\mathcal{C}_j$ is the execution time of job Γ_j. $\mathcal{I}_j$ is the interference on Γ_j of all the jobs of higher priority than job Γ_j, released after job Γ_j. Note that all the terms in the equation are random variables. This is the stochastic counterpart of a well known deterministic equation that provides the response time of a job under a preemptive priority-driven scheduling policy (see eq. (16) in [3]).

None of the jobs of priority less than P_j have any influence on the response time of job Γ_j. In addition, none of the jobs of priority P_j released after λ_j have any influence on the response time of job Γ_j. In order to simplify the notation, we assume all these jobs are removed to calculate $\mathcal{R}_j$, and the task indexes updated accordingly. In addition, we can remove the subindex P_j from $\mathcal{W}_{P_j}$ to simplify the notation, since all jobs considered have priority P_j or higher.

Figure 1 illustrates an example in which the response time of job Γ_3 is computed, following the algorithm described in [7]. The calculation starts with zero backlog at the release time of the first job, Γ_1, i.e., $\mathcal{W}(\lambda_1) = \mathbb{O}$, where $\mathbb{O}$ is a null random variable, with probability function

$$f_{\mathbb{O}}(w) = \begin{cases} 1 & \text{if } w = 0 \\ 0 & \text{if } w \neq 0 \end{cases} \qquad (1)$$

The backlog distribution at λ_2, denoted by $\mathcal{W}(\lambda_2)$, is calculated by convolving $f_{\mathcal{W}(\lambda_1)}$ with $f_{\mathcal{C}_1}$, shifting the result $(\lambda_2 - \lambda_1)$ time units left and accumulating negative values in zero, since negative backlog values are not possible (see fig.1). In the same way, the backlog distribution at λ_3, denoted by $\mathcal{W}(\lambda_3)$ can be calculated by convolving $f_{\mathcal{W}(\lambda_2)}$ with $f_{\mathcal{C}_2}$, shifting the result $(\lambda_3 - \lambda_2)$ time units left and accumulating negative values in zero.

Let us define the function $\textsc{shrink}(\mathcal{W}, \Delta)$, which produces a new random variable whose probability function is equal to the probability function of $\mathcal{W}$, left-shifted the amount Δ and with all values for negative abscissae accu-

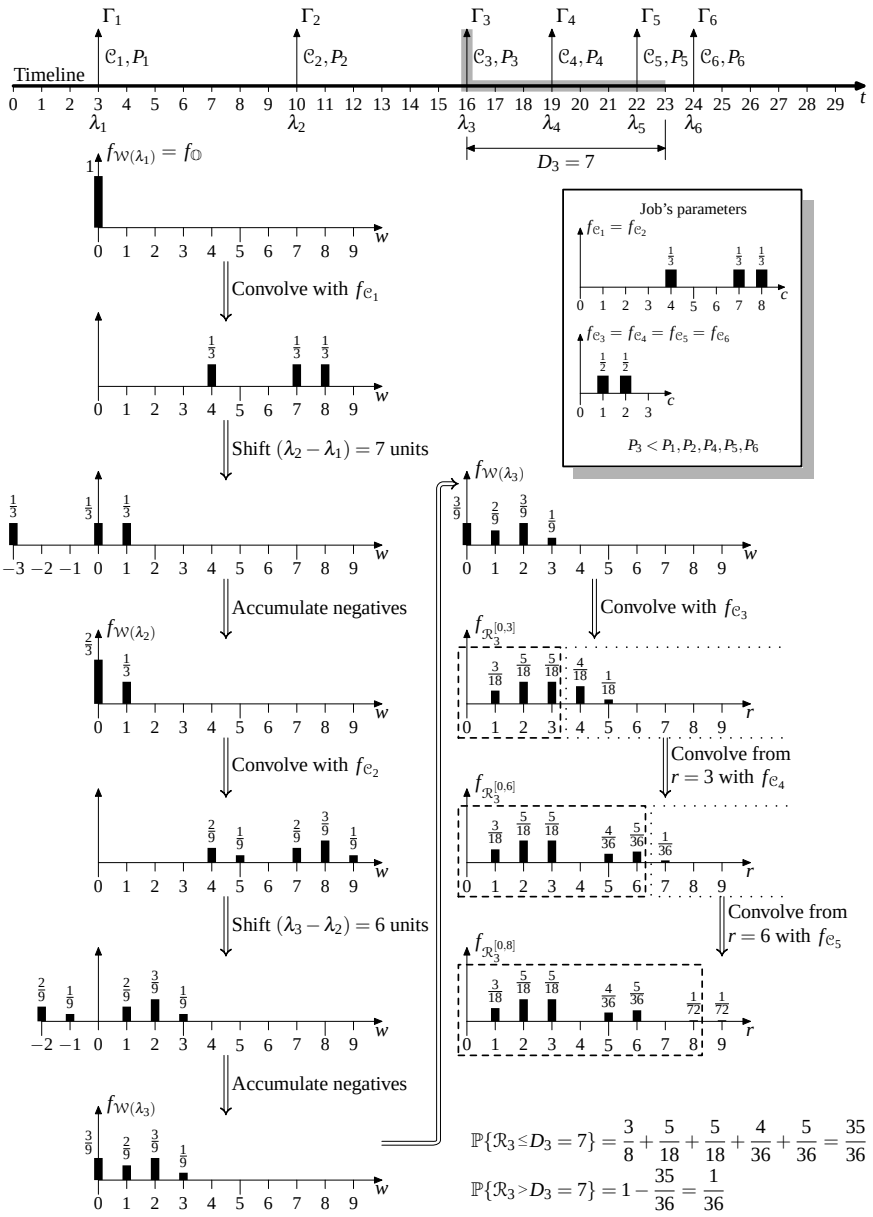

Figure 1. Example of calculation of response time probability function for a job

mulated at zero. That is:

$$f_{\text{SHRINK}(\mathcal{W},\Delta)}(x) = \begin{cases} 0 & \text{if } x < 0 \\ \sum_{w=-\infty}^{0} f_{\mathcal{W}}(w+\Delta) & \text{if } x = 0 \\ f_{\mathcal{W}}(x+\Delta) & \text{if } x > 0 \end{cases} \quad (2)$$

Using this function, the calculation of $\mathcal{W}(\lambda_j)$ can be iteratively expressed as

$$\begin{aligned} &\mathcal{W}(\lambda_1) = \mathbb{O} \\ &\mathcal{W}(\lambda_j) = \text{SHRINK}(\mathcal{W}(\lambda_{j-1}) + \mathcal{C}_{j-1}, \lambda_j - \lambda_{j-1}) \quad \text{for } j > 1 \end{aligned} \quad (3)$$

Note that the probability function of a sum of random variables is obtained by convolving their probability functions. Once $\mathcal{W}(\lambda_j)$ has been calculated, in order to calculate $\mathcal{R}_j$ it is necessary to add the execution time $\mathcal{C}_j$ and the interference of future jobs. Figure 1 depicts the process again.

Next, $f_{\mathcal{W}(\lambda_3)}$ is convolved with $f_{\mathcal{C}_3}$. The resultant distribution, denoted $\mathcal{R}_3^{[0,\lambda_4-\lambda_3]}$, is a random variable that provides the response time distribution of job Γ_3 assuming that jobs Γ_4 and subsequent do not exist (and therefore do not interfere with Γ_3). The probability function of $\mathcal{R}_3$ in the range $[0,\lambda_4-\lambda_3]$ coincides with that of $\mathcal{R}_3^{[0,\lambda_4-\lambda_3]}$ in the same

range.

Next, $f_{\mathcal{R}_3^{[0,\lambda_4-\lambda_3]}}$ is convolved from[2] $r=(\lambda_4-\lambda_3)$ with $f_{\mathcal{C}_4}$. The resultant distribution, denoted $\mathcal{R}_3^{[0,\lambda_5-\lambda_3]}$, is a random variable that provides the response time distribution of job Γ_3 assuming that jobs Γ_5 and subsequent do not exist (and therefore do not interfere with Γ_3). The probability function of $\mathcal{R}_3$ in the range $[0,\lambda_5-\lambda_3]$ coincides with that of $\mathcal{R}_3^{[0,\lambda_5-\lambda_3]}$ in the same range.

The iteration process continues until the relative deadline of Γ_j is included in the interval of one of the random variables. In the example of Figure 1, the iteration ends with the calculation of $\mathcal{R}_3^{[0,\lambda_6-\lambda_3]}$ for a relative deadline of value 7 for Γ_3. At that moment, the probability of Γ_3 meeting its deadline can be computed (of value $35/36$), and therefore the probability of missing its deadline (of value $1/36$).

Let us define the function $\mathrm{CF}(\mathcal{R},\Delta,\mathcal{C})$, which convolves $\mathcal{R}$ from Δ with $\mathcal{C}$. The result is a new random variable, whose probability function is obtained by

$$f_{\mathrm{CF}(\mathcal{R},\Delta,\mathcal{C})}(x)=\begin{cases} f_{\mathcal{R}}(x) & \text{for } x\le\Delta \\ \sum_{i=\Delta+1}^{\infty} f_{\mathcal{R}}(i)\cdot f_{\mathcal{C}}(x-i) & \text{for } x>\Delta \end{cases} \quad (4)$$

Using this function, the calculation of $\mathcal{R}_j$ can be iteratively expressed as

$$\begin{aligned} \mathcal{R}_j^{[0,\lambda_{j+1}-\lambda_j]} &= \mathcal{W}(\lambda_j)+\mathcal{C}_j \\ \mathcal{R}_j^{[0,\lambda_{k+1}-\lambda_j]} &= \mathrm{CF}(\mathcal{R}_j^{[0,\lambda_k-\lambda_j]},\lambda_k-\lambda_j,\mathcal{C}_k) \quad \text{for } k>j \end{aligned} \quad (5)$$

The iteration can stop when $\lambda_{k+1}-\lambda_j\ge D_j$.

In theory, the probability of a task missing its deadline is calculated by averaging the probabilities of all its jobs missing that deadline, but in practice the number of these jobs is infinite. However, when $\bar{U}<1$ the system becomes stable as proved in [9]. In the steady state, the probability of a job missing its deadline becomes constant for the same job released one, two or any number of hyperperiods later. This probability depends on the steady state backlog at the release instant of the job. Three methods were proposed in [7] to perform the calculation of this steady state backlog: an exact method based on obtaining the Markov matrix modelling the stochastic process, an approximate method based on iteration using equation (3), and another approximate method based on the truncation of the Markov matrix. Once the steady state backlog is known for the first job of the task in one hyperperiod, using equation (3) it is possible to calculate the steady state backlog for the rest of the jobs coming from the task released in the same hyperperiod. Next, the probabilities of missing the deadlines are calculated for all the jobs of the task released in that hyperperiod using equation (5), and their average provides the probability of deadline misses of the task.

2 The "convolve from" operation is formally defined in eq. (4)

In the case of EDF scheduling, the steady state backlog distribution is calculated for the first ground-job released in one hyperperiod. A ground-job is a job with the same or lower priority than all the jobs previously released. Once the steady state backlog for a ground-job is calculated using any of the three previous methods, the steady state backlog distribution can be calculated for any job of any task within the hyperperiod using equation (3). In the case of fixed priorities, the backlog should be calculated for the first job of each task within a steady state hyperperiod, and then, the backlog for the rest of jobs of the task within that hyperperiod can be obtained using eq. (3),

One interesting property of the steady state backlog calculation under EDF scheduling is that the steady state backlog is calculated only once (for the first ground-job). In the case of fixed-priority scheduling, it is necessary to calculate *n* different steady state backlogs (one for each priority level). In practice, the calculation of the steady state backlog is the most time-consuming operation in the analysis, so analyzing the probabilistic schedulability for EDF requires far less time than for fixed-priority.

4. The concept of pessimism in the stochastic analysis

In the deterministic analysis of hard real-time systems, all approximations are *pessimistic* in the sense that the response times obtained by the approximated analysis are guaranteed to be greater (i.e., worse) than the exact response times of the system. However, in the stochastic analysis, the response time is a random variable, and the real-time constraints are expressed in terms of probabilities of deadline misses. With these ideas in mind, we will define the "*worse than*" relationship among random variables in the context of real-time systems. Using this relationship, we will state that any random variable in the stochastic analysis is *pessimistic* if it is "worse than" the exact one.

The most important result of this section is that if pessimistic variables are introduced into the stochastic analysis, the response times provided by the analysis will be also pessimistic. In that case we simply state that the analysis is pessimistic. The pessimistic analysis is a *safe* approximation in the sense that the probabilities of deadline misses it provides are *guaranteed* to be greater than the exact ones.

Next we formally define the "worse than" relationship among random variables.

Definition 1. *Given two random variables $\mathcal{X}$ and $\mathcal{Y}$, we state that "$\mathcal{X}$ is worse than $\mathcal{Y}$", and denote it by $\mathcal{X}\succcurlyeq\mathcal{Y}$ if $F_{\mathcal{X}}(x)\le F_{\mathcal{Y}}(x)$ for all x.*

Graphically, this means that the curve $F_{\mathcal{X}}(\cdot)$ never goes above the curve $F_{\mathcal{Y}}(\cdot)$. Note that if the curves $F_{\mathcal{X}}(\cdot)$ and $F_{\mathcal{Y}}(\cdot)$ cross, the variables $\mathcal{X}$ and $\mathcal{Y}$ are not comparable, and

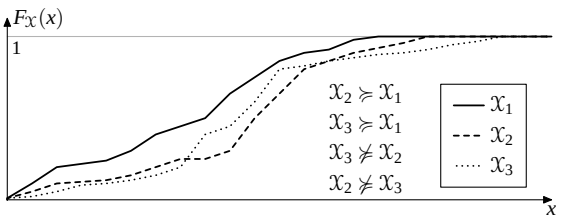

Figure 2. Graphical meaning of the "worse than" relationship

it is not true that $\mathcal{X} \succcurlyeq \mathcal{Y}$ nor $\mathcal{Y} \succcurlyeq \mathcal{X}$ (see fig. 2). Note that the relationship $\mathcal{X} \succcurlyeq \mathcal{Y}$ is the stochastic counterpart of the "greater than or equal to" relationship among deterministic variables ($X \geq Y$). Thus, strictly, the relationship $\mathcal{X} \succcurlyeq \mathcal{Y}$ should be named "worse than or equal to" (or, alternatively, "no-better-than"). However, we plainly use "worse than" for the sake of compactness and legibility.

The mathematical definition of *worse than*, given in definition 1, coincides with the mathematical definition of *first-order stochastic dominance* introduced in statistics and further used in economics [15]. For example, the wealth an investor receives from decisions A and B may be modelled by two random variables. If the probability of an investor receiving greater wealth investing in A is higher than investing in B, *for any wealth value*, we state that decision A first-order dominates decision B. We use the same mathematical definition, but in a different context.

Suppose that $\mathcal{R}'$ is the approximate response time of a task provided by the stochastic analysis, while $\mathcal{R}$ is the exact response time. In general they will not be equal, i.e. they will not have the same probability function, due to approximations in the model or the analysis. However, if $\mathcal{R}' \succcurlyeq \mathcal{R}$, then the analysis is pessimistic and thus safe, because this would imply $\mathbb{P}\{\mathcal{R}>D\} \leq \mathbb{P}\{\mathcal{R}'>D\}$ for all D. That is, for any deadline, the exact probability of deadline misses is less than the probability provided by the pessimistic analysis.

4.1. Properties of the stochastic analysis

The stochastic analysis described in Section 3 has two important properties, which are presented in this section in Theorems 1 and 2. The idea is intuitive: if pessimistic data is introduced in the stochastic analysis, the backlog and the response times resulting from the analysis become also pessimistic. This idea, almost trivial when all the system parameters are deterministic, has to be carefully proved when the parameters and the results are random variables.

In order to prove these theorems, some basic properties of the relationship "worse than" and some additional properties of the functions SHRINK() and CF() which were introduced in Section 3, are required. These properties are presented below, but their proof has been omitted for the sake of brevity. The interested reader can find them in [8].

Property 1. *Reflexivity: $\mathcal{A} \succcurlyeq \mathcal{A}$ for any $\mathcal{A}$.*

Property 2. *Transitivity: if $\mathcal{A} \succcurlyeq \mathcal{B}$ and $\mathcal{B} \succcurlyeq \mathcal{C}$, then $\mathcal{A} \succcurlyeq \mathcal{C}$.*

Property 3. *If $\mathcal{A} \succcurlyeq \mathcal{B}$, then for all $\mathcal{C} \succcurlyeq \mathbb{0}$, $\mathcal{A}+\mathcal{C} \succcurlyeq \mathcal{B}+\mathcal{C}$.*

Property 4. *For all positive $\mathcal{A} \succcurlyeq \mathbb{0}, \mathcal{B} \succcurlyeq \mathbb{0}$, it follows that $\mathcal{A}+\mathcal{B} \succcurlyeq \mathcal{A}$ and $\mathcal{A}+\mathcal{B} \succcurlyeq \mathcal{B}$.* That is, the result of adding two positive random variables is always worse than either of them.

Property 5. *For any $\mathcal{A} \succcurlyeq \mathbb{0}, \mathcal{B} \succcurlyeq \mathbb{0}, \mathcal{C} \succcurlyeq \mathbb{0}, \mathcal{D} \succcurlyeq \mathbb{0}$, such that $\mathcal{A} \succcurlyeq \mathcal{B}$ and $\mathcal{C} \succcurlyeq \mathcal{D}$, it follows that $\mathcal{A}+\mathcal{C} \succcurlyeq \mathcal{B}+\mathcal{D}$.*

Apart from the above general properties of the "worse than" relationship, the functions SHRINK() and CF() fulfil the following properties.

Property 6. *For any $\mathcal{A} \succcurlyeq \mathcal{B}$, and $\Delta \geq 0$, it follows that* $\text{SHRINK}(\mathcal{A},\Delta) \succcurlyeq \text{SHRINK}(\mathcal{B},\Delta)$. That is, the shrink function preserves pessimism

Property 7. *The function* CF() *defined in eq. (4) preserves pessimism. In particular:*
a) If $\mathcal{C}_1 \succcurlyeq \mathcal{C}_2$, then $\text{CF}(\mathcal{R},\Delta,\mathcal{C}_1) \succcurlyeq \text{CF}(\mathcal{R},\Delta,\mathcal{C}_2)$
b) If $\mathcal{R}_1 \succcurlyeq \mathcal{R}_2$, then $\text{CF}(\mathcal{R}_1,\Delta,\mathcal{C}) \succcurlyeq \text{CF}(\mathcal{R}_2,\Delta,\mathcal{C})$
c) If $\Delta \geq 0$ and $\mathcal{C} \succcurlyeq \mathbb{0}$, then $\text{CF}(\mathcal{R},\Delta,\mathcal{C}) \succcurlyeq \mathcal{R}$
d) If $\Delta_1 \leq \Delta_2$, then $\text{CF}(\mathcal{R},\Delta_1,\mathcal{C}) \succcurlyeq \text{CF}(\mathcal{R},\Delta_2,\mathcal{C})$

Now we can state and demonstrate the following theorems.

Theorem 1. *Let S and S' be two real-time systems with identical parameters, but different initial backlog $\mathcal{W}(0)$ and $\mathcal{W}'(0)$ respectively. If $\mathcal{W}'(0) \succcurlyeq \mathcal{W}(0)$, then $\mathcal{W}'(t) \succcurlyeq \mathcal{W}(t)$ for all $t \geq 0$.*

Proof. Let t_1 be equal to the arrival instant of the next job which contributes to the backlog. Since S and S' have the same parameters, this instant is the same for both. For all $t < t_1$, the backlog at instant t is obtained simply as $\text{SHRINK}(\mathcal{W}(0),t)$, so, by property 6, $\mathcal{W}'(t) \succcurlyeq \mathcal{W}(t)$ for $t < t_1$. At instant $t = t_1$, the backlog is increased by the execution time of the arriving job. Let $\mathcal{C}$ be the random variable which represents this execution time, which is the same for both systems S and S'. By property 3, $\mathcal{W}'(t_1)+\mathcal{C} \succcurlyeq \mathcal{W}(t_1)+\mathcal{C}$. Taking t_1 as the new time origin, the same reasoning can be repeated until reaching any future instant. □

This theorem implies that, when $\mathcal{W}(0) = \mathbb{0}$, the backlog worsens with time. This has important implications in the issue of obtaining the steady state backlog, which will be addressed in Section 5.

Theorem 2. *Let S and S' be two real-time systems, with identical parameters, except for one of the jobs, say Γ_k, whose execution time is $\mathcal{C}_k$ in system S and $\mathcal{C}'_k$ in system S'.*

If $\mathcal{C}'_k \succcurlyeq \mathcal{C}_k$, then the response times obtained by the stochastic analysis of these systems fulfil $\mathcal{R}'_j \succcurlyeq \mathcal{R}_j$ for all Γ_j.

Proof. All jobs with priority greater than P_k are not affected by Γ_k, so their response time remains the same, and trivially $\mathcal{R}'_j \succcurlyeq \mathcal{R}_j$ for these jobs (because the "worse than" relation is reflexive). So we will focus only on jobs with priority less than P_k. Let us consider an arbitrary job Γ_j.

If $j < k$, the backlog for any instant prior to λ_k is the same for both systems, because the sequence of arrivals and the execution times are the same. As a consequence, if job Γ_j cannot be preempted by Γ_k, the response time will be the same for both systems. If Γ_j can be preempted by Γ_k, then $\mathcal{R}'_j \succcurlyeq \mathcal{R}_j$, by the hypothesis $\mathcal{C}'_k \succcurlyeq \mathcal{C}_k$ and property 7(a).

If $j = k$, we are calculating the response time of job Γ_k. As in the previous case, the backlog is the same in both systems. But the response time $\mathcal{R}'_k$ will be worse than $\mathcal{R}_k$, because $\mathcal{C}'_k \succcurlyeq \mathcal{C}_k$, and because of properties 7(a) and 7(b) we will have $\mathcal{R}'_k \succcurlyeq \mathcal{R}_k$.

Finally, if $j > k$, $\mathcal{W}'(\lambda_j) \succcurlyeq \mathcal{W}(\lambda_j)$, because at instant λ_k the backlog $\mathcal{W}$ is increased by $\mathcal{C}_k$, while the backlog $\mathcal{W}'$ is increased in $\mathcal{C}'_k$, being $\mathcal{C}'_k \succcurlyeq \mathcal{C}_k$. In virtue of Theorem 1, the backlog will be worse for any future instant. Then, for any job Γ_j released after Γ_k, its initial backlog is worse in system S', and thus, by properties 7(a) and 7(b) again, we conclude $\mathcal{R}'_j \succcurlyeq \mathcal{R}_j$. □

This theorem implies that, if required, the analyst can replace the execution time of any job by a more pessimistic distribution. The results obtained after this replacement are pessimistic, but safe. This mechanism allow us to introduce approximations and extensions to the task set model.

Corollary 1. *Let S be a real-time system and S' the system obtained by adding a new job to S. Then, the response times obtained by the stochastic analysis of these systems fulfil $\mathcal{R}'_j \succcurlyeq \mathcal{R}_j$ for all Γ_j.*

Proof. This can be easily proved by assuming that system S has an additional job with $\mathcal{C} = \mathbb{O}$ (which does not alter the analysis), while system S' has the same job with $\mathcal{C}' \succcurlyeq \mathbb{O}$. Now, Theorem 2 can be directly applied. □

5. Problems in the analysis of periodic and independent task sets

The biggest challenge in the analysis of periodic and independent task sets is the calculation of the steady state backlog. Three methods were presented in [7] to perform the steady state backlog calculation:

- Calculation of the eigenvalues of the Markov matrix that models the stochastic process (which is an infinite matrix with a repetitive structure).
- Calculation of the eigenvalues of a truncated Markov matrix.
- Iteration on equation (3).

The calculation of the eigenvalues of the Markov matrix is useful from a theoretical perspective, since it provides valuable information about the expected kind of solutions. However, this approach has many practical problems. Firstly, the obtaining of the Markov matrix, and the finding of its eigenvalues, implies a computational cost unaffordable except for simple task sets like the one presented in [7]. Otherwise the periodic part of the Markov matrix becomes huge. Finally, the Markov matrix is badly conditioned which adds to the difficulties of the problem.

A partial solution to the computational problems related to the Markov matrix is its truncation. This reduces its range and provides an approximation to the backlog distribution. However, this approximation is not necessarily worse than the exact one, so the steady state backlog obtained by this method may be optimistic, which is inadequate for the analysis of probabilistic hard real-time systems.

The last method to calculate the steady state backlog comes from iterating on equation (3). The concept is simple: in order to calculate the steady state backlog for a given job within the hyperperiod, we calculate the backlog for that job in the first, second, etc, hyperperiods and we stop when we observe that the steady state backlog distribution converges (the convergence is guaranteed whenever $\bar{U} < 1$). For example, we may observe that the steady state backlog distribution at the release time of the first job within the hyperperiod is almost identical in hyperperiods 19 and 20 and stop. In practice, only a few hyperperiods are required to obtain convergence, except when $\bar{U}$ is close to 1.0, since theoretically the convergence is geometrically ergodic [7].

Therefore, except for simple task sets, only the iterative method can be applied. However, this method presents several problems:

1. When $U^{\max} > 1$, which represents the most interesting case, the steady state backlog is made up of an infinite number of points, with a tail which approaches zero asymptotically. Dealing with an infinite number of points is not computationally possible.
2. After iterating on a new hyperperiod, the backlog distribution becomes closer and closer to the steady state backlog. However, the steady state backlog distribution estimated by iteration is optimistic.
3. The execution time probability functions, if defined exactly, are made up of hundreds of points. Dealing with them rapidly overflows processor and memory resources.

The solution to the three previous problems in the analysis of periodic and independent task sets comes from the concept of pessimism in the stochastic analysis, as shown in the next subsections.

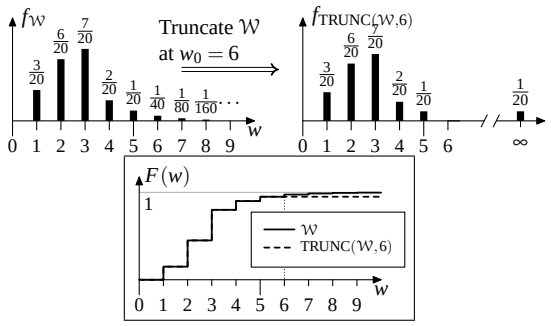

Figure 3. Truncation of a long tail

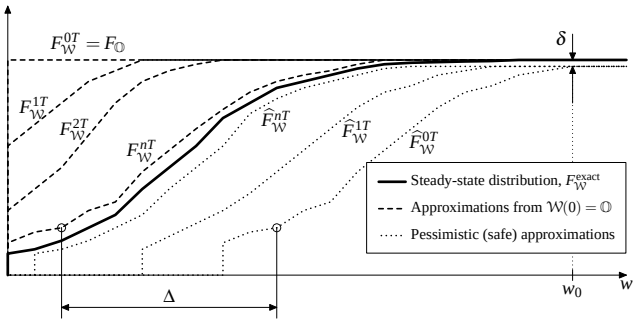

Figure 4. Approximating the steady state backlog by iteration

5.1. The problem of an infinite backlog tail

An obvious solution to the problem of the infinite length of the steady state backlog is to truncate its probability function at a given point. The backlog probability function has an infinite queue that decreases exponentially as the backlog increases. Therefore, cutting that queue at some point we obtain an approximate steady state backlog probability function defined by a finite number of points. Figure 3 depicts a steady state backlog which is truncated at $w = 6$. The resultant probability function corresponds to a new random variable, denoted by $\mathrm{TRUNC}(\mathcal{W}, 6)$. In general,

$$f_{\mathrm{TRUNC}(\mathcal{W},w_0)}(w) = \begin{cases} f_{\mathcal{W}}(w) & \text{if } w < w_0 \\ 0 & \text{if } w \geq w_0 \end{cases}$$

This truncated probability function does not sum 1, however, the "deficit" of probability can be assumed to be located at $w = \infty$. The CDF never reaches the value 1, so it is clear that $F_{\mathrm{TRUNC}(\mathcal{W},w_0)} \leq F_{\mathcal{W}}$ (see fig.3), and thus $\mathrm{TRUNC}(\mathcal{W}, w_0) \succcurlyeq \mathcal{W}$ for any w_0 and $\mathcal{W}$.

From Theorem 1, introducing a truncated (pessimistic) backlog probability function in any of the iterations of eq. (3), gives rise to subsequent more pessimistic backlog probability functions than those obtained without truncation. In order to compute the response time of a job, according to eq. (5), the backlog and the execution time of the job are added in the first step. Since the truncated backlog is worse than the exact one, the result of this first step will also be worse, by property 3. And, according to property 7(b), if the first step of the response time calculation is worse, all the subsequent steps will be also worse, so the response time probability functions derived from pessimistic backlog distributions are also pessimistic.

The reader should note that, even if the complete information about the tail of the distribution is lost, some information is retained, namely, the accumulated probability of the tail is left as a probability deficit, which will affect the results of the analysis. In addition, the truncated distribution is pessimistic, so it is safe to use it in the analysis. This makes truncation preferable over other analysis techniques, like simulation, which do not provide any information at all about the tail, and are not pessimistic.

Therefore, the truncation of any backlog probability function enables the use of finite length probability functions, but introduces pessimism in the analysis. The pessimism introduced into the analysis depends on the truncation point, w_0, but also depends on all the parameters of the task set. A high truncation point reduces the pessimism of the analysis, but increases the memory requirements and computational cost of the analysis tool.

5.2. The problem of zero initial backlog

The iterative method provides approximations to the steady state backlog probability function. After iterating for a few hyperperiods, the result of the iteration is frequently an approximate steady state backlog probability function close to the exact one.

The iteration starts with $f_{\mathcal{W}(\lambda_1)} = f_{\mathbb{0}}$, i.e., with zero initial backlog at the beginning of the first hyperperiod, as indicated in equation (3). However, according to Theorem 1, if we start from a null backlog, the backlog worsens with each new hyperperiod. Thus, in each iteration, we obtain a backlog which is worse than (or equal to) the one obtained in the previous iteration. Although the sequence converges towards a steady state distribution, this steady state backlog will be worse than that obtained in any of the iterations. Therefore, stopping at any iteration we obtain an *optimistic* estimation of the steady state backlog which is not admissible in probabilistic hard real-time systems. The iteration process is depicted in Figure 4. Terms $F^{kT}_{\mathcal{W}}$ are the backlog distribution functions at the beginning of k-th hyperperiod and $F^{\mathrm{exact}}_{\mathcal{W}}$ is the exact backlog distribution function in the steady state.

One solution to the problem of zero initial backlog is depicted in the same figure. Once convergence is detected, for example using the quadratic error between two subsequent iterations, the approximated steady state backlog distribution function obtained in the last iteration, $F_{\mathcal{W}}^{nT}$ in Figure 4, is shifted right Δ units and truncated after some point w_0, giving $\widehat{F}_{\mathcal{W}}^{0T}$, which is worse than the steady state backlog. Starting the iteration process with $\widehat{F}_{\mathcal{W}}^{0T}$ instead of $F_{\mathcal{W}}^{0T} = F_{\mathbb{O}}$ provides approximations to the steady state backlog distribution which are worse than the exact one. Thus, the stochastic analysis provides safe results in this case.

Nevertheless, there are two practical problems to solve in the previous approach. Firstly, to define the size of Δ, which depends on the difference between $F_{\mathcal{W}}^{nT}$ and $F_{\mathcal{W}}^{(n-1)T}$ and depends also on the convergence rate. Secondly, to define the truncation point w_0, so that $\widehat{F}_{\mathcal{W}}^{0T}$ is worse than the exact one but does not lack too much probability mass, δ.

5.3. The problem of dense execution time probability functions

The complexity of the analysis grows dramatically with the number of points defining the discrete probability functions of the execution times. Thus, a reliable method to simplify these distributions is needed. In addition, it is difficult in practice to estimate or measure the execution time probability functions exactly, so we need a safe method of estimation or measurement.

It is well known that moving the probabilities from low values of execution time to high values of execution time introduces pessimism. In fact, the deterministic analysis is based on worst-case execution times, which are obtained moving the probabilities of all the execution times to the worst-case execution time. We prove that introducing pessimistic execution times makes pessimistic the stochastic analysis. Figure 5 depicts an example of movement from $f_{\mathcal{C}}$ to $f'_{\mathcal{C}}$.

The distribution functions before and after the movement fulfils $F_{\mathcal{C}} \leq F'_{\mathcal{C}}$ (see fig. 5), and thus $\mathcal{C}' \succcurlyeq \mathcal{C}$. Therefore, as expected from Theorem 2, the stochastic analysis becomes pessimistic and can be applied to probabilistic hard real-time systems.

6. Applications of the concept of pessimism

This section presents some applications of the concept of pessimism, previously introduced. Firstly, it allow us to extend the stochastic analysis of periodic and independent task sets. In particular, section 6.1 presents the statistical analysis of dependent task sets that can block in shared resources. Secondly, section 6.2 proves that one of the optimal priority assignment algorithms in the deterministic scenario is also valid in the stochastic scenario.

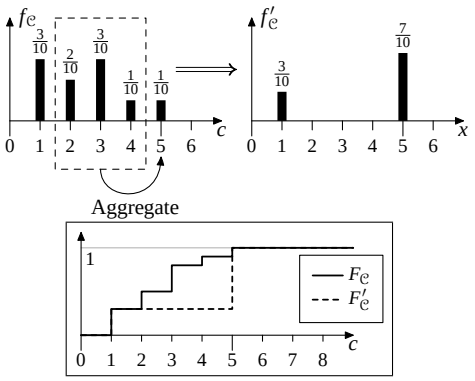

Figure 5. Moving probabilities towards higher execution times

6.1. Blocking in shared resources

Shared resources, like shared memory areas, are useful to communicate between tasks. Resource access protocols are used to preserve the consistency of the shared data, guaranteeing at the same time bounded blocking times. Examples of these protocols are the Priority Inheritance Protocol (PIP) and Priority Ceiling Protocol (PCP) for fixed priority scheduling [18], as well as the Stack Resource Policy (SRP) for fixed and non-fixed priority scheduling [4].

Under a deterministic analysis, the response time of a task τ_i that can suffer blocking is calculated by artificially increasing its execution time by B_i units, where B_i is the blocking time of the task. Since the exact blocking time can vary between different releases of the same task or be difficult to calculate, B'_i is used instead of B_i, where B'_i is a bound on the exact blocking time, B_i.

Under the stochastic analysis the situation is analogous. The execution time of a task τ_i, of execution time $\mathcal{C}_i$ should be increased by adding the blocking time $\mathcal{B}_i$, which is now a random variable. The result is a transformed task with execution time $\mathcal{C}_i + \mathcal{B}_i$ (being $f_{\mathcal{C}_i+\mathcal{B}_i} = f_{\mathcal{C}_i} \otimes f_{\mathcal{B}_i}$). Now the problem is analogous to that found in the deterministic analysis, i.e., how to calculate the exact distribution of the random variable $\mathcal{B}_i$. The solution is to find a bound valid for all scenarios. In stochastic terms, this means finding a random variable $\mathcal{B}'_i$, worse than the exact $\mathcal{B}_i$ for all possible scenarios. Let us define how to construct a random variable worse than any of a set of random variables.

Definition 2. *Given a set of random variables* $\{\mathcal{X}_i\}$*, we define the* supremum *of that set, and denote it as* $\sup\{\mathcal{X}_i\}$*, the random variable whose CDF is*

$$F_{\sup\{\mathcal{X}_i\}}(x) = \min_i F_{\mathcal{X}_i}(x) \tag{6}$$

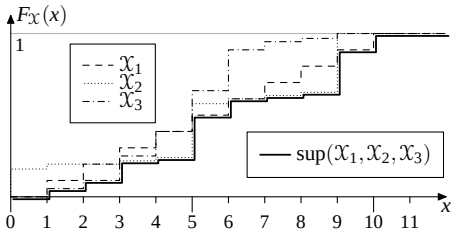

Figure 6. Construction of the supremum of a set of random variables

Figure 6 shows how the function is constructed[3], by taking the minimum of all $F_{\mathcal{X}_i}(\cdot)$. By construction, $\sup\{\mathcal{X}_i\} \succcurlyeq \mathcal{X}_i$ for all i. Thus, the idea of the supremum of a set of random variables is analogous to the maximum of a set of real numbers. Using this idea, the classical results for resource access protocols can be easily translated to the stochastic analysis as well.

The system model has to be extended to hold information about the set of semaphores (S_k) used by the system to guard the shared resources, and the length of the critical sections in the tasks. In the classical analysis, the typical information stored in the model consists of a set of real numbers $D_{i,k}$ which represent, for each pair (τ_i, S_k), the length of the longest critical section that task τ_i contains, guarded by semaphore S_k. From this set of real numbers, the maximum blocking time B'_i of each task is obtained. The way in which this is done depends on the resource sharing protocol. For example, PCP guarantees that each task is blocked only once by any task of less priority, while PIP only guarantees that a task cannot be blocked twice by the same semaphore or by the same task. These different properties lead to different algorithms to obtain B'_i.

These ideas can be translated to the stochastic case, using the concept of *supremum* defined above. The length of a critical section is now a random variable whose probability function is assumed known. Then, a random variable $\mathcal{D}_{i,k}$ is constructed as the supremum of the length of all the critical sections of task τ_i guarded by semaphore S_k. From these $\mathcal{D}_{i,k}$ an estimation $\mathcal{B}'_i$ of the blocking time $\mathcal{B}_i$ of each task can be obtained. Once $\mathcal{B}'_i$ is obtained, the stochastic analysis can be done as described in Section 3, but using $(\mathcal{B}'_i + \mathcal{C}_i)$ instead of $\mathcal{C}_i$. If we can guarantee that $\mathcal{B}'_i \succcurlyeq \mathcal{B}_i$, then the analysis will be pessimistic, because $\mathcal{B}'_i + \mathcal{C}_i \succcurlyeq \mathcal{B}_i + \mathcal{C}_i$, from property 3. Thus, by Theorem 2, the response times will be also pessimistic.

6.1.1. Priority Ceiling Protocol (PCP). It is well known that, under PCP, a task τ_i can be blocked only once by tasks of lower priority [18]. This property ensures that the maximum blocking time B'_i that a task τ_i can suffer coincides with the length of the longest critical section among all the lower priority tasks which can cause blocking to τ_i. Translating the deterministic method presented in [18] to the stochastic case, we will compute $\mathcal{B}'_i$ as the supremum of the set $\{\mathcal{D}_{j,k} | P_j < P_i, C(S_k) \geq P_i\}$, $C(S_k)$ being the priority ceiling of the semaphore S_k, defined as the highest priority among the tasks which use that semaphore.

3 Please, note that the plot of the supremum has been slightly shifted down for better legibility of the figure.

	$S_1(P_1)$	$S_2(P_1)$
τ_1	$\mathcal{D}_{1,1}$	$\mathcal{D}_{1,2}$
τ_2	$\mathcal{D}_{2,1}$	$\mathcal{D}_{2,2}$
τ_3	$\mathcal{D}_{3,1}$	$\mathcal{D}_{3,2}$

Table 1. Cumulative distributions of the critical sections for the blocking examples.

For example, consider a system with three tasks and two semaphores (Table 1). The priority ceiling of each semaphore is indicated in parentheses. Each cell in the table contains the length of the critical section of task τ_i guarded by semaphore S_k. If a task τ_i does not have any critical section guarded by semaphore S_k, then $\mathcal{D}_{i,k} = \mathbb{0}$. If a task τ_i contains several sections guarded by S_k, $\mathcal{D}_{i,k}$ is computed as the supremum of the lengths of these sections.

Task τ_1 can be blocked by any of the critical sections in lower priority tasks, because the priority ceiling of both semaphores is P_1. Then, $\mathcal{B}'_1 = \sup\{\mathcal{D}_{2,1}, \mathcal{D}_{2,2}, \mathcal{D}_{3,1}, \mathcal{D}_{3,2}\}$. The same applies to task τ_2, so $\mathcal{B}'_2 = \sup\{\mathcal{D}_{3,1}, \mathcal{D}_{3,2}\}$. Task τ_3 cannot suffer blocking because it is the lowest priority task, so $\mathcal{B}'_3 = \mathbb{0}$. Once all $\mathcal{B}'_i$ have been obtained this way, they are added to the corresponding $\mathcal{C}_i$ and the stochastic analysis is carried out as explained in Section 3.

Finally, the reader should note that the results presented for PCP are valid for the Stack Resource Policy (SRP) [4], since deterministic blocking times are calculated using the same algorithm for PCP and SRP. The only difference is that, under SRP, preemption levels have to be used instead of priorities, and the maximum of each resource ceiling, which happens when the number of remaining resource units is zero, has to be used instead of the "priority ceiling" of PCP. I.e., we will compute $\mathcal{B}'_i$ as the supremum of the set $\{\mathcal{D}_{j,k} | \pi_j < \pi_i, C_{S_k}(0) \geq \pi_i\}$, π_i being the preemption level of task τ_i, and $C_{S_k}(n)$ being the ceiling of the resource S_k, when n units are available.

6.1.2. Priority Inheritance Protocol (PIP). If the resource sharing protocol is PIP instead of PCP, the method for obtaining $\mathcal{B}'_i$ is different. Under PIP it has been proved [18] that any task τ_i will be blocked once at most by the same semaphore S_k, or by the same lower prior-

ity task τ_j ($P_j < P_i$). However, it is possible that the task gets blocked several times on different semaphores and by different lower priority tasks. In these cases, the blocking time will be the sum of the lengths of the critical sections which caused blocking. This implies that $\mathcal{B}'_i$ should be computed by examining all possible blocking scenarios, and taking the worst of all these blocking times.

For instance, consider again the example in Table 1. Task τ_1 can be blocked by tasks τ_2 and τ_3, but not by the same semaphore twice. Therefore, if it gets blocked by τ_2 on semaphore S_1, task τ_3 can only cause additional blocking on semaphore S_2, and vice-versa. As a consequence, $\mathcal{B}'_1 = \sup\{(\mathcal{D}_{2,1} + \mathcal{D}_{3,2}), (\mathcal{D}_{2,2} + \mathcal{D}_{3,1})\}$. Note that, since we are dealing with random variables, each sum requires a convolution of the probability functions. Task τ_2 can be blocked only by τ_3, because τ_3 is the only lower priority task. But, since it cannot be blocked twice by the same task, we conclude $\mathcal{B}'_2 = \sup\{\mathcal{D}_{3,1}, \mathcal{D}_{3,2}\}$. Finally, task τ_3 cannot suffer blocking by lower priority tasks, so $\mathcal{B}'_3 = \mathbb{O}$.

An exhaustive analysis of all blocking scenarios is possible. Nevertheless, it can be avoided using a different method, at the cost of introducing even more pessimism. This method is the stochastic counterpart of the one presented in [6] for the deterministic case. In order to obtain a pessimistic approximation, $\mathcal{B}''_i$, of the blocking time the following steps should be performed:

- For each j such that $P_j < P_i$, compute the supremum of the set $\{\mathcal{D}_{j,k} | C(S_k) \geq P_i\}$. Add all these supremi and call the result $\mathcal{B}l_i$.
- For each k such that $C(S_k) \geq P_i$, compute the supremum of the set $\{\mathcal{D}_{j,k} | P_j < P_i\}$. Add all these supremi and call the result $\mathcal{B}s_i$.
- Construct the random variable $\mathcal{B}''_i$ as one whose CDF is $F_{\mathcal{B}''_i}(x) = \max\{F_{\mathcal{B}l_i}(x), F_{\mathcal{B}s_i}(x)\}$. This concept is the inverse of the concept of supremum defined before, so we call it the *infimum*. Thus, $\mathcal{B}''_i = \inf\{\mathcal{B}l_i, \mathcal{B}s_i\}$.

Applying this approximation to the example on Table 1, $\mathcal{B}''_1 = \inf\big\{(\sup\{\mathcal{D}_{2,1}, \mathcal{D}_{2,2}\} + \sup\{\mathcal{D}_{3,1}, \mathcal{D}_{3,2}\}), (\sup\{\mathcal{D}_{2,1}, \mathcal{D}_{3,1}\} + \sup\{\mathcal{D}_{2,2}, \mathcal{D}_{3,2}\})\big\}$. It can be shown that $\mathcal{B}''_i \succcurlyeq \mathcal{B}'_i \succcurlyeq \mathcal{B}_i$. The proof, however, is not short and has been omitted. The reader can find it in [8].

6.2. Priority assignment

One of the pending problems in the stochastic analysis of real-time systems is how to assign priorities to tasks under fixed-priority scheduling. Using the concept of stochastic pessimism, we will prove that the deterministic optimal algorithm for assigning priorities to tasks presented in [3] is valid in the stochastic scenario.

The algorithm consists of $(n-1)$ iteration steps, where n is the number of tasks. We assume that priorities decrease as indexes increase, i.e., $P_1 > P_2 > \ldots, P_n$. In the first step, all the tasks belong to the subset of eligible tasks; one of them receives the lowest priority in the system, P_n, and is removed from the subset. In the second step one of the remaining eligible tasks receives the next priority, i.e., priority P_{n-1} and is removed from the subset of eligible tasks. The process is repeated until reaching the last step, step $(n-1)$. In this last step there are only two eligible tasks, one receives priority P_2, is removed from the set of eligible tasks, and so the other task receives priority P_1.

The problem is how to elect in each step the task that receives the priority associated with the step. In each step, all the eligible tasks are tested sequentially until finding one that is schedulable receiving the priority associated to the step, i.e., with lower priority than the rest of eligible tasks and higher priority than the tasks elected in the previous steps. If no task is found to be schedulable within the eligible task set, the task set as a whole is said to be unfeasible and the iteration process finishes. If all the steps complete successfully, the system is schedulable using fixed priorities and the priorities are those calculated.

The previous algorithm is optimal, i.e., it always finds a feasible assignment of priorities if one exists. The proof in [3] is based on the fact that increasing the priority of a task never decreases its schedulability, or conversely, decreasing a task priority never increases its schedulability. In the stochastic scenario, a task τ_i is schedulable if it fulfils its stochastic real-time constraint, i.e., if $\mathbb{P}\{\mathcal{R}_i > D_i\} \leq M_i$. Proving that decreasing a task priority never increases its schedulability, is the key in the proof of optimality. Next, we prove this intuitive result resorting to the properties of Section 4 again.

Lemma 1. *Decreasing the priority of a task τ_i in the stochastic scenario never increases its schedulability (i.e., the probability $\mathbb{P}\{\mathcal{R}_i \leq D_i\}$)*

Proof. Decreasing the priority of a task adds new jobs in the iterations of eq. (3) and (5). In both cases, it gives rise to more pessimistic response time distributions, as proved in Corollary 1. This implies higher priority of missing any deadline, and therefore lower stochastic schedulability. □

Theorem 3. *The algorithm in [3] for assigning priorities to fixed-priority tasks is optimal in the stochastic scenario.*

Proof. The proof is identical to that given in [3] using Lemma 1. □

7. Conclusions and future work

Exact stochastic analysis of real-time systems is a highly demanding memory and CPU activity, which can be afforded only for simple periodic and independent task sets. In order to deal with practical real-time systems we have to perform approximations. In the context of probabilistic

hard-real time systems, approximations are valid only when the results are on the safe side, i.e., if the analysis is pessimistic.

This paper has introduced the relation *worse than* between two distributions of a random parameter of the analysis, which defines a stochastic ordering in the context of real-time systems. This relation and its properties define a theoretical framework that opens the door to safe stochastic analysis approximations. Whenever the distribution of a parameter of the stochastic analysis is substituted by a worse distribution, the resultant response time distributions coming from the analysis are worse for all the tasks, i.e., the probabilities of missing deadlines are higher and so the analysis becomes safe.

The most interesting characteristic of the relation *worse than* is that it allows us to order different distributions of the same random parameter of the analysis. For example, it allows us to state that one execution time distribution is worse than another, that a blocking time distribution is worse than another, that a response time distribution is worse than another, etc. The ordering between random variables is a valuable tool that permits a rapid translation of well known real-time deterministic results to the stochastic scenario. Deterministic analysis becomes a particular case of the stochastic analysis. This way, any deterministic analysis is always more pessimistic than its stochastic counterpart. Using these translations we have introduced the analysis of task sets that can block on shared resources and the priority assignment, but many others are possible.

Future work will focus on practical issues about the theory presented in this paper. It would be interesting to quantify the degree of pessimism of all the transformations previously introduced, since it would determine their usability. In addition, more applications of the stochastic pessimism concept may be found. These could allow, for example, to deal safely with other task extensions, such as release jitter, stochastic dependencies, etc.

References

[1] L. Abeni and G. Buttazzo. Stochastic Analysis of a Reservation Based System. In *Proc. of the 9th Int. Workshop on Parallel and Distributed Real-Time Systems*, Apr. 2001.

[2] A. K. Atlas and A. Bestavros. Statistical Rate Monotonic Scheduling. In *Proc. of the 19th IEEE Real-Time Systems Symposium*, pages 123–132, Dec. 1998.

[3] N. C. Audsley. Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. Technical Report YCS 164, Dept. Computer Science, University of York, Dec. 1991.

[4] T. P. Baker. Stack-based scheduling of realtime processes. *Real-Time Systems*, 3(1), Mar. 1991.

[5] G. Bernat, A. Colin, and S. Petters. WCET Analysis of Probabilistic Hard Real-Time Systems. In *Proc. of the 23rd IEEE Real-Time Systems Symposium*, Dec. 2002.

[6] G. Buttazzo. *Hard Real-Time Computing Systems. Predictable Scheduling Algorithms and Applications*, chapter 7. Kluwer, Boston/Dordrecht/London, 1997.

[7] J. L. Díaz, D. F. García, K. Kim, C.-G. Lee, L. Lo Bello, J. M. López, S. L. Min, and O. Mirabella. Stochastic Analysis of Periodic Real-Time Systems in a Real-Time System. In *Proc. of the 23rd IEEE Real-Time Systems Symposium*, pages 289–300, Austin, Texas, Dec. 2002.

[8] J. L. Díaz and J. M. López. Safe extensions to the stochastic analysis of real-time systems. Technical report, Departamento de Informática, University of Oviedo, 2004. Also available at `http://www.atc.uniovi.es/research/SESARTS04.pdf`.

[9] J. L. Díaz, J. M. López, and D. F. García. Probabilistic Analysis of the Response Time in a Real-Time System. In *Proc. of the 1st CARTS Workshop on Advanced Real-Time Technologies*, Aranjuez, Spain, Oct. 2002. Also available as Technical Report at `http://www.atc.uniovi.es/research/PART01.pdf`.

[10] M. K. Gardner. *Probabilistic Analysis and Scheduling of Critical Soft Real-Time Systems*. PhD thesis, University of Illinois, Urbana-Champaign, 1999.

[11] M. K. Gardner and J. W. Liu. Analyzing Stochastic Fixed-Priority Real-Time Systems. In *Proc. of the 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Mar. 1999.

[12] J. P. Lehoczky. Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines. In *Proc. of the 11th IEEE Real-Time Systems Symposium*, pages 201–209, Dec. 1990.

[13] J. P. Lehoczky. Real-Time Queueing Theory. In *Proc. of the 17th IEEE Real-Time Systems Symposium*, pages 186–195, Dec. 1996.

[14] J. P. Lehoczky. Real-Time Queueing Network Theory. In *Proc. of the 18th IEEE Real-Time Systems Symposium*, pages 58–67, Dec. 1997.

[15] H. Levy. Stochastic dominance and expected utility: survey and analysis. *Management Science*, 38(4):555–593, 1992.

[16] L. Liu and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of ACM*, 20(1):46–61, 1973.

[17] S. Manolache, P. Eles, and Z. Peng. Memory and Time-Efficient Schedulability Analysis of Task Sets with Stochastic Execution Times. In *Proc. of the 13th Euromicro Conference on Real-Time Systems*, pages 19–26, Jun. 2001.

[18] L. Sha, R. Rajkumar, and J. P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Trans. Comput.*, 39(9):1175–1185, Sept. 1990.

[19] T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J.-S. Liu. Probabilistic Performance Guarantee for Real-Time Tasks with Varying Computation Times. In *Proc. of the Real-Time Technology and Applications Symposium*, pages 164–173, Chicago, Illinois, May 1995.

[20] K. Tindell, A. Burns, and A. J. Wellings. An Extendible Approach for Analyzing Fixed Priority Hard Real-Time Tasks. *Real-Time Systems*, 6:133–151, 1994.