Stochastic Analysis of Real-Time Systems under Preemptive Priority-Driven Scheduling

José María López José Luis Díaz Joaquín Entrialgo
Daniel García

Abstract

Exact stochastic analysis of most real-time systems under preemptive priority-driven scheduling is unaffordable in current practice. Even assuming a periodic and independent task model, the exact calculation of the response time distribution of tasks is not possible except for simple task sets. Furthermore, in practice, tasks introduce complexities such as release jitter, blocking in shared resources, etc., which cannot be handled by the periodic independent task set model.

In order to solve these problems, exact analysis must be abandoned for an approximated analysis. However, in the real-time field, approximations must not be optimistic, i.e. the deadline miss ratios predicted by the approximated analysis must be greater than or equal to the exact ones. In order to achieve this goal, the concept of pessimism needs to be mathematically defined in the stochastic context, and the pessimistic properties of the analysis carefully derived.

This paper provides a mathematical framework for reasoning about stochastic pessimism, and obtaining mathematical properties of the analysis and its approximations. This framework allows us to prove the safety of several proposed approximations and extensions. We analyze and solve some practical problems in the implementation of the stochastic analysis, such as the problem of the finite precision arithmetic or the truncation of the probability functions. In addition, we extend the basic model in several ways, such as the inclusion of shared resources, release jitter or non-preemptive sections.

1. Introduction and previous work

Traditionally, researchers in the field of real-time systems have used pessimistic assumptions in order to make the computational cost of the analysis affordable. For example, the execution time is modelled as a single value, the WCET, which is calculated so as to always be greater than the actual execution time. Techniques such as processor utilization analysis (Liu and Layland, 1973; Lehoczky, 1990) and response time analysis (Tindell et al., 1994), use this simplification. The advantage of this approach is that the computational cost of the analysis is very small. The disadvantage is that it is too pessimistic; while the WCET is very infrequent, the analysis assumes that it does occur in all instances of all tasks, thus giving rise to oversized real-time systems.

Recently, some researchers have suggested using random variables to model the tasks' execution times. Thus, the execution time is not a single value, but a collection of possible values, each one with an associated probability. The distribution of this random variable can be obtained by measurement, or using hybrid techniques (Bernat et al., 2002). The main problem with this approach is that the complexity and computational cost of the stochastic analysis is excessive. In order to reduce this complexity, some approaches to the random stochastic analysis require a special scheduling model to provide isolation between tasks, so that each task can be analyzed independently of other tasks in the system (Atlas and Bestavros, 1998; Abeni and Buttazzo, 2001). Other methods use common scheduling algorithms, but introduce worst-case assumptions to simplify the analysis: the critical instant assumption (Tia et al., 1995; Gardner, 1999; Gardner and Liu, 1999), restrictive load conditions like the heavy traffic condition (Lehoczky, 1996, 1997), or restrictions on preemption (Manolache et al., 2007).

In (Díaz et al., 2002), we proposed a technique for the analysis of periodic and independent tasks without assuming worst-case or restrictive conditions. This allows for the analysis of systems with a maximum system utilization higher than one, whenever the average system utilization remains lower than one. Provided with the exact distributions of the executions time of the tasks, the analysis will output the exact distributions of the response times of the tasks. From these distributions is trivial to obtain schedulability parameters such as the probability of missing any given deadline, etc. However, the exact analysis proposed in (Díaz et al., 2002) applies only to periodic and independent tasks, and the analysis techniques are computationally expensive.

In (Díaz et al., 2004) we investigated the use of pessimism in a stochastic context, in order to reduce the complexity of the analysis and open the door to extensions in the model. This new pessimistic stochastic analysis no longer produces the exact distributions of the response times, but distributions which are *pessimistic* or conservative, in the sense first defined in (Abeni and Buttazzo, 2001). Using the concept of pessimism, some practical problems of the analysis were solved, and the model was extended to allow for blocking in shared resources. However, the approach used in (Díaz et al., 2004) was not general enough, and it required complex proofs. Nor was it flexible enough to investigate other possible extensions for the model, such as including release jitter.

In this paper, we present a reelaboration and extension of (Díaz et al., 2004). Some of the material presented there, such as the formal definition of pessimism and the characterization of pessimistic analysis, is repeated here. However, a new mathematical framework for proving the properties of the stochastic analysis is developed. This new framework allows us to deal with new issues not addressed before, such as the impact of the finite precision arithmetic used by computers in the analysis results, or the inclusion of release jitter in the model. For completeness, the rest of the applications of pessimism are also listed here, but not developed in depth, since the details are published in (Díaz et al., 2004).

The rest of this article is organized as follows. Section 2 describes the basic system model, which exclusively considers periodic independent tasks. Section 3 summarizes the stochastic analysis presented in (Díaz et al., 2002) of the basic system model, introducing the notation and showing one complete example. Section 4 formalizes the concept of pessimism in the stochastic analysis and gives its motivation. Section 5 provides a general framework for reasoning about pessimism and performing safe stochastic approximations. Section 6 presents some applications of these ideas: the problem of priority assignment in the stochastic scenario is solved in Section 6.1, practical issues related to computer aided analysis are addressed in Section 6.2, and finally Section 6.3 extends the stochastic analysis to deal with blocking in shared resources, non-preemptive sections and release jitter. The pros and cons of stochastic analysis are discussed in Section 7. Finally, Section 8 presents our conclusions and future work.

2. Basic system model

The basic task model was introduced and discussed in (Díaz et al., 2004), and it is summarized here for the reader.

The system is composed of a set of N independent periodic tasks $\{\tau_1, \ldots, \tau_i, \ldots, \tau_N\}$. The parameters of each task τ_i are the period, T_i, its initial phase, Φ_i (also called offset), the execution time, $\mathcal{C}_i$, the real-time constraint of the task, expressed by the deadline, D_i, and the maximum allowable probability of deadline misses, M_i.

The execution time, $\mathcal{C}_i$, 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 $F_{\mathcal{C}_i}(x) = \sum_{c=0}^{x} f_{\mathcal{C}_i}(c)$.

Each periodic task τ_i gives rise to an infinite sequence of jobs, $\Gamma_{i,j}$, whose release time $\lambda_{i,j}$ is deterministic (in particular $\lambda_{i,j} = \Phi_i + jT_i$). The response time of a task τ_i is another random variable, denoted by $\mathcal{R}_i$, which is calculated from the response time of all its jobs. Task τ_i is said to be schedulable if $\mathbb{P}\{\mathcal{R}_i{>}D_i\} \leq M_i$. The reader should note that $M_i = 0$ in the deterministic analysis of hard real-time systems.

The scheduling policy we assume is a general, preemptive, priority-driven policy that assigns a static priority, P_j, to each job, Γ_j, and schedules jobs according to this priority. We are not concerned with the policy used to assign priorities to jobs, as long as the priority assignment is repeated periodically each hyperperiod. This model includes well known fixed priority policies such as *Deadline Monotonic* (DM), as well as the optimal priority assignment presented in section 6.1, and non-fixed priority policies such as *Earliest Deadline First* (EDF),

The basic task model will be extended in forthcoming sections, so that random release times in the form of release jitter are allowed. In addition, dependencies will be allowed through blocking in shared resources. Finally, the scheduling policy will be extended to allow for non-preemptive sections.

3. Stochastic Analysis

The stochastic analysis described in this section was first presented in (Díaz et al., 2002). A summary of this work with improved notation was presented in (Kim et al., 2005), and (Díaz et al., 2004). The basic procedure of the analysis is briefly repeated here, and, in order to facilitate its understanding, we give a complete step-by-step example of calculation of a periodic task random response time.

In order to simplify the notation in the following, we will not track the task to which each job belongs, but will use a single subindex, j, when referring to a job Γ_j and the parameters of that job such as its release time, λ_j, random execution time, $\mathcal{C}j$, priority, P_j, etc. This single index refers to the order of the job in the infinite sequence of jobs generated by all the instances of all tasks under analysis, assuming this sequence to be ordered by release time, that is, $j_1 < j_2 \Rightarrow \lambda_{j_1} \leq \lambda_{j_2}$.

The (random) response time of a job Γ_j is given by

$$\mathcal{R}_j = \mathcal{W}(\lambda_j) + \mathcal{C}_j + \mathcal{I}_j \tag{1}$$

In this equation all variables are random variables, but note that release times are still deterministic. $\mathcal{W}(\lambda_j)$ is the backlog at time λ_j; it represents the workload of priorities P_j and higher that have not yet been processed immediately prior to λ_j, the release time of Γ_j. Term $\mathcal{C}_j$ is the execution time of job Γ_j and $\mathcal{I}_j$ is the interference in Γ_j of all the jobs of higher priority than job Γ_j released at or after job Γ_j.

The backlog at the release time of any job Γ_j, denoted $\mathcal{W}(\lambda_j)$, can be calculated using the following iterative equation:

$$\begin{aligned} &\mathcal{W}(\lambda_{k_0}) = 0 \\ &\mathcal{W}(\lambda_k) = \text{SHRINK}(\mathcal{W}(\lambda_{k-1}) + \mathcal{C}_{k-1}, \lambda_k - \lambda_{k-1}) \quad \text{for } k > k_0,\ P_k \geq P_j \end{aligned} \tag{2}$$

λ_{k_0} being the release time of the first job with priority greater than or equal to P_j. Equation (2) starts with zero backlog and continues with a new iteration for each higher priority job released before Γ_j. Note that each iteration requires the addition of two random variables ($\mathcal{W}(\lambda_{k-1}) + \mathcal{C}_{k-1}$), which produces a new random variable whose probability function is obtained by convolution. Finally, the function $\textsc{shrink}()$ is applied to this random variable. $\textsc{shrink}(\cdot)$ produces a new random variable whose probability function is obtained by Equation (3):

$$f_{\textsc{shrink}(\mathcal{W},\Delta)}(x) = \begin{cases} 0 & \text{if } x < 0 \\ \sum_{w=-\infty}^{0} f_{\mathcal{W}}(w+\Delta) & \text{if } x = 0 \\ f_{\mathcal{W}}(x+\Delta) & \text{if } x > 0 \end{cases} \tag{3}$$

Graphically, this transformation consists of shifting the original PF in Δ units to the left, and accumulating in zero all the negative values after the shifting.

Once the backlog at the release time of Γ_j is known, it is added to its execution time $\mathcal{C}_j$, as given by Equation (1). The PF of the addition is calculated by convolving the PF of both random variables. This provides the *partial response time* $\mathcal{R}_j^{[0,\lambda_{j+1}-\lambda_j]} = \mathcal{W}(\lambda_j) + \mathcal{C}_j$, where λ_{j+1} is the release time of the first high priority job released at or after Γ_j. This partial response time $\mathcal{R}_j^{[0,\lambda_{j+1}-\lambda_j]}$ is equal to the final response time if there are no interfering jobs after λ_j, which is the case of the highest priority task under fixed priority scheduling. Otherwise, partial response times are random variables whose PF coincide with that of the final response time in the range shown in the superindex, therefore, $\mathbb{P}\{\mathcal{R}_j{=}r\} = \mathbb{P}\{\mathcal{R}_j^{[0,\lambda_{j+1}-\lambda_j]}{=}r\}$ for all $r \in [0, \lambda_{j+1} - \lambda_j]$.

Iterating over the sequence of interfering jobs, each new iteration extends the range in which the partial response time coincides with the final response time. When this range includes the deadline D_j, the iterative procedure can be stopped. Each new partial response time of Γ_j is calculated using Equation (4)

$$\begin{aligned} \mathcal{R}_j^{[0,\lambda_{j+1}-\lambda_j]} &= \mathcal{W}(\lambda_j) + \mathcal{C}_j \\ \mathcal{R}_j^{[0,\lambda_{k+1}-\lambda_j]} &= \mathrm{AF}(\mathcal{R}_j^{[0,\lambda_k-\lambda_j]}, \lambda_k - \lambda_j, \mathcal{C}_k) \quad \text{for } k > j,\ P_k > P_j \end{aligned} \tag{4}$$

The function AF() applies to two random variables $\mathcal{R}$ and $\mathcal{C}$ (partial response time and computation time) and one integer Δ (the time interval $\lambda_k - \lambda_j$),and produces as result a new random variable, whose PF is given by a partial convolution defined in Equation (5)

$$f_{\mathrm{AF}(\mathcal{R},\Delta,\mathcal{C})}(x) = \begin{cases} f_{\mathcal{R}}(x) & \text{for } x \le \Delta \\ \sum_{i=\Delta+1}^{\infty} f_{\mathcal{R}}(i) \cdot f_{\mathcal{C}}(x-i) & \text{for } x > \Delta \end{cases} \tag{5}$$

The partial convolution leaves the PF of $\mathcal{R}$ in interval $[0,\Delta]$ unchanged, convolving the PF of $\mathcal{R}$ with that of $\mathcal{C}$ only in interval $[\Delta+1,\infty]$.

The iterative procedure can stop when $D_j < \Delta = \lambda_{k+1} - \lambda_j$, and the probability of deadline misses for the job Γ_j can be obtained as $\mathbb{P}\{\mathcal{R}_j{>}D_j\} = 1 - \sum_{k=0}^{k=D_j} \mathbb{P}\{\mathcal{R}_j^{[0,\Delta]}{=}k\}$.

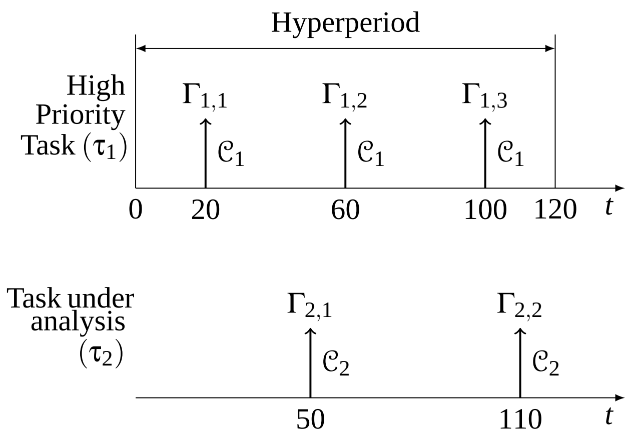
The probability of a task missing its deadline is calculated by averaging the probabilities of all its jobs missing that deadline, but the number of these jobs is infinite. However, when $\bar{U} < 1$ the system reaches a periodic steady-state (Díaz et al., 2002). In the steady-state the probability of a job missing its deadline becomes constant for the same job released one, two or any number of hyperperiods later. Thus, the probability of a task missing its deadline can be calculated by considering only those of its jobs released in one hyperperiod, namely, the steady-state hyperperiod.

Figure 1 provides a simple example, in which the random response time of task τ_2 is calculated. Note that in this figure jobs are subindexed according to the task to which they belong, but in the previous exposition of the analysis we used a single index, representing the order of arrival. Following the timeline from left to right in the figure, we discover that this ordering is $\Gamma_{1,1}$, $\Gamma_{2,1}$, $\Gamma_{1,2}$, $\Gamma_{1,3}$ and $\Gamma_{2,2}$, which would then become Γ_1, Γ_2, $\ldots$, Γ_5 respectively using the single index notation. This single index is a convenient way to specify the order in which the previous equations should be applied. However, for the sake of clarity, we maintain the more classical notation with two subindexes (task number, and job within the task number) in the figure and the explanations.

The steady-state backlog at the release time of jobs $\Gamma_{2,1}$ and $\Gamma_{2,2}$, which make up task τ_2, is calculated by iterating until convergence is reached (this convergence is guaranteed since the system's $\bar{U}$ is 0.94167, less than 1). A different approach would be to model the backlog as a Markovian process, defined by its Markov matrix, and calculate its eigenvalues. Next, the random response time of jobs $\Gamma_{2,1}$ and $\Gamma_{2,2}$ is calculated considering all the high priority jobs released after the arrival of the jobs, but before their absolute deadline. This step produces the cumulative distribution functions (CDFs) of these jobs. Finally, the CDF of the response time of task τ_2 is calculated by averaging the CDF of response times of $\Gamma_{2,1}$ and $\Gamma_{2,2}$.

Figure 2 shows the CDF of the backlog at the release time of $\Gamma_{2,1}$, at times $t = 20$, $t = 140$ (after one hyperperiod), time $t = 260$ (after two hyperperiods) and at the steady-state. Parameters in the example have been chosen to improve the visualization of backlogs, making convergence slower than usual. In addition, Figure 2 depicts the CDF of the random response times of $\Gamma_{2,1}$ and $\Gamma_{2,2}$, as well as their average. Introducing the deadline of task τ_2 as the abscissa in the CDF of the average, the probability of meeting the deadline is obtained.

For the case of EDF scheduling, the only difference is the computation of the initial backlog for each job. Once this initial backlog is obtained, the computation of the response time of the job follows the same algorithm outlined above. In order to obtain the initial backlog of a given job Γ_j, released at instant λ_j, we must consider the workload generated for all previous jobs with a greater priority, which in the case of EDF means with an earlier absolute deadline. This can be done easily by computing the steady-state backlog generated by all jobs in the system. From this system backlog, the job-level backlog for each job in the hyperperiod can be easily derived. In fact, the EDF case is simpler than the fixed-priority one, because the system backlog is computed only once, and valid for all jobs in the hyperperiod, while for the fixed-priority case it is necessary to compute



TASK PARAMETERS

c	$f_{\mathcal{C}_1}(c) = f_{\mathcal{C}_2}(c)$
10	0.1 ■
20	0.4 ■■■■
21	0.2 ■■
22	0.2 ■■
50	0.1 ■

$T_1 = 40$ $T_2 = 60$

$\Phi_1 = 20$ $\Phi_2 = 50$

$P_1 > P_2$ $D_2 = 90$

CALCULATION OF THE STEADY-STATE BACKLOG

$$\begin{aligned}
\mathcal{W}(20) &= 0\\
\mathcal{W}(50) &= \text{SHRINK}(\mathcal{W}(20) + \mathcal{C}_1,\ 50-20)\\
\mathcal{W}(60) &= \text{SHRINK}(\mathcal{W}(50) + \mathcal{C}_2,\ 60-50)\\
\mathcal{W}(100) &= \text{SHRINK}(\mathcal{W}(60) + \mathcal{C}_1,\ 100-60)\\
\mathcal{W}(110) &= \text{SHRINK}(\mathcal{W}(100) + \mathcal{C}_1,\ 110-100)\\
\mathcal{W}(140) &= \text{SHRINK}(\mathcal{W}(110) + \mathcal{C}_2,\ 140-110)
\end{aligned}$$

Compute one hyperperiod

$$\begin{aligned}
\mathcal{W}(20) &= \mathcal{W}(140)\\
\mathcal{W}(50) &= \text{SHRINK}(\mathcal{W}(20) + \mathcal{C}_1,\ 50-20)\\
&\vdots
\end{aligned}$$

Iterate

CALCULATION OF THE RESPONSE TIME OF JOB $\Gamma_{2,1}$

$$\begin{aligned}
\mathcal{R}_{2,1}^{[0,60-50]} &= \mathcal{W}(50) + \mathcal{C}_2 = \text{AF}(\mathcal{W}(50), 0, \mathcal{C}_2)\\
\mathcal{R}_{2,1}^{[0,100-50]} &= \text{AF}(\mathcal{R}_{2,1}^{[0,60-50]}, 100-50, \mathcal{C}_1)\\
\mathcal{R}_{2,1}^{[0,140-50]} &= \text{AF}(\mathcal{R}_{2,1}^{[0,140-50]}, 140-50, \mathcal{C}_1) \qquad \Longrightarrow \text{STOP (since } 140-50 \geq D_2)\\
F_{\mathcal{R}_{2,1}^{[0,t]}} &= \sum_{r=0}^{t} f_{\mathcal{R}_{2,1}^{[0,140-50]}}(t) \qquad \text{WITH} \quad t \leq 140-50 = 90
\end{aligned}$$

CALCULATION OF THE RESPONSE TIME OF JOB $\Gamma_{2,2}$

$$\begin{aligned}
\mathcal{R}_{2,2}^{[0,140-110]} &= \mathcal{W}(110) + \mathcal{C}_2 = \text{AF}(\mathcal{W}(110), 0, \mathcal{C}_2)\\
\mathcal{R}_{2,2}^{[0,180-110]} &= \text{AF}(\mathcal{R}_{2,2}^{[0,140-110]}, 180-110, \mathcal{C}_1)\\
\mathcal{R}_{2,2}^{[0,220-110]} &= \text{AF}(\mathcal{R}_{2,2}^{[0,180-110]}, 220-110, \mathcal{C}_2) \qquad \Longrightarrow \text{STOP (since } 220-110 \geq D_2)\\
F_{\mathcal{R}_{2,2}^{[0,t]}} &= \sum_{r=0}^{t} f_{\mathcal{R}_{2,2}^{[0,220-110]}}(t) \qquad \text{WITH} \quad t \leq 220-110 = 110
\end{aligned}$$

AVERAGING THE RESPONSE TIMES OF $\Gamma_{2,1}$ AND $\Gamma_{2,2}$

$$F_{\mathcal{R}_2^{[0,t]}} = 0.5(F_{\mathcal{R}_{2,1}^{[0,t]}} + F_{\mathcal{R}_{2,2}^{[0,t]}})$$

CALCULATION OF THE PROBABILITY OF MEETING THE DEADLINE

$$\mathbb{P}\{\mathcal{R}_2 \leq D_2\} = F_{\mathcal{R}_2^{[0,t]}}(D_2) = 0.496$$

Figure 1: Example of calculation of the random response time of a task.

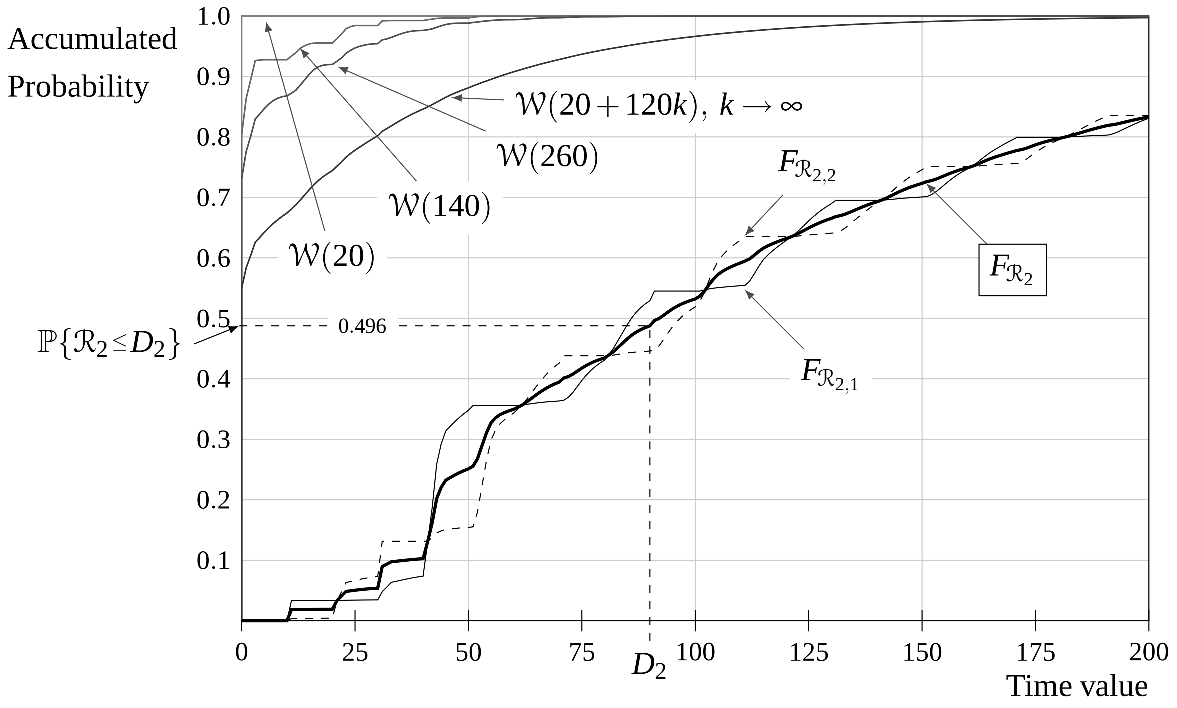


Figure 2: Cumulative distribution functions of backlogs and response times for τ_2.

one system backlog for each priority level. More details and one example can be found in (Díaz and López, 2007).

4. Pessimism in the stochastic context

In the deterministic analysis of real-time systems, all approximations are *pessimistic* in the sense that the response times obtained by the approximated analysis are guaranteed to be greater than the exact response times of the system. However, in stochastic analysis, the response times are random variables and real-time constraints are expressed in terms of probabilities of deadline misses. With these ideas in mind, we will define the relationship "*greater than*" among random variables. In this way, we will be able to reason about the pessimism in the stochastic analysis, and say that certain types of approximations are pessimistic in the sense that the random response times given by the analysis are "*greater than*" the real random response times. Thus, an approximation which leads to a pessimistic analysis is a safe approximation.

The concepts in this section were first introduced in (Díaz et al., 2004). They are repeated and refined here since they are important to understand the rest of the article.

Definition 1. *Let $\mathcal{X}$ and $\mathcal{X}'$ be two random variables. We state that $\mathcal{X}'$ is greater than $\mathcal{X}$ (or alternatively, $\mathcal{X}$ is less than $\mathcal{X}'$), and denote it by $\mathcal{X}' \succ \mathcal{X}$ (alternatively, $\mathcal{X} \prec \mathcal{X}'$) if $\mathbb{P}\{\mathcal{X}' \le D\} \le \mathbb{P}\{\mathcal{X} \le D\}$ for any D, and the two random variables are not identically distributed.*

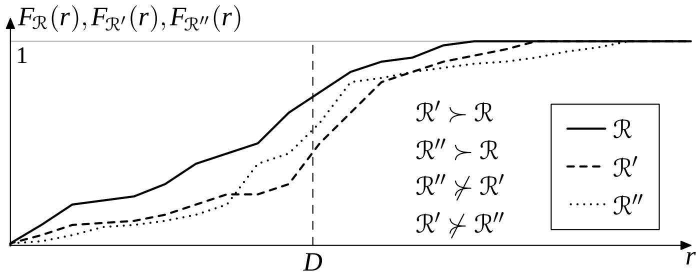


Figure 3: Graphical interpretation of the "greater than" relationship among response time distributions.

Note that, if $\mathcal{X}$ represents an exact response time, and $\mathcal{X}'$ is an approximation, the above definition guarantees that the probability of deadline misses for $\mathcal{X}'$ is greater than that of $\mathcal{X}$, for any deadline D. We state that $\mathcal{X}'$ is a pessimistic approximation, and therefore, the stochastic analysis providing $\mathcal{X}'$ is pessimistic.

The relationship *greater than* between two random response times is a stochastic order, analogous to the concept of *first stochastic order*, defined in statistics (Levy, 1992), and also used informally in the context of real-time systems in (Abeni and Buttazzo, 2001).

Likewise, we state that random variable $\mathcal{X}_2$ is greater than or equal to the random variable $\mathcal{X}_1$ ($\mathcal{X}_2 \succcurlyeq \mathcal{X}_1$ or $\mathcal{X}_1 \preccurlyeq \mathcal{X}_2$), if $F_{\mathcal{X}_2}(x) \leq F_{\mathcal{X}_1}(x)$. Note that the restriction "not identically distributed" has been removed.

The definition has a simple graphical interpretation: random variable $\mathcal{X}'$ is greater than random variable $\mathcal{X}$ if the cumulative probability function (CDF) of $\mathcal{X}'$ is always below the CDF of $\mathcal{X}$. Figure 3 graphically compares three response time random distributions. In this figure, $\mathcal{R}' \succ \mathcal{R}$, i.e, $\mathcal{R}'$ is greater than $\mathcal{R}$, since $F_{\mathcal{R}'}(\cdot)$ is below $F_{\mathcal{R}}(\cdot)$. Therefore, the probability of missing any deadline D calculated from $\mathcal{R}'$ is greater than when calculated from $\mathcal{R}$. However, $F_{\mathcal{R}''}(\cdot)$ and $F_{\mathcal{R}'}(\cdot)$ cross, so it is not true that $\mathcal{R}'' \succ \mathcal{R}'$ nor $\mathcal{R}'' \prec \mathcal{R}'$.

Once a way of comparing random variables has been formally defined, we can also define other useful concepts, such as the *supremum* (a random variable which is *greater than* any other in a given set), and the *infimum* (a random variable which is *less than* any other in a given set).

Definition 2. *Given a set of random variables $\{\mathcal{X}_i\}$, we define the supremum of that set, denoted* $\sup\{\mathcal{X}_i\}$*, as the random variable with CDF*

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

Definition 3. *Given a set of random variables $\{\mathcal{X}_i\}$, we define the infimum of that set, denoted* $\inf\{\mathcal{X}_i\}$*, as the random variable with CDF*

$$F_{\sup\{\mathcal{X}_i\}}(x) = \max_i F_{\mathcal{X}_i}(x) \tag{7}$$

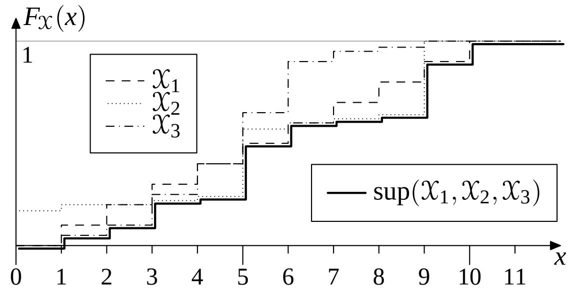


Figure 4: Construction of the supremum of a set of random variables

Note that the *supremum* (and the *infimum*) of a set is a new random variable, not necessarily in the set, defined by an artificially constructed probability function. Figure 4 graphically shows how this construction is achieved, by taking the minimum of all $F_{\mathcal{X}_i}(\cdot)$ (note that the plot of the supremum has been slightly shifted down for better legibility of the figure). By construction, $\sup\{\mathcal{X}_i\} \succcurlyeq \mathcal{X}_i$, and $\inf\{\mathcal{X}_i\} \preccurlyeq \mathcal{X}_i$ for all i. Thus, the supremum of a set of random variables is analogous to the maximum of a set of real numbers, and the infimum is analogous to the minimum. These concepts will be useful when translating classical results of the deterministic analysis to the stochastic scenario.

5. Pessimism preservation in the analysis

The main idea in this section is to find out how the mathematical properties of the relationship "*greater than*" interact with the mathematical procedures used in the analysis, leading to conclusions such as "if pessimistic execution times for the tasks are introduced in the model, pessimistic response times would be obtained in the analysis". Although some of these properties were already presented in (Díaz et al., 2004), in this paper we introduce a much more general approach.

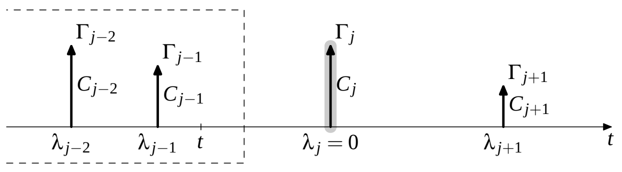
First, in section 5.1 we will provide a general framework to deal with pessimistic approximations in *deterministic* analysis, since this is the basis of pessimistic approximations in the stochastic analysis. In fact, stochastic analysis can be seen as a summary of all the possible deterministic scenarios making up the sample space.

Next, in section 5.2 we will provide a general rule that transforms pessimistic deterministic approximations into pessimistic stochastic approximations. Finally, using this rule and the deterministic approximations of Section 5.1, the basic stochastic approximations are presented at the end of Section 5.2.

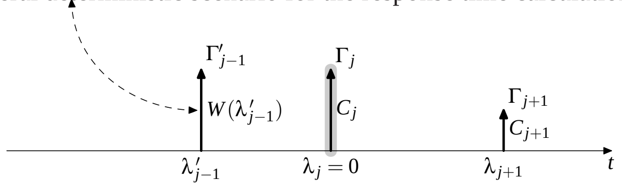
5.1. Pessimistic approximations in deterministic analysis

The objective of this subsection is to define pessimistic approximations in the deterministic analysis, which we will transform later into their equivalent stochastic approximations. Throughout this subsection, we will use the word "increases" with the meaning of “does not change or increases”, for the sake of brevity.

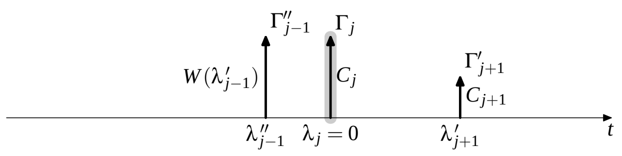
Figure 5(a) depicts the general deterministic scenario under priority-driven scheduling, in which the response time of job Γ_j is to be calculated.



(a) General deterministic scenario for the response time calculation of Γ_j.



(b) Equivalent scenario after replacing jobs Γ_{j-2} and Γ_{j-1} by the resultant backlog at λ'_{j-1}.



(c) Delaying the release time of Γ'_{j-1} from λ'_{j-1} to $\lambda''_{j-1} \leq \lambda_j$ and advancing the release time of Γ_{j+1} from λ_{j+1} to $\lambda'_{j+1} \geq \lambda_j$ increases the response time of Γ_j.

Figure 5: Approximations in deterministic analysis.

For the time being, we will assume that priorities are independent of release times. This assumption is valid under fixed priority scheduling, but not under EDF scheduling. Later, this restriction will be removed.

None of the jobs of priority less than Γ_j have any influence on the response time of job Γ_j. In order to simplify the notation, we assume all these jobs were previously removed. In addition, release times are relative to the release time of Γ_j. Therefore, the release time of Γ_j, the job under analysis, becomes $\lambda_j = 0$.

In general, the response time of Γ_j under preemptive priority-driven scheduling is exclusively a function of all execution and release times. This relationship can be expressed as

$$R_j = F(\dots,(\lambda_{j-1},C_{j-1}),C_j,(\lambda_{j+1},C_{j+1}),\dots) \tag{8}$$

It is well-known in deterministic analysis that increasing the execution time of one or more jobs in the system increases the response time of any job, i.e, $F(\cdot)$ is a monotonic increasing function of any C_k, with $k = (\dots, j-1, j, j+1, \dots)$. Therefore, any approximation of execution times that provides higher execution times than the exact ones is a pessimistic approximation.

A direct consequence of the previous result is that adding a new job, Γ_k, to the job set introduces pessimism. The proof is simple by comparing the new job set, including Γ_k, with the original job set, after adding a job of zero execution time released at the same time as Γ_k.

It is possible to simplify the job set and therefore Equation (8), as shown in Figure 5(b). All the jobs released before t, with $t \leq 0$, can be replaced by an equivalent job of release time t and execution time equal to the backlog at time t, denoted $W(t)$. In general, the equivalent job released at t summarizes the whole history before t. Now, R_j can be calculated from

$$R_j = F((t, W(t)), \ldots, (\lambda_{j-1}, C_{j-1}), C_j, (\lambda_{j+1}, C_{j+1}) \ldots) \quad \text{with } t < \lambda_j = 0 \tag{9}$$

Instant t is renamed λ'_{j-1} in Figure 5(b), since the equivalent job is the nearest job released before Γ_j.

Since $F(\cdot)$ is a monotonic increasing function of the execution times coming from real or equivalent tasks, any approximation that substitutes the exact backlog, $W(t)$, at any time $t < 0$ for a higher backlog increases the response time R_j, and so is a pessimistic approximation. For example, this approximation is carried out implicitly when jobs missing their deadlines are forced to finish in the real system, but are not removed in the analysis. Thus, we conclude that $F(\cdot)$ is a monotonic increasing function of $W(t)$, for any $t < 0$.

Let us focus now on pessimistic approximations related to release times. Let Γ_k be a job released before Γ_j, i.e, with $\lambda_k < 0$. If the release time of Γ_k is delayed to λ'_k and in the new situation Γ_k is not released after Γ_j, then the response time of Γ_j increases. This transformation is shown in Figure 5(c) for job Γ'_{j-1}, which becomes Γ''_{j-1}. The proof is simple, bearing in mind that the response time of Γ_j can be expressed as

$$R_j = W(0) + C_j + I_j \tag{10}$$

where $W(0)$ is the backlog at the release time of Γ_j, C_j is its execution time, and I_j is the interference on Γ_j of the jobs released at or after Γ_j. Making one or more of the jobs released before Γ_j closer to Γ_j on the left, delays their processing, increments the backlog $W(0)$ and, indirectly, the interference I_j, and thus increments the response time of Γ_j. Therefore, we conclude that $F(\cdot)$ is a monotonic increasing function of any λ_k, whenever $\lambda_k < 0$.

Under EDF scheduling the priority of a job decreases as its release time is delayed, since its absolute deadline is also delayed. In this case, the priority of the job should be calculated with regard to the initial release time, instead of the delayed release time. Otherwise, the analysis may not be pessimistic.

An analogous result is derived for any job Γ_k released after Γ_j, i.e, fulfilling $\lambda_k > 0$. If the release time of Γ_k is advanced to $\lambda'_k < \lambda_k$, and in the new situation Γ_k is not released before Γ_j, then the response time of Γ_j will increase. The transformation is shown in Figure 5(c) for job Γ_{j+1}, which becomes Γ'_{j+1}. This result can be proved again from Equation (10). Making one or more of the jobs released after Γ_j closer to Γ_j without crossing to its left, increments the interference I_j, and thus increments the response time of Γ_j. Therefore, we conclude that $F(\cdot)$ is a monotonic decreasing function of any λ_k, whenever $\lambda_k > 0$.

Under EDF scheduling the priority of a job increases as its release time is advanced. Thus, the previous approximation is also pessimistic under EDF scheduling.

An alternate expression to Equation (8) is Equation (11) below, which summarizes the whole history before any time λ_k, with $\lambda_k \geq \lambda_j = 0$, by the partial response time $R_j^{[0,\lambda_k]}$.

$$R_j = G(R_j^{[0,\lambda_k]}, (\lambda_k, C_k), (\lambda_{k+1}, C_{k+1}), \dots) \quad \text{with } \lambda_k \geq \lambda_j = 0 \tag{11}$$

Term $R_j^{[0,\lambda_k]}$ is a partial response time of Γ_j, which assumes that jobs Γ_k and subsequent do no exist, so it is a valid response time for Γ_j in interval $[0, \lambda_k]$. It is well-known in deterministic analysis that by artificially increasing any partial response time of the task, the final response time of the job increases, since it increments the interference of job Γ_k and subsequent jobs. Therefore, function $G(\cdot)$ is a monotonic increasing function of $R_j^{[0,\lambda_k]}$ with $\lambda_k \geq 0$. Thus, any approximation that provides partial response times greater than the exact ones is a pessimistic approximation.

Now, let us transform these results regarding pessimism in the deterministic analysis into their stochastic counterparts.

5.2. Pessimistic approximations in stochastic analysis

A random response time $\mathcal{R}'$ is pessimistic when it is greater than the exact one, $\mathcal{R}$, i.e, when $\mathcal{R}' \succ \mathcal{R}$. The problem is how to perform approximations within stochastic analysis parameters, while guaranteeing that the resultant random response times are pessimistic or identically distributed, thus making the approximated stochastic analysis safe.

Proposition 1 is a simple but powerful result, which translates deterministic approximations, like those given in Section 5.1, into their stochastic counterparts.

Proposition 1. *Let $y = g(x_1, \dots, x_k, \dots, x_m)$ be a real-valued function of real variables $(x_1, \dots, x_k, \dots, x_m)$ and $\mathcal{Y}$ a random variable evaluated as $\mathcal{Y} = g(\mathcal{X}_1, \dots, \mathcal{X}_k, \dots, \mathcal{X}_m)$, where $(\mathcal{X}_1, \dots, \mathcal{X}_k, \dots, \mathcal{X}_m)$ are discrete and independent random variables. Let $\mathcal{Y}' = g(\mathcal{X}_1, \dots, \mathcal{X}'_k, \dots, \mathcal{X}_m)$ and $\mathcal{Y}'' = y(\mathcal{X}_1, \dots, \mathcal{X}''_k, \dots, \mathcal{X}_m)$, where $\mathcal{X}''_k \succ \mathcal{X}'_k$. The following asserts hold:*

- *If $g(x_1, \dots, x_k, \dots, x_m)$ is a monotonic increasing function of x_k, then it follows that $\mathcal{Y}'' \succcurlyeq \mathcal{Y}'$.*

- *If $g(x_1, \dots, x_k, \dots, x_m)$ is a monotonic decreasing function of x_k, then it follows that $\mathcal{Y}'' \preccurlyeq \mathcal{Y}'$.*

Proof. The CDF of random variables $\mathcal{Y}'$ and $\mathcal{Y}''$ can be calculated from

$$F_{\mathcal{Y}'}(p) = \mathbb{P}\{\mathcal{Y}' \le p\} = \sum_{q=-\infty}^{+\infty} \mathbb{P}\{\mathcal{Y}' \le p | \mathcal{X}'_k = q\} \cdot \mathbb{P}\{\mathcal{X}'_k = q\}$$
$$F_{\mathcal{Y}''}(p) = \mathbb{P}\{\mathcal{Y}'' \le p\} = \sum_{q=-\infty}^{+\infty} \mathbb{P}\{\mathcal{Y}'' \le p | \mathcal{X}''_k = q\} \cdot \mathbb{P}\{\mathcal{X}''_k = q\}$$

Probabilities $\mathbb{P}\{\mathcal{Y}' \le p | \mathcal{X}'_k = q\}$ and $\mathbb{P}\{\mathcal{Y}'' \le p | \mathcal{X}''_k = q\}$ only depend on random variables $\mathcal{X}_j$, with $j \neq k$, so they are identical for $F_{\mathcal{Y}'}(p)$ and $F_{\mathcal{Y}''}(p)$. Let us rename these probabilities $(\dots, w_1, w_2, \dots, w_q, \dots)$, where $w_q = \mathbb{P}\{\mathcal{Y}' \le p | \mathcal{X}'_k = q\} = \mathbb{P}\{\mathcal{Y}'' \le p | \mathcal{X}''_k = q\}$. Therefore, probabilities $\mathbb{P}\{\mathcal{Y}' \le p\}$ and $\mathbb{P}\{\mathcal{Y}'' \le p\}$ are calculated as a weighted sum of probability functions

$$(\dots, \mathbb{P}\{\mathcal{X}'_k = 1\}, \mathbb{P}\{\mathcal{X}'_k = 2\}, \dots, \mathbb{P}\{\mathcal{X}'_k = q\}, \dots), \text{ and}$$
$$(\dots, \mathbb{P}\{\mathcal{X}''_k = 1\}, \mathbb{P}\{\mathcal{X}''_k = 2\}, \dots, \mathbb{P}\{\mathcal{X}''_k = q\}, \dots)$$

respectively, where the weights are given by $(\dots, w_1, w_2, \dots, w_q, \dots)$, with $\sum_{q=-\infty}^{\infty} w_q = 1$.

The proposition states that $\mathcal{X}''_k \succ \mathcal{X}'_k$, so it follows that $\mathbb{P}\{\mathcal{X}''_k \le q\} \le \mathbb{P}\{\mathcal{X}'_k \le q\}$, i.e, probabilities in $\mathcal{X}'_k$ are located at values of q lower than in $\mathcal{X}''_k$.

If $y(x_1, \dots, x_k, \dots, x_m)$ is a monotonic increasing function of x_k, then it can be proved that $(\dots \ge w_1 \ge w_2 \ge \dots \ge w_q \ge \dots)$. Therefore, the weighted sum gives more weight to low values of q than to high values of q and it follows that $\mathbb{P}\{\mathcal{Y}' \le p\} \ge \mathbb{P}\{\mathcal{Y}'' \le p\}$ for any p, i.e, $\mathcal{Y}'' \succcurlyeq \mathcal{Y}'$.

If $y(x_1, \dots, x_k, \dots, x_m)$ is a monotonic decreasing function of x_k, then it can be proved that $(\dots \le w_1 \le w_2 \le \dots \le w_q \le \dots)$. Therefore, the weighted sum gives higher weight to high values of q than to low values of q and it follows that $\mathbb{P}\{\mathcal{Y}' \le p\} \le \mathbb{P}\{\mathcal{Y}'' \le p\}$ for any p, i.e, $\mathcal{Y}'' \preccurlyeq \mathcal{Y}'$. □

To illustrate Proposition 1, let us consider a random experiment in which we throw two dice and calculate the sum of the values provided by both dice, which is modelled by random variable $\mathcal{Y}$. The result provided by either die is a number in $[1, 6]$, the first die modelled by $\mathcal{X}_1$ and the second by $\mathcal{X}_2$, so $\mathcal{Y} = \mathcal{X}_1 + \mathcal{X}_2$. In this case, the real-valued function is simply $y = g(x_1, x_2) = x_1 + x_2$. If the first die is substituted for another die that provides a result in $[2, 7]$, modelled by random variable $\mathcal{X}'_1$, then we can state that $\mathcal{X}'_1 \succ \mathcal{X}_1$ and then $\mathcal{Y}' = \mathcal{X}'_1 + \mathcal{X}_2 \succcurlyeq \mathcal{Y} = \mathcal{X}_1 + \mathcal{X}_2$, since $y(x_1, x_2) = x_1 + x_2$ is a monotonic increasing function of x_1.

Next, Theorem 1 provides seven pessimistic stochastic approximations, which are proved by applying Proposition 1 to the pessimistic approximations for the deterministic analysis.

Theorem 1. *Let* $\{\dots, \Gamma_{j-1}, \Gamma_j, \Gamma_{j+1}, \dots\}$ *be a set of jobs made up of* Γ_j *and all the jobs of higher priority than* Γ_j*, of random release times* $\{\dots, \mathcal{L}_{j-1}, \mathcal{L}_j = 0, \mathcal{L}_{j+1}, \dots\}$ *relative to the release of* Γ_j*, and random execution times* $\{\dots, \mathcal{C}_{j-1}, \mathcal{C}_j, \mathcal{C}_{j+1}, \dots\}$*. Let* $\mathcal{R}_j$ *be the random response time of job* Γ_j*. If any of the following transformations is performed:*

1. *The random execution time of any job Γ_k is substituted for another random execution time $\mathcal{C}'_k$, such that $\mathcal{C}'_k \succ \mathcal{C}_k$.*

2. *A high priority job is added to the system.*

3. *The random backlog at any relative time $t \leq 0$, denoted $\mathcal{W}(t)$, is substituted for another random backlog $\mathcal{W}'(t)$, such that $\mathcal{W}'(t) \succ \mathcal{W}(t)$.*

4. *The partial response time $\mathcal{R}_j^{[0,\mathcal{L}_k]}$, at relative time $\mathcal{L}_k \succcurlyeq 0$, is substituted for another partial response time $\mathcal{R}_j'^{[0,\mathcal{L}_k]}$, such that $\mathcal{R}_j'^{[0,\mathcal{L}_k]} \succ \mathcal{R}_j^{[0,\mathcal{L}_k]}$.*

5. *The relative random release time $\mathcal{L}_k$, of any job Γ_k with $\mathcal{L}_k \prec 0$, i.e, released before Γ_j, is substituted for another relative random release time $\mathcal{L}'_k$, such that $\mathcal{L}_k \prec \mathcal{L}'_k \preccurlyeq 0$.*

6. *The relative random release time $\mathcal{L}_k$, of any job Γ_k with $\mathcal{L}_k \succ 0$, i.e, released after Γ_j, is substituted for another relative random release time $\mathcal{L}'_k$, such that $0 \preccurlyeq \mathcal{L}'_k \prec \mathcal{L}_k$.*

7. *The relative random release time $\mathcal{L}_k$ of any job Γ_k is substituted by a relative random release time $\mathcal{L}'_k = 0$.*

then the new random response time of Γ_j, denoted $\mathcal{R}'_j$, fulfils $\mathcal{R}'_j \succcurlyeq \mathcal{R}_j$, so all the above transformations can be considered pessimistic approximations.

Proof. Equations (12), (13) and (14) are the stochastic counterparts of Equations (8), (9) and (11), respectively.

$$\mathcal{R}_j = F(\ldots,(\mathcal{L}_{j-1},\mathcal{C}_{j-1}),\mathcal{C}_j,(\mathcal{L}_{j+1},\mathcal{C}_{j+1}),\ldots) \tag{12}$$

$$\mathcal{R}_j = F((t,\mathcal{W}(t)),\ldots,(\mathcal{L}_{j-1},\mathcal{C}_{j-1}),\mathcal{C}_j,(\mathcal{L}_{j+1},\mathcal{C}_{j+1})\ldots), \text{ with } t \preccurlyeq \mathcal{L}_j = 0 \tag{13}$$

$$\mathcal{R}_j = G(\mathcal{R}_j^{[0,\mathcal{L}_k]},(\mathcal{L}_k,\mathcal{C}_k),(\mathcal{L}_{k+1},\mathcal{C}_{k+1}),\ldots), \text{ with } \mathcal{L}_k \succcurlyeq \mathcal{L}_j = 0 \tag{14}$$

The proof of statements 1 to 6 is direct from Proposition 1 and Equations (8), (9), (11), (12), (13) and (14), taking into account the following deterministic results provided in Section 5.1:

- $F(\cdot)$ in Equations (8) and (9) is a monotonic increasing function of: C_k, $W(t)$ with $t < 0$, and λ_k with $\lambda_k < 0$.

- $F(\cdot)$ in Equation (8) is a monotonic decreasing function of λ_k with $\lambda_k > 0$.

- $G(\cdot)$ is a monotonic increasing function of $R_j^{[0,\lambda_k]}$ with $\lambda_k \geq 0$.

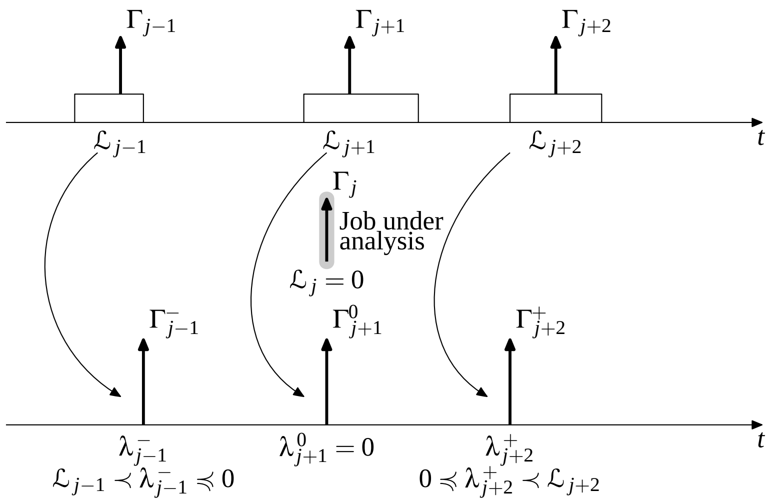


Figure 6: Illustration of the transformations 5, 6 and 7 in Theorem 1.

Statement 7 requires more elaboration. Let us divide the sample space in three subspaces: $\mathcal{L}_k \prec 0$, $\mathcal{L}_k = 0$ and $\mathcal{L}_k \succ 0$. It follows that

$$\begin{aligned}\mathbb{P}\{\mathcal{R}_j \le r\} =& \mathbb{P}\{\mathcal{R}_j \le r | \mathcal{L}_k \prec 0\} \cdot \mathbb{P}\{\mathcal{L}_k \prec 0\} + \\ & \mathbb{P}\{\mathcal{R}_j \le r | \mathcal{L}_k = 0\} \cdot \mathbb{P}\{\mathcal{L}_k = 0\} + \\ & \mathbb{P}\{\mathcal{R}_j \le r | \mathcal{L}_k \succ 0\} \cdot \mathbb{P}\{\mathcal{L}_k \succ 0\}\end{aligned}$$

From statements 5 and 6 it follows that $\mathbb{P}\{\mathcal{R}_j \le r | \mathcal{L}_k \prec 0\} \ge \mathbb{P}\{\mathcal{R}_j \le r | \mathcal{L}_k = 0\}$ and $\mathbb{P}\{\mathcal{R}_j \le r | \mathcal{L}_k \succ 0\} \ge \mathbb{P}\{\mathcal{R}_j \le r | \mathcal{L}_k = 0\}$. Bearing in mind that $\mathbb{P}\{\mathcal{L}_k \prec 0\} + \mathbb{P}\{\mathcal{L}_k = 0\} + \mathbb{P}\{\mathcal{L}_k \ge 0\} = 1$, it follows that

$$\mathbb{P}\{\mathcal{R}_j \le r\} \ge \mathbb{P}\{\mathcal{R}_j \le r | \mathcal{L}_k = 0\} = \mathbb{P}\{\mathcal{R}'_j \le r\}$$

and thus, $\mathcal{R}'_j \succcurlyeq \mathcal{R}_j$. □

Theorem 1 will be the basis of all the approximations performed in the next sections.

Approximations 5 to 7 in Theorem 1 are related to release times and deserve special attention. Figure 6 depicts three high priority jobs, $\Gamma_{j-1}, \Gamma_{j+1}, \Gamma_{j+2}$, and a low priority job Γ_j, whose random response time has to be calculated. Jobs $\Gamma_{j-1}, \Gamma_{j+1}$ and Γ_{j+2} are released at random times $\mathcal{L}_{j-1}, \mathcal{L}_{j+1}$ and $\mathcal{L}_{j+2}$, indicated by rectangles. The earliest release time with non-zero probability coincides with the left end of the rectangle, while the latest release time with non-zero probability coincides with the right end.

Γ_{j-1} is released before than Γ_j with probability one, since the right end of its rectangle is prior to the release time of Γ_j. From pessimistic approximation 5, we can delay Γ_{j-1}, obtaining Γ^-_{j-1}, which is released at λ^-_{j-1}, the latest release time of Γ_{j-1}. Note that we have transformed job Γ_{j-1}, with a random release time, into a job with a deterministic release time (it has no rectangle around it), and the transformation is pessimistic. This is the final objective of all the release time approximations: to transform the random release times into pessimistic deterministic release times, making the stochastic analysis both affordable and safe.

Γ_{j+2} is released after Γ_j with probability one, since the left end of its rectangle is posterior to the release time of Γ_j. Using pessimistic approximation 6 on Γ_{j+2}, this job is transformed into job Γ^+_{j+2}, released at deterministic time λ^+_{j+2}, which coincides with the earliest release time of Γ_{j+2}. Finally, Γ_{j+1} may be released before or after Γ_j, since its rectangle includes the release time of Γ_j. Using approximation 7 on Γ_{j+1} we obtain job Γ^0_{j+1}, which is released at a deterministic time equal to that of Γ_j. Note that we did not apply this approximation to jobs Γ_{j-1} and Γ_{j+2} since it would be too pessimistic, and approximations 5 and 6 are less pessimistic and equally safe.

6. Applications of the concept of pessimism

6.1. Priority assignment

The analysis algorithm presented in Section 3 does not assume any policy of priority assignment, as long as the assignment is repeated each hyperperiod. One possible policy for assigning static priorities is *Rate Monotonic*. However, according to (Audsley, 1991) and (Tindell et al., 1994), this assignment is not optimal when the existence of a critical instant is not guaranteed, nor when the deadlines are not equal to the periods. For these cases, they provide an alternative algorithm, described in detail in (Audsley, 1991).

For the deterministic case, the algorithm is optimal, i.e., it always finds a feasible assignment of priorities if one exists. The proof (Audsley, 1991) 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 proves that the algorithm is also optimal for stochastic analysis, since the rest of the proof coincides with the proof of deterministic analysis. This result can be proved directly from Theorem 1, statement 2. Decreasing the priority of any task τ_i introduces high priority jobs coming from other tasks into the analysis, which worsens the random response time of all the jobs making up τ_i, and increments the probability of missing the deadline of τ_i. Therefore, the algorithm described in (Audsley, 1991) is also optimal in the stochastic scenario.

6.2. Computational problems

There are a number of computational problems in order to implement the analysis presented in Section 3:

- Computing the steady-state backlog by iterating until convergence is reached, presents the problem of how to choose the “initial backlog” which leads to a quick conver-

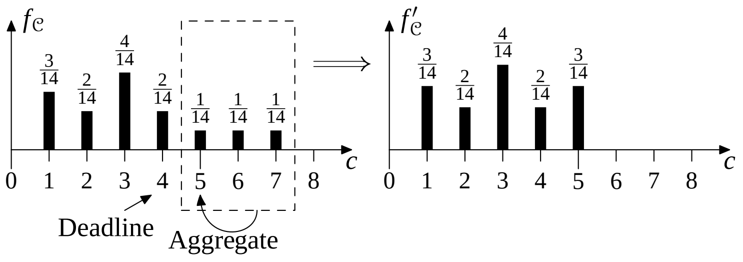


Figure 7: Moving probabilities towards lower execution times.

gence and does not produce an optimistic approximation of the steady-state backlog. This problem was addressed in (Díaz et al., 2004).

- The theoretical steady-state backlog has a probability function with an infinite number of points. Even if we use the approximation found by iteration, which is not infinite, the number of points can be too large to be stored in a computer. The obvious solution is to truncate the probability function. However, this gives rise to one question: if we use truncated versions of the probability functions instead of the complete ones, is the analysis valid? More important, is it safe (i.e: pessimistic)? In (Díaz et al., 2004) it is proved that this kind of approximation is indeed pessimistic, and thus the analysis using truncated PFs is still valid.

- The complexity of the analysis depends largely on the number of non-zero points of the probability functions used. It is necessary to find a way of reducing this size, without affecting the validity of the analysis, i.e., preserving the pessimism. In (Díaz et al., 2004) it is shown that clusters of probabilities can be aggregated towards the worst execution time of the cluster, and that this kind of transformation is pessimistic and reduces the number of points to store in memory.

  Another simplification can be carried out in the following context: if a task of deadline D_i is dropped just at the moment it misses its deadline, then its execution time can not be higher than $(D_i + 1)$. Therefore, the probabilities of execution times $(D_i + 2), (D_i + 3), \ldots, C_i^{\max}$ can be moved to execution time $(D_i + 1)$, as shown in Figure 7. Note that this transformation is not mandatory, but highly recommended, since it reduces the analysis time and pessimism of the results.

- The computer representation of real numbers using finite precision formats, such as IEEE-754 floating point standard, may cause round-off errors. How can we guarantee that those errors never introduce optimism in the results? This problem has not been addressed before, and we analyze it in the rest of this section.

6.2.1. Using finite precision arithmetic From an arithmetic point of view, stochastic analysis is a long sequence of additions and multiplications of probabilities, i.e, real numbers in the range $[0, 1]$. Usually, the calculation of a single random response time requires millions of additions and multiplications, which can be carried out in two ways: finite precision arithmetic or infinite precision arithmetic. Infinite precision arithmetic provides exact results for all the operations, but the memory and processing time required are unaffordable in current practice. For example, if significands (fractional parts) of execution time probabilities are coded using n bits, after convolving two execution time probability functions, $2n$ bits are required to code each probability of the result exactly. Therefore, after thousands or millions of convolutions, the number of bits required to code a single probability of the result would become prohibitively huge.

The solution to this problem is the use of finite precision arithmetic. Commonly, finite precision arithmetic of real numbers is carried out using floating point arithmetic, but fixed point arithmetic implemented on an integer unit is also possible. All the results provided in this section are valid for both fixed and floating point arithmetic.

Finite precision arithmetic solves the computational problems of infinite precision arithmetic, but introduces round-off errors, so the analysis is no longer exact. The problem now is how to round off the results so that the analysis is guaranteed to be pessimistic.

Let us analyze the effects of the usual rounding modes: round to nearest, round towards zero, round towards $-\infty$ and round towards $+\infty$.

Round to nearest is the most common rounding mode and provides the representable number closest to the exact result. However, this rounding mode does not provide pessimistic results. For example, during the calculation of the steady-state backlog we convolve the current backlog with a random execution time, giving a new backlog, defined by an array of time points and their probabilities. Using round to nearest, the probability of the minimum backlog for the resultant backlog distribution may be higher than the exact distribution. In that case, the resultant random variable would not be greater than or equal to the exact one.

Round towards zero and *round towards* $-\infty$ are equivalent for the stochastic analysis, since probabilities are numbers in the range $[0,1]$. Round towards 0 makes results pessimistic, since all the probabilities are less than or equal to the exact. The probability deficit coming from the rounding can be assumed to be located at $+\infty$. Figure 8 depicts the exact probability function and one approximated probability function after rounding towards 0.

Rounding towards zero solves the problem of pessimistic rounding, but introduces a new one. Each addition and multiplication reduces the probability mass of the resultant distribution, since part of this mass goes to infinity, and so it never returns to finite values. For example, let us assume that 10^8 floating point operations are involved in the calculation of a probability value of the steady-state backlog distribution. Using IEEE-754 single precision format to store the probabilities, the average probability deficit generated by any of these operations is $\varepsilon = 0.5 \cdot 2^{-24} \approx 3 \cdot 10^{-8}$, since this format uses 24 bits for the significand, 23 explicit and 1 implicit. Therefore, after 10^8 floating point operations on the same point, the resultant steady-state backlog would be zero or close to zero, since the

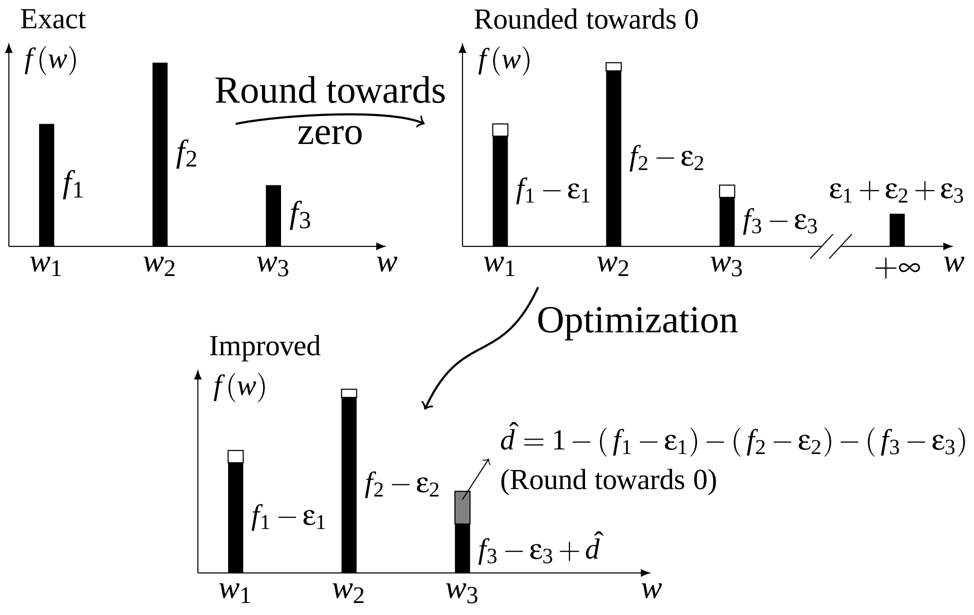


Figure 8: Effects of rounding towards 0.

whole probability mass would be located at infinity.

Of course, a simple solution is the use of longer floating point formats, such as the double and extended double formats, at the cost of more memory and processing time for the analysis tool. However, this is not a solution, but only postpones its onset.

A simple but effective way to alleviate probability leakage, is to estimate the probability deficit, denoted d, by subtracting from 1.0 the sum of all the probabilities in the resultant distribution, also using round towards zero in the subtractions. The estimation, denoted $\hat{d}$, is an underestimation of the probability deficit d, because of the rounding towards zero, which is then added to the probability of the maximum value of the distribution with non-zero probability, using round towards zero in the addition. Figure 8 also depicts this optimization.

In general, the convolution of two arrays of probabilities of m points each require about $2m^2$ operations in total[2], so the probability deficit using rounding towards zero and IEEE-754 single format is about $2m^2 \times 0.5 \times 2^{-24}$. Applying the previous optimization, the probability deficit is reduced to $m \times 0.5 \times 2^{-24}$, since there are m operations in the calculation of the initial deficit and its correction, incrementing the probability of the maximum value with non-zero probability.

It should be noted that the previous optimization should be applied only when the queue of the distribution has not been truncated. Otherwise, the optimization would be pointless, since the whole probability deficit should be added to the infinity point.

It is also possible to perform pessimistic rounding using round towards $+\infty$. Initially, there would be probability excess instead of probability deficit, which is overestimated using again rounding towards $+\infty$, and removed by starting from the lowest point with non-zero probability, using round towards zero in this case. Therefore, a probability deficit would once again be reached.

6.3. Extensions to the basic stochastic analysis

This section presents other applications of the concept of pessimism in stochastic analysis, extending the basic model and the stochastic analysis of periodic independent task sets. Firstly, Section 6.3.1 presents the stochastic analysis of dependent task sets that can block in shared resources. Section 6.3.2 allows for non-preemptive sections in the tasks. Finally, Section 6.3.3 considers random release times coming from release jitter.

6.3.1. Blocking in shared resources When two or more tasks use shared resources, they often need to orchestrate the access in such a way that they never access the shared resource simultaneously. Usually, these *critical sections* in which the resource access must be mutually exclusive are guarded by locks or semaphores. This may cause a *priority inversion* problem, in which a task with higher priority is blocked when trying to enter a critical section whose semaphore is held by a task of lower priority. This situation is especially critical in the real-time field since, in order to find the worst-case scenario, the duration of the blocking times must have an upper limit. Some algorithms have been devised for providing such a guarantee, such as the Priority Inheritance Protocol (PIP) and Priority Ceiling Protocol (PCP) for fixed priority scheduling (Sha et al., 1990), as well as the Stack Resource Policy (SRP) for fixed and non-fixed priority scheduling (Baker, 1991).

The deterministic analysis of real-time systems which use these protocols is performed by computing the value of the worst-case blocking time of each task, adding this blocking time to the execution time of the tasks, and then performing the classical analysis using these augmented execution times. The value of the worst-case blocking time can be derived from the worst-case duration of each critical sections, although the precise way in which this derivation is performed depends on the protocol used (PIP, PCP or SRP).

Most of the previous ideas are applicable to the stochastic scenario, bearing in mind that in this case the lengths of the critical sections, and thus the duration of the blocking suffered by higher priority tasks, are random variables instead of single-valued worst-case values. Let us summarize these ideas:

- The blocking time a task τ_i can suffer is a random variable $\mathcal{B}_i$, but an adequate shared resource protocol ensures that its value is bounded (i.e: $\mathbb{P}\{\mathcal{B}_i > x_b\} = 0$ for a finite x_b, equal to the worst-case blocking time of the deterministic case). The shared resource protocol which ensures this can be one of those previously mentioned for the deterministic case (i.e: PIP, PCP, SRP)

- The exact distribution of $\mathcal{B}_i$ is difficult (if not impossible) to obtain, but an approximation $\mathcal{B}'_i$ can be computed and it can be shown that $\mathcal{B}'_i \succcurlyeq \mathcal{B}$, so it is conservative to use the pessimistic approximation.

- The distribution of the random variable $\mathcal{B}'_i$ can be derived from the lengths of the critical sections of the tasks (these lengths also being random variables). The exact way in which this can be done depends on the shared resource protocol considered, but the procedure is a straightforward translation of the one used for the deterministic case, by using the concepts of supremum and infimum (see definitions 6 and 7) in place of maximum and minimum. For details see (Díaz et al., 2004).

- The execution time of the task $\mathcal{C}_i$ is artificially increased by adding $\mathcal{B}'_i$ to it (this involves performing the convolution of their probability functions). It is worth noting that only the execution time of the job under analysis is incremented by the blocking time. Incrementing the execution time of the other jobs of the task would introduce unnecessary pessimism.

- The stochastic analysis presented in Section 3 can be applied to the new system using these augmented $\mathcal{C}_i$

6.3.2. Non-preemptive sections The system model presented in Section 2 assumes preemption. If one high priority job is released while a low priority task is executing, a context switch takes place, moving the low priority job to the pending queue and executing the higher priority job.

There are situations in which it is necessary to execute a set of instructions atomically. The set of instructions executed atomically is called a non-preemptive section.

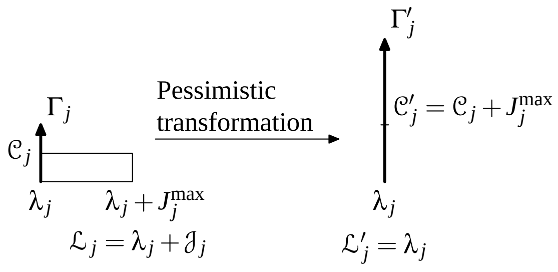
If a high priority job is released while a low priority job is executing within a non-preemptive section, the high priority job will suffer blocking. In the worst case, the blocking will be equal to the length of the non-preemptive section. The situation is analogous to that of blocking while accessing protected shared resources.

Non-preemptive sections can be introduced in the analysis as if they were pseudo-critical sections protected by a single binary pseudo-semaphore with a ceiling equal to the maximum priority (or preemption level) in the system. Once a task enters in a non-preemptive section, pseudo-semaphore is locked and its ceiling receives the maximum preemption level in the system. Therefore, the task can not be preempted until it leaves the pseudo-critical section.

In general, critical sections and non-preemptive sections should be kept short, since they introduce pessimism in the stochastic analysis.

6.3.3. Release jitter There are situations in which the actual release time of a task differs from its theoretical release time. The difference between the theoretical and actual release times is called release jitter. Release jitter is not a fixed quantity, but varies between zero and a maximum jitter.

The simplest approach to account for release jitter in the deterministic analysis is to increment the execution time of all the jobs by their release jitters. Therefore, using this approach, the deterministic response time of any job increases monotonically with release jitter, which requires the use of the maximum jitter. For example, a job of theoretical release time λ_j, maximum jitter $J_j^{\max}$ and execution time C_j, can be safely replaced by a



Γ_j
$\mathcal{C}_j$
λ_j
$\lambda_j + J_j^{\max}$
$\mathcal{L}_j = \lambda_j + \mathcal{J}_j$
Pessimistic transformation
Γ'_j
$\mathcal{C}'_j = \mathcal{C}_j + J_j^{\max}$
λ_j
$\mathcal{L}'_j = \lambda_j$

Figure 9: Simple job transformation to account for release jitter.

job without jitter released at λ_j, with execution time $(C_j + J_j^{\max})$. Under stochastic analysis, the random execution time $\mathcal{C}_j$ would be replaced by $(\mathcal{C}_j + \mathcal{J}_j)$. Bearing in mind that $J_j^{\max} \geqslant \mathcal{J}_j$ and Proposition 1 on the addition function, we conclude that jitter can be safely removed in the stochastic analysis using the random execution time $(\mathcal{C}_j + J_j^{\max})$, as depicted in Figure 9.

However, this approach is suitable only if the maximum release jitter is small, as otherwise the pessimism introduced in the analysis may be excessive.

Another approach that would reduce the previous pessimism would be to model the jitter between the theoretical and real release time as a random variable, and perform an exact stochastic analysis that deals jointly with random release times and random execution times. However, this stochastic analysis is far more complex than the current stochastic analysis, which works with deterministic release times. In addition, jitter is usually a small fraction of time, which is difficult to measure or estimate, especially if we need its random distribution.

A practical solution to this problem is to modify the basic stochastic analysis, so that results are pessimistic using the maximum release jitter, independently of the exact distribution of jitter. Theorem 2 follows this idea, providing a simple transformation that allow us to deal safely with jitter in the stochastic analysis, assuming moderate pessimism.

Theorem 2. *Let $\{\ldots, \Gamma_{j-1}, \Gamma_j, \Gamma_{j+1}, \ldots\}$ be a set of jobs scheduled using fixed priorities, of random release times $\{\ldots, (\lambda_{j-1} + \mathcal{J}_{j-1}), (\lambda_j + \mathcal{J}_j), (\lambda_{j+1} + \mathcal{J}_{j+1}), \ldots\}$, where λ_j is the deterministic theoretical release time of job Γ_j, and $\mathcal{J}_j$ its random jitter of maximum value J_j^{max}. Let $\{\ldots, \mathcal{C}_{j-1}, \mathcal{C}_j, \mathcal{C}_{j+1}, \ldots\}$ be the random execution times, and let $\mathcal{R}_j$ be the random response time of job Γ_j.*
If the release times of all the jobs are transformed so that:

- *Any job Γ_k with $\lambda_k + J_k^{max} < \lambda_j$, becomes released at deterministic time $\lambda_k^- = \lambda_k + J_k^{max}$, and is denoted Γ_k^-.*
- *Any job Γ_k with $\lambda_k - J_j^{max} > \lambda_j$, becomes released at deterministic time $\lambda_k^+ = \lambda_k - J_j^{max}$, and is denoted Γ_k^+.*
- *Any job Γ_k with $\lambda_k + J_k^{max} \geq \lambda_j \geq \lambda_k - J_j^{max}$ (including Γ_j) becomes released at deterministic time $\lambda_k^0 = \lambda_j$, and is denoted Γ_k^0.*

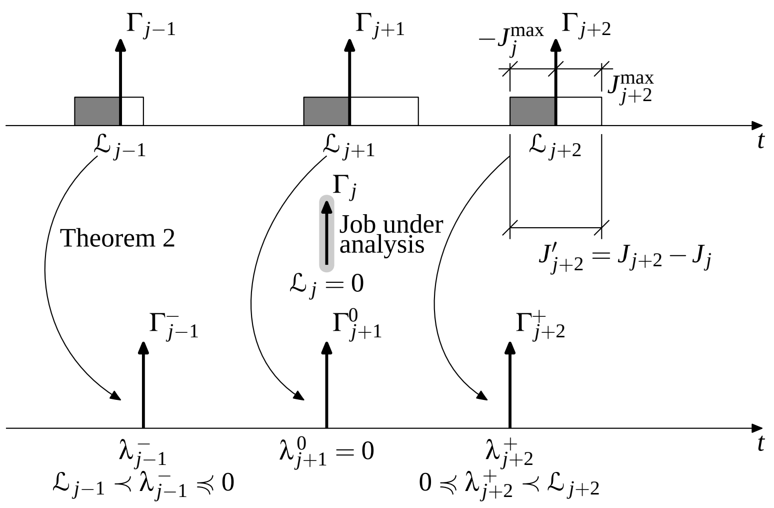


Figure 10: Pessimistic approximations under random releases coming from release jitter.

It follows that the new random response time of Γ_j, *denoted* $\mathcal{R}'_j$, *fulfils* $\mathcal{R}'_j \succcurlyeq \mathcal{R}_j$.

Proof. Making release times relative to the release time of Γ_j does not change the response time of Γ_j. The relative release of any job Γ_k becomes $(\lambda_k + \mathcal{J}_k) - (\lambda_j + \mathcal{J}_j) = (\lambda_k - \lambda_j) - \mathcal{J}_j + \mathcal{J}_k$. In particular, the release time of Γ_j becomes zero. There are two components of jitter in the relative release time: a negative component, $-\mathcal{J}_j$, induced by the release jitter of Γ_j, represented by shaded rectangles in Figure 10, and a positive component, $\mathcal{J}_k$, coming from job Γ_k itself, represented by white rectangles in the same figure. Therefore, any job Γ_k is randomly released in the relative interval $[\lambda_k - \lambda_j - J_j^{\max}, \lambda_k - \lambda_j + J_k^{\max}]$.

If $\lambda_k - \lambda_j + J_k^{\max} < 0$ then job Γ_k is released before Γ_j with probability one, so applying statement 5 of Theorem 1, the first transformation guarantees that the new response time of Γ_j fulfils $\mathcal{R}'_j \succcurlyeq \mathcal{R}_j$ when Γ_k is released at relative time $\lambda_k - \lambda_j + J_k^{\max}$, i.e., at absolute time $\lambda_k^- = \lambda_k + J_k^{\max}$.

If $\lambda_k - \lambda_j - J_j^{\max} > 0$ then job Γ_k is released after Γ_j with probability one, so applying statement 6 of Theorem 1, the second transformation guarantees that the new response time of Γ_j fulfils $\mathcal{R}'_j \succcurlyeq \mathcal{R}_j$ when Γ_k is released at relative time $\lambda_k - \lambda_j - J_j^{\max}$, i.e., at absolute time $\lambda_k^+ = \lambda_k - J_j^{\max}$.

Finally, if $\lambda_k - \lambda_j - J_j^{\max} \leq 0 \leq \lambda_k - \lambda_j + J_k^{\max}$, applying statement 7 of Theorem 1, the third transformation guarantees that the new response time of Γ_j fulfils $\mathcal{R}'_j \succcurlyeq \mathcal{R}_j$ when Γ_k is released at relative time zero, i.e., at absolute time $\lambda_k^0 = \lambda_j$. □

Figure 10 depicts the transformations of Theorem 2. White rectangles are used to represent the random jitter of the jobs, while shaded rectangles represent negative jitter, induced by the random release jitter of Γ_j, after making release times relative to the release of Γ_j. It should be noted that the delay of job Γ_{j-1} in Figure 10 is produced by its worst jitter in the calculation of the random response time of Γ_j, and cannot be improved by any approximation in which only the maximum release jitter is known.

The problem now is how to apply Theorem 2 to calculate in practice the pessimistic response times of periodic tasks with release jitter. The response time of any job Γ_j is calculated from $\mathcal{R}_j = \mathcal{W}(\lambda_j) + \mathcal{C}_j + \mathcal{I}_j$, as explained in Section 3. Term $\mathcal{W}(\lambda_j)$ is calculated for the first job of each task in a steady-state hyperperiod, considering that it is released with zero release jitter, and considering all the jobs of type Γ_k^- (including those coming from the same task as Γ_j) to be released at deterministic times $\lambda_k + J_k^{\max}$, i.e., at their latest release times. The calculation of $\mathcal{R}_j$ is completed using the jobs of types Γ_k^0, released at λ_j, and types Γ_k^+, released at $\lambda_k - J_j^{\max}$, i.e., at their earliest release times taking into account the negative jitter induced by job Γ_j.

The previous results for jitter are also valid under EDF scheduling, assuming that deadlines are defined relative to the theoretical release times of the tasks, making job priorities independent of jitter.

Finally, the reader should observe that jitter worsens the response time of the tasks, so it should be kept as low as possible.

7. Stochastic analysis: pros and cons

The ideal response time analysis would be one that predicts exactly the complete sequence of response times we would measure in the real system for each task. Deterministic analysis is the simplest analysis, since it summarizes the whole sequence in a single worst-case value. Stochastic analysis of course is not ideal, but is closer to the ideal, since it summarizes the sequence into a set of values and frequencies of appearance (probabilities).

The biggest problem of deterministic analysis is that worst-case execution times are extremely pessimistic in practice. Worst-case execution times rarely appear and are one or more orders of magnitude higher than the typical execution times. Working with worst-case execution times gives rise to excessively oversized systems. Stochastic analysis deals with a configurable number of points for the execution times, reducing system oversizing and cost. However, it involves a design cost, as shown below.

The exact calculation of worst-case execution times is not possible in many cases because of hardware/software complexities, which require the use of pessimistic worst-case execution times. The problem of finding execution times becomes even worse under stochastic analysis, since we should introduce into the analysis not only single worst-case execution times, but worst-case distributions of execution times.

Current techniques of hardware optimization introduce execution time dependencies. For example, cache memory makes the execution time of the highest priority job dependent on the previously released jobs. Deterministic and stochastic analyses assume independent tasks (although some special dependencies like blocking in shared resources can be dealt with successfully). Dependencies can be removed using pessimistic execution times, but finding pessimistic execution times is a complex problem, especially if execution times are modelled as random variables.

Another of the cons the stochastic analysis is the time required to perform the analysis. Obviously, it is several orders of magnitude higher than the time required for the deterministic analysis. Nonetheless, we have the theoretical tools to perform approximations that allow us to trade-off computational cost and pessimism. One of the keys to make stochastic analysis possible is the correct choice of tasks periods during system design. In general, periods should be chosen to have common factors, which reduces the length of the hyperperiod and the computational cost of the stochastic analysis. For example, in the context of control systems, sampling periods, which define task periods, can be shortened in order to reduce the hyperperiod length. This increments the computational cost of the control system, but improves the controllability , and makes stochastic analysis applicable. Thus, the dimension and cost of the system may be reduced.

A common misconception about the stochastic analysis is that it is applicable only to trivial task sets and that it lacks of accuracy. However it is not the case. In fact, the time required to analyze a task set made up of 35 periodic tasks, of periods: 100, 200, 250, 400, 500, 600 and 1000, and execution time probability functions each made up of 100 points, was about 25 minutes in a low-end 32 bit workstation with a single 2 GHz CPU and 512 MBytes of RAM. The response time probability functions were calculated with an error about 10^{-14}. In addition, system utilization was about 0.95 and maximum system utilization about 13.0. The interested reader can find on the web this example along with a stochastic analyzer called *stochan*[3], which implements the analysis described in this article.

8. Conclusions and future work

Stochastic analysis is a valuable tool that greatly improves deterministic analysis. In fact, deterministic analysis is a particular case of stochastic analysis, as it is the most pessimistic stochastic analysis. As the improvement strongly depends on task sets characteristics, we have not presented comparative results between deterministic and stochastic analysis.

Current theory allows us to analyse periodic tasks, both independent and communicating through shared resources, with or without jitter and with or without non-preemptive sections. Stochastic analysis may be extended in future to other types of tasks, using the theoretical framework about stochastic pessimism introduced in the article.

Most of the article is about the relationships "$\succ$" and "$\succcurlyeq$" between the random variables of a real-time system. Thanks to these relationships, it is possible to state that a random execution time distribution is higher (worse) than another, that a random blocking time distribution is higher than another, etc. The ordering between random variables allows us a direct translation of well known real-time deterministic results to the stochastic scenario.

Stochastic analysis is costly in computational terms, but can be configured to trade-off computational cost and pessimism. In addition, task periods should be chosen carefully during the design stage in order to obtain reasonable hyperperiod lengths.

Major problems to make the stochastic analysis applicable in practice are obtaining execution time distributions and the problem of non-independent execution times due to hardware dependencies, such as cache memories. Both problems are common to deterministic analysis, but become even more difficult in the stochastic analysis.

Future work will address the following theoretical issues:

- The analysis of sporadic tasks. They are common in real-time systems, so must be included in the model and analysis.
- Finding suboptimal phase assignments in order to improve the fulfillment of the stochastic real-time constraints. It is our belief that phases have great influence on schedulability in the stochastic analysis, even higher than in the deterministic analysis.
- A study of the applicability of stochastic analysis to strictly hard-real time systems. In this case, the probability of missing the deadlines is so low, that we may face with serious precision problems in the stochastic analysis, or conclude that deterministic analysis is a better choice.
- Extend stochastic analysis to simple distributed real-time systems.

References

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.

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.

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, December 1991.

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

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.

J. L. Díaz and J. M. López. Some notes on stochastic analysis using EDF scheduling. Technical report, Departamento de Informática, University of Oviedo, 2007. Also available at `http://www.atc.uniovi.es/research/SNSAUES07.pdf`.

J. L. Díaz, D. F. García, Kanghee Kim, Chang-Gun Lee, Lucia Lo Bello, J. M. López, Sang Lyul Min, and Orazio 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, December 2002.

José Luis Díaz, José María López, Manuel García, Antonio Manuel Campos, Kanghee Kim, and Lucia Lo Bello. Pessimism in the stochastic analysis of real-time systems: Concept and applications. In *Proc. of the 25rd IEEE Real-Time Systems Symposium*, Lisboa, Portugal, December 2004.

M. K. Gardner and J. W.S. 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.

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

K. Kim, J. L. Díaz, L. Lo Bello, J. M. López, C. G. Lee, and S. L. Min. An exact stochastic analysis of priority-driven periodic real-time systems and its approximations. *IEEE Transactions on Computers*, 54(11), November 2005.

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

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

John 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, December 1990.

Haim Levy. Stochastic dominance and expected utility: survey and analysis. *Management Science*, 38(4):555–593, 1992. ISSN 0025-1909.

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

Sorin Manolache, Petru Eles, and Zebo Peng. *Real-Time Applications with Stochastic Task Execution Times Analysis and Optimisation*. Springer, P.O. Box 17, 3300 AA Dordrecht, The Netherlands, 2007. ISBN 978-1-4020-5509-6 (e-book).

Lui Sha, Ragunathan Rajkumar, and John P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9): 1175–1185, September 1990.

T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J.W.-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.

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.

Notes

Affiliation of authors: José M. López, José L. Díaz, Joaquín Entrialgo and Daniel García ({chechu, jldiaz, joaquin, dfgarcia}@uniovi.es), *Depto. de Informática*, *Universidad de Oviedo* (33204, Gijón, Spain)

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

[2] In theory, convolutions can be performed using the FFT, thus reducing the complexity from m^2 to $m \log m$. In practice, it is effective only when the distributions have similar number of points. This is not our case, since the number of points of execution time probability functions is usually much lower than the number of points of backlog or partial response times.

[3]Available from `http://www.atc.uniovi.es/rsa/starts/tools.php`