Probabilistic Analysis for Mixed Criticality Systems using Fixed Priority Preemptive Scheduling

Dorin Maxim, Robert Davis, Liliana Cucu-Grosjean, Arvind Easwaran

Probabilistic Analysis for Mixed Criticality Systems using Fixed Priority Preemptive Scheduling

Dorin Maxim
LORIA - University of Lorraine, Nancy, France
dorin.maxim@loria.fr

Robert I. Davis
University of York, UK & Inria, Paris, France
rob.davis@york.ac.uk

Liliana Cucu-Grosjean
Inria, Paris, France
liliana.cucu@inria.fr

Arvind Easwaran
Nanyang Technological University, Singapore
arvinde@ntu.edu.sg

ABSTRACT

This paper introduces probabilistic analysis for fixed priority preemptive scheduling of mixed criticality systems on a uniprocessor using the Adaptive Mixed Criticality (AMC) and Static Mixed Criticality (SMC) schemes. We compare this analysis to existing deterministic methods, highlighting the performance gains that can be obtained by utilising more detailed information about worst-case execution time estimates described in terms of probability distributions. Besides improvements in schedulability, we also demonstrate significant gains in terms of the budgets that can be allocated to LO-criticality tasks.

CCS CONCEPTS

•**Computer systems organization** → **Real-time systems;**
•**Software and its engineering** → **Real-time schedulability;**
•**Mathematics of computing** → *Probabilistic representations;*

KEYWORDS

Real-Time Systems; Mixed Criticality; Schedulability Analysis; Probabilities; Fixed Priority;

ACM Reference format:
Dorin Maxim, Robert I. Davis, Liliana Cucu-Grosjean, and Arvind Easwaran. 2017. Probabilistic Analysis for Mixed Criticality Systems using Fixed Priority Preemptive Scheduling. In *Proceedings of RTNS '17, Grenoble, France, October 4–6, 2017,* 10 pages.
DOI: 10.1145/3139258.3139276

PRELIMINARY PUBLICATION

A preliminary version [26] of the research described in this paper was published in the Workshop on Mixed Criticality Systems (WMC) in 2016. In this paper, we correct the analysis given in [26], ensuring that the schedulability of HI-criticality tasks does not depend on the behavior of LO-criticality tasks. Further, we provide an alternative analysis (in Section 4.4) and show how support for LO-criticality tasks can be improved via increased execution time budgets (in Section 4.6).

RTNS '17, Grenoble, France

DOI: 10.1145/3139258.3139276

1 INTRODUCTION

In 2007, Vestal [32] introduced a mixed criticality task model, where each task is represented by multiple Worst-Case Execution Time (WCET) estimates that are determined at different levels of assurance. For example, the WCET estimates $C(LO)$ and $C(HI)$ are the estimates for low-assurance and high-assurance respectively in a dual-criticality system. By contrast, in this paper we use a richer model based on probabilistic Worst-Case Execution Time (pWCET) distributions. Figure 1 illustrates this richer model. The thick line on the graph denotes the 1-CDF (Complementary Cumulative Distribution Function or *exceedance function*) for the pWCET distribution of a task. From the exceedance function, it is possible to read off for a specified probability, an execution time that has no higher probability of being exceeded on any single run or job of the task. The high assurance estimate $C(HI)$ may be obtained using an exceedance probability (on the y-axis) of for example $p(HI) = 10^{-12}$, where 10^{-12} denotes an acceptable threshold on the failure probability for each job of the task at a high assurance level. Here, the scheduler can ignore any execution demand beyond $C(HI)$ because its probability of occurrence is below the threshold required. Similarly, at a lower level of assurance (say with an acceptable threshold of $p(LO) = 10^{-8}$), the scheduler can ignore any execution demand beyond $C(LO)$.

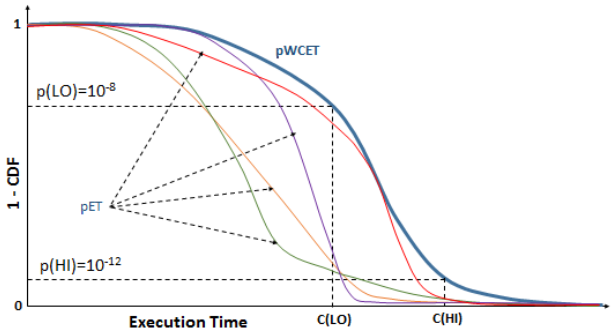


Figure 1: Exceedance function or 1-CDF for the pWCET distribution of a task and the pET distributions of its jobs.

In this paper, we focus on the development of a probabilistic schedulability analysis framework for mixed criticality systems under Fixed-Priority Preemptive Scheduling (FPPS). There are two well-known FPPS schemes and corresponding deterministic schedulability analyses in the literature on mixed criticality scheduling; Static Mixed Criticality (SMC) [6] and Adaptive Mixed

Criticality (AMC) [5]. We develop probabilistic analyses for each of these schemes. A brief description of them is given in Section 2.

1.1 Fundamental concepts

We assume that the tasks we are interested in execute on a time-randomized hardware platform; for example with an instruction cache that implements a random replacement policy [11]. Each *specific job* of a task is defined by a fixed combination of inputs, software state, and hardware state, excluding the random variables (emanating from a random number generator in the hardware) which give rise to execution time variability.

Definition 1.1. The probabilistic Execution Time (pET) distribution for a specific job is the distribution obtained by running the task with the corresponding fixed combination of inputs, software state, and hardware state (excluding random variables) an infinite number of times.

Definition 1.2. The probabilistic Worst-Case Execution Time (pWCET) distribution for a task is an upper bound, in the sense of the greater than or equal to operator defined by Diaz et al. [14], on the pET distribution of every possible specific job of the task.

Viewed in terms of its 1-CDF, the pWCET distribution of a task is never lower than any of the pET distributions for its specific jobs, as illustrated in Figure 1.

Definition 1.3. Two random variables $\mathcal{X}$ and $\mathcal{Y}$ are *independent* if they describe two events such that the outcome of one event does not have any impact on the outcome of the other.

We assume that the pET distributions for each specific job of a task are *independent* of the pET distribution for any other specific job of the same or different task. This is the case when the only contribution to variation in the execution time of a specific job comes from independent random variables (i.e. is due to the behaviour of the hardware random number generator). In that case, the pWCET distribution for the task, which upper bounds the pET distribution of all of the specific jobs, can be used to model the execution time of the task as an *independent random variable*. We note that the actual execution times for a sequence of jobs may show strong correlations and dependencies; it is the *representation* of execution times via an appropriate pWCET distribution which provides independence.

We assume that Static Probabilistic Timing Analysis (SPTA) [13, 21, 22] is used to analyse each task, and that this analysis provides a valid pWCET distribution as defined above, enabling the modeling of task execution times as independent random variables.

1.2 Related work

A number of different probabilistic timing analysis techniques have been proposed to provide pWCET distributions. Here, we focus on Static Probabilistic Timing Analysis (SPTA). For time-randomized hardware using a random replacement cache, initial work by Cucu-Grosjean et al. [12] in 2012 introduced a simple SPTA based on re-use distances that is valid for single path programs, and assumes an evict-on-access random replacement policy. In 2013, Davis et al. [13] derived a similar SPTA based on re-use distances for the more effective evict-on-miss policy, extended the approach to multipath programs, and showed how the effects of preemptions could be accounted for. More precise SPTA based on the concept of cache contention and focussed blocks was introduced by Altmeyer et al. in 2014 [2, 3], and subsequently also extended to multipath programs by Lesage et al. in 2015 [22], [21]. In 2014, Griffin et al. [17] derived an alternative approach using lossy compression techniques.

Probabilistic schedulability analysis, i.e. determining the probability of deadline failure when at least one task parameter (e.g. execution time) is described by an independent random variable, was investigated by Woodbury and Shin in 1988 [33]. In 1995, Tia et al. [31] proposed a Probabilistic Time-Demand Analysis addressing the problem of providing probabilistic guarantees for periodic tasks. Some of the limitations of this work were lifted by Gardner et al. in 1999 [16]. They proposed a Stochastic Time-Demand Analysis for FPPS, with assumptions made regarding the critical instant. Analysis by Diaz et al. in 2002 [14, 15], later refined by Lopez et al. in 2008 [23]; addressed the issue of backlog at the end of each hyperperiod. This analysis is difficult to use in practice, due to its computational complexity. (Improvements based on re-sampling were proposed by Maxim et al. [27]). In 2009, Ivers and Ernst [19] addressed the problem of dependencies between the execution times of jobs. In 2013, Maxim and Cucu-Grosjean [24] presented a probabilistic response time analysis, proving properties regarding the critical instant, and extending the analysis to tasks with inter-arrival times and deadlines also described by random variables. In 2015, Tanasa et al. [30] provided a probabilistic response time analysis using continuous functions to approximate the execution time distributions.

Previous work on probabilistic analysis for mixed criticality systems initially considered EDF scheduling. In 2015, Santinelli and George [29] presented preliminary work that investigated the probabilistic C-space, showing how schedulability varies with task execution times. Subsequently, Guo et al. [18] extended the mixed criticality task model with a single exceedance probability value for the low assurance budget of each HI-criticality task, and used probabilistic analysis to improve schedulability. In 2016, Draskovic et al. [28] examined FPPS of mixed criticality periodic task systems with execution times described by random variables. They employed the method of Diaz et al. [15] to compute the probability of a deadline miss for every job in the hyperperiod. Draskovic et al. also computed the expected time before a change to HI-criticality mode. They showed that this expected time depends on the LO-criticality execution time budget allocated to HI-criticality tasks. A smaller budget results in a lower probability of deadline failure, but a shorter expected time before a transition to HI-criticality mode. In 2017, Abdeddaim and Maxim [1] derived probabilistic response time analysis for mixed criticality tasks under FPPS, computing the probability of deadline misses for each task in each criticality mode. Their work does not assume any monitoring, hence they assume that lower criticality tasks continue to execute in higher criticality modes.

An overview of the research into mixed criticality real-time scheduling emanating from the seminal paper of Vestal [32], can be found in the survey on mixed criticality systems [10].

2 SYSTEM MODEL

In this paper, we are interested in the fixed priority preemptive scheduling of a mixed criticality system comprising a static set of n sporadic tasks which execute on a single processor. We assume

without loss of generality that each task τ_i has a unique priority. We further assume a discrete time model where all task parameters are described using integers.

Each task τ_i is defined by its period (or minimum inter-arrival time), relative deadline, worst-case execution time, and level of criticality (defined by the system engineer responsible for the entire system): (T_i, D_i, C_i, L_i). We restrict our attention to constrained-deadline systems in which $D_i \leq T_i$ for all tasks. Further, we assume that the processor is the only resource that is shared by the tasks, and that the overheads due to the operation of the scheduler and context switch costs can be bounded by a constant, and hence included within the worst-case execution times attributed to each task.

In a mixed criticality system, further information is needed in order to perform schedulability analysis. In this paper we are concerned with dual criticality systems, with criticality levels LO and HI. Using a *deterministic* representation each LO-criticality task τ_i has a single worst-case execution time estimate $C_i(LO)$, while each HI-criticality task τ_i has two worst-case execution time estimates $C_i(LO)$ and $C_i(HI)$ with $C_i(HI) \geq C_i(LO)$. We use $p(LO)$ (and $p(HI)$) to denote the worst-case probability that $C_i(LO)$ (resp. $C_i(HI)$) is exceeded during the execution of any single job of the task. Note, in this paper, we assume that these probabilities are the same for all tasks and so drop the index. We use exemplar values for $p(LO)$ and $p(HI)$ of 10^{-8} and 10^{-12} respectively. Note for ease of presentation, we also drop the index for $C(LO)$ and $C(HI)$ when using these terms in a generic way; nevertheless, these values are specific to each task.

By contrast, using a *probabilistic* representation, each task τ_k (of LO- or HI-criticality) has a probabilistic worst-case execution time (pWCET) distribution $\mathcal{C}_k$. Further, there is a correspondence between the probabilistic and deterministic representations. Considering the 1-CDF (exceedance function), $f(\mathcal{C}_k)$ for task τ_k, $C_k(LO)$ (resp. $C_k(HI)$) in the deterministic representation corresponds to the value of $f(\mathcal{C}_k)$ at a probability of exceedance of $p(LO)$ (resp. $p(HI)$), as illustrated in Figure 1. Note beyond this correspondence, we make no assumptions in our analysis about the form of the distribution. We assume that the WCET values of two tasks C_i and C_j are independent and so too are the upper bounds on their execution time behaviours, which are modelled as independent random variables characterised by pWCET distributions $\mathcal{C}_i$ and $\mathcal{C}_j$.

The SMC and AMC scheduling schemes investigated in this paper both use budget enforcement by the Real-Time Operating System (RTOS) to ensure that LO-criticality tasks cannot execute for more than their LO-criticality execution time budget $C(LO)$. With the AMC scheme, the RTOS also uses the $C(LO)$ budget for each HI-criticality task to determine if a mode change should take place. The system moves from LO- to HI-criticality mode if this budget is reached without the job completing. Once HI-criticality mode is entered, then with AMC, any jobs of LO-criticality tasks that have already started can continue to execute; however, no further jobs of LO-criticality tasks can be released. (This simple extension to the original AMC scheme, which called for jobs of LO-criticality tasks to be aborted on entering HI-criticality mode, is permitted by the analysis [5] and was proposed in [8]). With SMC, jobs of LO-criticality tasks continue to be released in HI-criticality mode.

We assume that any job of a HI-criticality task that executes for its $C(HI)$ budget without completing is executing erroneously and is therefore aborted by the RTOS. Similar to [24], we also assume that any job that does not complete by its deadline is aborted.

In any processor busy period, where all jobs of HI-criticality tasks complete without exceeding their LO-criticality budgets, the system is said to be in *LO-criticality mode*; otherwise it is said to be in *HI-criticality mode*. (We assume that at an idle instant when there are no jobs with outstanding execution, the system may revert back to LO-criticality mode. Other more sophisticated recovery policies could however be used [8]). There are different requirements on schedulability that apply in the different modes of the system.

We use R_i to refer to the *deterministic* worst-case response time (WCRT) of task τ_i, and $\mathcal{R}_i$ to refer to the probabilistic worst-case response time (pWCRT) distribution which may be computed using pWCET values.

In LO-criticality mode, jobs of LO-criticality tasks must have a Worst-Case Deadline Miss Probability (WCDMP) that is no greater than a specified threshold $H(LO)$ (for example 10^{-8}). We assume these thresholds are the same value for all LO-criticality tasks and so drop the index. Using deterministic analysis, this requirement may be satisfied by showing that the tasks are schedulable i.e. have a worst-case response time $R_i(LO) \leq D_i$, computed using execution times of $C(LO)$. Using the probabilistic analysis developed in this paper, the requirement may be satisfied directly by determining the worst-case deadline miss probability. This is achieved by computing the probabilistic worst-case response time distribution (pWCRT) and determining the value of the 1-CDF (Cumulative Distribution Function) at a response time corresponding to the task's deadline D_i, thus computing an upper bound on the probability of missing the deadline.

Jobs of HI-criticality tasks must have a worst-case deadline miss probability that is no greater than a specified threshold $H(HI)$ (for example 10^{-12}). This requirement applies to all modes, and may be met via deterministic methods or via calculating the appropriate probabilistic worst-case response time distribution and comparing it with the task's deadline. In both cases, due account needs to be taken of interference from LO-criticality tasks.

Jobs of both LO- and HI-criticality tasks may also fail to meet their timing requirements by not completing within their execution time budgets of $C(LO)$ and $C(HI)$ respectively, which are enforced by the RTOS. The probability of such budget overruns are upper bounded by $p(LO)$ and $p(HI)$ respectively, and are assumed to be acceptable for both LO- and HI-criticality tasks. (This is the case for both deterministic and probabilistic analyses).

Finally we note an important point about using pWCET distributions and probabilistic analysis, which does not occur with deterministic analysis. When we analyse LO-criticality tasks, we can use low assurance information e.g. $C(LO)$ values and pWCET distributions for LO-criticality tasks. However, when we analyse HI-criticality tasks, we must be sure to use *only* high assurance information e.g. pWCET distributions for HI-criticality tasks, and rely on the high assurance RTOS to enforce $C(LO)$ budgets for LO-criticality tasks. We return to this point in Section 4.1.

3 RECAP OF EXISTING ANALYSES

In this section, we recapitulate the deterministic schedulability analysis for SMC [6] and AMC [5], and also existing probabilistic analysis for FPPS [24].

3.1 Deterministic Schedulability Analysis

Static Mixed Criticality (SMC) scheduling [6] is based on Vestal's original approach using fixed priorities, extended using run-time monitoring. Thus, if a job of a LO-criticality task does not complete execution by its budget $C(LO)$, then it is aborted.

The response time R_i of task τ_i under SMC may be computed using the following fixed point iteration, which is a simple adaptation of standard Response Time Analysis [4, 20]. Recall that L_i is the criticality level of task τ_i.

$$R_i = C_i(L_i) + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil min(C_j(L_i), C_j(L_j)) \quad (1)$$

where $hp(i)$ is the set of all tasks with priority higher than that of task τ_i.

With Adaptive Mixed Criticality (AMC) scheduling [5], if a job of a HI-criticality task executes for its $C(LO)$ budget without signaling completion, then the system enters HI-criticality mode. In this mode, previously released jobs of LO-criticality tasks are completed; however, any subsequent releases of LO-criticality tasks are not started.

The analysis for AMC first computes the worst-case response times for tasks in the LO-criticality mode via the following fixed point iteration:

$$R_i(LO) = C_i(LO) + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i(LO)}{T_j} \right\rceil C_j(LO) \quad (2)$$

On a criticality change, the only concern is HI-criticality tasks; for these tasks:

$$R_i(HI) = C_i(HI) + \sum_{\forall j \in hpH(i)} \left\lceil \frac{R_i(HI)}{T_j} \right\rceil C_j(HI) + \sum_{\forall k \in hpL(i)} \left\lceil \frac{R_i(LO)}{T_k} \right\rceil C_k(LO) \quad (3)$$

where $hpH(i)$ is the set of HI-criticality tasks with priority higher than that of task τ_i and $hpL(i)$ is the set of LO-criticality tasks with priority higher than that of task τ_i.

Equation (3) limits the interference from LO-criticality tasks by noting that no further jobs of these tasks can be released after the change to the HI-criticality mode which must occur at or before $R_i(LO)$. We note however that the interference from every LO-criticality job that executes is assumed to be its entire budget $C(LO)$. Similarly $C(HI)$ is assumed for every job of a HI-criticality task, even though execution for this much time may be a rare event.

3.2 Probabilistic Schedulability Analysis

In this section, we recap on the probabilistic response time analysis for FPPS derived in [24] for tasks with both execution times and inter-arrival times described by random variables. Since in this paper the inter-arrival time for each task is a constant, we present only a simplified version of this analysis. First we recap the basic terminology and operators used.

We distinguish between *full* distributions and *partial* distributions. A *full* distribution $\mathcal{Z}$ has probabilities which sum to 1. Such a distribution may be split into two (or more) *partial* distributions $\mathcal{X}$ and $\mathcal{Y}$ such that $\forall v$ $P(\mathcal{Z} = v) = P(\mathcal{X} = v) + P(\mathcal{Y} = v)$. We say that $\mathcal{Z} = \mathcal{X} \oplus \mathcal{Y}$ where $\oplus$ is the *coalescence* of the two distributions via the addition of the probabilities for each value. In contrast, the sum $\mathcal{Z}$ of two *independent* random variables $\mathcal{X}$ and $\mathcal{Y}$ is given by their *convolution* $\mathcal{X} \otimes \mathcal{Y}$ where $P\{\mathcal{Z} = z\} = \sum_{k=-\infty}^{k=+\infty} P\{\mathcal{X} = k\} P\{\mathcal{Y} = z - k\}$.

We now outline how the pWCRT distribution $\mathcal{R}_i$ of task τ_i can be computed.

In [24], Maxim and Cucu-Grosjean proved that considering all valid patterns of job releases, the worst-case response time distribution of a job of task τ_i occurs for the first job of τ_i released simultaneously with jobs of all higher priority tasks, which are then re-released as soon as possible. (Note, this is the case for the task model considered in [24] and in this paper, where jobs are aborted if they have not completed by their deadline). We can therefore compute an upper bound on the pWCRT distribution $\mathcal{R}_i$ of task τ_i as follows.

The worst-case response time distribution for task τ_i is first initialized to:

$$\mathcal{R}_i^0 = \mathcal{B}_i \otimes \mathcal{C}_i \quad (4)$$

where the backlog $\mathcal{B}_i$ at the release of τ_i is given by:

$$\mathcal{B}_i = \bigotimes_{j \in hp(i)} \mathcal{C}_j \quad (5)$$

The worst-case response time is then updated iteratively for each preemption as follows:

$$\mathcal{R}_i^m = (\mathcal{R}_i^{m-1,head} \oplus (\mathcal{R}_i^{m-1,tail} \otimes \mathcal{C}_k^{pr})) \quad (6)$$

Here, m is the index of the iteration. $\mathcal{R}_i^{m-1,head}$ is the part of the distribution $\mathcal{R}_i^{m-1}$ that is not affected by the preemption under consideration (i.e. it only contains values $\leq t_m$ where t_m is the time of the preemption). $\mathcal{R}_i^{m-1,tail}$ is the remaining part of the distribution $\mathcal{R}_i^{m-1}$ that may be affected by the preemption. Finally, $\mathcal{C}_k^{pr}$ is the pWCET distribution of the preempting task τ_k.

Iteration ends when there are no releases left from jobs of higher priority tasks at time instants smaller than the largest value in the response time distribution currently obtained. Iteration may also be terminated once any new preemptions are beyond the deadline of the task.

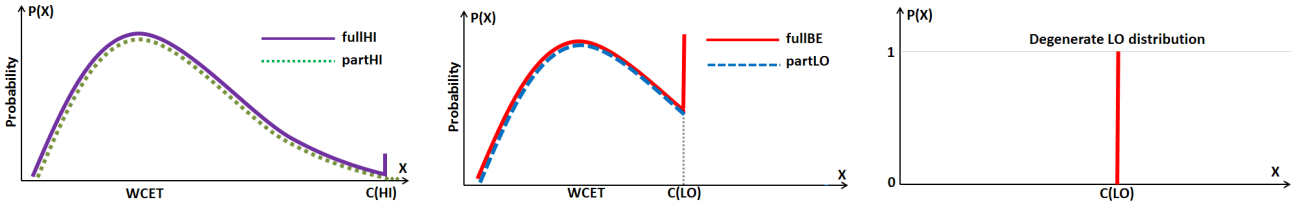
Once iteration is complete, the worst-case deadline miss probability valid for any job of task τ_i is given by:

$$WCDMP_i = P(\mathcal{R}_i > D_i). \quad (7)$$

Worked examples of the analysis described in the following section, which are based on the above method, can be found in Appendix A of the technical report [25] on which this paper is based.

4 PROBABILISTIC ANALYSIS

In this section, we introduce probabilistic response time analysis for the SMC and AMC scheduling schemes, referred to as pSMC and pAMC analysis respectively.



(a) *partHI* (green) and *fullHI* (purple) distribution for a HI-criticality task.

(b) *partLO* (blue) and *fullBE* (red) probability distributions for a LO-criticality task.

(c) *degen* distribution for a LO-criticality task.

Figure 2: Illustrations of the full, partial, and degenerate distributions used in probabilistic analysis.

4.1 pWCET Distributions Used

In our analysis, we make use of different types of pWCET distribution which we now describe. We use the term *partLO* to denote a partial distribution $\mathcal{X}$ formed from the full pWCET distribution $\mathcal{Z}$ of a task by taking only those values that represent completion of the task in no more than its LO-criticality execution time budget $C(LO)$. Thus $\mathcal{X}$ is such that $\forall v \leq C(LO)$, $P(\mathcal{X} = v) = P(\mathcal{Z} = v)$ and $\forall v > C(LO)$, $P(\mathcal{X} = v) = 0$. Similarly, we use the term *partHI* to denote a partial distribution formed by taking only those values that represent completion of the task in no more than its HI-criticality execution time budget $C(HI)$. Since the RTOS enforces the $C(HI)$ budget, we form a full distribution accounting for this, referred to as *fullHI*, which is truncated at $C(HI)$ and accumulates the probabilities for all higher execution times at that point i.e. $P(\mathcal{Y} = C(HI)) = P(\mathcal{Z} \geq C(HI))$. The *partHI* and *fullHI* distributions for a HI-criticality task are illustrated in Figure 2a. (Note, the shape of these distributions is for illustration purposes only; no specific distribution is assumed by the analysis).

Figure 2b illustrates the *partLO* and *fullBE* distributions for a LO-criticality task. The *partLO* distribution describes the behaviour expected of the task by the system designer. In addition, with both the SMC and AMC schemes, if a job of a LO-criticality task executes for its execution time budget $C(LO)$ without signaling completion, then the job will be aborted as a result of budget enforcement by the RTOS. We need to also take this behaviour into account when computing the interference on other tasks of lower priority. Thus we form a full distribution accounting for budget exceedance, referred to as *fullBE*, which is truncated at $C(LO)$ and accumulates the probabilities for all higher execution times at that point i.e. $P(\mathcal{Y} = C(LO)) = P(\mathcal{Z} \geq C(LO))$. This is also illustrated in Figure 2b.

When we analyse HI-criticality tasks, we cannot trust the pWCET distributions obtained for LO-criticality tasks. The reason for this is that LO-criticality tasks are not developed to the same rigourous standards as HI-criticality tasks. Thus, in the analysis we provide for the HI-criticality tasks, we must not make any assumptions about their correct behaviour. Instead, we make a conservative worst-case assumption. We assume that such tasks may execute erroneously and so *always* attempt to overrun their $C(LO)$ budget and therefore need to be aborted by the RTOS, which is itself a trusted component that has been developed to the standards required for HI-criticality operation. When we are analysing HI-criticality tasks, we therefore assume that LO-criticality tasks have a *degen* (meaning full degenerate) distribution which has a single value equivalent to the maximum time that the task can execute i.e $C(LO)$ with probability 1 (as illustrated in Figure 2c).

pWCRT distributions are composed from pWCET distributions using the analysis described in Section 3.2. When partial pWCET distributions are used (for example representing LO-criticality mode only), then the resultant pWCRT distribution is also a partial distribution, giving probabilities that are conditional on being in that mode.

When analysing the worst-case deadline miss probability for LO-criticality tasks, we use the *partLO* distributions, since we are only interested in the probability that tasks exceed their deadlines *and* the system remains in LO-criticality mode, i.e. $C(LO)$ budgets are not exceeded. The component of a full pWCRT distribution that is lost represents those scenarios where the system enters HI-criticality mode, and hence where there is no requirement for LO-criticality tasks to meet their deadlines.

As a building block for the analysis in the following sections, we use the function $\text{pRTA}(\tau_i, \delta, \delta LO, \delta HI, t)$ defined as follows.

Definition 4.1. $\text{pRTA}(\tau_i, \delta, \delta LO, \delta HI, t)$ is a function which returns a (full or partial) pWCRT distribution for task τ_i computed using the analysis described in Section 3.2, i.e. via (4), (5), and (6). This analysis starts from an initial distribution of type δ for task τ_i, and uses pWCET distributions of types δLO and δHI for preempting (i.e. higher priority) LO-criticality and HI-criticality tasks respectively. Further, jobs of higher priority LO-criticality tasks are only included in the computation of the pWCRT distribution if their release times are no later than the *cutoff* time t. The parameters δ, δLO, and δHI may take values *partLO*, *partHI*, *fullBE*, *degen*, and *fullHI* when the function is used.

4.2 pSMC Analysis

Recall that with the SMC scheme, LO-criticality tasks may execute in HI-criticality mode, the only constraint on their execution being budget enforcement.

For both LO- and HI-criticality tasks, there are two ways in which the tasks can fail to meet their timing requirements: (i) they can fail to complete within their budgets, (ii) they can fail to meet their deadlines. We assume that the budgets ($C(LO)$ and $C(HI)$) have been set such that the probability ($p(LO)$ or $p(HI)$) of each job of a task failing to complete within its budget is acceptable. We therefore focus only on the worst-case probability of deadline misses, assuming that the task of interest does not exceed its execution time budget. (Note this is similar to the view taken by deterministic schedulability analysis which assumes that the

probability of budget exceedance is acceptable and computes schedulability assuming absolute values for $C(LO)$ and $C(HI)$).

For HI-criticality tasks, we need to determine schedulability (i.e. upper bound the worst-case probability of deadline failure) in both LO- and HI-criticality modes. By contrast for LO-criticality tasks, we need only determine their schedulability in LO-criticality mode.

For a LO-criticality task τ_l, we can determine schedulability in LO-criticality mode using the *partLO* distribution for τ_l, and the $fullBE$ distribution for higher priority, LO-criticality tasks, since they may overrun their budgets, but there is no behaviour of these tasks that can cause HI-criticality mode to be entered. Finally, we need only consider the *partLO* distribution for HI-criticality tasks, as the remaining part of the full distribution for these tasks implies that the system enters HI-criticality mode, hence by using the function pRTA() from Definition 4.1 we have:

$$\mathcal{R}_l(LO) = \text{pRTA}(\tau_l, partLO, fullBE, partLO, \infty) \tag{8}$$

When analysing a HI-criticality task τ_h executing in LO-criticality mode, we cannot trust the behaviour of LO-criticality tasks[1]. Thus we must use degenerate distributions equating to the execution time budget $C(LO)$ for higher priority, LO-criticality tasks, thus we have:

$$\mathcal{R}_h(LO) = \text{pRTA}(\tau_h, partLO, degen, partLO, \infty) \tag{9}$$

Equations (8) and (9) provide the partial pWCRT distribution for each task conditional on the system operating in LO-criticality mode and the task not exceeding its own LO-criticality budget.

When analysing a HI-criticality task τ_h we need to determine the probability that it will miss its deadline irrespective of the criticality mode. Again we cannot trust the behaviour of LO-criticality tasks. Thus we must use degenerate distributions equating to the execution time budget $C(LO)$ for higher priority, LO-criticality tasks. To compute the pWCRT distribution irrespective of mode, we begin with the *partHI* distribution for the HI-criticality task τ_k (since we are interested in the case where it completes within its budget, but nevertheless misses its deadline) and include the $fullHI$ distribution for preempting higher priority, HI-criticality tasks.

$$\mathcal{R}_h(HI) = \text{pRTA}(\tau_h, partHI, degen, fullHI, \infty) \tag{10}$$

4.3 pAMC Analysis

The AMC and SMC schemes have identical behaviour in LO-criticality mode, hence the analysis given in Section 4.2 also provides pAMC analysis for both LO- and HI-criticality tasks in LO-criticality mode. That leaves pAMC analysis of HI-criticality tasks irrespective of mode.

To compute the pWCRT distribution for a HI-criticality task τ_h in both HI- and LO-criticality modes, we begin with the *partHI* distribution for the task and include the $fullHI$ distribution for preempting higher priority, HI-criticality tasks. As before, we assume that the behaviour of LO-criticality tasks cannot be trusted and therefore again make use of the *degen* distributions for those tasks.

$$\mathcal{R}_h(HI) = \text{pRTA}(\tau_h, partHI, degen, fullHI, R_h(LO)) \tag{11}$$

Note, we limit the jobs of higher priority LO-criticality tasks to those released by the LO-criticality response time $R_h(LO)$ of task τ_h given by deterministic analysis of AMC i.e. by (2). (The rationale for this is that if task τ_h is still executing beyond $R_h(LO)$ then it must be the case that the system has entered HI-criticality mode, and so no further releases of LO-criticality tasks are permitted). If $R_h(LO)$ cannot be obtained by deterministic analysis i.e. the task is unschedulable according to that analysis and $R_h(LO) > T_h$ then $R_h(LO)$ may be assumed to be infinite and the pAMC analysis (11) reduces to the pSMC analysis (10).

Comparing (11) with (10), it is easy to see that the pAMC analysis dominates pSMC. The only difference is the discounting of LO-criticality preemptions after time $R_h(LO)$ in the case of pAMC. Thus all task sets that are deemed schedulable[2] by pSMC analysis are also schedulable according to the pAMC analysis. Further, we note that the pAMC analysis dominates deterministic analysis for AMC, and similarly, pSMC analysis dominates deterministic analysis for SMC. This can be seen by considering the distributions used in the probabilistic analyses. These distributions and the resulting pWCRT distributions satisfy the *limit condition* [24]. The maximum values in each input distribution are the same as the values used in the corresponding deterministic analysis, thus the maximum value in the output pWCRT distributions are the same as the deterministic worst-case response times.

4.4 pAMC2 Alternative Analysis

A weakness of the pAMC analysis described in the previous section relates to its use of the value of $R_h(LO)$ computed via deterministic analysis. The analysis for pAMC is only able to outperform that for pSMC when a valid value of $R_h(LO)$ can be obtained. To address this problem, we provide an alternative probabilistic analysis for AMC, referred to as pAMC2. The analysis for LO- and HI-criticality tasks in LO-criticality mode remains unchanged and is given in Section 4.2, while that for a HI-criticality task irrespective of both mode is described below.

For the HI-criticality task of interest τ_h, let $R_h^*(LO)$ be a value chosen from the partial pWCRT distribution for LO-criticality mode, given by (9), such that the probability that the response time of the task in LO-criticality mode exceeds $R_h^*(LO)$ is $E = H(HI)/10$, i.e. one tenth of the acceptable threshold on the deadline miss probability for HI-criticality tasks[3]. From our choice of $R_h^*(LO)$, we know that if task τ_h has not completed execution by $R_h^*(LO)$ after it is released, then the probability that the task, and hence the system, is still executing in LO-criticality mode is no greater than E.

We now split the analysis for task τ_h into two cases:

Case 1: Represents scenarios where there are LO-criticality tasks released, i.e. the system is still in LO-criticality mode, more than $R_h^*(LO)$ time units after the release of task τ_h. We pessimistically assume that all of these scenarios lead to deadline misses for task τ_h; however, since by definition of $R_h^*(LO)$ the probability of these scenarios occurring is no greater than E they contribute at most E to the overall worst-case deadline miss probability for task τ_h.

Case 2: Scenarios where there are no LO-criticality tasks released more than $R_h^*(LO)$ time units after the release of task τ_h. To simplify

[1] The analysis given in [26] did not take this into account and is corrected here.

[2] Recall that by schedulable we mean that LO-criticality tasks must not exceed their WCDMP threshold in LO-criticality mode, and HI-criticality tasks must not exceed their WCDMP threshold in either mode.

[3] We choose the value of $E = H(HI)/10$ as a pragmatic means of both keeping $R_h^*(LO)$ small and E well below the threshold $H(HI)$.

the analysis, ignoring case 1, we pessimistically assume that the probability of these scenarios occurring is 1.

We now derive the pWCRT distribution for Case 2. Similar to the pSMC analysis, we compute the pWCRT distribution irrespective of mode. Here, we begin with the *partHI* distribution for task τ_h and similarly include the $fullHI$ distribution for preempting higher priority, HI-criticality tasks. As this analysis is for a HI-criticality task τ_h, we assume that the behaviour of LO-criticality tasks cannot be trusted and therefore again make use of the degenerate distributions for those tasks. Since the analysis is for Case 2, we limit the jobs of higher priority LO-criticality tasks to those released by time $R_h^*(LO)$.

$$\mathcal{R}_h(HI) = \text{pRTA}(\tau_k, partHI, degen, fullHI, R_h^*(LO)) \quad (12)$$

To account for Case 1, we later simply add E to the worst-case deadline miss probability of the task (see Section 4.5). Together with (8), this completes the analysis for *pAMC2*.

With the exception of the additional term E, the pAMC2 analysis would dominate both the pSMC analysis and also the pAMC analysis. Comparing (10) with (12), the former would hold since $R_h^*(LO) \leq \infty$. Comparing (11) with (12), the latter would hold since $R_h^*(LO) \leq R_h(LO)$ due to the limit condition. However, with the additional term E, dominance is no longer assured. We note that a dominant approach could be achieved by the simple expedient of declaring a task set schedulable if it is deemed schedulable under either the pAMC or pAMC2 analysis.

The pAMC2 analysis does however dominate deterministic analysis for AMC. This can be seen by considering any task that is schedulable according to deterministic analysis of AMC. $R_h^*(LO)$, as defined in the pAMC2 analysis cannot be greater than $R_h(LO)$ obtained by deterministic analysis in (2), since the distribution that $R_h^*(LO)$ is taken from satisfies the limit condition. As $R_h^*(LO) \leq R_h(LO)$ it follows that the pWCRT distribution characterising the behaviour of a HI-criticality task τ_h satisfies the limit condition with respect to the value $R_h(HI)$ derived by deterministic analysis in (3), i.e. it has no values $> R_h(HI)$. Since $R_h(HI) \leq D_k$, the only non-zero contribution to the overall worst-case deadline miss probability is E which is less than $H(HI)$ and so the task is also schedulable according to pAMC2 analysis.

4.5 Probabilistic schedulability

For a LO-criticality task τ_l to meet its timing requirements, then its execution time must not exceed its LO-criticality budget with more than a specified probability. This is guaranteed, as with deterministic analysis, by setting its execution time budget no lower than $C(LO)$. Secondly, its pWCRT distribution conditional on not exceeding its budget and that the system remains in LO-criticality mode must give a worst-case deadline miss probability that does not exceed the specified threshold, i.e. $P(\mathcal{R}_l(LO) > D_l) \leq H(LO)$. This can be determined for pSMC, pAMC, and pAMC2 using (8).

For a HI-criticality task τ_h to meet its timing requirements, then its execution time must not exceed its HI-criticality budget with more than a specified probability. This is again guaranteed, as with deterministic analysis, by setting its execution time budget no lower than $C(HI)$. Further, its pWCRT distribution (valid irrespective of mode), and conditional on not exceeding its budget, must result in a worst-case deadline miss probability that does not exceed the specified threshold i.e. $P(\mathcal{R}_h(HI) > D_h) \leq H(HI)$, where $\mathcal{R}_h(HI)$ can be determined for pSMC using (10), and for pAMC using (11). For the pAMC2 analysis, $P(\mathcal{R}_h(HI) > D_h) + E \leq H(HI)$ where $E = H(HI)/10$ and $\mathcal{R}_h(HI)$ is determined via (12).

4.6 Improved Support

$C(LO)$ represents the low assurance estimate of the WCET of a task; however, the use of this value at runtime is as an execution time budget which for clarity we now denote separately as $C(BU)$. This budget is used with respect to LO-criticality jobs to indicate when such a job should be aborted if it has not yet completed execution. In the case of HI-criticality jobs the budget is used to indicate when HI-criticality mode should be entered and thus with the AMC scheme when further LO-criticality jobs may no longer be released.

Using the pWCET distributions for both LO- and HI-criticality tasks, we can improve support for LO-criticality execution by increasing the $C(BU)$ budgets used for both LO- and HI-criticality tasks above the original $C(LO)$ values. This has a number of effects:

(i) It decreases the probability that a LO-criticality job will overrun its budget and be aborted before completing;
(ii) It reduces the probability that HI-criticality mode will be entered;
(iii) It increases the computed pWCRT distributions for both LO- and HI-criticality tasks leading to higher worst-case deadline miss probabilities.

Improved support for LO-criticality tasks can be obtained by tightening their timing requirements, i.e. by decreasing the permitted probability of exceeding the execution time budget, and the threshold on the worst-case deadline miss probability. Thus we may decrease these values from $p(LO)$ and $H(LO)$ to some new values $p(BU)$ and $H(BU)$, which we assume are equal. The LO-criticality execution time budget for each task (i.e. $C(BU)$) can then be read off from the 1-CDF of its pWCET distribution, corresponding to the value $p(BU)$. We modify the $C(BU)$ values for HI-criticality tasks in the same way. The pSMC, pAMC, and pAMC2 analyses can then be applied, using these new $C(BU)$ values in place of $C(LO)$ and the new distributions that they imply, to determine schedulability for LO-criticality tasks with respect to the new threshold $H(BU)$, and for HI-criticality tasks with respect to the original threshold $H(HI)$.

Since schedulability is monotonically decreasing for both LO- and HI-criticality tasks with respect to increases in execution time budgets $C(BU)$ and decreases in the threshold $H(BU)$, we may use a binary search to determine the smallest value of $p(BU)(= H(BU))$ commensurate with schedulability. The initial values for the search are $p(BU) = 1$ which corresponds to a LO-criticality execution time of zero, and $p(BU) = p(HI)$, which corresponds to a LO-criticality execution time equal to $C(HI)$.

With deterministic analysis, we may also use a binary search to determine the smallest value of $p(BU)(= H(BU))$ commensurate with schedulability. In this case we can use the $C(BU)$ value obtained from the pWCET distribution for each task in place of $C(LO)$. We note that increasing $C(BU)$ for HI-criticality tasks reduces schedulability in LO-criticality mode, and in the case of AMC also in HI-criticality mode via an increase in the value of $R(LO)$.

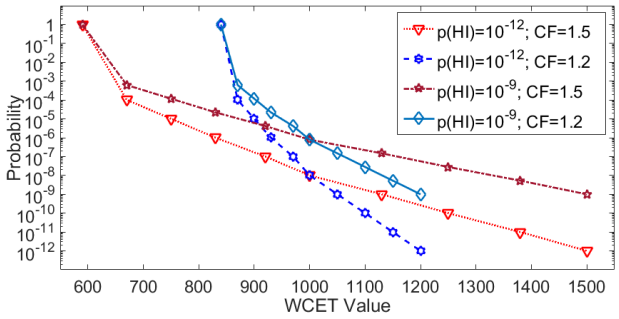


Figure 3: Example of possible pWCET distributions.

In Section 5.4 we quantify the relative improvements in support for LO-criticality execution which can be obtained using probabilistic and deterministic analysis techniques.

5 EVALUATION

In this section we evaluate the effectiveness of the probabilistic analysis techniques introduced in this paper. In particular, we examine the performance improvements that can be obtained with respect to the corresponding deterministic analyses.

5.1 Task set parameter generation

The task set parameters used in our experiments were randomly generated as follows:

- Task utilisations ($U_i = C_i/T_i$) (for LO-criticality mode) were generated using the UUnifast algorithm [9].
- Task periods were generated according to a log-uniform distribution. By default, the range of task periods was two orders of magnitude $r = 2$, e.g. from 10ms to 1000ms.
- The LO-criticality execution time of each task was set based on the utilisation and period: $C_i(LO) = U_i T_i$.
- The HI-criticality execution time of each task was given by $C_i(HI) = CF \cdot C_i(LO)$, where CF is the Criticality Factor (default $CF = 1.5$).
- The probability that a generated task was a HI-criticality task was given by the parameter CP (default $CP = 0.5$).
- Task deadlines were constrained, chosen from a uniform distribution in the range $[CF.C(LO), T]$.
- Task priorities were set in deadline monotonic order.

We generated the pWCET distribution for each task via extrapolation from the $C(LO)$ and $C(HI)$ parameter values. We assumed that the probability of exceeding $C(LO)$ is $p(LO) = 10^{-8}$, and that the probability of executing for, but not exceeding $C(HI)$ is $p(HI) = 10^{-12}$. To determine intermediate points, we assumed that the pWCET distributions have an exponential tail. Thus we assumed that the 1-CDF of the pWCET was a straight line on an exceedance graph with probabilities given on a log scale, as depicted in Figure 3. Note the longer lines are for a Criticality Factor of 1.5 and thus show more execution time variation than the shorter lines which are for CF = 1.2. Further, the lower lines are for the default settings of $p(HI) = 10^{-12}$ and $p(LO) = 10^{-8}$, whereas the upper lines are for $p(HI) = 10^{-9}$ and $p(LO) = 10^{-5}$, as an example of one of the pairs of values used in a later experiment. (Note the left most point on each line collects the remaining part of the distribution so that the probability mass sums to 1).

The Thresholds $H(LO)$ and $H(HI)$ on the maximum acceptable worst-case deadline miss probabilities for LO- and HI-criticality tasks were set to 10^{-8} and 10^{-12} respectively.

5.2 Schedulability tests

We investigated the performance of the following techniques and associated schedulability tests.

- pSMC: Probabilistic SMC analysis (Section 4.2).
- pAMC and pAMC2: Probabilistic AMC analysis (Sections 4.3 and 4.4).
- dSMC: Deterministic SMC analysis [5] (Section 3.1).
- dAMC: Deterministic AMC analysis [5] (Section 3.1).

In addition we include two further tests:

- dUB: Task sets pass this 'test' if they are schedulable according to deterministic analysis of FPPS in each of the individual LO-criticality and HI-criticality modes with priorities in deadline monotonic order. This is a deterministic necessary test for any fixed priority preemptive mixed criticality scheduling algorithm [5].
- pUB: Task sets pass this 'test' if they are schedulable according to probabilistic analysis of FPPS in each of the individual LO-criticality and HI-criticality modes with priorities in deadline monotonic order.

The dominance relationships between the algorithms and tests implies that in all figures there is an ordering to the lines: pUB dominates pAMC which dominates pSMC. Similarly dUB dominates dAMC which dominates dSMC. Further pUB dominates dUB, pAMC dominates dAMC, and pSMC dominates dSMC. As explained earlier, pAMC2 does not strictly dominate pAMC or pSMC; however, it does dominate dAMC and hence also dSMC. pUB dominates pAMC2. The purpose of the experiments is to examine the relative performance of the different schemes.

5.3 Baseline experiment

In our baseline experiment, the LO-criticality utilisation was varied from 0.05 to 1 in steps of 0.05. For each utilisation value, 1000 task sets were generated and the schedulability of those task sets determined for the different schemes.

Figure 4 plots the percentage of task sets generated that were deemed schedulable for a system of 10 tasks, with on average 50% of those tasks having HI-criticality (CP = 0.5) and each task having a HI-criticality execution time that is 1.5 times its LO-criticality execution time (CF = 1.5). The utilisation values (x-axis) are computed using the $C(LO)$ values and periods for each task. (Note, the graphs are best viewed in an electronic version of this paper in colour).

Figure 4 shows that the probabilistic analyses (pAMC2, pAMC and pSMC) provide substantially improved performance compared to the deterministic analyses (dAMC and dSMC), with many more task sets deemed schedulable. This is because the probabilistic analysis is able to account for the full extent of the pWCET distributions, and thus the very small probability that multiple jobs take long execution times leading to a very long response time. We note that pAMC2, pAMC and pSMC are able to deem some tasks sets with LO-criticality utilisation equal to 1 schedulable. This is correct, and is a reflection of the shape of the pWCET distributions (see Figure 3). Recall that $C(LO)$ has a probability of exceedance of 10^{-8} thus once the distributions for a number of tasks are

convolved, the probability that all of them execute for $C(LO)$ or more becomes very small.

We observe that there is only a small difference between the results for pSMC and those for pAMC; the difference is visible for utilisation values from 0.7 to 1. This is because there is only a difference between the pWCRT distributions calculated by the two methods when the cutoff time for pAMC, given by $R(LO)$, can be computed deterministically (i.e. when it is less than the task's period). In the cases where this is possible, the task set is often also schedulable according to pSMC.

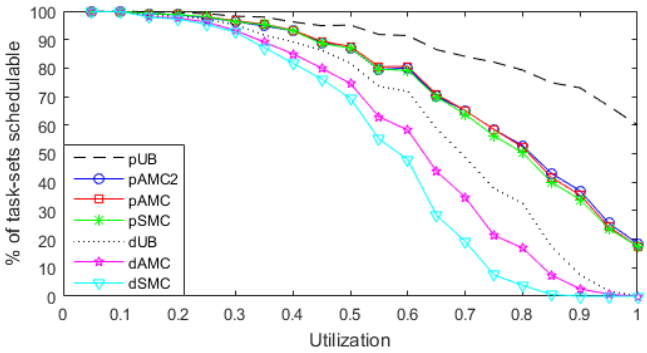


Figure 4: Percentage of task sets deemed schedulable according to the different schedulability tests.

Overall, in this experiment, we found only 146 task sets (less than 1%) that were schedulable according to pAMC, but not schedulable according to pSMC. For utilizations greater than 0.7, there were 57 task sets that were schedulable according to pAMC2, but not according to pAMC. In contrast, for utilizations less than 0.7, pAMC outperformed pAMC2; with 49 task sets that were schedulable according to pAMC, but not according to pAMC2. Similarly, comparing pAMC2 and pSMC, for utilizations greater than 0.45, there were 163 task sets that were schedulable according to pAMC2, but not according to pSMC. However, for utilizations less than 0.45, pSMC slightly outperformed pAMC2; with 9 task sets that were schedulable according to pSMC, but not according to pAMC2. These results clearly show the incomparability between pAMC2 and both pAMC and pSMC. Out of a total of 20000 task-sets that we analyzed, 14855 (74.3%) were schedulable according to pAMC2, 14847 (74.2%) were schedulable according to pAMC, and 14701 (73.5%) according to pSMC. By comparison, the figures for dSMC and dAMC were 10596 (53%) and 11619 (58.1%) respectively.

In Appendix B of the technical report [25] on which this paper is based, we provide additional results for weighted schedulability [7] experiments showing how the performance of the analysis techniques varies with: the Criticality Factor (CF), the proportion of HI-criticality tasks (CP), the number of orders of magnitude range between the minimum and maximum task period, the task set cardinality, and the probability threshold used for $p(HI)$.

5.4 Quantifying the improvements

In this subsection, we report on a further experiment quantifying the improvements in LO-criticality execution (i.e. lower likelihood of entering HI-criticality mode or aborting LO-criticality tasks due to budget overruns) that can be obtained by using probabilistic analysis versus deterministic analysis, as outlined in Section 4.6.

In this experiment, we used a search over each possible value of $C(BU)$ corresponding to the different probabilities $p(BU)$ in the pWCET distribution (see Figure 3) and recorded the smallest value of $p(BU)$ for which each task set was schedulable according to the scheduling schemes and analysis techniques (i.e. dSMC, dAMC, pSMC, pAMC, and pAMC2).

We generated task sets with a LO-criticality utilisation of 0.7 using the default parameter settings (10 tasks, $CF = 1.5$, $CP = 0.5$ etc.). Only task sets that were schedulable according to dSMC with $C(LO)$ determined for $p(BU) = 10^{-1}$ were included in the results.

Figure 5 shows the minimum schedulable $p(BU)$ value (from which LO-criticality budgets $C(BU)$ are derived) for each of the analysis techniques. Note we have ordered the 100 task sets studied according to the ease with which they could be scheduled from easiest to hardest, based on the minimum schedulable $p(BU)$ values for the different analysis techniques. From the graph, we can observe that the first 65 out of the 100 task sets were schedulable according to pAMC2 with $p(BU) = 10^{-12}$ and the associated execution time budgets $C(BU)$, whereas none of the task sets were schedulable with those values according to the deterministic methods. In fact with $p(BU) = 10^{-10}$ no task sets were schedulable according to dSMC and only 9 according to dAMC. Further, the median value for minimum schedulable $p(BU)$ was 10^{-4} for dSMC, 10^{-7} for dAMC, and 10^{-12} for pSMC, pAMC, and pAMC2. The results of this experiment clearly show the much greater support for LO-criticality execution that is achievable by applying probabilistic analysis techniques. Given the relationship between $C(BU)$ and $p(BU)$ defined by the criticality factor $CF = 1.5$, in this experiment, pSMC, pAMC, and pAMC2 support LO-criticality execution time budgets that are typically more than 50% higher than with dSMC and dSMC.

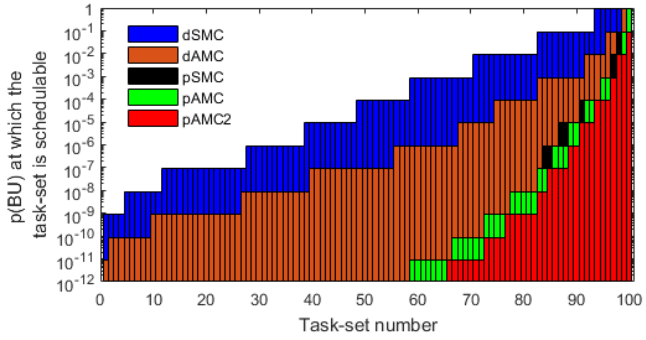


Figure 5: Minimum $p(BU)$ value needed for each of 100 task sets to be schedulable under the different analyses.

The differences in performance between the various techniques are highlighted by the different coloured areas in Figure 5. Thus blue highlights cases where dAMC outperforms dSMC, while orange plus black highlights cases where pAMC outperforms dAMC. Note the difference, shown in black, between the performance of pSMC and pAMC is small. There were just 7 out of 100 task sets where pAMC gave improved performance over pSMC. In contrast, the

green area shows that there were 35 out of 100 task sets where pAMC2 outperformed pAMC.

6 CONCLUSIONS

In this paper we introduced probabilistic analysis for fixed priority preemptive scheduling of mixed criticality systems under the SMC [6] and AMC schemes [5]. This analysis makes use of probabilistic worst-case execution time (pWCET) distributions to compute probabilistic worst-case response time distributions (pWCRT) and thus the worst-case deadline miss probability (WCDMP) for each task under SMC and AMC. Provided that the worst-case deadline miss probability is below the appropriate threshold for that task, then it is declared schedulable.

The main contributions of this paper are as follows:

(i) Introducing probabilistic analysis of mixed criticality systems based on a richer mixed criticality model using pWCET distributions.

(ii) Deriving probabilistic worst-case response time analysis for both LO- and HI-criticality tasks under the SMC and AMC schemes.

(iii) Showing via an experimental evaluation that the probabilistic analyses, pSMC, pAMC and pAMC2, substantially out-perform their deterministic counterparts.

(iv) Showing that the the probabilistic analysis framework is able to provide improved support for LO-criticality tasks based on the observation that the LO-criticality WCET estimates used in the deterministic representation are in reality tunable execution time budgets at different levels of assurance. The more effective probabilistic analysis enables these budgets to be increased for both LO- and HI-criticality tasks, thus reducing the chance that HI-criticality mode is entered and lowering the likelihood that LO-criticality jobs have to be dropped or aborted.

We found that the baseline probabilistic analysis (pSMC) was so much more effective than the deterministic methods that using the LO-criticality response time $R(LO)$ computed via deterministic techniques as a cutoff for AMC was largely ineffective in improving performance in the probabilistic case. In other words, pSMC performed very nearly as well as pAMC. This lead us to refine the analysis in pAMC2, making use of a cutoff $R^*(LO)$ derived from probabilistic analysis. This approach resulted in an improvement in performance over pAMC and pSMC for high utilisation task sets.

Acknowledgements

The research in this paper was partially funded by the EUROSTARS RETINA Project, the FR BGLE funded Departs project, the FR LEOC Capacites project, the FR FUI Waruna project, the ESPRC grant MCCps (EP/P003664/1), the Inria International Chair program, and the Singapore Ministry of Education Tier-2 grant, ARCS9/14. EPSRC Research Data Management: No new primary data was created during this study.

REFERENCES

[1] Y. Abdeddaim and D. Maxim. Probabilistic schedulability analysis for fixed priority mixed criticality real-time systems. In *Proceedings of DATE*, 2017.

[2] S. Altmeyer, L. Cucu-Grosjean, and R.I. Davis. Static probabilistic timing analysis for real-time systems using random replacement caches. *Real-Time Systems*, 51(1):77–123, 2015.

[3] S. Altmeyer and R.I. Davis. On the correctness, optimality and precision of static probabilistic timing analysis. In *Proceedings of DATE*, pages 26:1–26:6, 2014.

[4] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8:284–292, 1993.

[5] S. Baruah, A. Burns, and R.I. Davis. Response-Time Analysis for Mixed Criticality Systems. In *Proceedings of RTSS*, pages 34–43, 2011.

[6] S. Baruah and S. Vestal. Schedulability Analysis of Sporadic Tasks with Multiple Criticality Specifications. In *Proceedings of ECRTS*, pages 147–155, 2008.

[7] A. Bastoni, B. Brandenburg, and J. Anderson. Cache-related preemption and migration delays: Empirical approximation and impact on schedulability. In *Proceedings of the Workshop on OSPERT*, pages 33–44, 2010.

[8] I. Bate, A. Burns, and R.I. Davis. An enhanced bailout protocol for mixed criticality embedded software. *IEEE Transactions on Software Engineering*, PP(99):1–1, 2016.

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

[10] A. Burns and R.I. Davis. A survey of research into mixed criticality systems. *ACM Computing Surveys*, (to appear):35, 2017.

[11] F. Cazorla, E. Quiñones, T. Vardanega, L. Cucu, B. Triquet, G. Bernat, E. Berger, J. Abella, F. Wartel, M. Houston, L. Santinelli, L. Kosmidis, C. Lo, and D. Maxim. PROARTIS: probabilistically analyzable real-time systems. *ACM Trans. Embedded Comput. Syst.*, 12(2s):94, 2013.

[12] L. Cucu-Grosjean, L. Santinelli, M. Houston, C. Lo, T. Vardanega, L. Kosmidis, J. Abella, E. Mezzetti, E. Quiones, and F. J. Cazorla. Measurement-based probabilistic timing analysis for multi-path programs. In *Proceedings of ECRTS*, pages 91–101, 2012.

[13] R.I. Davis, L. Santinelli, S. Altmeyer, C. Maiza, and L. Cucu-Grosjean. Analysis of probabilistic cache related pre-emption delays. In *Proceedings of ECRTS*, pages 168–179, 2013.

[14] J. L. Diaz, J. M. Lopez, M. Garcia, A. M. Campos, Kanghee Kim, and L. L. Bello. Pessimism in the stochastic analysis of real-time systems: concept and applications. In *Proceedings of RTSS*, pages 197–207, 2004.

[15] J.L Díaz, D.F. Garcia, K. Kim, C.G. Lee, L.L. Bello, López J.M., and O. Mirabella. Stochastic analysis of periodic real-time systems. In *Proceedings of RTSS*, 2002.

[16] M.K. Gardner and J.W. Lui. Analyzing stochastic fixed-priority real-time systems. In *proceedings of TACAS*, 1999.

[17] D. Griffin, B. Lesage, A. Burns, and R. I. Davis. Static probabilistic timing analysis of random replacement caches using lossy compression. In *Proceedings of RTNS*, pages 289–298, 2014.

[18] Z. Guo, L. Santinalli, and K. Yang. Edf schedulability analysis on mixed-criticality systems with permitted failure probability. In *Proceedings of RTCSA*, 2015.

[19] M. Ivers and R. Ernst. Probabilistic network loads with dependencies and the effect on queue sojourn times. In *proceedings QSHINE*, pages 280–296, 2009.

[20] M. Joseph and P. Pandya. Finding Response Times in a Real-Time System. *The Computer Journal*, 29(5):390–395, May 1986.

[21] B. Lesage, D. Griffin, S. Altmeyer, L. Cucu-Grosjean, and R. I. Davis. On the analysis of random replacement caches using static probabilistic timing methods for multi-path programs. In *Real-Time Systems*, 2017. to appear.

[22] B. Lesage, D. Griffin, S. Altmeyer, and R.I. Davis. Static probabilistic timing analysis for multi-path programs. In *Proceedings of RTSS*, 2015.

[23] J.M. Lopez, J. L. Diaz, J. E., and D. Garcia. Stochastic analysis of real-time systems under preemptive priority-driven scheduling. *Real-Time Systems*, 40(2), 2008.

[24] D. Maxim and L. Cucu-Grosjean. Response time analysis for fixed-priority tasks with multiple probabilistic parameters. In *Proceedings of RTSS*, pages 224–235, 2013.

[25] D. Maxim, R.I. Davis, L. Cucu-Grosjean, and A. Easwaran. Probabilistic analysis for mixed criticality systems using fixed priority preemptive scheduling. Technical report, Department of Computer Science, University of York. "https://www.cs.york.ac.uk/ftpdir/reports/2017/YCS/505/YCS-2017-505.pdf".

[26] D. Maxim, R.I. Davis, L. Cucu-Grosjean, and A. Easwaran. Probabilistic analysis for mixed criticality scheduling with smc and amc. In *Proceedings of WMC*, 2016.

[27] D. Maxim, M. Houston, L. Santinelli, L. Cucu-Grosjean, and R.I. Davis. Re-Sampling for Statistical Timing Analysis of Real-Time Systems. In *Proceedings of RTNS*, 2012.

[28] P. Huang S. Draskovic and L. Thiele. On the safety of mixed-criticality scheduling. In *Proceedings of WMC*, 2016.

[29] L. Santinelli and L. George. Probabilities and mixed-criticalities: the probabilistic c-space. In *Proceedings of WMC*, 2015.

[30] B. Tanasa, U. D. Bordoloi, P. Eles, and Z. Peng. Probabilistic response time and joint analysis of periodic tasks. In *Proceedings of ECRTS*, pages 235–246, July 2015.

[31] 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 *Proceedings of RTAS*, 1995.

[32] S. Vestal. Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution Time Assurance. In *Proceedings of RTSS*, 2007.

[33] M. H. Woodbury and K. G. Shin. Evaluation of the probability of dynamic failure and processor utilization for real-time systems. In *Proceedings of RTSS*, pages 222–231, Dec 1988.