Real-Time Systems (2021) 57:397–442
https://doi.org/10.1007/s11241-021-09365-4

Schedulability of probabilistic mixed-criticality systems

Stefan Draskovic[1] · Rehan Ahmed[2] · Pengcheng Huang[3] · Lothar Thiele[1]

Accepted: 16 January 2021 / Published online: 21 February 2021

Abstract

Mixed-criticality systems often need to fulfill safety standards that dictate different requirements for each criticality level, for example given in the 'probability of failure per hour' format. A recent trend suggests designing this kind of systems by jointly scheduling tasks of different criticality levels on a shared platform. When this is done, the usual assumption is that tasks of lower criticality are degraded when a higher criticality task needs more resources, for example when it overruns a bound on its execution time. However, a way to quantify the impact this degradation has on the overall system is not well understood. Meanwhile, to improve schedulability and to avoid over-provisioning of resources due to overly pessimistic worst-case execution time estimates of higher criticality tasks, a new paradigm emerged where task's execution times are modeled with random variables. In this paper, we analyze a system with probabilistic execution times, and propose metrics that are inspired by safety standards. Among these metrics are the probability of deadline miss per hour, the expected time before degradation happens, and the duration of the degradation. We argue that these quantities provide a holistic view of the system's operation and schedulability.

Keywords Mixed-criticality scheduling · Probabilistic execution times · Stochastic analysis

✉ Stefan Draskovic
stefan.draskovic@tik.ee.ethz.ch

Rehan Ahmed
rehan.ahmed@itu.edu.pk

Pengcheng Huang
pengcheng.huang@ch.abb.com

Lothar Thiele
lothar.thiele@tik.ee.ethz.ch

[1] Computer Engineering and Networks Laboratory (TIK), ETH Zürich, Zurich, Switzerland

[2] Computer Engineering Department, Information Technology University (ITU), Lahore, Pakistan

[3] ABB Research Center Switzerland, Baden, Switzerland

Table 1 Failure rate specification for different criticality levels

Level	Failure condition	Failure rate
A	Catastrophic	10^{-9}/h
B	Hazardous	10^{-7}/h
C	Major	10^{-5}/h
D	Minor	10^{-3}/h
E	No effect	n/a

1 Introduction

Mixed-criticality (MC) systems are real-time systems that feature tasks of different criticality levels. Typical application domains include avionics and automotive (Burns and Davis 2017). In MC systems, each task has an associated criticality level. Depending on the criticality level, a failure of a task, for example due to deadline miss, can have a more or less severe impact on the overall safety of the system. Due to possible catastrophic consequences of a system failure, MC systems for some application domains are subject to certification standards. For example, DO-178C (Rtca/do-178c 2012) is a standard for avionics systems. It defines five criticality levels, 'A' to 'E', with 'A' being the highest criticality level. Here, a failure of a task of criticality 'A' can have a negative impact on the overall safety of the aircraft, while a failure of a task of criticality 'D' may only slightly increase the aircraft crew's workload. Quantitatively, an application's criticality correlates to a tolerable failure rate under a given certification standard. The failure rates of all tasks, under their respective criticality levels, have to be guaranteed for certification of the overall system. As an example, Table 1 states the tolerable failure rates for DO-178B.[1]

Traditionally, industry favors physical segregation of tasks based on their criticality level (Tămaş-Selicean and Pop 2015). This implies, for example, that tasks of each criticality level execute on their own hardware, and tasks of different criticality levels do not interfere. However, such a physical separation based on criticality levels can lead to system under-utilization and complex distributed multi-processor architectures. Recently, there has been a push towards integrating tasks of different criticality levels on a single hardware platform (Burns and Davis 2017). The advantages for such consolidation include reduction in cost, power dissipation, weight, as well as maintenance.

Unfortunately, this consolidation of criticality levels makes isolating tasks of different criticality levels problematic. Essentially, a low criticality level 'D' task may hinder the execution of a higher criticality level 'B' task, possibly resulting in a deadline miss—which can be considered as a type of failure. To counter this, researchers have proposed several schemes which are covered in detail in Sect. 2. Broadly speaking, the approaches are based on an execution time abstraction proposed by Vestal (2007). Vestal's model builds on the Worst-case Execution Time

[1] DO-178B was replaced by DO-178C in 2012.

(WCET) abstraction. He assumes that tasks have a set of WCET estimates with different levels of confidence. The system is required to meet the deadline of a criticality level 'A' task for the highest confidence and most pessimistic WCET estimates. For lower criticality tasks, correct execution needs to be guaranteed for less pessimistic WCET estimates. Prominent proposed approaches that build on Vestal's model feature mode-based scheduling schemes that ensure that the system executes tasks of all criticality levels correctly when less pessimistic WCETs estimates are not overrun, while reduced service to tasks of lower criticality levels is in place when this is not the case.

In this paper, instead of taking a single WCET estimate as in the traditional real-time model, or a criticality dependent set of WCET estimates as per Vestal's model, we assume a stochastic model of execution times. For each task, the execution time is modeled with an independent random variable. This additional information on the execution time allows us to have improved schedulability due to the so called multiplexer gain, i.e., the likelihood of high execution times of many tasks occurring simultaneously is very small. Under the proposed scheme there is a non-zero probability of a high criticality task missing its deadline. If the probability is less than the failure rate specification of the criticality level, see for example Table 1, then the MC system can still be schedulable according to the probabilistic bounds on deadline misses.

Individual tasks are assumed to be periodic with constrained deadlines. The platform is assumed to have a single core. We assume a dual-critical model, where the criticality of tasks can be either LO or HI. The system is also assumed to have two modes of operation: LO- and HI-criticality mode. In the LO-criticality mode, all tasks are executed normally. In the HI-criticality mode, newly released jobs of LO tasks are starting in a degraded mode so that preference is given to HI tasks.

The application of stochastic execution to MC systems is not new and several recent works exist (Maxim et al. 2017; Masrur 2016; Guo et al. 2015). However, existing results do not provide a holistic scheduling scheme and analysis covering all execution modes and transitions. A detailed accounting of existing schemes and their limitations is given in Sect. 2. In the following, we suppose that a MC scheduling scheme fulfills the following requirements:

- Schedulability analysis of tasks is provided for each criticality level in each system mode.
- Conditions that should trigger a mode switch are defined.
- Analysis of the time spent in each system mode is provided.
- A method to consolidate these individual components and compute a metric comparable to the Probability of Failure per Hour for tasks of each criticality level is given.

In this paper, we address all of these individual components. Specifically, we make the following contributions:

1. We propose conditions that trigger a mode switch, both from LO- to HI-criticality mode (LO → HI), and from HI- to LO-criticality mode (HI → LO).
2. We provide a detailed stochastic analysis of LO-criticality mode. Using the analysis, the *Probability of Deadline Miss per Hour* in this mode is computed for tasks of both criticality levels.
3. We provide a first stochastic analysis of HI-criticality mode. Using the analysis, the maximal time spent in HI-criticality mode is obtained, along with the *Probability of Deadline Miss per Hour* for tasks of both criticality levels. Also taken into account is the probability the system enters HI-criticality mode.
4. Using contributions 1–3, we compute the overall *Probability of Deadline Miss per Hour* values for all tasks by consolidating the respective values for LO- and HI-criticality mode. This allows us to compare these probabilities with the permitted ones found in typical certification standards.
5. We determine the probability that a LO task is started in its degraded mode.

Due to these contributions, we claim that this is the first work which provides a system-wide approach to MC scheduling, while considering a stochastic model of task execution times.

Organization: This paper is organized as follows: Sect. 2 highlights the related research in Mixed-criticality scheduling and in stochastic analysis. It also highlights the limitations of existing research which are addressed by this work. Section 3 states our system model. The model includes the task model and the model of the MC system. This is followed by Sect. 4, which states and explains important definitions and operations for stochastic analysis of systems with non-deterministic execution times. Section 5 covers the proposed analysis for getting *Probability of Deadline Miss per Hour* values, both for all LO and for all HI tasks. This section also has important intermediate results such as the duration of LO- and HI-criticality mode, and the probability of each event that causes a system mode switch. Results are covered in Sect. 6. In this section, we evaluate various schedulability metrics and design trade-offs for MC systems. Conclusion is given in Sect. 7, followed by references.

2 Related work

Vestal's paper (2007) is the first paper that presents the MC model, where safety-critical tasks have multiple WCET estimates with different levels of assurance. Based on the model, a preemptive fixed priority scheduling scheme for sporadic task sets is presented: Static Mixed Criticality (SMC). In the widely examined dual-criticality case, hard guarantees are given to HI tasks, but LO jobs might miss their deadline if a HI job overruns its optimistic WCET. As well as this, a LO job is de-scheduled if it overruns its WCET.

Baruah et al. (2011) introduced an important fixed priority scheduling scheme, Adaptive Mixed Criticality (AMC), which defines a system that can operate in different modes. The system starts in LO-criticality mode where all tasks are scheduled to execute according to their optimistic WCET estimates. If any job overruns its optimistic WCET, a switch to HI-criticality mode happens, where all LO tasks are de-scheduled. This way, HI tasks are guaranteed to meet their deadlines all the time, whereas LO tasks have this guarantee only in LO-criticality mode.

EDF scheduling has been adapted to Vestal's model as well. Baruah et al. (2011) propose a scheduling scheme for sporadic task sets based on EDF, called EDF-VD. In this scheme, the deadlines of all HI tasks are scaled down by a single scaling factor so that an overrun is detected early. Once an overrun is detected, the system enters HI-criticality mode where all LO tasks are de-scheduled. In this scheme, all tasks meet their deadlines if no optimistic WCET is overrun, while only HI tasks meet their deadlines if some of them are overrun. Ekberg and Yi (2012) use demand-bound functions to scale the deadlines of HI tasks individually, by a heuristic search strategy. Deadlines are chosen so that the schedulability of the system is maximized. The LO- and HI-criticality mode model in this scheme is similar to the one used in Baruah et al. (2011). Huang et al. (2014) amend EDF-VD to include degraded service for low criticality tasks while the system is in HI-criticality mode. The paper also presents an upper bound on the duration of this mode. Park and Kim (2011) present another EDF-based scheme, CBEDF. Here, high criticality tasks are always guaranteed to execute, while some guarantees are given to tasks of low criticality using offline empty slack location discovery. Vestal's model with two modes of operation was also investigated for time-triggered scheduling, most notably in Baruah and Fohler (2011). For a comprehensive overview of research into Mixed Criticality, we refer the reader to the review by Burns and Davis (2017), while for a discussion on the applicability of Mixed Criticality systems to industry and its safety-critical practices see Ernst and Di Natale (2016).

As for probabilistic MC systems, related work often models them with probabilistic Worst-Case Execution Time (pWCET) distributions, which are seen as extending Vestal's model such that each task has a large number of WCETs with various levels of confidence (Burns and Davis 2017; Davis and Cucu-Grosjean 2019). A pWCET distribution comes from either the randomness inherent in a system and its environment, or the lack of knowledge we have about a system, or possibly both (Davis et al. 2017). To derive these distributions, well established methods like static probabilistic timing analysis (Devgan and Kashyap 2003), or measurement based probabilistic timing analysis techniques (Cucu-Grosjean et al. 2012) already exist. Ideally, modeling tasks with pWCET distributions removes dependency between them, meaning any task-set can be analyzed as though all tasks had independent execution times. In practice, by using pWCET distributions, these dependencies are reduced but not removed completely. This still poses a major problem in applying pWCET methodologies for real-time computing. For an extensive survey of timing analysis techniques, we refer the reader to Davis and Cucu-Grosjean (2019). In this paper we assume that tasks' execution times are modeled with random variables which are given, and these random variables can be seen as an abstraction of ideal pWCETs.

For the analysis of probabilistic MC systems, obtaining probabilistic response times is key. The survey on probabilistic schedulability analyses by Davis and Cucu-Grosjean (2019) lists various approaches to response time analysis. Our paper builds mainly upon the work of Díaz et al. (2002, 2004), as their analysis of real-time systems is pessimistic. Using probabilistic analysis, existing work often presents scheduling schemes where individual tasks have certain permissible deadline miss probabilities. Examples are Maxim et al. (2017) and Abdeddaïm and Maxim (2017), were SMC and AMC scheduling are adapted to a probabilistic MC model, demonstrating the improvement in schedulability. Masrur (2016) proposes a scheme with no mode switches, where LO tasks have a soft guarantee on meeting their deadline as well. Alahmad and Gopalakrishnan (2016, 2018) use a Markov decision process to provide probabilistic guarantees to jobs, and also formulate an optimization problem that provides the scheduling policy. Santinelli and George (2015), Santinelli and Guo (2018), and Santinelli et al. (2016) examine probabilistic MC systems by doing a sensitivity analysis, which focuses on the impact made by varying execution times. However, we observe that a holistic characterization of probabilistic mixed-criticality systems remains largely unexplored in the state-of-the-art. Deadline miss probabilities of individual jobs are often not aggregated into system-wide metrics, for example in Masrur (2016) and Maxim et al. (2017). We note that giving soft guarantees to individual tasks is not equivalent to guaranteeing a probability of deadline miss per hour. Another related work, Guo et al. (2015), analyzes a simple probabilistic model, where a HI task has just two WCETs and their corresponding probabilities of occurrence. Using the model, they propose a EDF-based scheduling algorithm which has an allowed probability of a timing fault happening system-wide. Finally, Küttler et al. (2017) consider a model where some guarantees are available to tasks of lower criticality. They propose lowering the priorities of lower criticality tasks in certain modes of operation. Still, without characterizing the duration of modes, we believe that the impact of degradation of LO tasks can not be properly quantified.

Finally, our own previous work (Draskovic et al. 2016) addresses the probability of deadline miss in LO-criticality mode of a dual mode system, while also commenting on the time before a transition to HI-criticality mode happens. However, a system-wide overview of the system is not given as HI-criticality mode is not analyzed. In this paper, we address the aforementioned limitations of the state-of-the-art.

3 System model

We start this section with an informal overview of our system model, before precise definitions are presented. The model is an extension of Vestal's original model (2007), and as is with Adaptive Mixed Criticality (Baruah et al. 2011), there are two modes of operation, LO- and HI-criticality mode.

LO-criticality mode can be considered a normal mode of operation, and the system is expected to operate in this mode most of the time. HI-criticality mode can be considered an emergency mode, where newly instantiated LO jobs are started and running in degraded mode so preference is given to the execution of HI jobs. More specifically, HI criticality tasks are not affected by the mode of operation, these task are always released and executed until their completion. LO criticality tasks have two variants: each LO job can be released in degraded or regular mode. They always finish in the mode they started with. Though LO tasks are never dropped, they are released with degradation when the system is in HI-criticality mode. In practice, this means that there are two implementations of each task, and the degraded variant offers a reduced functionality. For example, the numerical result is computed with less precision. Vestal's original model specifies dropping LO jobs when HI jobs need more resources, and our model can be seen as a generalization where not executing a job is the extreme case.

The system starts in LO-criticality mode, and remains there until a mode switching event occurs. The first mode switching event is the only one discussed for non-probabilistic MC systems, and is thus found in previous work, for example (Baruah et al. 2011; Ekberg and Yi 2012; Huang et al. 2014; Maxim et al. 2017): a HI job's execution lasts longer than a provided threshold. The second mode switching event is when a HI job misses its deadline. It is introduced to reduce the probability of consecutive deadline misses of HI jobs. Note that a HI job might miss its deadline without overrunning its threshold execution time, for example because it was blocked by jobs of higher priority. Finally, the third mode switch event is when a long backlog of LO jobs accumulates, which could in turn produce an arbitrarily high backlog when entering HI mode. Once in HI-criticality mode, the system switches back to LO-mode the first time it is idle.

Using this model, we say a task-set to be schedulable using fixed priority preemptive scheduling, if the probability that any job misses its deadline during an hour of operation is sufficiently small, and if the ratio of LO jobs released in degraded mode is acceptable.

General notation on random variables This work deals with discrete random variables, and they are denoted using calligraphic symbols, for example $\mathcal{A}$. The probability function of $\mathcal{A}$, noted $p_{\mathcal{A}}(\cdot)$, tells us the probability that $\mathcal{A}$ takes a specific value u: $p_{\mathcal{A}}(u) = \mathbb{P}(\mathcal{A} = u)$. Without loss of generality, we assume that the possible values of all random variables span the full range of natural numbers. If the maximal and minimal values with non-zero probability of $\mathcal{A}$ exist, and are noted $u_{\max}$ and $u_{\min}$, then the probability function can be represented in vector notation:

$$p_{\mathcal{A}} = [p_{\mathcal{A}}(u_{\min}), \dots, p_{\mathcal{A}}(u_{\max})]^{\intercal}. \tag{1}$$

Let us define a relation to compare two random variables $\mathcal{A}$ and $\mathcal{B}$, as was done by Díaz et al. (2002).

Definition 1 (*First-Order Stochastic Dominance*) $\mathcal{A}$ is greater or equal than $\mathcal{B}$, written as $\mathcal{A} \succeq \mathcal{B}$, if and only if

$$\forall l \geq 0 : \quad \sum_{u=l}^{\infty} p_{\mathcal{A}}(u) \geq \sum_{u=l}^{\infty} p_{\mathcal{B}}(u). \tag{2}$$

Note that probability densities can be incomparable.

We introduce a shorthand notation for the probability that a variable modeled with random variable $\mathcal{A}$ has a value greater than scalar s. Instead of the cumbersome expression $\sum_{s<i} \mathbb{P}(i = \mathcal{A})$, we use $\mathbb{P}(s < \mathcal{A})$.

Finally, we introduce a simple notation $[s]^1$ to indicate that a scalar or expression s is limited to a maximum value of 1, $[s]^1 = \min(s, 1)$.

Task model A task-set Π consists of N independent tasks. Each task is periodic, constrained deadline, with an initial phase and a criticality level. A single task τ_i

is characterized by tuple $(T_i, D_i, \phi_i, \chi_i, \mathcal{C}_i)$, where T_i is the period, D_i is the relative deadline, ϕ_i is the phase, $\chi_i \in \{\text{LO}, \text{HI}\}$ is the task's criticality level, and $\mathcal{C}_i$ models the probabilistic execution time. $\mathcal{C}_i$ has a maximal value with non-zero probability, which is the WCET, noted $C_i^{\max}$. Tasks with criticality level LO and HI are referred to as 'LO tasks' and 'HI tasks', respectively. An instance j of task τ_i is called a job, and denoted as $\tau_{i,j}$. Each job $\tau_{i,j}$ has its release time $r_{i,j} = \phi_i + (j-1) \cdot T_i$, and its absolute deadline $d_{i,j} = r_{i,j} + D_i$. The hyperperiod HP of a set of tasks is defined to be the least common multiple of all task periods.

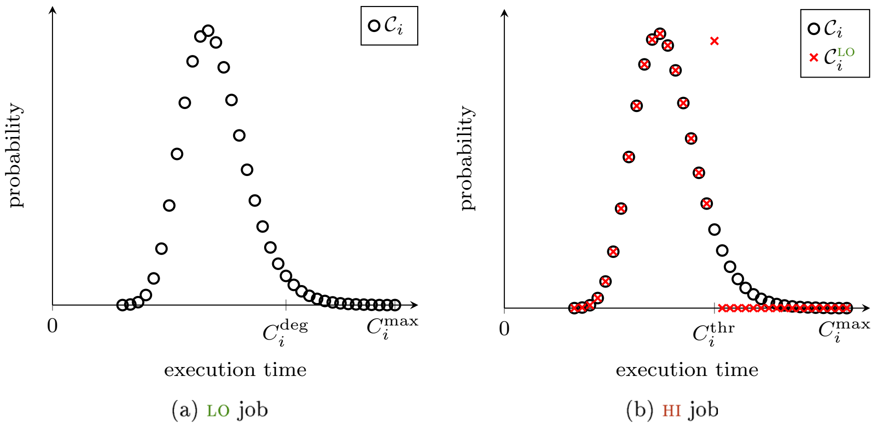
We model the execution times of each task τ_i with known independent and identically distributed random variables $\mathcal{C}_i$. This means that there is no dependency between the execution times of any two jobs, regardless of whether they are of the same task or not, and execution times of all jobs of one task are modeled with the same random variable. However, the provided analysis is safe, i.e., if the computed bounds hold for a given set of probabilistic execution times, they also hold if the execution times are smaller or equal according to Definition 1. Therefore, the probabilistic execution times $\mathcal{C}_i$ can also be regarded as ideal probabilistic worst case execution times (pWCETs), which would remove the requirement that execution times of jobs are independent.

In the standard MC model (Vestal 2007), HI tasks have an optimistic and a pessimistic WCET estimate, and LO tasks are executed by the processor only if HI tasks meet their optimistic WCET estimates during operation. The reasoning behind this is the assumption that most of the time HI tasks will not execute for longer than their optimistic WCET estimate, so less computational resources are needed for the correct operation of the system. In this paper, we assume that the distribution of the execution time of each task $\mathcal{C}_i$ is known. Therefore, instead of the optimistic WCET estimate, for each HI task we define a threshold execution time value C_i^{thr}. We assume this value is a given design choice. Note that the probability that a HI task executes for longer than this threshold is $\mathbb{P}(\mathcal{C}_i > C_i^{\text{thr}})$. The precise way this threshold is used in scheduling of jobs is described later in this section. Additionally, instead of not executing LO jobs in order to free up resources, we introduce that each LO job can be released in degraded or regular mode. If it executes with degradation, its WCET is C_i^{deg}. The C_i^{deg} value is assumed to given as a design choice. It could be zero if the task is not to be run in HI-criticality mode, or it can be any value less than its WCET: in this case it is assumed that a lower functionality is provided.

For the execution time of HI tasks, it is useful to introduce the following random variable that describes a worst-case behavior as long as the analyzed system is still in LO-critical mode.

Definition 2 (*Trimmed Execution Time*) Random variable $\mathcal{C}_i^{\text{LO}}$ models the execution time of HI tasks τ_i, but modified such that they do not execute for longer than C_i^{thr}:

$$p_{\mathcal{C}_i^{\text{LO}}}(u) = \begin{cases} p_{\mathcal{C}_i}(u) & u < C_i^{\text{thr}}, \\ \sum_{v=C_i^{\text{thr}}}^{C_i^{\max}} p_{\mathcal{C}_i}(v) & u = C_i^{\text{thr}}, \\ 0 & u > C_i^{\text{thr}}. \end{cases} \tag{3}$$



(a) LO job

(b) HI job

Fig. 1 Task execution times, with named values and trimmed execution time $\mathcal{C}_i^{\text{LO}}$

Figure 1a illustrates the $\mathcal{C}_i$ of a LO task, as well as the WCET denoted as C_i^{deg} in degraded mode. Figure 1b illustrates the $\mathcal{C}_i$ of a HI task as well as the trimmed execution time $\mathcal{C}_i^{\text{LO}}$ with the corresponding C_i^{thr} and C_i^{max} values.

This definition differs from the one found in many related works, i.e. Draskovic et al. (2016), where the execution time of HI tasks in LO-critical mode is defined as the conditional probability $\mathbb{P}(p_{\mathcal{C}_i}(u) = u | u \leq C_i^{\text{thr}})$, often called 'truncated' execution time. The 'trimmed' execution times, as defined in this paper, are by definition greater or equal to the equivalent 'truncated' execution times. This paper uses 'trimmed' execution times because they simplify the analysis of HI-criticality mode, namely by simplifying initial conditions noted by Definition 12. The cost of this simplification is that it introduces pessimism in the LO-criticality mode analysis, however this has been found to be numerically negligible through simulations. Nevertheless, using the 'truncated' execution times option with a more complex analysis is also possible. For more information, see the comment on future work in the conclusion.

The response time of job $\tau_{i,j}$ is modeled with random variable $\mathcal{R}_{i,j}$. The way this variable can be obtained and upper-bounded is presented in Sect. 4. The deadline miss probability of job $\tau_{i,j}$ is the probability that this job finishes after its deadline $\mathsf{DMP}_{i,j} = \mathbb{P}(\mathcal{R}_{i,j} > d_{i,j})$.

Schedulability In this paper, we consider a single-core platform. A simple execution model is used, where task preemption overhead is zero.

As in the standard MC model, the system is defined to operate in two modes of operation, LO- and HI-criticality mode. When the system is operating in LO-criticality mode, both LO and HI jobs are released. When the system is operating in HI-criticality mode, HI jobs are released normally, while LO jobs are released in degraded mode.

In this paper the definition of schedulability is inspired by the probability-of-failure-per-hour notion. Therefore, we first define the probability of deadline miss per hour, before defining schedulability. We also define the probability of degraded job, a proportion of how many LO jobs execute in degraded mode in the long run.

Definition 3 (*Failure Probabilities*) The *probability of deadline miss per time interval T for* HI *or* LO *jobs* is denoted as $\mathsf{DMP}_{\mathrm{HI}}(T)$ or $\mathsf{DMP}_{\mathrm{LO}}(T)$, respectively. It is the probability that at least one HI or LO job misses its deadline during a time interval of length T.

Formally, we define $\mathsf{DMP}_{\mathrm{HI}}(T)$ and $\mathsf{DMP}_{\mathrm{LO}}(T)$ as:

$$\mathsf{DMP}_{\chi}(T) = \max_{\forall t} \ \mathbb{P}\big(\exists \tau_{i,j} \in S_{\chi}(t) \ : \ \tau_{i,j} \text{ misses its deadline}\big), \tag{4}$$

where $\chi = \{\mathrm{LO}, \mathrm{HI}\}$, and $S_{\chi}(t) = \{\tau_{i,j} \mid \chi_i = \chi \ \wedge \ t \leq r_{i,j} < t + T\}$.

Definition 4 (*Probability of Degraded Job*) The *probability of degraded* LO *jobs* $\mathsf{PDJ}_{\mathrm{deg}}$ is the probability that any individual LO job is released in degraded mode:

$$\mathsf{PDJ}_{\mathrm{deg}} = \max_{\forall t} \frac{|S_{\text{LO-DEG}}(t)|}{|S_{\mathrm{LO}}(t)|}, \tag{5}$$

where

$$S_{\mathrm{LO}}(t) = \{\tau_{i,j} \mid \chi_i = \mathrm{LO} \ \wedge \ t \leq r_{i,j} < t + T\}$$
$$S_{\text{LO-deg}}(t) = \{\tau_{i,j} \mid \chi_i = \mathrm{LO} \ \wedge \ t \leq r_{i,j} < t + T \ \wedge \ \tau_{i,j} \text{ is in degraded mode}\}$$

Definition 5 (*Schedulability*) A MC system is $(\sigma_{\mathrm{HI}}, \sigma_{\mathrm{LO}}, \sigma_{\mathrm{deg}})$-schedulable if $\mathsf{DMP}_{\mathrm{HI}}(1h) \leq \sigma_{\mathrm{HI}}$, $\mathsf{DMP}_{\mathrm{LO}}(1h) \leq \sigma_{\mathrm{LO}}$, and $\mathsf{PDJ}_{\mathrm{deg}} \leq \sigma_{\mathrm{deg}}$, where $1h$ denotes the duration of 1 h.

The probabilistic MC scheduling scheme used in this paper can now be defined:

Definition 6 (*Probabilistic MC Scheduling*) In LO-criticality mode, all tasks are scheduled using a provided fixed-priority preemptive schedule. The system starts in LO-criticality mode, and remains in it until one of the following events causes a transition to HI-criticality mode:

1. A HI job overruns its threshold execution time C_i^{thr}.
2. A HI job misses its deadline.
3. The system-level backlog, meaning the amount of pending execution, becomes higher than a predefined threshold $B_{\max}$.

In HI-criticality mode, the same fixed-priority preemptive schedule is used, but LO jobs are released with degradation in order to free up the processor. LO jobs starting in LO-criticality mode are still continuing in their normal mode with execution time C_i. The system remains in HI-criticality mode until it becomes idle for the first time.

4 Preliminaries

With tasks having probabilistic execution times, a set of computational primitives are required to perform the schedulability analysis. A probabilistic analysis of real-time systems, on which our analysis is based, was described by Díaz et al. (2002, 2004). We summarize the analysis technique in this section. The analysis and its primitives are used extensively in the following sections to perform the schedulability analysis of mixed-criticality systems.

The analysis requires computation of the *backlog*, i.e., the sum of pending execution times of all ready jobs. For each priority level i there is a backlog containing the execution times of all pending jobs with priority i or higher. When a new job with priority i arrives, all backlogs with level i or lower are increased by adding its execution time. Adding the execution time random variable to a backlog is done using *convolution*. Executing a job decreases the backlogs of all levels i that are equal or smaller than the priority of the job. Decreasing the backlog is done using *shrinking*.

Definition 7 (*Backlog*) The ith priority backlog at time t, $\mathcal{B}_i(t)$, is a random variable that describes the sum of all remaining execution times of pending jobs of priority not less than i, at time t. The backlog $\mathcal{B}_i(t-)$ is the same as $\mathcal{B}_i(t)$, except it does not take into account jobs released at time t.

Using convolution to compute backlog after arrival of a job Suppose that a job $\tau_{i,j}$ is released at time $r_{i,j}$, and $\mathcal{B}_k(r_{i,j}-)$ is the kth priority backlog at time $r_{i,j}$, but excludes the newly released job. Assuming that $i \geq k$, and that no other job is released at the same time, backlog $\mathcal{B}_k(r_{i,j})$ can be computed using the convolution operator $\otimes$:

$$p_{\mathcal{B}_k(r_{i,j})} = p_{\mathcal{B}_k(r_{i,j}-)} \otimes p_{\mathcal{C}_i}. \tag{6}$$

Backlog reduction due to execution of highest priority job Let us assume that in the interval $t_0 < t < t_1$ there are no job arrivals. During this interval, the backlog is decreased as the processor executes pending jobs. If $\mathcal{B}_i(t_0)$ is the ith priority backlog at time t_0, the corresponding backlog at time t can be computed using the so-called *shrinking* operation. Specifically, for computing backlog at time $t_0 < t < t_1$, the following equation can be used:

$$p_{\mathcal{B}_i(t)}(u) = \begin{cases} \sum_{j=0}^{t-t_0} p_{\mathcal{B}_i(t_0)}(j) & u = 0, \\ p_{\mathcal{B}_i(t_0)}(u + t - t_0) & u > 0. \end{cases} \tag{7}$$

In other words, the backlog after an execution of $t - t_0$ time units is computed by left-shifting the initial backlog by $t - t_0$, while truncating at zero since the processor is idle when no pending execution is present. For brevity, we define the corresponding shrinking function of a random variable $\mathcal{B}$:

$$\text{shrink}(\mathcal{B}, m)(u) = \begin{cases} \sum_{j=0}^{m} p_{\mathcal{B}}(j) & u = 0, \\ p_{\mathcal{B}}(u + m) & u > 0. \end{cases} \tag{8}$$

Backlog State Space Exploration First, we define the function `bsse` for computing the backlog at some time $t + u$ given the backlog at time t.

Definition 8 (*Backlog Computation*) $\texttt{bsse}\big(\mathcal{B}_i(t), \Pi, i, t, u\big)$ is a function for computing the ith priority backlog at time $t+u$, i.e., $\mathcal{B}_i(t+u)$. We assume that the ith priority backlog at time t is $\mathcal{B}_i(t)$, and that the task arrivals and execution times in the interval $[t, t+u)$ are in accordance with task set Π.

The computation of `bsse` can be done by applying the definition of a task set as well as the previously described operations, namely convolution and shrinking. We demonstrate this using the following example.

Example Task-set Π is given, and consists of task $\tau_1 = (T_1 = 5, \pi_1 = 0, D_1 = 5, \mathcal{C}_1)$, and of task $\tau_2 = (10, 0, 10, \mathcal{C}_2)$. Task τ_2 has a higher priority. The backlogs at time $0-$ at priority levels 1 and 2 are given as $\mathcal{B}_1(0-)$ and $\mathcal{B}_2(0-)$, respectively. For this set-up, find the backlog at time $10-$ at priority level 1, as well as the backlog at time 7 at priority level 2.

Solution. The following combination of convolution and shrinking computes $\mathcal{B}_1(10-) = \texttt{bsse}\big(\mathcal{B}_1(0-), \Pi, 2, 0-, 10-\big)$, by taking into account the execution times of all jobs:

$$\begin{aligned}
p_{\mathcal{B}_1(0)} &= p_{\mathcal{B}_1(0-)} \otimes p_{\mathcal{C}_1} \otimes p_{\mathcal{C}_2},\\
p_{\mathcal{B}_1(5-)} &= \text{shrink}(\mathcal{B}_1(0), 5),\\
p_{\mathcal{B}_1(5)} &= p_{\mathcal{B}_1(5-)} \otimes p_{\mathcal{C}_1},\\
p_{\mathcal{B}_1(10-)} &= \text{shrink}(\mathcal{B}_1(5), 5).
\end{aligned}$$

For computing the highest priority backlog, task τ_1 is ignored. Using the same procedure, we obtain $\mathcal{B}_2(7) = \texttt{bsse}\big(\mathcal{B}_2(0-), \Pi, 1, 0-, 7\big)$:

$$\begin{aligned}
p_{\mathcal{B}_2(0)} &= p_{\mathcal{B}_2(0-)} \otimes p_{\mathcal{C}_2},\\
p_{\mathcal{B}_2(7)} &= \text{shrink}(\mathcal{B}_2(0), 7).
\end{aligned}$$

4.1 Upper bound of backlog

In order to provide a holistic schedulability analysis, we need to determine upper bounds of the backlogs for all time instances within any future hyperperiod, i.e., we are interested in a set of random variables $\overline{\mathcal{B}}_i(t)$ such that $\mathcal{B}_i(n \cdot \text{HP} + t) \preceq \overline{\mathcal{B}}_i(t)$ for all priority levels i, future hyperperiods $n \geq 0$ and time instances within a hyperperiod $0 \leq t < \text{HP}$. We start by computing the steady-state backlog and proceed by showing that it provides the desired upper bound.

Computation of the steady state backlog The ith priority backlog at the start of the nth hyperperiod is $\mathcal{B}_i(n \cdot \text{HP})$, but this backlog may be different for each n. However, the sequence of random variables $\{\mathcal{B}_i(n \cdot \text{HP})\}$ can be viewed as a

Markov process as shown by Díaz et al. (2002). Specifically, they present the following theorem about the existence of a limit to the above mentioned sequence, including the corresponding proof:

Theorem 1 (Section 4.2 of Díaz et al. 2002) *The sequence of backlogs* $\{\mathcal{B}_i(n \cdot \mathrm{HP})\}$ *for* $n \geq 0$, *where i is a priority level, has a limit if the average system utilization is less than one, and if the sequence of jobs remains the same each hyperperiod. If it exists, this limit is called the ith priority steady state backlog at the beginning of the hyperperiod, and noted* $\overline{\mathcal{B}}_i(0)$.

For computing the steady state backlog at the start of a hyperperiod $\overline{\mathcal{B}}_i(0)$, Diaz et al. propose three methods. The first method is an exact one stated in Sect. 4.3.2 of Díaz et al. (2002) and exploits the structure of the infinite dimension transition matrix $\boldsymbol{P}$. A second method (Sect. 4.3.3 of Díaz et al. (2002)) finds an approximate value of $\overline{\mathcal{B}}_i(0)$ by truncating $\boldsymbol{P}$ to make its dimension finite. Finally, a third method is to iterate over hyperperiods until the following relaxed steady state condition is satisfied:

$$\max_{i,x} \left\{ \left| p_{\mathcal{B}_i(k \cdot \mathrm{HP})}(x) - p_{\mathcal{B}_i((k-1) \cdot \mathrm{HP})}(x) \right| \right\} < \epsilon. \tag{9}$$

This condition states that the maximum difference between all ith priority backlogs must not exceed a configurable small value ϵ. This method does not require computation nor truncation of the transition matrix $\boldsymbol{P}$. For further details on choosing appropriate initial backlogs, please refer to Díaz et al. (2004).

Pessimism of the steady state backlog Assuming that the initial backlog is zero at every priority level, and that the sequence of jobs remains the same each hyperperiod, it has been shown in Díaz et al. (2004) that the ith priority steady state backlog is an upper bound to all ith priority backlogs at the start of the hyperperiod. The following two Lemmas can be used to show that the backlogs at the beginning of a hyperperiod are increasing from hyperperiod to hyperperiod. They state that the operations of convolution and shrinking preserve the partial ordering of random variables.

Lemma 1 (Property 3 in Díaz et al. 2004) *Given three positive random variables* $\mathcal{A}, \mathcal{B}, \mathcal{C}$. *If* $\mathcal{A} \preceq \mathcal{B}$, *then* $\mathcal{A} + \mathcal{C} \preceq \mathcal{B} + \mathcal{C}$.

Lemma 2 (Property 6 in Díaz et al. 2004) *Given two positive random variables* $\mathcal{A}, \mathcal{B}, \mathcal{C}$. *If* $\mathcal{A} \preceq \mathcal{B}$, *then* $\mathrm{shrink}(\mathcal{A}, m) \preceq \mathrm{shrink}(\mathcal{B}, m)$.

Algorithm 1 Computing response time of a job

```
1: procedure rta ($\mathcal{B}_i(r_{i,j})$, $\Pi$, $\tau_{i,j}$)
2:     $p_{\mathcal{R}_{i,j}} \leftarrow \mathcal{B}_i(r_{i,j}) \otimes C_i$
3:     $t \leftarrow 0$
4:     while $t \leq D_{i,j}$ do
5:         for each preempting job $\tau_{k,l}$ that arrives at $r_{i,j} + t$ do
6:             $\{R_l, R_u\} = \mathrm{Split}(\mathcal{R}_{i,j}, t)$
7:             $p_{\mathcal{R}_{i,j}}(0, \ldots, t-1) \leftarrow R_l$
8:             $p_{\mathcal{R}_{i,j}}(t, \ldots) \leftarrow \mathcal{R}_u \otimes \mathcal{C}_k$
9:         $t \leftarrow t + 1$
       return $\mathcal{R}_{i,j}$
10: function Split($\mathcal{X}$, m)
11:     $X_l \leftarrow [p_{\mathcal{X}}(0), p_{\mathcal{X}}(1), \ldots p_{\mathcal{X}}(m-1), 0, 0, \ldots 0]$
12:     $X_u \leftarrow [0, 0, \ldots 0, p_{\mathcal{X}}(m), p_{\mathcal{X}}(m+1), \ldots p_{\mathcal{X}}(\hat{\mathcal{X}})]$
13:     return $X_l$, $X_u$
```

Now, the following Theorem can be shown by means of the above considerations: We have, by definition, $\overline{\mathcal{B}}_i(t) = \lim_{n\to\infty} \mathcal{B}_i(t + n \cdot \text{HP})$ for all $n \geq 0$ and $0 \leq t < \text{HP}$, and we know from Theorem 1 that $\mathcal{B}_i(n \cdot \text{HP}) \preceq \overline{\mathcal{B}}_i(0)$ for all $n \geq 0$.

Theorem 2 (Theorem 1 in Díaz et al. 2004) *Assuming that the initial backlog is zero, and that the sequence of jobs remains the same each hyperperiod, the ith priority backlog at time t inside every hyperperiod is upper bounded by the ith priority steady state backlog at time t inside the hyperperiod*:

$$\forall i \;:\; p_{\mathcal{B}_i(0)}(0) = 1 \;\Rightarrow\; \forall t \in [0, \text{HP}),\ \forall n \in \mathbb{N} \;:\; \mathcal{B}_i(n \cdot \text{HP} + t) \preceq \overline{\mathcal{B}}_i(t)$$

In summary, if the initial backlog is zero, the steady-state backlog $\overline{\mathcal{B}}_i(t)$ provides an upper bound for all backlogs within any future hyperperiod. This result will be used extensively in the the response time analysis described next.

4.2 Response time analysis

The response time of a job $\mathcal{R}_{i,j}$ tells us when this job will finish its execution, relative to its release time. We summarize the procedure as proposed by Díaz et al. (2002). The response time of a given job $\tau_{i,j}$ is influenced by the initial backlog at its release time $B_i(r_{i,j})$, and the computation times of all jobs that preempt the job. Therefore we can define a function:

$$\mathcal{R}_{i,j} = \texttt{rta}\big(\mathcal{B}_i(r_{i,j}),\, \Pi,\, \tau_{i,j}\big). \tag{10}$$

The pseudocode for computing response times is given in Algorithm 1. For a given job $\tau_{i,j}$, first $\mathcal{C}_i$ is convolved with the the current ith priority backlog (line 2). This would provide us with the response time of $\tau_{i,j}$, if there were no preempting jobs. When a preempting job is released at a given point in time, then the probability function vector of $\tau_{i,j}$'s response time is split in two portions (line 6): the part before preemption (R_l), and the part after preemption (R_u). The part after preemption is convolved with the probability function vector of the preempting job's computation time, and the result is added to R_l in order to get $\tau_{i,j}$'s response time after this preemption (lines 7 and 8). The probability function of $\mathcal{R}_{i,j}$ is only computed until the job's deadline $d_{i,j}$.

Next, we present a theorem that we will use to obtain the worst-case hourly deadline miss probability. Beforehand, the Lemma shows that the response time function rta is monotone in the backlog at the release time of the job.

Lemma 3 (Theorem 1, Property 3 of López et al. 2008) *Given two random variables* $\mathcal{A}, \mathcal{B}$. *If* $\mathcal{A} \preceq \mathcal{B}$, *then* $\texttt{rta}\big(\mathcal{A}, \Pi, \tau_{i,j}\big) \preceq \texttt{rta}\big(\mathcal{B}, \Pi, \tau_{i,j}\big)$.

As the steady-state backlog at any time within a hyperperiod is always greater than or equal to the backlog at the corresponding time within any hyperperiod, the following Lemma can be obtained.

Lemma 4 *Assuming the initial backlog is zero, substituting any backlog* $\mathcal{B}_i(r_{i,j})$ *with the appropriate steady state backlog* $\overline{\mathcal{B}}_i(r_{i,j})$ *in the response time analysis, produces a value greater or equal to the response time.*

$$\forall i \ : \ p_{\mathcal{B}_i(0)}(0) = 1 \ \Rightarrow \ \texttt{rta}\big(\mathcal{B}_i(r_{i,j}), \Pi, \tau_{i,j}\big) \preceq \texttt{rta}\Big(\overline{\mathcal{B}}_i(r_{i,j} \bmod \text{HP}), \Pi, \tau_{i,j}\Big).$$

Proof This Lemma is a direct consequence of Lemma 3 and Theorem 2 as well as the results in López et al. (2008). □

The value $\texttt{rta}\Big(\overline{\mathcal{B}}_i(r_{i,j} \bmod \text{HP}), \Pi, \tau_{i,j}\Big)$ will be named the steady state response time, and denoted as $\overline{\mathcal{R}}_{i,j}$. Note that use of the steady-state backlog $\overline{\mathcal{B}}_i$ leads to an upper bound of the response time $\mathcal{R}_{i,j}$. Based on these results, we can now determine an upper bound on the response time of each job. Due to the fact that we defined the steady-state (worst case) hyperperiod, we can finally determine the worst-case deadline miss probability of a job $\tau_{i,j}$ within any hyperperiod. Instead of using the modulo operation as in Lemma 4 we can also just look at jobs $\tau_{i,j}$ within the single worst case hyperperiod with $0 \leq j < \text{HP}/T_i$.

Theorem 3 *The deadline miss probability of a job* $\tau_{i,j}$ *denoted as* $\mathsf{DMP}_{i,j}$ *can be bounded as follows*:

$$\forall i, 0 \leq j < \text{HP}/T_i \ : \ \mathsf{DMP}_{i,j} \leq \overline{\mathsf{DMP}}_{i,j} = \mathbb{P}\Big(d_{i,j} < \texttt{rta}\Big(\overline{\mathcal{B}}_i(r_{i,j}), \Pi, \tau_{i,j}\Big)\Big).$$

Proof The proof follows directly from the results described in López et al. (2008) as well as Lemma 4. □

5 Analysis of mixed-criticality systems with stochastic task execution times

In this section, we determine the $(\sigma_{\text{HI}}, \sigma_{\text{LO}}, \sigma_{\text{deg}})$-schedulability of a mixed-critical task set Π as defined in Definition 5. To this end, we compute upper bounds on probabilities that there is at least one deadline miss of a LO or HI job within 1 h, i.e., $\mathsf{DMP}_{\text{HI}}(T)$ or $\mathsf{DMP}_{\text{LO}}(T)$, respectively, for a time interval of length $T = 1$ h. In addition, we will compute an upper bound on the probability that a LO job operates in degraded mode $\mathsf{PDJ}_{\text{deg}}$. The underlying concept of the forthcoming analysis is described next.

Let us start with the computation of the probability $\mathsf{PDJ}_{\text{deg}}$ that a LO job operates in degraded mode. This probability can be upper bounded by noting that LO jobs are executed only in their degraded mode if their release time $r_{i,j}$ happens during HI-criticality mode. Therefore, we will first determine the maximal length $\Delta^{\text{HI}}_{\max}$ of any HI -criticality mode execution. In addition, we determine an upper bound on the probability, that there is at least one mode switch within a single hyperperiod denoted as $\mathsf{P}^{\text{HP}}_{\text{LO}\to\text{HI}}$. Using these two values, we can bound the relative time the system is in HI mode and therefore, the probability that a LO job operates in degraded mode.

To determine upper bounds on probabilities $\mathsf{DMP}_{\text{HI}}(1h)$, $\mathsf{DMP}_{\text{LO}}(1h)$ that there is at least one deadline miss of a LO or HI job within 1 h, we first look at upper bounds on the probabilities that at least one LO or HI job misses its deadline during any HI -criticality mode execution that is started within a hyperperiod, denoted as $\mathsf{DMP}^{\text{HI}}_{\text{HI}}$ or $\mathsf{DMP}^{\text{HI}}_{\text{LO}}$, respectively. Note that the upper index denotes the mode, whereas the lower one denotes the criticality of the jobs we are considering. In addition, we determine an upper bound on the probability that at least one LO or HI job misses its deadline during a hyperperiod under the conditions that first, no mode switches take place and second, HI jobs do not overrun their threshold C^{thr}. We denote these values as $\mathsf{DMP}^{\text{LO}}_{\text{HI}}$ or $\mathsf{DMP}^{\text{LO}}_{\text{LO}}$, respectively. Again, the upper index concerns the mode and the lower one the criticality of the considered jobs. Now we can determine the desired probabilities $\mathsf{DMP}_{\text{HI}}(T)$ and $\mathsf{DMP}_{\text{LO}}(T)$ by combining (a) the w.c. probabilities $\mathsf{DMP}^{\text{LO}}_{\text{HI}}$ and $\mathsf{DMP}^{\text{LO}}_{\text{LO}}$ that a deadline miss happens during a hyperperiod if the system is in LO -criticality mode, (b) the w.c. probabilities $\mathsf{DMP}^{\text{HI}}_{\text{HI}}$ or $\mathsf{DMP}^{\text{HI}}_{\text{LO}}$ that at least one LO or HI job misses its deadline during any HI-criticality mode started within a hyperperiod.

We will now first determine bounds $\mathsf{PDJ}_{\text{deg}}$ and $\mathsf{DMP}_{\chi}(1h)$ using the above defined quantities: $\Delta^{\text{HI}}_{\max}$, $\mathsf{P}^{\text{HP}}_{\text{LO}\to\text{HI}}$, $\mathsf{DMP}^{\text{HI}}_{\chi}$ and $\mathsf{DMP}^{\text{LO}}_{\chi}$ for HI and LO jobs, i.e., for $\chi \in \{\text{LO}, \text{HI}\}$. Afterwards, we will explain how these quantities can be determined.

5.1 Probability of job degradation

In this section, we will compute an upper bound on the probability that a LO job operates in degraded mode, i.e., $\mathsf{PDJ}_{\text{deg}}$. As described above, we will make use of the maximal duration of a HI-criticality mode execution and the probability that there is no mode switch within a hyperperiod.

Definition 9 (*Maximal Duration of High-Criticality-Mode*) The quantity $\Delta^{\text{HI}}_{\text{max}}$ denotes the maximal duration the system is continuously executing in HI-criticality mode.

Definition 10 (*Mode Switch Probability*) The quantity $\text{P}^{\text{HP}}_{\text{LO}\rightarrow\text{HI}}$ denotes an upper bound on the probability that there is at least one mode switch LO → HI within a single hyperperiod.

Using these definitions, we can determine an upper bound on the desired quantity.

Theorem 4 *The probability of degradation of a* LO *job can be bounded as follows*:

$$\text{PDJ}_{\text{deg}} \leq \left\lceil \frac{\Delta^{\text{HI}}_{\text{max}}}{\text{HP}} + 1 \right\rceil \text{P}^{\text{HP}}_{\text{LO}\rightarrow\text{HI}}. \tag{11}$$

Proof We obtain this value by multiplying the probability that HI-criticality mode is entered during one hyperperiod, with the the number of LO jobs that are released in degraded mode when it does.

First, note that there is some constant number K of LO jobs with that are released every hyperperiod. From the moment one HI-criticality mode is entered, it executes at least partly in at most $\lceil 1 + \Delta^{\text{HI}}_{\text{max}} / \text{HP} \rceil$ hyperperiods. Therefore, what ever the number of mode switches is inside one hyperperiod, in the worst case, all LO jobs from this and the next $\lceil \Delta^{\text{HI}}_{\text{max}} / \text{HP} \rceil$ hyperperiods are executed in degraded mode. In other words, $K \cdot \lceil \Delta^{\text{HI}}_{\text{max}} / \text{HP} + 1 \rceil$ LO jobs are degraded.

Second, let us note that there is at least one mode switch within a hyperperiod with probability $\text{P}^{\text{HP}}_{\text{LO}\rightarrow\text{HI}}$. Combining this probability with the number of LO jobs that are degraded if a mode switch happens, we get:

$$\begin{aligned} \text{PDJ}_{\text{deg}} &\leq \left(K \cdot \left\lceil \frac{\Delta^{\text{HI}}_{\text{max}}}{\text{HP}} + 1 \right\rceil \text{P}^{\text{HP}}_{\text{LO}\rightarrow\text{HI}} + 0 \cdot (1 - \text{P}^{\text{HP}}_{\text{LO}\rightarrow\text{HI}}) \right) K^{-1} \\ &= \left\lceil \frac{\Delta^{\text{HI}}_{\text{max}}}{\text{HP}} + 1 \right\rceil \text{P}^{\text{HP}}_{\text{LO}\rightarrow\text{HI}}. \end{aligned}$$

□

This upper bound on the probability of degradation of a LO job may be overly pessimistic in the case when the hyperperiod is much larger than the maximal duration of HI-criticality mode, $\text{HP} \gg \Delta^{\text{HI}}_{\text{max}}$. Still, in practical scenarios, it is not considered usual practice to design a system with a very long hyperperiod. We therefore accept the upper bound as satisfactory.

The necessary quantities $\Delta^{\text{HI}}_{\text{max}}$ and $\text{P}^{\text{HP}}_{\text{LO}\rightarrow\text{HI}}$ will be determined later as part of our analysis of the HI- and LO-criticality modes.

5.2 Probabilities of deadline misses

Let us now determine the deadline miss probabilities of $\mathsf{DMP}_{\mathrm{HI}}(T)$ and $\mathsf{DMP}_{\mathrm{LO}}(T)$, i.e., the probabilities that at least one HI criticality job or one LO criticality job misses its deadline within the time interval T. With $T = 1$ h we get the quantities as required by the schedulability test according to Definition 5. For the following theorem, let us suppose that $\chi \in \{\mathrm{LO}, \mathrm{HI}\}$ denotes the criticality of jobs in the deadline miss probabilities.

In principle, the analysis investigates two coupled systems. The first one which is denoted as the LO-system never does a mode switch, i.e., all mode switch events are ignored. In addition, it uses modified execution time probabilities of HI criticality jobs such that the LO-system pessimistically describes the behavior of the original system if operating in LO-criticality mode. In particular, all execution times of HI jobs that are higher than the threshold are trimmed to it, see Definition 2. The worst-case steady-state probability that at least one χ job misses its deadline during a hyperperiod in the LO-system is denoted as $\mathsf{DMP}^{\mathrm{LO}}_{\chi}$. This probability is determined using the worst-case steady-state backlog and response-time analysis as provided in Lemma 4, but using the trimmed execution times of HI jobs. The other system is denoted as the HI-system and considers the case that at least one LO → HI mode switch happened within a hyperperiod, i.e., at least one HI-criticality mode is executed.

Definition 11 (*Deadline Miss Probabilities in Different Modes*) The worst-case probability that at least one χ critical job misses its deadline during any HI-criticality mode started in a single hyperperiod is denoted as $\mathsf{DMP}^{\mathrm{HI}}_{\chi}$. The worst-case steady-state probability that at least one χ critical job misses its deadline during a hyperperiod in a system where (a) all mode switch events are ignored and (b) execution times of HI jobs are trimmed to their threshold according to Definition 2 is denoted as $\mathsf{DMP}^{\mathrm{LO}}_{\chi}$.

Note that $\mathsf{DMP}^{\mathrm{LO}}_{\chi}$ can be computed according to Lemma 4. Using these definitions, we can determine bounds on the requested deadline miss probabilities using the following result. The desired probabilities per hour can be obtained by setting $T = 1$ h.

Theorem 5 (Deadline Miss Probabilities) *The deadline miss probabilities* $\mathsf{DMP}_{\chi}(T)$ *for* $\chi \in \{\mathrm{LO}, \mathrm{HI}\}$ *can be bounded as follows*:

$$\mathsf{DMP}_{\chi}(T) \leq 2 - \left(1 - \mathsf{DMP}^{\mathrm{LO}}_{\chi}\right)^{\left\lceil \frac{T}{\mathrm{HP}} \right\rceil} - \left(1 - \mathsf{DMP}^{\mathrm{HI}}_{\chi}\right)^{\left\lceil \frac{T}{\mathrm{HP}} \right\rceil}. \tag{12}$$

Proof It needs to be proven that the probability that there is no deadline miss of any χ job within time interval T is bounded by

$$1 - \mathsf{DMP}_{\chi}(T) \geq \left(1 - \mathsf{DMP}^{\mathrm{LO}}_{\chi}\right)^{\left\lceil \frac{T}{\mathrm{HP}} \right\rceil} + \left(1 - \mathsf{DMP}^{\mathrm{HI}}_{\chi}\right)^{\left\lceil \frac{T}{\mathrm{HP}} \right\rceil} - 1.$$

There is no deadline miss within T if there is no deadline miss when the system executes in LO-criticality mode and there is no deadline miss if it operates in HI-criticality mode. Suppose the first event is named a and the second one b, then we know that $p(a \cap b) = p(a) + p(b) - p(a \cup b) \geq p(a) + p(b) - 1$ even if both events are not independent. Therefore, the theorem is true if

$$\left(1 - \mathsf{DMP}_{\chi}^{\mathrm{LO}}\right)^{\left\lceil \frac{T}{\mathrm{HP}} \right\rceil}$$

lower bounds the probability that there is no deadline miss when the system is in LO -criticality mode and

$$\left(1 - \mathsf{DMP}_{\chi}^{\mathrm{HI}}\right)^{\left\lceil \frac{T}{\mathrm{HP}} \right\rceil}$$

lower bounds the probability that there is no deadline miss when the system is in HI -criticality mode.

Let us first look at the LO-criticality mode. At first, note that $\lceil T/\mathrm{HP} \rceil$ is the number of hyperperiods that completely cover an interval of length T. Therefore, we can safely assume that our interval has the length of $\lceil T/\mathrm{HP} \rceil$ full hyperperiods. Remember that the backlogs during a steady-state computation are monotonically increasing, see Theorem 2. In a similar way, response times of jobs are monotonically increasing from hyperperiod to hyperperiod, see Lemma 4. As a result, the deadline miss probabilities of jobs are increasing from hyperperiod to hyperperiod as well and $\mathsf{DMP}_{\chi}^{\mathrm{LO}}$ is a safe upper bound for *every* hyperperiod in our modified LO-system. We model the system as a worst-case Bernoulli process, acting from hyperperiod to hyperperiod. As a result, $\left[1 - \mathsf{DMP}_{\chi}^{\mathrm{LO}}\right]^{\left\lceil \frac{T}{\mathrm{HP}} \right\rceil}$ is a lower bound on the probability that there is no deadline miss in the LO-system, i.e. all switching events are disabled and the execution times of HI jobs are trimmed.

It remains to be shown that the response times in our LO-system are always larger or equal than those in the original system when it is in LO-criticality mode. This is certainly true as after a HI → LO mode switch, the backlogs are 0 for sure and therefore, they are lower than those in the modified LO-system. Due to Lemma 4, the response times are larger in the modified LO-system. Moreover, trimming of execution times of HI criticality jobs has no influence on the backlogs as long as there is no HI → LO mode switch, i.e., the original system operates in LO-mode.

Now let us look at the HI-mode. Again note, that $\lceil T/\mathrm{HP} \rceil$ is the number of hyperperiods that completely cover an interval of length T. The worst-case probability that at least one χ critical job misses its deadline during any HI-criticality mode started in a single hyperperiod is denoted as $\mathsf{DMP}_{\chi}^{\mathrm{HI}}$, see Definition 11. Therefore, $\left[1 - \mathsf{DMP}_{\chi}^{\mathrm{HI}}\right]^{\left\lceil \frac{T}{\mathrm{HP}} \right\rceil}$ is a lower bound on the probability that there is no deadline miss caused by a LO → HI switch within a hyperperiod.

This concludes the proof as we considered the case that the systems operates in LO-criticality mode somewhere within a hyperperiod (bounded by the case that it is always in this mode during the hyperperiod) and the case that one or more HI

-criticality modes are started within a hyperperiod (all corresponding deadline misses are accounted for in the hyperperiod where the HI-criticality mode was started). □

Now we will determine the quantities $\Delta^{\mathrm{HI}}_{\max}$, $\mathrm{P}^{\mathrm{HP}}_{\mathrm{LO}\to\mathrm{HI}}$, $\mathsf{DMP}^{\mathrm{LO}}_{\chi}$ and $\mathsf{DMP}^{\mathrm{HI}}_{\chi}$ required to compute $\mathsf{PDJ}_{\mathrm{deg}}$, $\mathsf{DMP}_{\mathrm{HI}}(T)$ and $\mathsf{DMP}_{\mathrm{LO}}(T)$. We start by analyzing the behavior of the MC system in LO-criticality mode.

5.3 LO-criticality mode

The analysis of the LO-criticality mode will allow us to determine some of the required quantities, namely the worst case probability $\mathrm{P}^{\mathrm{HP}}_{\mathrm{LO}\to\mathrm{HI}}$ of at least one LO → HI mode switch within a hyperperiod and the worst-case probability $\mathsf{DMP}^{\mathrm{LO}}_{\chi}$ that at least one χ critical job misses its deadline within a hyperperiod if operating in the modified LO-system, see Sect. 5.2. Moreover, we will determine the worst-case probability of a LO → HI mode switch at time instance $t \in \{0, \dots, \mathrm{HP} - 1\}$ within any hyperperiod, as this quantity will allow us to analyse the χ-critical mode later on.

Lemma 5 *Given a modified task system where no LO → HI mode switch is executed and all* HI *critical jobs are trimmed to their execution time threshold C^{thr}_i, see Definition* 2. *Then,*

$$\mathsf{DMP}^{\mathrm{LO}}_{\chi} = \left[\sum_{\tau_{i,j} \in S} \overline{\mathsf{DMP}}_{i,j} \right]^1,$$

$$S = \{\tau_{i,j} \mid \chi_i = \chi \ \wedge \ 0 \le j < \mathrm{HP}/T_i\}$$

is an upper bound on the probability of at least one deadline miss of any χ job during LO*-criticality mode execution within any hyperperiod, where $\overline{\mathsf{DMP}}_{i,j}$ denotes an upper bound on the deadline miss probability of job $\tau_{i,j}$ according to Theorem* 5. *Note,* $[\dots]^1$ *indicates the expression is limited to a maximum value of* 1.

Proof We will show that the response times in the modified system are always larger or equal than those in the original system when it is in LO-criticality mode. According to Theorem 3, the upper bound on the deadline miss probability $\overline{\mathsf{DMP}}_{i,j}$ holds for any hyperperiod. On the other hand, we can not assume that the miss probabilities for the jobs are within a hyperperiod are independent. Therefore, we upper bound the probability of the union of events by their sum. It remains to be shown that the modified LO-system with all LO → HI mode switches disabled and the trimmed execution times of HI critical jobs provides upper bounds on the original system when operating in LO-criticality mode. This is certainly true as after a HI → LO mode switch in the original system, the backlogs are 0 for sure and therefore, they are lower than those in the modified LO-system. Due to Lemma 4, the response times are larger in the modified LO-system. Moreover, trimming of execution times of HI criticality jobs has no influence on the backlogs as long as there is no HI → LO mode switch, i.e., the original system operates in LO-mode.

The bounding of the value $\mathsf{DMP}^{\text{LO}}_{\chi}$ to 1 is safe, as for any summation of events we have $p(a \cup b) \leq p(a) + (b)$ and $p(a \cup b) \leq 1$ leading to $p(a \cup b) \leq \min(1, p(a) + (b))$. □

Now, we will determine an upper bound on the worst-case probability $\mathrm{P}_{\text{LO}\rightarrow\text{HI}}(t)$ of a LO → HI mode switch at time instance $t \in \{0, \dots, \text{HP} - 1\}$ within any hyperperiod. Remember that there are three triggering events for a LO → HI mode switch, namely (a) a HI critical job misses its deadline (b) the system-level backlog, meaning the amount of pending executions, becomes higher than a predefined threshold $B_{\max}$ and (c) a HI critical job overruns its threshold execution time C_{thr}. We will analyze the three different mechanisms one after the other and finally combine the results.

Let us start with the deadline miss probability at time instance $0 \leq t < \text{HP}$ which we denote as $\mathrm{P}_{dm}(t)$.

Lemma 6 *Given a modified task system where no* LO → HI *mode switch is executed and all* HI *critical jobs are trimmed to their execution time threshold* C_i^{thr}, *see Definition 2. Then,*

$$\forall 0 \leq t < \text{HP} \;\; : \; \mathrm{P}_{dm}(t) = \left[\sum_{\tau_{i,j} \in S(t)} \overline{\mathsf{DMP}}_{i,j} \right]^1$$
$$S(t) = \{\tau_{i,j} \mid \chi_i = \text{HI} \, \wedge \, d_{i,j} = t\}$$

is an upper bound on the probability of at least one deadline miss of any HI *critical job during* LO*-criticality mode execution at time* t, $0 \leq t < \text{HP}$, *where* $\overline{\mathsf{DMP}}_{i,j}$ *denotes an upper bound on the deadline miss probability of job* $\tau_{i,j}$ *in the modified task system according to Theorem 3. Note,* $[\dots]^1$ *indicates the expression is limited to a maximum value of* 1.

Proof We can not assume that the deadline miss probabilities at time t are independent. Therefore we use as an upper bound of the union of events the sum of the individual probabilities.

The bounding of the value $\mathrm{P}_{dm}(t)$ to 1 is safe, as for any summation of events we have $p(a \cup b) \leq p(a) + (b)$ and $p(a \cup b) \leq 1$ leading to $p(a \cup b) \leq \min(1, p(a) + (b))$. $S(t)$ denotes the set of all HI critical jobs with deadline at time t. □

We continue with the probability that at time instance $0 \leq t < \text{HP}$ the total backlog exceeds the upper bound $B_{\max}$ which we denote as $\mathrm{P}_{be}(t)$.

Lemma 7 *Given a modified task system where no* LO → HI *mode switch is executed and all* HI *critical jobs are trimmed to their execution time threshold* C_i^{thr}, *see Definition 2. Then,*

$$\forall 0 \le t < \textsc{hp} \;:\; \mathrm{P}_{be}(t) = \mathbb{P}(\overline{\mathcal{B}}_N(t) > B_{\max})$$

is an upper bound on the probability that the total backlog at time t exceeds $B_{\max}$ *during* LO*-criticality mode execution within any hyperperiod, where* $\overline{\mathcal{B}}_N(t)$ *denotes an upper bound on the lowest priority backlog in the modified task system according to Theorem 2.*

Proof The total backlog equals $\overline{\mathcal{B}}_N(t)$ according to Definition 7. Then, the Lemma directly follows from Theorem 2. □

Unfortunately, the computation of the probability $\mathrm{P}_{ov}(t)$ that at time instance $0 \le t < \textsc{hp}$ at least one HI critical job overruns its threshold execution time C_i^{thr} is more involved. Whereas the overrun probability $\mathbb{P}(\mathcal{C}_i > C_i^{\mathrm{thr}})$ can be simply calculated, it is more complex to understand at what time instance such an event happens, due to interference from other jobs. We will first compute the upper bound on the backlog for our modified LO-system as usual. Based on this, we now consider each HI critical job individually and compute its response time if the job would have the execution time C_i^{thr}. If this response time plus the release time $r_{i,j}$ of the job equals t, then the job overruns at t under the condition that it overruns at all. The following Lemma summarizes the corresponding result.

Lemma 8 *Given a modified task system where no* LO → HI *mode switch is executed and all* HI *critical jobs are trimmed to their execution time threshold* C_i^{thr}*, see Definition 2. Then,* $\forall 0 \le t < \textsc{hp}$

$$\mathrm{P}_{ov}(t) = \left[\sum_{\tau_{i,j} \in S} \mathbb{P}\Big((\mathtt{rta}\Big(\overline{\mathcal{B}}_i(r_{i,j}), \Pi, \tau_{i,j}^{\mathrm{ov}}\Big) + r_{i,j}) \mod \textsc{hp} = t \Big) \cdot \mathbb{P}(\mathcal{C}_i > C_i^{\mathrm{thr}}) \right]^1,$$
$$S = \{\tau_{i,j} \mid \chi_i = \textsc{hi}\}$$

is an upper bound on the probability that at time instance $0 \le t < \textsc{hp}$ *at least one* HI *critical job overruns its threshold execution time* C_i^{thr}*. Here,* $\overline{\mathcal{B}}_i(t)$ *denotes an upper bound on the level i backlog in the modified task system according to Theorem 2 and* $\tau_{i,j}^{\mathrm{ov}}$ *denotes a modified job* $\tau_{i,j}$ *with a deterministic computation time of* C_i^{thr}*. Note,* $[\ldots]^1$ *indicates the expression is limited to a maximum value of* 1.

Proof At first note that we do not assume that the probabilities of overrunning the threshold execution time C_i^{thr} are independent. Therefore, the union of at least one overrun at time t is bounded by the sum of individual probabilities for each HI job, see the definition of S. Moreover, $\mathbb{P}(a) = \mathbb{P}(a|b) \cdot \mathbb{P}(b)$ for events a and b. In our case, $\mathbb{P}(b) = \mathbb{P}(\mathcal{C}_i > C_i^{\mathrm{thr}})$, i.e., the event that task $\tau_{i,j}$ has a overrun of its threshold execution time. We now need to show that the term $\mathbb{P}(\mathtt{rta}\Big(\overline{\mathcal{B}}_i(r_{i,j}), \Pi, \tau_{i,j}^{\mathrm{ov}}\Big) + r_{i,j}) \mod \textsc{hp} = t)$ denotes the probability that an overrun due to task $\tau_{i,j}$ happens at time t under condition that the overrun happens at all, i.e., it represents $\mathbb{P}(a|b)$. Note that the term $[\mathtt{rta}\Big(\overline{\mathcal{B}}_i(r_{i,j}), \Pi, \tau_{i,j}^{\mathrm{ov}}\Big) + r_{i,j}]$ denotes the finishing time of task $\tau_{i,j}$ if using the worst-case steady-state backlogs $\overline{\mathcal{B}}$ and the execution time C_i^{thr}. Therefore, under the assumption that the task overruns, it determines the distribution of the time when the overrun actually happens. As this time may be in the next hyperperiod, we use the modulo operation.

The bounding of the value P_{ov} to 1 is safe, as for any summation of events we have $p(a \cup b) \leq p(a) + (b)$ and $p(a \cup b) \leq 1$ leading to $p(a \cup b) \leq \min(1, p(a) + (b))$. □

Based on the previous three Lemmas we can conclude this section with the desired worst-case probability $\mathrm{P}_{\mathrm{LO}\rightarrow\mathrm{HI}}(t)$ of a LO → HI mode switch at time instance $0 \leq t < \mathrm{HP}$ within any hyperperiod.

Theorem 6 $\mathrm{P}_{\mathrm{LO}\rightarrow\mathrm{HI}}(t)$ *is an upper bound on the worst-case probability of a* LO → HI *mode switch at time instance* $0 \leq t < \mathrm{HP}$ *within any hyperperiod with*

$$\forall 0 \leq t < \mathrm{HP} \ : \ \mathrm{P}_{\mathrm{LO}\rightarrow\mathrm{HI}}(t) = \left[\mathrm{P}_{dm}(t) + \mathrm{P}_{be}(t) + \mathrm{P}_{ov}(t)\right]^1,$$

where $\mathrm{P}_{dm}(t)$, $\mathrm{P}_{be}(t)$ *and* $\mathrm{P}_{ov}(t)$ *are computed according to Lemmas* 6, 7 *and* 8, *respectively. An upper bound on the probability of at least one* LO → HI *mode switch within a hyperperiod can be determined as*

$$\mathrm{P}_{\mathrm{LO}\rightarrow\mathrm{HI}}^{\mathrm{HP}} = \left[\sum_{0 \leq t < \mathrm{HP}} \mathrm{P}_{\mathrm{LO}\rightarrow\mathrm{HI}}(t)\right]^1.$$

Note, $[\ldots]^1$ *indicates the expression is limited to a maximum value of* 1.

Proof The Theorem is a simple consequence of the previous Lemmas as we can not assume independence of events within a hyperperiod.

□

As a simple corollary to the above Theorem, one can compute a lower bound on the expected length of a single LO-criticality mode execution as

$$\Delta_{exp}^{\mathrm{LO}} = \left(\left\lceil \frac{1}{\mathrm{P}_{\mathrm{LO}\rightarrow\mathrm{HI}}^{\mathrm{HP}}} \right\rceil - 1\right) \cdot \mathrm{HP}.$$

This results concludes the analysis of the LO-criticality mode and we are now analysing the HI-criticality mode in order to determine the remaining quantities as necessary for Theorems 4 and 5.

5.4 HI-criticality mode

We are still missing the computation of the maximal duration of a HI-criticality mode execution quantity $\Delta_{\max}^{\mathrm{HI}}$, as well as the worst-case probability $\mathsf{DMP}_{\chi}^{\mathrm{HI}}$ of at least one deadline miss of any χ job during any HI-criticality mode started within a hyperperiod, where $\chi \in \{\mathrm{LO}, \mathrm{HI}\}$.

To this end, we will determine HP different worst-case HI-criticality mode scenarios, one for each starting time $0 \leq t < \text{HP}$ relative to the beginning of a hyperperiod. In other words, we will investigate HP different HI-criticality mode executions and then use the maximum of their durations as $\Delta^{\text{HI}}_{\text{max}}$, and the maximum of their deadline miss probabilities to determine upper bounds that at least one HI or LO task misses its deadline during a single HI-criticality mode execution. These quantities will then be combined with the probability $\text{P}_{\text{LO}\rightarrow\text{HI}}(t)$ that a LO → HI switch happens at relative starting time t in order to determine $\mathsf{DMP}^{\text{HI}}_{\chi}$, i.e., the worst-case probability of at least one deadline miss of any χ critical job during any HI-criticality mode started within a hyperperiod.

Broadly speaking, HI-criticality mode has three differences with LO-criticality mode. First, jobs released in HI-mode have different execution times: LO jobs are released in degraded mode, and HI jobs do not have the condition that they do not overrun their C^{thr}_{i} execution time threshold. Second, 'carry-over' jobs, which are released in LO-criticality mode but whose deadlines are after the mode switch, are present in HI-criticality mode and they need to be accounted for. Third, the initial system-level backlog is not zero, but depends on the mode switch time trigger. To account for these differences, we present the following worst-case HI-critical execution task-set. It is created such that it is pessimistic what ever the mode switch trigger may be, and it accounts for both carry-over jobs and jobs released during HI-mode.

The worst-case HI-mode scenario for starting at time t will be defined as follows:

Definition 12 (*Worst-Case HI-Criticality Execution*) We define HP task set $\widehat{\Pi}(t)$, one for each starting time $0 \leq t < \text{HP}$. It differs from the original task set Π as follows:

1. The phase offsets ϕ_i are implicitly changed such that all jobs are already available in $0 \leq t < \text{HP}$, i.e., we allow for negative job indices j.
2. We consider all jobs with starting times after t, i.e., $j \geq (t - \phi_i)/T_i + 1$. They have a known execution time $\widehat{\mathcal{C}}_i$ which is not larger than the degraded mode WCET C^{deg}_{i} for LO criticality jobs, and a known execution time $\widehat{\mathcal{C}}_i = \mathcal{C}_i$ for HI criticality jobs.
3. We consider jobs whose release time is smaller than t and deadline is larger than t. These included jobs $\tau_{i,j} \in \widehat{T}$ with $(t - \phi_i)/T_i + 1 < j < (t + D_i - \phi_i)/T_i + 1$ have execution times $\widehat{\mathcal{C}}_i = \mathcal{C}_i$ for both LO and HI criticality jobs; i.e. for LO jobs the execution times are not degraded, and for HI jobs they may or may not overrun their C^{thr}_{i} threshold.
4. In addition, for each HI-criticality mode starting time t, $0 \leq t < \text{HP}$, we introduce the initial backlog at time t and priority levels $1 \leq i \leq N$, $\widehat{\mathcal{B}}_i(t)$. If a overrun can not happen at time t, due to the fact there is no HI job released whose deadline has passed by time t, the initial backlog is as follows:

$$\mathbb{P}(\widehat{\mathcal{B}}_i(t) = u) = \begin{cases} \mathbb{P}(\overline{\mathcal{B}}_i(t) = u) & u < B_{\max} \\ \sum_{v=B_{\max}}^{\infty} \mathbb{P}(\overline{\mathcal{B}}_i(t) = v) & u = B_{\max} \\ 0 & u > B_{\max} \end{cases}$$

where $\overline{\mathcal{B}}_i(t)$ denotes an upper bound on the ith priority backlog in the modified LO-criticality system according to Theorem 2. If an overrun can happen at time t, due to at least one HI job having its release time before t and its deadline after, then the initial backlog at time t is the following:

$$\mathbb{P}(\widehat{\mathcal{B}}_i(t) = u) = \begin{cases} \mathbb{P}(\overline{\mathcal{B}}_i^{\mathsf{ov}}(t) = u) & u < B_{\max} \\ \sum_{v=B_{\max}}^{\infty} \mathbb{P}(\overline{\mathcal{B}}_i^{\mathsf{ov}}(t) = v) & u = B_{\max} \\ 0 & u > B_{\max} \end{cases}$$

where $\overline{\mathcal{B}}_i^{\mathsf{ov}}(t)$ denotes an upper bound on the ith priority backlog in the modified LO-criticality system according to Theorem 2, but with the added condition that at least one of the released HI jobs whose deadline is after time t has overrun its threshold execution time C_i^{thr}.

Let us now describe how $\overline{\mathcal{B}}_i^{\mathsf{ov}}(t)$ can be computed.

To this end, we solve

$$\mathbb{P}(\overline{\mathcal{B}}_i^{\mathsf{no+ov}}(t) = u) = \mathbb{P}(\mathsf{no}) \cdot \mathbb{P}(\overline{\mathcal{B}}_i(t) = u) + \mathbb{P}(\mathsf{ov}) \cdot \mathbb{P}(\overline{\mathcal{B}}_i^{\mathsf{ov}}(t) = u).$$

Here, $\overline{\mathcal{B}}_i(t)$ denotes an upper bound on the ith priority backlog in the modified LO-criticality system according to Theorem 2. $\overline{\mathcal{B}}_i^{\mathsf{no+ov}}(t)$ is also an upper bound on the ith priority backlog according to Theorem 2, but the system used for its computation is slightly modified. It is the LO-criticality system with the difference that HI jobs released before time t whose deadlines are after that time have no condition on whether they overrun their C_i^{thr} execution time or not—we use their normal execution times $\mathcal{C}_i$ in calculating the backlog. The probability that none of these HI jobs overrun their respective C_i^{thr} execution times is noted $\mathbb{P}(\mathsf{no})$, while the $\mathbb{P}(\mathsf{ov}) = 1 - \mathbb{P}(\mathsf{no})$ is the probability that at least one of these HI jobs overruns. $\mathbb{P}(\mathsf{no})$ is obtained directly from execution times of these HI jobs, $\mathbb{P}(\mathsf{no}) = \sum_{\tau_{i,j} \in S} \mathbb{P}\big(\mathcal{C}_i > C_i^{\mathrm{thr}}\big)$, where $S = \big\{\tau_{i,j} \mid \chi_i = \mathrm{HI} \land r_{i,j} \le t \land d_{i,j} > t\big\}$.

Condition 2 includes all tasks which are released during HI-criticality mode, noting that LO jobs are degraded and HI jobs have $\mathcal{C}_i$ execution times. The third condition deals with carry-over jobs from LO- to HI-criticality mode, whose deadline misses have not yet been accounted for in the LO-criticality mode analysis. Note that here the worst case comes from the assumption that all HI jobs may overrun. Finally, condition 4 includes the worst-case backlog at the starting time t, as it is the backlog with the condition that an overrun of at least one HI job occurred, but also it is limited by the maximal backlog $B_{\max}$. Simpler constructions of the worst-case task-set lead to high overestimations to the length and deadline miss probabilities of HI-criticality mode.

Starting from the worst-case scenarios for the HI-mode for each time instant t, $0 \leq t < \text{HP}$, we now evaluate each scenario and determine the corresponding worst-case durations as well as the deadline miss probabilities. To do this, we apply the results from Sect. 4 and use the function $\texttt{bsse}\left(\widehat{\mathcal{B}}_i(t),\ \widehat{\Pi},\ i,\ t,\ u\right)$ to compute all relevant backlogs for the task sets from Definition 12. The successive computation of the backlogs stops whenever the system gets idle for the first time: $\widehat{\mathcal{B}}_i(t_s) = 0$ for all priority levels i. This time is an upper bound on the HI → LO switching time. Using the response time analysis, see (10), we can finally determine all jobs that miss their deadline during the HI-mode. Additionally, for the response time analysis for calculating the deadline miss probabilities of HI carry-over jobs, we substitute the execution time of the carry-over job under analysis $\widehat{\mathcal{C}}_i$ with the conditional execution time $\mathbb{P}(\mathcal{C}_i > C_i^{\text{thr}})$, in order to get the deadline miss probability with the condition that the HI carry-over job overran its C_i^{thr} execution time threshold.

Lemma 9 *The first time t_{idle}, the execution of the task set $\widehat{\Pi}(t)$ from Definition 12 yields a system-level backlog which is zero, determines an upper bound $\Delta^{\text{HI}}_{\max}(t)$ on the duration of a HI-criticality mode starting at time t relative to the beginning of any hyperperiod of the original task system Π:*

$$\forall 0 \leq t < \text{HP} \ \ : \ \ \Delta^{\text{HI}}_{\max}(t) = t_{\text{idle}} - t.$$

Let us define the probability $p_{i,j}(t)$ that some job $\tau_{i,j}$ of task set $\widehat{\Pi}(t)$ from Definition 12 misses its deadline in the time interval $[t, t + \Delta^{\text{HI}}_{\max}(t)]$. Then $\mathsf{DMP}^{\text{HI}}_{\chi}(t)$ is an upper bound on the probability that there is at least one deadline miss of any χ critical job with $\chi \in \{\text{LO}, \text{HI}\}$ within a HI-criticality mode execution starting at time t relative to the beginning of any hyperperiod in the original task system Π:

$$\forall 0 \leq t < \text{HP} \ \ : \ \ \mathsf{DMP}^{\text{HI}}_{\chi}(t) = \left[\sum_{\tau_{i,j} \in S(t)} p_{i,j}(t) \right]^1$$
$$S(t) = \{\tau_{i,j} \in \widehat{\Pi}_i(t) \mid \chi_i = \chi\}.$$

Note, $[\ldots]^1$ *indicates the expression is limited to a maximum value of* 1.

Proof The main part of the proof is to show that the task set $\widehat{\Pi}(t)$ indeed defines a worst-case scenario in terms of duration and deadline miss probabilities, when the HI-criticality mode starts at time t relative to the beginning of any hyperperiod. Note that the second condition in Definition 12 ensures that all tasks which are released during a HI-criticality mode in the worst case, are included in the HI-criticality task set as well. Moreover, we consider the exact execution times for all of these jobs, namely the degraded execution times $\widehat{\mathcal{C}}_i$ which are not longer than C_i^{deg} for LO criticality jobs, and $\widehat{\mathcal{C}}_i = \mathcal{C}_i$ for HI criticality jobs. The third condition adds the worst-case carry-over jobs from LO- to HI-criticality mode whose deadline misses have not yet been accounted for in the LO-mode analysis. All jobs who missed their deadline before the LO → HI mode switch have been considered already in the LO-mode analysis, but their possible backlog at t will be considered. Therefore, we just need to explicitly include jobs whose release time is before and whose deadline is after the LO → HI mode switch. The corresponding execution times are taken as worst-case as well, namely for each carry-over HI job individually, for calculating its deadline miss probability we assume it overruns its execution time threshold. Finally, we look at the worst-case backlog at the starting time t. It encompasses the remaining execution times of jobs who were released before t but not yet finished. Due to the triggering condition of a mode switch, we assume the worst-case that at least one HI job has overrun its C_i^{thr} execution time. Also according to triggering conditions, the backlog is never larger then $B_{\max}$ for all priority levels. Note that the backlog also contains jobs whose deadline is within the HI-mode, i.e., the carry-over jobs who have been explicitly included as tasks.

In order to determine the upper bound on the deadline miss probability $\mathsf{DMP}_{\chi}^{\text{HI}}(t)$ of any χ-critical job we again do not assume independence of individual miss events and use the sum of the corresponding probabilities as an upper bound. □

As a result of this Lemma we can determine the desired quantities, namely maximal duration and upper bound on deadline misses, for each time point t relative to the starting of a hyperperiod. The computations are based on simple simulations of HP executions of worst-case HI-criticality mode scenarios. The simulation times are finite as long as there exists a finite time in $\widehat{\Pi}(t)$ when the system gets the first time idle. The following Lemma leads to a necessary and sufficient condition.

Lemma 10 *A set of finite bounds $\Delta_{\max}^{\text{HI}}(t)$ on the duration of* HI*-criticality modes exists if and only if the maximal system utilization in HI-criticality mode in the original system is less than one.*

Proof Let us look at the modified task set $\widehat{\Pi}(t)$ starting at time t. If the maximal system utilization in HI-criticality mode is less than one, then the maximal system level backlog at time $t + (n + 1) \cdot \text{HP}$ is strictly smaller than the maximal system level backlog at time $t + n \cdot \text{HP}$ for $n > 1$, because the arriving jobs in time interval $[t + n \cdot \text{HP}, t + (n + 1) \cdot \text{HP})$ are identical for all $n > 1$ and there is less additional accumulated computation time from all arriving jobs than its length HP. Therefore, a time instance will exist when the maximal system level backlog is zero and the system is idle. If the maximal system utilization in HI-criticality mode is larger or equal than one, then the maximal system level backlog at time $t + (n + 1) \cdot \text{HP}$ could be equal or greater than the maximal system level backlog at time $t + n \cdot \text{HP}$. Therefore, in the worst case, the system level backlog never gets to zero and the HI-criticality mode could last for ever. □

Based on these results, we can now aggregate the computed quantities in order to determine the maximal duration of a HI-criticality mode execution quantities $\Delta_{\max}^{\text{HI}}$ as well as the worst-case probability $\mathsf{DMP}_{\chi}^{\text{HI}}$ of at least one deadline miss of any χ job during any HI-criticality mode started within a hyperperiod, where $\chi \in \{\text{LO}, \text{HI}\}$.

Table 2 Scheduling schemes used throughout Sect. 6

Deterministic schemes		
DMPO	Deadline Monotonic Priority Ordering	Audsley et al. (1991)
AMC	Adaptive Mixed Criticality	Baruah et al. (2011)
UB-HL	Upper Bound on Fixed Priority Preemptive MC Schemes	Baruah et al. (2011)
Probabilistic schemes		
pDMPO	Probabilistic Deadline Monotonic Priority Ordering	Díaz et al. (2002)
pMC	Probabilistic Mixed Criticality	*this work*

Theorem 7 $\Delta^{\mathrm{HI}}_{\max}$ *is an upper bound on the maximal duration of any* HI*-criticality mode in the original task system* Π, *where*

$$\Delta^{\mathrm{HI}}_{\max} = \max_{0 \le t < \mathrm{HP}} \Delta^{\mathrm{HI}}_{\max}(t).$$

$\mathsf{DMP}^{\mathrm{HI}}_{\chi}$ *is a bound on the worst-case probability of at least one deadline miss of any* χ *critical job with* $\chi \in \{\mathrm{LO}, \mathrm{HI}\}$ *during any* HI*-criticality mode started within a hyperperiod in the original task system*, *where*

$$\mathsf{DMP}^{\mathrm{HI}}_{\chi} = \left[\sum_{0 \le t < \mathrm{HP}} \mathrm{P}_{\mathrm{LO}\to\mathrm{HI}}(t) \cdot \mathsf{DMP}^{\mathrm{HI}}_{\chi}(t) \right]^{1}$$

with $\mathrm{P}_{\mathrm{LO}\to\mathrm{HI}}(t)$ *as determined in Theorem* 6. *Note*, $[\ldots]^1$ *indicates the expression is limited to a maximum value of* 1.

Proof According to Lemma 9, $\Delta^{\mathrm{HI}}_{\max}(t)$ is an upper bound on the duration of a HI -criticality mode starting at relative time t within a hyperperiod. Clearly, the maximum for all relative time instances provides the maximal duration for any time instance. The probability of a deadline miss within a HI-mode execution is the probability of the union of deadline misses at any time instance within the hyperperiod. As we cannot assume independence, we upper bound this probability by the sum of individual probabilities. The probability of a deadline miss within a HI-mode starting at relative time t is clearly the probability that a a mode switch happens, i.e., $\mathrm{P}_{\mathrm{LO}\to\mathrm{HI}}(t)$, times the probability that a deadline miss happens within the HI-mode, i.e., $\mathsf{DMP}^{\mathrm{HI}}_{\chi}(t)$.

□

This concludes the schedulability analysis of probabilistic Mixed-Criticality Systems according to Definition 5, as all required quantities for Theorems 4 and 5 have been determined in Sects. 5.3 and 5.4 .

Of course, the tightness of the analysis can be improved through various approaches. Some of them as well as limitations of the described analysis are noted in the conclusion.

6 Experimental results

In order to illustrate our probabilistic Mixed Criticality (pMC) schedulability analysis, this section first shows one sample task-set. The sample task-set is inspired by applications from the avionics industry. Then, experiments on randomly generated task-sets are used to compare pMC scheduling with other schemes: a probabilistic but non-Mixed Criticality scheme 'Probabilistic Deadline Monotonic Priority Ordering' pDMPO, the deterministic 'Adaptive Mixed Criticality' scheme (AMC), and a deterministic non-MC 'Deadline Monotonic Priority Ordering' scheme. These are all listed in Table 2, and described in detail below. For the experiments, we generated randomized task-sets with all but one parameter the same, in order to see the effect this one parameter has.

Three experiments are conducted. The first experiment serves to show the impact of the system utilization, the second experiment varies the probability each HI task overruns its C_i^{thr} execution time threshold $\mathbb{P}(\mathcal{C}_i > C_i^{\mathrm{thr}})$, and finally the impact of the maximal system-level backlog is visualized in the third experiment. In general, we show that pMC dominates all other schemes, except in situations when HI-criticality mode is entered too often. In these cases, we find that there is too much degradation of LO jobs, therefore scheduling using the probabilistic but non-Mixed Criticality pDMPO yields better results.

Baseline schemes To evaluate pMC scheduling, we have used three deterministic and one probabilistic baseline scheme, as listed in Table 2. All schemes are based on fixed-priority preemptive scheduling. The first deterministic scheme is a non-Mixed Criticality one, Deadline Monotonic Priority Ordering (DMPO). As the name suggests, tasks are prioritized only by their deadlines, and scheduled according to their C_i^{max} WCETs.

The next scheme is Adaptive Mixed Criticality (AMC), as described by Baruah et al. (2011). The scheme features two modes of operation. The system starts in LO-criticality mode where HI tasks are scheduled according to their C_i^{thr} threshold execution times. If any HI job overruns this value, a switch to HI-criticality mode happens, where all LO tasks are released in degraded mode. The scheme does not quantify the duration of these two modes, only the schedulability of them.

As a deterministic baseline scheme we introduce the UB-HL bound (Baruah et al. 2011). The bound is a necessary test for all fixed priority preemptive MC schemes, and such it provides an upper bound on the performance of all fixed priority preemptive deterministic MC schemes.

Finally, the Probabilistic Deadline Monotonic Priority Ordering (pDMPO) scheme represents the analysis as introduced by Díaz et al. (2002). In pDMPO, tasks are given priorities based on their deadlines, they are scheduled using their complete $\mathcal{C}_i$ execution times, and there is only one mode of operation. The scheme can be viewed as a border case of pMC, where HI-criticality mode is never entered.

Task Execution Times To model task execution times $\mathcal{C}_i$, Weibull distributions were used, with a condition that they do not take values greater than the task's WCET C_i^{max}. These distributions have been used in related work for modeling the distribution of long but unlikely execution times (Cucu-Grosjean et al. 2012).

Table 3 The sample system's parameters

Task τ_i	χ_i	Pr.	Period T_i	$C_i^{\max} \cdot f_c^{-1}$	$C_i^{\text{thr}} \cdot f_c^{-1}$	$C_i^{\text{deg}} \cdot f_c^{-1}$
`sens_c1`	HI	5	7.5 ms	648 µs	259 µs	–
`loc_c1`	HI	4	7.5 ms	365 µs	146 µs	–
`loc_c2`	HI	3	60 ms	13 µs	5 µs	–
`loc_c3`	HI	2	60 ms	73 µs	24 µs	–
`loc_c4`	HI	1	60 ms	13 µs	5 µs	–
`engine`	LO	16	750 µs	23 µs	–	8 µs
`elevator`	LO	15	750 µs	22 µs	–	8 µs
`aircraft_dynamics`	LO	14	750 µs	161 µs	–	54 µs
`h_filter`	LO	13	1.5 ms	11 µs	–	4 µs
`az_filter`	LO	12	1.5 ms	12 µs	–	4 µs
`Vz_filter`	LO	11	1.5 ms	12 µs	–	4 µs
`q_filter`	LO	10	1.5 ms	11 µs	–	4 µs
`Va_filter`	LO	9	1.5 ms	12 µs	–	4 µs
`altitude_hold`	LO	8	3 ms	6 µs	–	2 µs
`Vz_control`	LO	7	3 ms	6 µs	–	2 µs
`Va_control`	LO	6	3 ms	6 µs	–	2 µs

Note that values relating to execution times are a function of parameter f_c

Weibull distributions are functions of parameters k and λ. To generate an execution time, we first choose k uniformly from [1.5, 3]. Then, the parameter λ is computed the following way. For LO tasks, λ was computed such that the cumulative density function at the task's WCET $C_i^{\max}$ is $1 - 10^{-8}$. Similarly, for HI tasks, we choose λ so the cumulative density function at the task's execution time threshold C_i^{thr} is $1 - 10^{-8}$, unless stated otherwise. This is the way we set the probability a HI task overruns its threshold execution time. Finally, all values of the probability density function above $C_i^{\max}$ are set to be 0, and the rest of the distribution is normalized. This way, we have a valid execution time modeled by a Weibull distribution, with the condition it never exceeds the task's WCET $C_i^{\max}$, and for which the probability a HI task overruns its execution time threshold is C_i^{thr}.

6.1 Sample system

Here we introduce a task-set modelling a sample system, to which we applied our proposed schedulability analysis. We explored the task-set, first by varying execution times of all tasks, and then by varying deadlines. This was done to illustrate probabilistic Mixed Criticality scheduling. We present the three schedulability values: $\mathsf{DMP}_{\text{HI}}(1h)$, $\mathsf{DMP}_{\text{LO}}(1h)$, and $\mathsf{PDJ}_{\text{deg}}$, and we also show the expected duration of LO-criticality mode Δ_{exp}^{LO}, and the maximal duration of HI-criticality mode $\Delta_{\max}^{\text{HI}}$.

The system's LO and HI tasks are inspired by the ROSACE (Pagetti et al. 2014) and FMS (Durrieu et al. 2014) applications, respectively. The HI tasks are inspired by an industrial implementation of the flight management system (FMS). This application

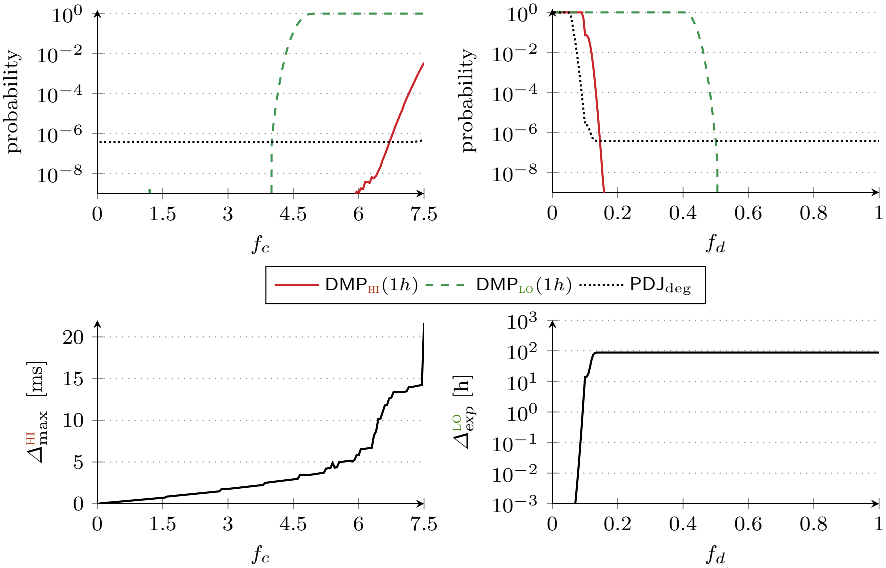


Fig. 2 Metrics characterizing the sample task-set. Left: with fixed deadlines $f_d = 1$ but various utilization. Right: with fixed utilization $f_c = 2$ but scaling all deadlines

consists of one task which reads sensor data, and four tasks that compute the location of the aircraft. For LO tasks, the open source avionic benchmark ROSACE was modeled. It is made up of three tasks which simulate pilot's instructions, and eight tasks implementing a controller.

Setup Table 3 lists the tasks' periods and execution time values: worst-case execution times (WCETs) $C_i^{\max}$, thresholds for HI tasks C_i^{thr}, and degraded WCETs C_i^{deg} for LO tasks. Execution time values are functions of the parameter f_c, which we vary from 0.05 to 7.5 in 0.05 steps. Note that for HI tasks, $C_i^{\max}$ values are 2.5 times larger than the corresponding C_i^{thr}, while for LO tasks the worst-case execution time in degraded mode is $C_i^{\text{deg}} = 0.33 \cdot C_i^{\max}$, rounded up to the nearest integer. The deadline of each task has been constrained by a factor of f_d, $D_i = T_i \cdot f_d$, where f_d is varied from 0.005 to 1 in steps of 0.005. Next, initial phases for tasks are 0, while tasks' priority assignments are given in the table. Note that we use deadline monotonic priority assignment.

We model probabilistic execution times of tasks with Weibull distributions, as described in the beginning of this section. The probability that a HI task executes for longer than its threshold execution time C_i^{thr} is $\mathbb{P}(\mathcal{C}_i > C_i^{\text{thr}}) = 10^{-8}$, for every HI task. For the maximal system-level backlog, we used $B_{\max} = 5$ms. The hyperperiod lasts for 60ms, and inside one there are 500 LO jobs and 19 HI jobs. Regardless of the parameter f_c, the utilization of LO tasks is 5.73 times higher than the utilization of HI tasks.

In Fig. 2, the two left plots have results when deadlines are fixed ($f_d = 1$), but execution times values from Table 3 are varied with $f_c \in (0, 7.5]$. In the two right plots of Fig. 2, shown are results when deadlines are varied $f_d \in (0, 1]$, but all execution time values are fixed ($f_c = 2$).

Results As expected, the deadline miss probability per hour for both HI and LO jobs, $\mathsf{DMP}_{\mathrm{HI}}(1h)$ and $\mathsf{DMP}_{\mathrm{LO}}(1h)$, increases as the utilization increases, or as the deadlines become more constrained. In this example, $\mathsf{DMP}_{\mathrm{LO}}(1h)$ is larger than $\mathsf{DMP}_{\mathrm{HI}}(1h)$, even though HI criticality tasks have the lowest priority. This is mainly because there are more LO than HI jobs, i.e. 500 versus 19 jobs per hyperperiod. As for the probability that a LO job is released in degraded mode, $\mathsf{PDJ}_{\mathrm{deg}}$, we notice it follows a similar trend. In this experiment, the value never goes to zero, because there is always a non-zero probability a LO → HI criticality mode switch occurs.

In the bottom right plot of Fig. 2, the expected duration of LO-mode is shown to resemble the inverse of $\mathsf{PDJ}_{\mathrm{deg}}$. Except when the deadlines are very constrained ($f_d < 0.12$), LO-criticality mode lasts for an expected $\Delta^{\mathrm{LO}}_{exp} = 88\mathrm{h}$ before a trigger event occurs. The maximal duration of HI-criticality mode $\Delta^{\mathrm{HI}}_{\max}$ depends only on the system utilization. This is shown in the bottom left plot as a function of f_c. The value is 1.1ms for $f_c = 2$, and 21.7ms for $f_c = 7.5$. Both values are smaller than $\Delta^{\mathrm{LO}}_{exp}$ by orders of magnitude.

6.2 Randomized systems

Now we continue, and present three further experiments. They demonstrate the impact of three design parameters on schedulability: the system utilization, the probability that a HI tasks overruns its execution time threshold C_i^{thr}, and the choice of the maximal system-level backlog.

More specifically, the first experiment shows whether task-sets of different system utilizations are $(\sigma_{\mathrm{HI}}, \sigma_{\mathrm{LO}}, \sigma_{\mathrm{deg}})$-schedulable using probabilistic Mixed Criticality (pMC) scheduling, as well as other scheduling schemes.

The second and third experiments compare pMC with the probabilistic but non-MC scheme pDMPO. They demonstrate that pMC leads to improved schedulability, except when HI-criticality mode is entered too often, either because of the first or the third mode switch trigger, respectively.

For all three experiments, tasks were randomly generated as described below.

Task-Set Generation For each of the three experiments presented, the UUnifast-Discard algorithm (Davis and Burns 2011) was used to randomly generate task-sets, with the following parameters we found reasonable.

- First, periods and maximal execution times in LO-criticality mode (C_i^{thr} values for HI tasks and $C_i^{\max}$ for LO tasks) were generated by the UUnifast algorithm. Periods were chosen between $\{50\,\mu s, 100\,\mu s, 200\,\mu s, 250\,\mu s, 500\,\mu s, 1000\,\mu s\}$.
- All initial phases were set to 0, and tasks' deadlines are equal to their period.
- Then, every task's criticality is assigned to be HI with a probability of 0.5 (i.e. parameter $CP = 0.5$).

- For every HI task, the worst case execution time (WCET) $C_i^{\max}$ is a fixed multiplier of the corresponding threshold C_i^{thr}, $C_i^{\max} = 1.5 \cdot C_i^{\text{thr}}$ (i.e. parameter $CF = 1.5$). For LO tasks, their degraded WCET is set to be a third of their actual WCET, $C_i^{\deg} = 0.33 \cdot C_i^{\max}$.
- To model task execution times $\mathcal{C}_i$, we have used Weibull distributions as explained at the beginning of this Section. The probability each HI job $\tau_{i,j}$ overruns its execution time threshold is $\mathbb{P}(\mathcal{C}_i > C_i^{\text{thr}}) = 10^{-8}$, unless stated otherwise.
- The number of tasks per task-set is 60.
- Finally, the maximum backlog $B_{\max}$ is 500 µs, unless stated otherwise.

For the system utilization and other details, we refer the reader to the setup section of each experiment.

Priority Assignment For the probabilistic scheduling schemes pMC and pDMPO, we have used deadline monotonic priority assignment. Note that (Maxim et al. 2011) shows that this assignment is in general not optimal for probabilistic systems, they suggest instead Audsley's priority assignment algorithm. For the deterministic scheduling schemes, AMC uses Audsley's priority assignment which is optimal for this scheme, while DMPO by definition uses deadline monotonic priorities.

6.2.1 'Utilization' experiment

In this first experiment, we examine the schedulability of systems with various system utilizations. More precisely, we check whether randomly generated systems of utilization 0.1 through 2.0 are $(\sigma_{\text{HI}}, \sigma_{\text{LO}}, \sigma_{\deg}) = (10^{-8}, 10^{-6}, 10^{-5})$-schedulable under probabilistic Mixed Criticality (pMC) scheduling, under a probabilistic but non-MC scheme (pDMPO), as well as under deterministic baseline schemes: DMPO, AMC, and UB-HL. We also examine the values relevant to pMC scheduling as functions of maximum system utilization: the probability of deadline miss per hour for HI or LO jobs $\mathsf{DMP}_{\text{HI}}(1h)$ and $\mathsf{DMP}_{\text{LO}}(1h)$, and the probability of degraded LO jobs $\mathsf{PDJ}_{\deg}$.

Setup We ranged the system utilization from 0.1 to 2.0 with 0.1 steps, and for each step we created 1000 task-sets according to the previously given description. To reiterate, the following parameters were used: the ratio between the WCET $C_i^{\max}$ and execution time threshold C_i^{thr} for every HI task is $CF = C_i^{\max}/C_i^{\text{thr}} = 1.5$, the probability each task is assigned HI criticality is $CP = 0.5$, the probability a HI job overruns its execution time threshold $\mathbb{P}(\mathcal{C}_i > C_i^{\text{thr}}) = 10^{-8}$, the degradation of LO tasks is $C_i^{\deg} = 0.33 \cdot C_i^{\max}$, there are 60 tasks in each task-set, and the maximal system-level backlog is $B_{\max} = 500$ µs.

Tasks' execution times $\mathcal{C}_i$ depend on the utilization and task-set in question. We found the mean of the execution times to be between 2.84 and 16.38 µs, with the maximal execution time $C_i^{\max}$ among all tasks in a task-set being between 21 and 387 µs.

Results Figure 3 presents the most important result of our experiments. For task-sets of different system utilizations, the $(10^{-8}, 10^{-6}, 10^{-5})$-schedulability under various scheduling schemes is given in Fig. 3 Top. To understand better how utilization

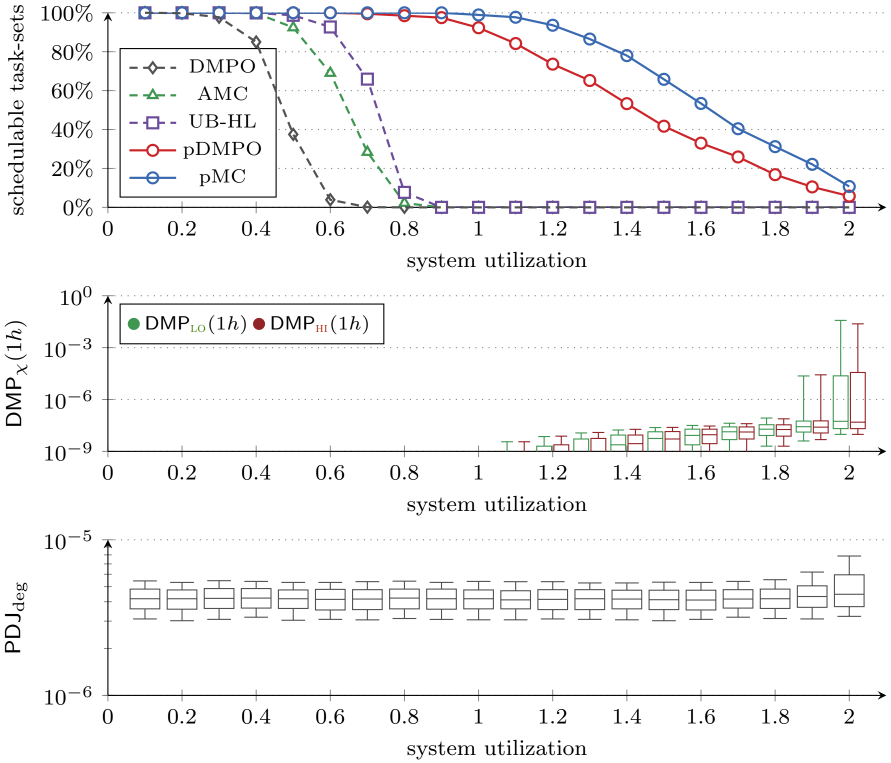


Fig. 3 The $(10^{-8}, 10^{-6}, 10^{-5})$-schedulability of task-sets, as a function of utilization under pMC and other schemes (Top), and the impact utilization has on $\mathsf{DMP}_{\mathrm{LO}}(1h)$, $\mathsf{DMP}_{\mathrm{HI}}(1h)$ (Middle) and $\mathsf{PDJ}_{\mathrm{deg}}$ (Bottom)

impacts pMC schedulability, Figure 3 Middle and Bottom show statistics on the $\mathsf{DMP}_{\mathrm{HI}}(1h)$, $\mathsf{DMP}_{\mathrm{LO}}(1h)$ and $\mathsf{PDJ}_{\mathrm{deg}}$ metrics. The box-plots visualize the 10th, 25th, 50th, 75th, and 90th percentile of each metric.

Regarding the three deterministic schemes, we see that they perform similarly as in related work, for example (Baruah et al. 2011). Remember that for deterministic schemes, a task-set is either ‘completely’ schedulable or it is not, as there is no notion of probabilities.

In Fig. 3 Top, we can see that deadline monotonic priority ordering (DMPO) has the lowest schedulability among all tested schemes. This is because DMPO attempts to schedule a task-set using only WCET ($C_i^{\max}$) values. The adaptive Mixed Criticality (AMC) scheme performs better, as it performs a LO → HI mode switch every time HI jobs need more execution time. Still, the schedulability of deterministic fixed priority preemptive schemes is upper-bounded by the UB-HL bound.

For the probabilistic schemes pDMPO and pMC, we can confirm that they outperform deterministic schemes. Probabilistic schemes allow a system with a utilization greater than one to be schedulable, because they take into account the low probability that a long execution time is observed. Let us first focus on probabilistic deadline monotonic priority ordering (pDMPO). We understand from Díaz et al. (2002) that deadline misses under pDMPO happen when the backlog is large, i.e. when one or more jobs take a long time to execute. The bigger the utilization is, the likelier it is that the backlog is large.

As for probabilistic Mixed Criticality (pMC), it features three LO → HI mode switch triggers. All three triggers are indicators that the backlog is large: the first trigger activates when a HI job executes for a long time, the second trigger indicates that a HI job missed its deadline due to a large backlog blocking its execution, and finally the third trigger explicitly notes that the system-level backlog is too large. After detecting these high-backlog situations, the system under pMC transitions to HI-criticality mode where LO jobs are degraded, and thus the backlog is decreased. This ensures that deadline miss probabilities of both LO and HI tasks are reduced, at the cost of having some LO jobs released in degraded mode. Most importantly, this is demonstrated in Fig. 3 Top, where pMC outperforms pDMPO as well as all other schemes. Furthermore, in Fig. 3 Middle, we see how both $\mathsf{DMP}_{\mathrm{HI}}(1h)$ and $\mathsf{DMP}_{\mathrm{LO}}(1h)$ increase gradually with the increase of utilization. The small difference between $\mathsf{DMP}_{\mathrm{HI}}(1h)$ and $\mathsf{DMP}_{\mathrm{LO}}(1h)$ comes from the fact that the system switches to HI-criticality mode whenever a HI jobs overruns its C_i^{thr} threshold, which helps HI jobs keep their deadline. Finally, Fig. 3 Bottom shows the probability a LO job is released with degradation. This slight increase is a sign of being in HI-criticality mode more often, and this quantifies the cost of probabilistic Mixed Criticality scheduling.

6.2.2 'Execution Threshold' experiment

In this experiment, we varied a design parameter relating to tasks' execution times $\mathcal{C}_i$: the probability that a HI job overruns its execution time threshold C_i^{thr}. We then inspected how this impacts schedulability under probabilistic Mixed Criticality (pMC) and the probabilistic non-Mixed Criticality pDMPO scheme. Because we used a utilization of 1.4, deterministic schemes could not schedule any task-sets. The probability each HI job $\tau_{i,j}$ overruns its execution time threshold $\mathbb{P}(\mathcal{C}_i > C_i^{\mathrm{thr}})$ is ranged from $5 \cdot 10^{-12}$ to 10^{-4}. Ultimately, this experiment demonstrates that it makes sense to use probabilistic Mixed Criticality scheduling if HI-criticality mode is not entered too often, and the importance of the $\mathsf{PDJ}_{\mathrm{deg}}$ metric is justified.

Setup A total of 16 configurations, each with 1000 task-sets, were generated for this experiment. The configurations have the same parameters, except for the probability each HI job $\tau_{i,j}$ overruns its execution time threshold $\mathbb{P}(\mathcal{C}_i > C_i^{\mathrm{thr}})$. The following values for $\mathbb{P}(\mathcal{C}_i > C_i^{\mathrm{thr}})$ were used: $\{5 \cdot 10^{-12}, 10^{-11}, 5 \cdot 10^{-11}, ..., 10^{-4}\}$. Besides this, the system utilization for all configurations is 1.4, while all other parameters are according to the description mentioned at the beginning of Sect. 6.2.

Regardless of the fact that $\mathbb{P}(\mathcal{C}_i > C_i^{\mathrm{thr}})$ is varied by 8 orders of magnitude, we found that the mean execution time per configuration changes little. It is between 8.69 and 8.70 µs. Among all tasks in every task-set, the worst case execution time $C_i^{\max}$ is 287 µs.

Results The results of this experiment are shown in Fig. 4. In the top figure, we present $(10^{-8}, 10^{-6}, 10^{-5})$- and $(10^{-8}, 10^{-6}, 1)$-schedulability under pMC, as well as $(10^{-8}, 10^{-6}, -)$-schedulability under pDMPO. Since by definition $\mathsf{PDJ}_{\mathrm{deg}} \leq 1$, we can

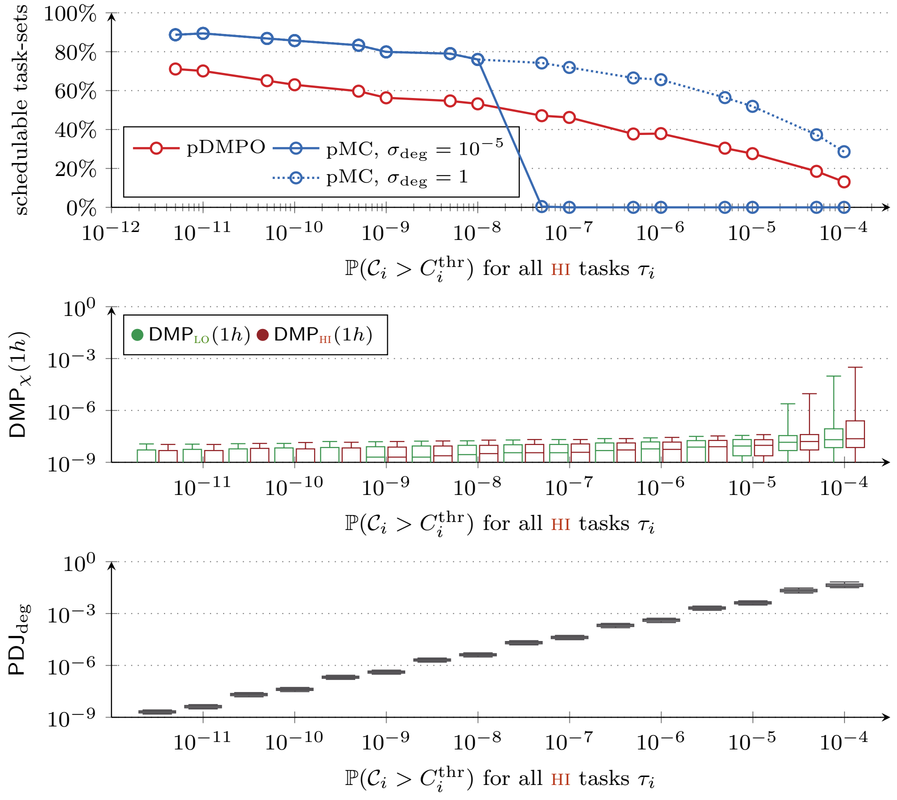


Fig. 4 $(10^{-8}, 10^{-6}, 10^{-5})$-schedulability of task-sets under pMC and pDMPO, and $(10^{-8}, 10^{-6}, 1)$ -schedulability under pMC, for various probabilities that a HI job overruns its execution time threshold $\mathbb{P}(\mathcal{C}_i > C_i^{\text{thr}})$ (Top), and the impact this value has on $\mathsf{DMP}_{\text{LO}}(1h)$, $\mathsf{DMP}_{\text{HI}}(1h)$ (Middle) and $\mathsf{PDJ}_{\text{deg}}$ (Bottom)

interpret $(\sigma_{\text{HI}}, \sigma_{\text{LO}}, \sigma_{\text{deg}} = 1)$-schedulability under pMC as a schedulability test which ignores the $\mathsf{PDJ}_{\text{deg}}$ metric. In the middle and bottom figures, the box-plots visualize the 10th, 25th, 50th, 75th, and 90th percentile of each metric.

In Fig. 4 Top, let us first focus on comparing pDMPO and pMC when $\sigma_{\text{deg}} = 1$. In this case, when the $\mathsf{PDJ}_{\text{deg}}$ metric is ignored, we see that more task-sets are always schedulable under pMC than under pDMPO. The reasons pMC scheduling is better in this case are the same reasons as in the 'utilization' experiment: by switching to HI-criticality mode after certain triggering events, the system under pMC scheduling reduces the backlog in these situations, which ultimately makes deadline misses less likely.

Now, let us examine pMC with a realistic $\mathsf{PDJ}_{\text{deg}}$ bound, i.e. $\sigma_{\text{deg}} = 10^{-5}$. As shown in the top figure, it is clear that there exists a limit after which pMC scheduling is not useful at all, as it leads to too much degradation. This can be understood by viewing Fig. 4 Bottom, where we see the cost of switching to HI-mode. On one

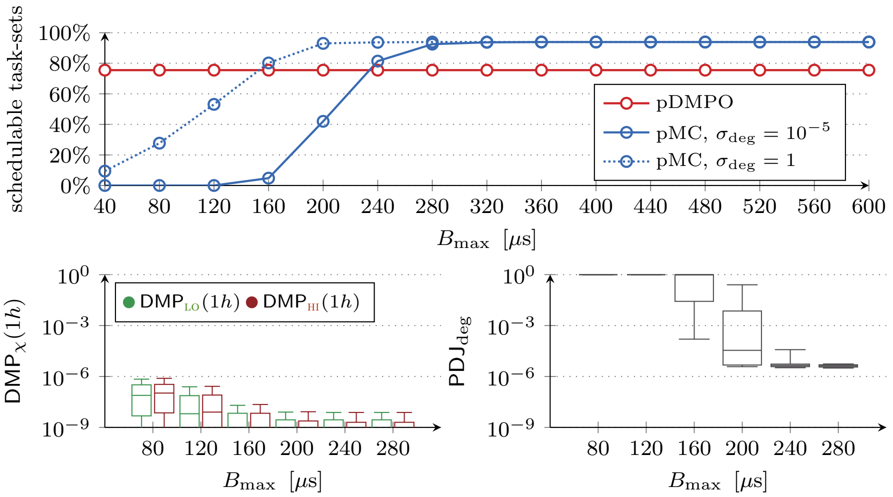


Fig. 5 $(10^{-8}, 10^{-6}, 10^{-5})$-schedulability of task-sets under pMC and pDMPO, and $(10^{-8}, 10^{-6}, 1)$-schedulability under pMC, for various $B_{\max}$ values (Top), and the impact this value has on $\mathsf{DMP}_{\text{LO}}(1h)$, $\mathsf{DMP}_{\text{HI}}(1h)$, and $\mathsf{PDJ}_{\deg}$ (Bottom)

extreme case, when $\mathbb{P}(\mathcal{C}_i > C_i^{\text{thr}}) = 10^{-4}$, the system switches to HI-mode often, on average once every 48.93ms (not shown in figure). Then, an average ratio of 0.046 of LO jobs are released in degraded mode. In a moderate case, for $\mathbb{P}(\mathcal{C}_i > C_i^{\text{thr}}) = 10^{-8}$, HI jobs overrun their execution time threshold C_i^{thr} less often, and LO-mode lasts on average 8.34 min. Here, an average ratio of $4.19 \cdot 10^{-6}$ of LO jobs are degraded. Finally, on the other extreme case, when $\mathbb{P}(\mathcal{C}_i > C_i^{\text{thr}}) = 5 \cdot 10^{-12}$, LO-mode lasts for 278.00 h on average, and only a tiny fraction of $2.09 \cdot 10^{-9}$ LO jobs are released in degraded mode. For many realistic applications, there exists a limit on the degradation which can be tolerated, before a complete loss of function happens. Thus we argue that this experiment demonstrates why the $\mathsf{PDJ}_{\deg}$ metric is crucial for probabilistic Mixed Criticality scheduling.

Finally, let us comment on $\mathsf{DMP}_{\text{LO}}(1h)$ and $\mathsf{DMP}_{\text{HI}}(1h)$, found in Fig. 4 Middle. These are similar, except $\mathsf{DMP}_{\text{HI}}(1h)$ is larger for higher $\mathbb{P}(\mathcal{C}_i > C_i^{\text{thr}})$ values. We have found that this increase in $\mathsf{DMP}_{\text{HI}}(1h)$ appears as a result of pessimistic assumptions introduced in Definition 12. We comment more about this pessimism the next experiment.

6.2.3 'Maximal Backlog' experiment

In the final experiment on randomized systems, the maximum system-level backlog $B_{\max}$ was varied. This affects how often HI-criticality mode is entered, while it has no effect on the LO-criticality mode. When the occurrence of HI-criticality mode is artificially increased, we can see the pessimism in the analysis of that mode—which we found mostly to be introduced by pessimistic assumptions on the initial conditions in HI-mode, as per Definition 12. As in the previous experiment, we tested the $(10^{-8}, 10^{-6}, 10^{-5})$-schedulability of task-sets under pMC and pDMPO, and $(10^{-8}, 10^{-6}, 1)$-schedulability under pMC scheduling.

Setup For this experiment, we first generated 1000 task-sets with a system utilization of 1.2. This high utilization guarantees that no deterministic scheme can be used to schedule task-sets. All parameters except for the maximum system-level backlog are according to the description at the beginning of this section. Then, the maximum system-level backlog $B_{\max}$ was varied from 40 to 600 µs, and all of the 1000 task-sets are analyzed for every $B_{\max}$ value. Each generated task-set has 60 tasks, the mean execution time among all tasks in every task-set is 10.61 µs, while the maximum execution time overall is 255 µs.

Results Figure 5 visualizes the results of this experiment. As done in the previous experiment, we conducted a $(10^{-8}, 10^{-6}, 10^{-5})$-schedulability test under pMC and pDMPO, as well as a schedulability test under pMC when the $\mathsf{PDJ}_{\text{deg}}$ metric is ignored (i.e. $\sigma_{\text{deg}} = 1$). The box-plots visualize the 10th, 25th, 50th, 75th, and 90th percentile of each evaluated metric. By definition, the maximum system-level backlog $B_{\max}$ does not impact scheduling under pDMPO at all, so the schedulability under this scheme is constant.

Regarding the impact on pMC scheduling, specifically on $\mathsf{DMP}_{\text{HI}}(1h)$ and on $\mathsf{DMP}_{\text{LO}}(1h)$, we see two cases. On the one hand, when the maximum system-level backlog $B_{\max}$ is sufficiently large, i.e. $\geq$ 200 µs, we see that it has a negligible impact on $\mathsf{DMP}_{\text{HI}}(1h)$ and $\mathsf{DMP}_{\text{LO}}(1h)$ values. On the other hand, when a small $B_{\max}$ causes HI-mode to be entered often, $\mathsf{DMP}_{\text{HI}}(1h)$ and $\mathsf{DMP}_{\text{LO}}(1h)$ both increase. Ideally, how often HI-mode is entered should not impact $\mathsf{DMP}_{\text{HI}}(1h)$ and $\mathsf{DMP}_{\text{LO}}(1h)$. The increase is a result of pessimism introduced in point 4 of Definition 12. As the reader recalls, there we make a pessimistic assumption that all HI jobs are overrunning their execution time thresholds C_i^{thr} at the time of the mode switch. This pessimistic assumption is mainly introduced to reduce the number of cases under which HI-criticality mode is analyzed.

The impact the backlog $B_{\max}$ has on $\mathsf{PDJ}_{\text{deg}}$ is straightforward. As HI-mode is entered more often, $\mathsf{PDJ}_{\text{deg}}$ increases. Because of this increase, we find that few task-sets are $(10^{-8}, 10^{-6}, 10^{-5})$-schedulable under pMC for $B_{\max}$ values less than 200 µs. We can therefore conclude thus the pessimism of HI-criticality mode analysis does not play a major role in the schedulability analysis of task-sets under realistic requirements for the maximal permitted degradation of LO jobs σ_{deg}. Finally, we observe again the main result from the 'execution threshold' experiment: probabilistic Mixed Criticality (pMC) scheduling is better than the non-MC scheme pDMPO, except when HI-criticality mode is entered too often.

7 Conclusion

Modeling tasks' execution times with random variables in Vestal's mixed-criticality model allows for a schedulability analysis based on the 'probability of deadline miss per hour'. We presented a dual-criticality system which operates in either LO- or HI-criticality mode. In LO-criticality mode, both LO and HI jobs run normally, but certain optimism towards HI jobs exists: they are required not to overrun their C_i^{thr} execution time threshold, a value analogues to the optimistic WCET in Vestal's model. HI-criticality mode is entered when a violation of this optimistic condition is detected, or when one of the following two events happen: a HI job misses its deadline, or the system-level backlog exceeds its maximal value. In this mode, LO jobs are degraded by having a shorter time budget for execution, so HI jobs have more resources available. This mode lasts until the system becomes idle.

To characterize such a system, we first defined $(\sigma_{\mathrm{HI}}, \sigma_{\mathrm{LO}}, \sigma_{\mathrm{deg}})$-schedulability, which quantifies the soft schedulability of a probabilistic mixed-criticality system. The schedulability conditions determine whether the *probability of deadline miss per hour for* HI *jobs*, the *probability of deadline miss per hour for* LO *jobs* and the *probability a* LO *job is started in its degraded mode* are less that the given $(\sigma_{\mathrm{HI}}, \sigma_{\mathrm{LO}}, \sigma_{\mathrm{deg}})$ limits.

Then, we presented an analysis approach. This was done by splitting the system into two—the LO- and the HI-criticality mode system—and combining the results. On one hand, a steady state analysis was carried out for LO-criticality mode, in which the system is expected to stay for a long time. This enabled us to pessimistically bound the deadline miss probability of each job, which we then used to find the probability that any job misses its deadline while in LO-mode in a certain time period. On the other hand, a simulation of the transient HI-criticality mode was used to bound its duration, and to obtain the probability of deadline miss of jobs inside it. This, together with the probability a LO → HI mode switch happens, enabled us to find the probability any job misses its deadline while in HI-mode in a certain time period.

Finally, simulation results illustrate all of the metrics on a sample task-set, and experiments involving schedulability analysis show how various design choices impact schedulability. Here, we show how probabilistic Mixed Criticality scheduling compares to other schemes, and make a clear case that using pMC makes sense for most cases, except when the amount of LO job degradation is too high.

Limitations and Future Work Our analysis applies for fixed priority preemptive scheduling, but it could be extended to dynamic scheduling schemes as well. On the one hand, probabilistic response-time calculus already exists for dynamic schemes (Díaz et al. 2002). In addition, dynamic-priority Mixed-Criticality schemes are found to be relevant (Baruah et al. 2011; Guo et al. 2015).

Regarding our proposed scheme, its main limitation is the pessimism of the analysis of HI-criticality mode. This pessimism is due to the fact that we have a single analysis whatever the reason for making the LO → HI transition was.

In a future work, it would be possible to do a less pessimistic analysis of HI-mode by deconstructing the analysis into three sub-classes, one for each LO → HI mode switch reason. For example, if a mode switch was caused by a maximal system-level backlog exceedance, the initial backlog would surely be exactly B_{max}. If the mode switch was not caused by an overrunning job, there would be no need to assume that carry-over jobs of HI criticality surely overrun. If the mode switch was caused by an overrunning HI job, one could introduce cases depending on which job cause the mode switch.

Table 4 pMC analysis runtimes, for different number of jobs in a hyperperiod n

Tasks per task-sets	13	25	60	75	100
Jobs in hyperperiod n	50–125	150–200	350–475	450–650	650–825
Utilization	0.5	1.0	1.2	1.5	2.0
Average runtime	5s	19s	1min 59s	4min 21s	23min 42s

The pessimism of the analysis for the LO-criticality mode could be reduced as well, but arguably this would bear less fruit. One idea here is to estimate the percentage of time a system spends in LO-criticality mode. In calculating $\mathsf{DMP}_\chi(T)$ in our work, we assumed the system is in LO-mode all the time. Replacing this assumption with a better estimate would bring improvement, however only for systems which spend a non-negligible amount of time in HI-criticality mode, which is usually not assumed to be the case. Another idea is to use a less pessimistic model of HI tasks in LO-mode, by modeling their executions with conditional 'truncated' execution times as is done in several related works (Draskovic et al. 2016; Maxim et al. 2017). However, this would require performing two LO-mode analyses: the one presented here would be used to calculate initial conditions in HI-mode, and the other with the less pessimistic model of HI tasks would be used to calculate deadline miss probabilities in LO-criticality mode.

Appendix: Computational complexity of the analysis

Here we comment on the computational complexity of our proposed probabilistic Mixed Criticality (pMC) schedulability analysis. Algorithm A presents a high-level recapitulation of the analysis, where all pseudo-commands are as explained in Sect. 5.

The computational complexity of the analysis is $\mathcal{O}(n^2 \cdot \text{HP} \cdot c \log c)$, where n is the number of jobs in one hyperperiod, HP is the length of one hyperperiod, and c is the length or number of values in the execution time distributions.

In the analysis, the most complex atomic command is the convolution. When using FFT, one convolution has a cost of $\mathcal{O}(c \log c)$.

Let us now comment on the complexity of the analysis in detail. According to Sect. 4.1, the steady state backlog is approximated by $\mathcal{B}_i(k \cdot \text{HP})$, where k is the smallest natural number satisfying inequality (9). To calculate $\mathcal{B}_i(k \cdot \text{HP})$, a convolution is needed for every one of the $n \cdot k$ jobs, thus the cost of line 2 is $\mathcal{O}(n \cdot k \cdot c \log c)$. Similarly, according to point 4 of Definition 12, backlog $\widehat{\mathcal{B}}_i(t)$ is defined as a combination of two steady state backlogs, and the cost of line 14 is also $\mathcal{O}(n \cdot k \cdot c \log c)$. The number k depends on the required numerical precision (9), but we have found it to be in the same order of magnitude as n, $k \sim n$.

To compute deadline miss probabilities, i.e. lines 4 and, 17, response time analysis is used as defined by Algorithm 1. Line 6 is based on response time analysis as well (Lemma 8). To find the response time of a job, we need to do as many convolutions as there are jobs preempting the said job. Thus, the cost of these lines is $\mathcal{O}(n \cdot c \log c)$.

Finally, when analyzing HI-mode, the maximal duration of the mode $\Delta^{\text{HI}}_{\max}$ plays a role. When calculating $\Delta^{\text{HI}}_{\max}$ in line 15, and when computing deadlines miss probabilities of jobs in lines 16 and 17, we need to take into account all jobs that are released in HI-mode. Regardless on when HI-mode is entered or exited, the number of these jobs is at most $n \cdot \Delta^{\text{HI}}_{\max} / \text{HP}$. For schedulable systems, we found that $\Delta^{\text{HI}}_{\max}$ is in the same order of magnitude as HP, $\Delta^{\text{HI}}_{\max} \sim \text{HP}$ and $\Delta^{\text{HI}}_{\max} / \text{HP} \sim 1$.

Algorithm A High-Level Overview of pMC Schedulability Analysis

```
1:  procedure LO-criticality Mode Analysis                                         ▷ $\mathcal{O}(n^2 \cdot c \log c)$
2:      Compute steady state backlog $\overline{\mathcal{B}}_i(0)$ (Theorem 1)                     ▷ $\mathcal{O}(n \cdot k \cdot c \log c)$
3:      for each job $\tau_{i,j}$ in HP do                                             ▷ $\mathcal{O}(n^2 \cdot c \log c)$
4:          Compute deadline miss prob. $\overline{\mathsf{DMP}}_{i,j}$ (Theorem 3, Algorithm 1)    ▷ $\mathcal{O}(n \cdot c \log c)$
5:          if $\tau_{i,j}$ is HI job then
6:              Compute time of $C_i^{\text{thr}}$ overrun detection (Lemma 8)              ▷ $\mathcal{O}(n \cdot c \log c)$
7:      Get $\mathsf{DMP}^{\text{LO}}_{\chi}$ (Lemma 5)                                            ▷ $\mathcal{O}(n)$
8:      Get $\text{P}_{dm}(t)$ for each $t$ in HP (Lemma 6)                                ▷ $\mathcal{O}(\text{HP} \cdot n)$
9:      Get $\text{P}_{be}(t)$ for each $t$ in HP (Lemma 7)                                ▷ $\mathcal{O}(\text{HP} \cdot c)$
10:     Get $\text{P}_{ov}(t)$ for each $t$ in HP(Lemma 8)                                 ▷ $\mathcal{O}(\text{HP} \cdot n)$
11:     Get $\text{P}^{\text{HP}}_{\text{LO}\to\text{HI}}$ (Theorem 6)                                         ▷ $\mathcal{O}(1)$
12: procedure HI-criticality Mode Analysis                                         ▷ $\mathcal{O}(\text{HP} \cdot n^2 \cdot c \log c)$
13:     for each time $t$ in HP do                                                 ▷ $\mathcal{O}(\text{HP} \cdot n^2 \cdot c \log c)$
14:         Compute initial backlog $\widehat{\mathcal{B}}_i(t)$ (Definition 12)                   ▷ $\mathcal{O}(n \cdot k \cdot c \log c)$
15:         Get $\Delta^{\text{HI}}_{\max}(t)$ (Lemma 9)                                       ▷ $\mathcal{O}(n \cdot \Delta^{\text{HI}}_{\max} / \text{HP} \cdot c \log c)$
16:         for each job $\tau_{i,j}$ in $[t, t + \Delta^{\text{HI}}_{\max}(t))$ do                   ▷ $\mathcal{O}(n^2 \cdot \Delta^{\text{HI}}_{\max} / \text{HP} \cdot c \log c)$
17:             Compute deadline miss prob. (Lemma 9)                              ▷ $\mathcal{O}(n \cdot c \log c)$
18:         Get $\mathsf{DMP}^{\text{HI}}_{\chi}(t)$ (Lemma 9)                                       ▷ $\mathcal{O}(n)$
19:     Get $\mathsf{DMP}^{\text{HI}}_{\chi}$, $\Delta^{\text{HI}}_{\max}$ (Theorem 7)                                   ▷ $\mathcal{O}(\text{HP})$
20: Get $\mathsf{DMP}_{\text{HI}}(1h)$, $\mathsf{DMP}_{\text{LO}}(1h)$ (Theorem 5)                                 ▷ $\mathcal{O}(1)$
21: Get $\mathsf{PDJ}_{\text{deg}}$ (Theorem 4)                                                 ▷ $\mathcal{O}(1)$
```

Event though the computational complexity of this scheme is high, we find it to be acceptable. The analysis only needs to be done offline, while designing the system. Furthermore, parts of Algorithm A are parallelizable. Each iteration of the for-loop in line 13 can be run independently, meaning that the analysis of HI-mode can be done in parallel on HP processes, each of complexity $\mathcal{O}(n^2 \cdot c \log c)$. Consequently, this would be the computational complexity of the whole Algorithm, if we were to have unlimited resources.

Runtimes For Sect. 6.2, we ran the analysis of each task-set on a single core of a Dual Deca-Core Intel Xeon E5-2690 v2, running at 3.00GHz. As defined in the task-set generation, all task-sets have $\text{HP} = 1000$ and $c \sim 1000$. In Table 4, we noted the average analysis runtimes for task-sets of different utilizations and number of jobs.

Appendix: Notation

See Table 5.

Table 5 Notations

Random variable	$\mathcal{A}$
Probability function	$p_{\mathcal{A}}(u) = \mathbb{P}(\mathcal{A} = u)$
Vector representation	$p_{\mathcal{A}} = [p_{\mathcal{A}}(u_{\min}) \dots p_{\mathcal{A}}(u_{\max})]^{\intercal}$
Comparison of random variables	$\mathcal{A} \succeq \mathcal{B}$
Upper bound on random variable	$\overline{\mathcal{A}} \succeq \mathcal{A}$
Task set, number of tasks, task i, job j	Π, N, τ_i, $\tau_{i,j}$
Task period	T_i
Length of a hyperperiod	HP
Relative deadline of task i	D_i
Absolute deadline of $\tau_{i,j}$	$d_{i,j}$
Release time of $\tau_{i,j}$	$r_{i,j}$
Initial phase	ϕ_i
Criticality level	$\chi_i \in \{\mathrm{LO}, \mathrm{HI}\}$
Probabilistic execution time	$\mathcal{C}_i \preceq C_i^{\max}$
Threshold of execution time for HI-jobs	C_i^{thr}
Maximal execution time for degraded LO-jobs	C_i^{deg}
Trimmed execution time of HI-jobs	$\mathcal{C}_i^{\mathrm{LO}}$
Threshold of backlog	$B_{\max}$
Response time of $\tau_{i,j}$	$\mathcal{R}_{i,j}$
Deadline miss probability of $\tau_{i,j}$	$\mathsf{DMP}_{i,j}$
Upper bound on deadline miss probability of $\tau_{i,j}$	$\overline{\mathsf{DMP}}_{i,j}$
Deadline miss prob. within T for HI- and LO-jobs	$\mathsf{DMP}_{\mathrm{HI}}(T), \mathsf{DMP}_{\mathrm{LO}}(T)$
One hour	$1h$
Degradation probability of LO-jobs	$\mathsf{PDJ}_{\mathrm{deg}}$
Backlog at priority level i and time t	$\mathcal{B}_i(t)$
Convolution operator	$\oplus$
Shrinking function of $\mathcal{B}$ at value m	$\mathrm{shrink}(\mathcal{B}, m)(u)$
Backlog function	$\mathtt{bsse}\big(\mathcal{B}_i(t),\ \Pi,\ i,\ t,\ u\big)$
Response time function	$\mathtt{rta}\big(\mathcal{B}_i(r_{i,j}),\ \Pi,\ \tau_{i,j}\big)$
Maximum duration of any HI-mode execution	$\Delta_{\max}^{\mathrm{HI}}$
Expected length of a LO-mode execution	$\Delta_{exp}^{\mathrm{LO}}$
w.c. prob. of a LO → HI mode switch at time $t \in \{0, \dots \mathrm{HP} - 1\}$ within a hyperperiod	$\mathrm{P}_{\mathrm{LO}\rightarrow\mathrm{HI}}(t)$
w.c. prob. of a LO → HI mode switch at time $t \in \{0, \dots \mathrm{HP} - 1\}$ due to deadline miss	$\mathrm{P}_{dm}(t)$
w.c. prob. of a LO → HI mode switch at time $t \in \{0, \dots \mathrm{HP} - 1\}$ due to backlog exceedance	$\mathrm{P}_{be}(t)$
w.c. prob. of a LO → HI mode switch at time $t \in \{0, \dots \mathrm{HP} - 1\}$ due to execution time overrun	$\mathrm{P}_{ov}(t)$
w.c. prob. of at least one LO → HI mode switch within a hyperperiod	$\mathrm{P}_{\mathrm{LO}\rightarrow\mathrm{HI}}^{\mathrm{HP}}$
w.c. prob. of at least one deadline miss of any χ job during any HI mode started within a hyperperiod	$\mathsf{DMP}_{\chi}^{\mathrm{HI}}$
w.c. prob. of at least one deadline miss of any χ job during a hyperperiod during LO mode execution	$\mathsf{DMP}_{\chi}^{\mathrm{LO}}$

Table 5 (continued)

w.c. task set of HI-mode if starting at t	$\widehat{\Pi}(t)$
w.c. duration of HI-mode if starting at t	$\Delta^{\mathrm{HI}}_{\max}(t)$
w.c. prob. of at least one deadline miss of any χ job during any HI mode started at t	$\mathsf{DMP}^{\mathrm{HI}}_{\chi}(t)$

Funding Open Access funding provided by ETH Zurich.

References

Rtca/do-178c software considerations in airborne systems and equipment certification (2012)

Abdeddaïm Y, Maxim D (2017) Probabilistic schedulability analysis for fixed priority mixed criticality real-time systems. In: Design, automation and test in Europe conference and exhibition (DATE), 2017. IEEE, pp 596–601

Alahmad B, Gopalakrishnan S (2016) A risk-constrained Markov decision process approach to scheduling mixed-criticality job sets. In: Workshop on mixed criticality systems (WMC 2016)

Alahmad BN, Gopalakrishnan S (2018) Risk-aware scheduling of dual criticality job systems using demand distributions. Leibniz Trans Embed Syst 5(1);01-1

Audsley NC, Burns A, Richardson MF, Wellings AJ (1991) Hard real-time scheduling: the deadline-monotonic approach. IFAC Proc Vol 24(2):127–132

Baruah S, Fohler G (2011) Certification-cognizant time-triggered scheduling of mixed-criticality systems. In: 2011 IEEE 32nd real-time systems symposium (RTSS). IEEE, pp 3–12

Baruah SK, Bonifaci V, D'Angelo G, Marchetti-Spaccamela A, Van Der Ster S, Stougie L (2011) Mixed-criticality scheduling of sporadic task systems. In: ESA. Springer, pp 555–566

Baruah SK, Burns A, Davis RI (2011) Response-time analysis for mixed criticality systems. In: 2011 IEEE 32nd real-time systems symposium (RTSS). IEEE, pp 34–43

Burns A, Davis RI (2017) A survey of research into mixed criticality systems. ACM Comput Surv (CSUR) 50(6):1–37

Cucu-Grosjean L, Santinelli L, Houston M, Lo C, Vardanega T, Kosmidis L, Abella J, Mezzetti E, Quinones E, Cazorla FJ (2012) Measurement-based probabilistic timing analysis for multi-path programs. In: 2012 24th Euromicro conference on real-time systems (ECRTS). IEEE, pp 91–101

Davis RI, Burns A (2011) Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. Real Time Syst 47(1):1–40

Davis RI, Burns A, Griffin D (2017) On the meaning of pWCET distributions and their use in schedulability analysis. In: Real-time scheduling open problems seminar at ECRTS

Davis RI, Cucu-Grosjean L (2019) A survey of probabilistic schedulability analysis techniques for real-time systems. Leibniz Trans Embed Syst 6(1):1–53

Davis RI, Cucu-Grosjean L (2019) A survey of probabilistic timing analysis techniques for real-time systems. Leibniz Trans Embed Syst 6(1):03-1

Devgan A, Kashyap C (2003) Block-based static timing analysis with uncertainty. In: Proceedings of the 2003 IEEE/ACM international conference on computer-aided design. IEEE Computer Society, p 607

Díaz JL, García DF, Kim K, Lee CG, Bello LL, López JM, Min SL, Mirabella O (2002) Stochastic analysis of periodic real-time systems. In: 23rd IEEE real-time systems symposium, 2002. RTSS 2002. IEEE, pp 289–300

Díaz JL, López JM (2004) Safe extensions to the stochastic analysis of real-time systems. Technical report. Departamento de Informatica, University of Oviedo. http://www.atc.uniovi.es/research/SESARTS04.pdf. Accessed 1 Oct 2019

Díaz JL, López JM, Garcia M, Campos AM, Kim K, Bello LL (2004) Pessimism in the stochastic analysis of real-time systems: concept and applications. In: 25th IEEE international real-time systems Symposium, 2004. Proceedings. IEEE, pp 197–207

Draskovic S, Huang P, Thiele L (2016) On the safety of mixed-criticality scheduling. In: Workshop on mixed criticality systems (WMC 2016)

Durrieu G, Faugere M, Girbal S, Pérez DG, Pagetti C, Puffitsch W (2014) Predictable flight management system implementation on a multicore processor. In: Embedded real time software (ERTS'14)

Ekberg P, Yi W (2012) Outstanding paper award: bounding and shaping the demand of mixed-criticality sporadic tasks. In: 2012 24th Euromicro conference on real-time systems (ECRTS). IEEE, pp 135–144

Ernst R, Di Natale M (2016) Mixed criticality systems—a history of misconceptions? IEEE Des Test 33(5):65–74

Guo Z, Santinelli L, Yang K (2015) EDF schedulability analysis on mixed-criticality systems with permitted failure probability. In: 2015 IEEE 21st international conference on embedded and real-time computing systems and applications (RTCSA). IEEE, pp 187–196

Huang P, Giannopoulou G, Stoimenov N, Thiele L (2014) Service adaptions for mixed-criticality systems. In: 2014 19th Asia and South Pacific design automation conference (ASP-DAC). IEEE, pp 125–130

Küttler M, Roitzsch M, Hamann CJ, Volp M (2017) Probabilistic analysis of low-criticality execution. In: Workshop on mixed criticality systems (WMC 2017)

López JM, Díaz JL, Entrialgo J, García D (2008) Stochastic analysis of real-time systems under preemptive priority-driven scheduling. Real Time Syst 40(2):180

Masrur A (2016) A probabilistic scheduling framework for mixed-criticality systems. In: Proceedings of the 53rd annual design automation conference. ACM, p 132

Maxim D, Buffet O, Santinelli L, Cucu-Grosjean L, Davis RI (2011) Optimal priority assignment algorithms for probabilistic real-time systems. In: RTNS, pp 129–138

Maxim D, Davis RI, Cucu-Grosjean L, Easwaran A (2017) Probabilistic analysis for mixed criticality systems using fixed priority preemptive scheduling. In: Proceedings of the 25th international conference on real-time networks and systems. ACM, pp 237–246

Pagetti C, Saussié D, Gratia R, Noulard E, Siron P (2014) The ROSACE case study: from simulink specification to multi/many-core execution. In: 2014 IEEE 20th real-time and embedded technology and applications symposium (RTAS). IEEE, pp 309–318

Park T, Kim S (2011) Dynamic scheduling algorithm and its schedulability analysis for certifiable dual-criticality systems. In: Proceedings of the ninth ACM international conference on Embedded software. ACM, pp 253–262

Santinelli L, George L (2015) Probabilities and mixed-criticalities: the probabilistic c-space. In: Proceedings of the workshop on mixed criticality systems (WMC 2015)

Santinelli L, Guo Z (2018) A sensitivity analysis for mixed criticality: trading criticality with computational resource. In: 2018 IEEE 23rd international conference on emerging technologies and factory automation (ETFA), vol 1. IEEE, pp 313–320

Santinelli L, Guo Z, George L (2016) Fault-aware sensitivity analysis for probabilistic real-time systems. In: 2016 IEEE international symposium on defect and fault tolerance in VLSI and nanotechnology systems (DFT), pp 69–74

Tămaş-Selicean D, Pop P (2015) Design optimization of mixed-criticality real-time embedded systems. ACM Trans Embed Comput Syst 14(3):50

Vestal S (2007) Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In: 28th IEEE international real-time systems symposium, 2007. RTSS 2007. IEEE, pp 239–243

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Stefan Draskovic received his M.Sc. degree in Electrical Engineering and Information Technology from ETH Zurich in 2015. He is currently working toward a Ph.D. degree in the Computer Engineering Group of the same university. His research is mainly in the real time systems domain, particularly the design and analysis of energy harvesting embedded systems, wireless communication, and mixed-criticality systems.

Rehan Ahmed is currently an Assistant Professor at the Computer Engineering Department, Information Technology of the Punjab, Pakistan. He received his B.S. (2007) in Electrical Engineering from University of Engineering and Technology Lahore, Pakistan. He received his M.S. (2010) and Ph.D. (2015) in Electrical Engineering from University of Wisconsin Madison, USA. His research focus during M.S./Ph.D. was on thermal constrained scheduling of real-time systems. From 2015–2019, he was a postdoctoral researcher at Computer Engineering and Networks Laboratory (TIK), Eidgenössische Technische Hochschule (ETH) Zürich. His main research interests are design/analysis of cyber-physical and energy harvesting systems.

Pengcheng Huang holds a Ph.D. degree in Computer Engineering from ETH Zurich. After his Ph.D., he spent three years (2017–2020) at Zumtobel innovating IoT and machine learning systems. He is currently a scientist at ABB working on AI and Cyber Physical Systems.

Lothar Thiele joined ETH Zurich, Switzerland, as a full Professor of Computer Engineering, in 1994. His research interests include models, methods and software tools for the design of real-time embedded systems, internet of things, cyberphysical systems, sensor networks, embedded software and bioinspired optimization techniques. In 1986 he received the “Dissertation Award” of the Technical University of Munich, in 1987, the "Outstanding Young Author Award" of the IEEE Circuits and Systems Society, in 1988, the Browder J. Thompson Memorial Award of the IEEE, and in 2000–2001, the “IBM Faculty Partnership Award”. In 2004, he joined the German Academy of Sciences Leopoldina. In 2005, he was the recipient of the Honorary Blaise Pascal Chair of University Leiden, The Netherlands. Since 2010, he is a member of the Academia Europaea. In 2013, he joined the National Research Council of the Swiss National Science Foundation SNF. Lothar Thiele received the “EDAA Lifetime Achievement Award” in 2015. Lothar Thiele has been elected IFIP Fellow by the International Federation for Information Processing (IFIP) as part of its first cohort of fellows in 2019.