Analysis of Deadline Miss Rates for Uniprocessor Fixed-Priority Scheduling

Kuan-Hsun Chen, Georg von der Brüggen, and Jian-Jia Chen
Department of Informatics, TU Dortmund University, Germany
https://ls12-www.cs.tu-dortmund.de/

Citation: 10.1109/RTCSA.2018.00028

Analysis of Deadline Miss Rates for Uniprocessor Fixed-Priority Scheduling

Kuan-Hsun Chen, Georg von der Brüggen, and Jian-Jia Chen
Department of Informatics, TU Dortmund University, Germany
E-mail: {kuan-hsun.chen, georg.von-der-brueggen, jian-jia.chen} @tu-dortmund.de

***Abstract*—Timeliness is an important feature for many embedded systems. Although soft real-time embedded systems can tolerate and allow certain deadline misses, it is still important to quantify them to justify whether the considered systems are acceptable. In this paper, we provide a way to safely over-approximate the expected deadline miss rate for a specific sporadic real-time task under fixed-priority preemptive scheduling in uniprocessor systems. Our approach is compatible with the existing results in the literature that calculate the probability of deadline misses either based on the convolution-based approaches or analytically. We demonstrate our approach by considering randomly generated task sets with an execution behavior that simulates jobs that are subjected to soft errors incurred by hardware transient faults under a given fault rate. To empirically gather the deadline miss rates, we implemented an event-based simulator with a fault-injection module and release the scripts. With extensive simulations under different fault rates, we evaluate the efficiency and the pessimism of our approach. The evaluation results show that our approach is effective to derive an upper bound of the expected deadline miss rate and efficient with respect to the required computation time.**

Keywords - Soft Real-Time Systems; Deadline Miss Rate

I. Introduction

Timeliness is an important feature for many embedded systems. Specifically, for safety-critical systems, *hard* real-time guarantees are of importance to ensure that the results are not just functionally correct but also *always* delivered within given timing constraints. In such hard real-time systems, it is assumed that any deadline miss can lead to a catastrophe and must be avoided. By contrast, *soft* real-time systems can tolerate a certain amount of deadline misses. However, intuitively, deadline misses in soft real-time systems should still be avoided as much as possible, and it is important to quantify the deadline misses to justify whether the considered systems are acceptable.

The existing approaches can in general be classified into static and statistical approaches. A *static approach*, e.g., [10], quantifies the *maximum* tardiness (or lateness) of a task, defined as the difference between the worst-case response time of the task and its relative deadline. Such a static quantification provides a means to justify the maximum over-run with respect to the deadline, but it does not provide the *frequency* of deadline misses. Alternatively, a *statistical* approach provides statistical quantification such as *deadline miss rates*, e.g., [16], *probabilistic response-time analyses*, e.g., [2, 3, 11, 19, 22], and the *probability of deadline misses*, e.g., [8, 19, 24].

Both the deadline miss rate as well as the probability of deadline misses are important performance indicators to evaluate the extent of requirements compliance for soft real-time systems. Nevertheless, existing probabilistic approaches, i.e., [2, 3, 11, 19, 22], focus on finding the probability of the first deadline miss, and it is assumed that after a deadline miss the system either discards the job missing its deadline or reboots itself. Therefore, the probability of one deadline miss directly relates to the deadline miss rate since all jobs can be considered individually. However, those very restrictive assumptions often do not hold in practice, as aborting jobs or rebooting the system after a deadline miss is not always an option. If this is the case, the additional workload due to a deadline miss may trigger further deadline misses. Hence, the actual deadline miss rate may be greater than the probability of the first deadline miss as shown in the following example.

Consider two implicit-deadline periodic tasks τ_1 and τ_2 under fixed-priority preemptive scheduling. Task τ_1 has a worst-case execution time (WCET) of 2 and a period of 3. Task τ_2 has a period of 5 and two distinct versions identified by the different resulting WCETs, which is either 1 or 2.25. Assume that for each job of τ_2 one of the two versions is executed with 50% probability. Since the first job of τ_2 will meet its deadline if it is executed for 1 time unit and miss the deadline if it is executed for 2.25 time units, the probability of deadline miss for the first job is 50%. However, as shown in Figure 1, once the first job of τ_2 executes 2.25 time units, which leads to its deadline miss, the second job definitely misses its deadline as well. In this example, the probability that the first job of τ_2 misses its deadline is 50% and therefore at least 2 of the 3 jobs of τ_2 miss their deadline in this case. Obviously, the occurrence of a pattern that leads to a greater deadline miss rate than 50% does not mean the actual miss rate is greater than 50% as well, since all other possible patterns and the related possibility must be considered. Furthermore, a deadline miss at time 15 will propagate into the next hyper-period[1], which complicates the calculation. Hence, to estimate the actual deadline miss rate in the displayed scenario, we deployed the aforementioned setting in our

[1]The hyper-period is the least common multiple of all task periods.

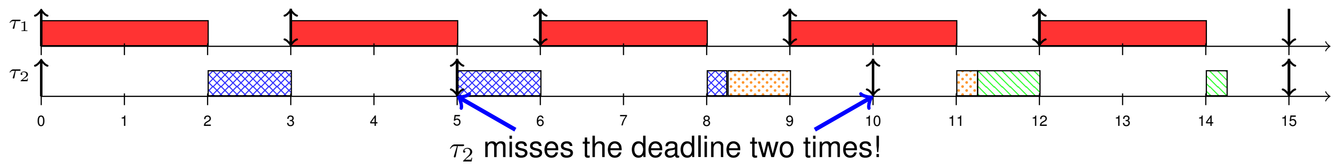


Figure 1: An example schedule showing that a deadline miss of the first job of τ_2 (at time 5) directly leads to an additional deadline miss of the second job of τ_2 (at time 10). For τ_2, same color blocks represent the same j-th job. The upward arrows represent the arrival time for each job, whereas the next downward arrow represent the related deadline.

event-based simulator (to be detailed in Section VI). The empirical results show that the deadline miss rate on average was 93.04% over 100 simulations where in each simulation five million jobs of τ_2 were considered. This shows that it is necessary to analyze more than just the first deadline miss of a task when considering the deadline miss rate if aborting jobs or rebooting the system after a deadline miss is not possible.

Regarding the deadline miss rate, most results in this direction, e.g., [5, 12, 13, 15, 21], consider measurement-based evaluations. They assume fixed job arrival times and have to run the system up to a significant number of jobs to reach a statistically meaningful conclusion. Therefore, such evaluations are time consuming and may be difficult to be applied in practice. To the best of our knowledge, the only exception is the convolution-based analysis in [16], in which Manolache et al. quantified the deadline miss rate analytically. However, their convolution-based analysis is only applicable for a very restrictive model, i.e., the tasks in the considered system are strictly periodic and scheduled under non-preemptive fixed-priority scheduling. Under these restrictions, the deadline miss rate can be derived by enumerating all possible transitions with convolution-based analyses. However, as recently shown in [8, 24], convolution-based analyses are computationally expensive when determining the deadline miss probability of a task. This implies that calculating the deadline miss rate with a convolution based approach suffers from the high computational complexity as well. A more detailed review of the related work can be found in Section II.

Overall, most existing approaches only focus on the probability of one deadline miss, but do not to provide any quantification on the deadline miss rate. An intuitive way is to deploy a simulation to exhaustively derive a tight but safe estimation of the deadline miss rate, i.e, close to the real value without under estimating it. However, as Butler and Finelli stated in [6], *Life-testing of ultrareliable software is infeasible*, i.e., the amount of time needed to perform the simulations is too large. Therefore, a statistical quantification that can efficiently derive the deadline miss rate is desired. To the best of our knowledge, this is the first paper providing a safe upper bound on the expected deadline miss rate of a specific task. Through extensive simulations, we evaluate the analysis time and the results of our approach, compared to our event-based simulator, under different fault rates.

Contributions: In this paper, we consider the deadline miss rate for soft real-time task system without rebooting after deadline misses. We demonstrate our approach considering soft errors incurred by transient faults. Our main contributions are listed as follows:

- To analyze the deadline miss rate, we show how jobs of a specific task in all possible schedules can be partitioned into busy intervals with different numbers of consecutive deadline misses in Section IV.
- In Section V we first show how this probability can be over-approximated. Afterwards, we discuss how to leverage on approaches in the literature that over-approximate the deadline miss probability of individual jobs to derive a safe upper bound on the expected deadline miss rate.
- To empirically gather deadline miss rates, we also implement an event-driven simulator with the distributed execution times among tasks in Section VI.
- Considering randomized task sets where each task has two distinct WCETs, we analyze the expected miss rates derived by our approach and compare them to the empirical deadline miss rates derived by the event-driven simulator under different settings for the system utilization and the fault rate in Section VII.

II. Related Work

Several results that based on probabilistic response-time analyses approximate the probability that the system has a deadline miss can be found in the literature [2, 3, 11, 19, 22]. For periodic real-time task systems, Diaz et al. [11] provided a framework to determine the deadline miss probability. Tanasa et al. [22] adopted the Weierstrass Approximation and applied a customized decomposition procedure to derive the deadline miss probability among all the possible combinations. However, both of them only work for small examples, i.e., 7 and 25 jobs in the hyper-period, respectively. For sporadic real-time task systems, Axer et al. [2] iterated over the activations of released jobs to evaluate the response-time distribution for non-preemptive fixed-priority scheduling. Maxim and Cucu-Grosjean [19] proposed a probabilistic response time analysis and probabilistic minimum inter-arrival times based on job-level convolution.

Recently, von der Brüggen et al. [24] proposed an analytical probabilistic response time analysis based on a task-level convolution. Ben-Amor et al. [3] provided a probabilistic response time analysis under precedence constraints. The concept of probabilistic scheduling analysis has been further extended to handle mixed-criticality real-time systems in [1, 18]. Chen and Chen [8] presented probabilistic schedulability tests to derive the probabilities of (ℓ-consecutive) deadline misses. Probabilistic methods based on the Hoeffding inequality and the Bernstein inequality were introduced by von der Brüggen et al. in [24].

Overall, the aforementioned studies all focus on the probability of deadline misses by finding the first deadline miss. Manolache et al. [16] presented a stochastic approach for obtaining the expected deadline miss rate analytically and addressed the problem of task priority assignment and task mapping in [17]. However, they consider only non-preemptive scheduling to reduce the computational complexity of their convolution based approach. Most studies in the literature [5, 12, 13, 15, 21] use the deadline miss rate as the performance metric to empirically evaluate their proposed approaches. To the best of our knowledge, this is the first paper analyzing the expected deadline miss rate without convolution and without very restricted system models.

III. System Model and Notation

In this section, we first review the task and scheduling models considered in this paper. Next, we introduce the distributed execution time model. Afterwards, the studied objective, i.e., the deadline miss rate, is formally defined.

A. Task and Scheduling Models

We consider a soft real-time task system with n independent periodic or sporadic tasks $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n\}$ in a uniprocessor system. We assume a preemptive fixed-priority scheduling policy where the priority of a task cannot be changed during runtime. The tasks are indexed from 1 to n where τ_1 has the highest priority and τ_n has the lowest priority.

Each task τ_i releases an infinite number of task instances, called jobs, under a minimum inter-arrival time constraint (or period) T_i, which specifies the minimum time between two consecutive job releases of τ_i. Each task is also associated with a relative deadline D_i. Therefore a job of task τ_i released at time t_a must be completed not later than the absolute deadline $t_a + D_i$ and the next job of task τ_i must be released exactly at (or not earlier than) $t_a + T_i$ for periodic (or sporadic) tasks. We consider: 1) *Implicit-deadline* task sets, i.e., $D_i = T_i\ \forall \tau_i \in \Gamma$, and 2) *Constrained-deadline* task sets, i.e., $D_i \leq T_i\ \forall \tau_i \in \Gamma$. Let $hp(\tau_k)$ be the set of tasks with higher priority than τ_k and let $hep(\tau_k)$ be $hp(\tau_k) \cup \{\tau_k\}$.

B. Distributed Execution Time Model

We consider each task τ_i has several (but finite) possible values of execution time $C_{i,j}$, and each of them is associated to a probability $P_{i,j}$. This model is often used in the literature to formalize systems with multiple possible execution times, e.g., [3, 8, 11, 19, 23].

For the simplicity of presentation, in this paper we only show one of the applicable cases, i.e., that the considered system uses software fault-tolerant techniques, where each task has two different worst case execution times with one given probability for each task. *The proposed approaches are still applicable for any general probabilistic distributions, since they are not limited to the number of execution times and corresponding probabilities.* We discuss how does this assumption affect our analyses in Section V-D.

We assume that soft errors induced by transient faults only affect the execution time of tasks in Γ without unnoticed faults, i.e., silent data corruption, or even system crash. Two distinct worst case execution times (WCETs) are assumed for each task τ_i. When no fault occurs during the execution of task τ_i and therefore error recovery is not necessary, the execution is considered to be a *normal* execution with a smaller WCET value, denoted as C_i^N. If a fault is detected in a job of task τ_i, the related job has a longer WCET denoted as C_i^A for potential error recovery, called an *abnormal* execution, i.e., $C_i^A \geq C_i^N\ \forall \tau_i$. The fault detection is assumed to perform perfectly and be done at predefined checkpoints or the end of a job execution. This additional computation time for the fault detection is integrated into C_i^N. The values of C_i^A and C_i^N depend on the adopted fault tolerance technique.

We assume that the occurrence of soft errors can be modeled by a given probability P_i^A, i.e., the probability is P_i^A that τ_i is executed abnormally. The probability of executing a job normally is thus $P_i^N = 1 - P_i^A$ for each job of τ_i. We assume that P_i^A is independent from previous errors and executions, as similar assumptions are used in [8, 11, 19, 23].

C. Deadline Miss Rate

For a schedule of a sequence of jobs of τ_k the deadline miss rate is formally defined as follows:

Definition 1 (Miss Rate): The miss rate of a task $\tau_k \in \Gamma$ for a given schedule S is the number of jobs missing their relative deadline D_k in S divided by the number of released jobs of task τ_k in S.

The expected miss rate of τ_k is defined as:

Definition 2 (Expected Miss Rate): The expected miss rate of a task $\tau_k \in \Gamma$, denoted by $\mathbb{E}_k$, is the probability that a job of τ_k misses its deadline.

Theoretically, the expected miss rate for a task τ_k can be determined by counting the number of jobs of τ_k that miss the deadline and the number of total releases in an infinitely long sequence of jobs, considering all tasks in Γ under the constraint given by Γ, the related scheduling algorithm, and the given fault rate. We focus on calculating a safe upper bound on the expected miss rate of a task τ_k, denoted as

$\hat{\mathbb{E}}_k$. By definition, $\hat{\mathbb{E}}_k \geq \mathbb{E}_k$. This means that we show how to over approximate the probability that a job of task τ_k misses its deadline, assuming that the probability P_i^A that a fault occurs during the execution of task τ_i is known for all tasks $\tau_i \in \Gamma$. This approximation is done based on the assumption that the system is never restarted when a deadline miss happens. In addition, we assume that jobs are never aborted, i.e., if a job misses its deadline, the remaining part of the job still has to be executed before the next job of the task can start executing. As discussed in [23], there are many reasons to apply this assumption in real contexts as long as the system safety is not jeopardized.

IV. Partition into Busy Intervals

To calculate the expected value of the deadline miss rate for a specific task τ_k, we first partition those deadline misses into different exclusive events for a schedule.

Definition 3 (Busy Interval of τ_k): An interval $[t_a, t_b]$ is a busy interval of task τ_k, if no job of τ_k is present in the system right before t_a, a job of τ_k arrives at t_a, a job of τ_k finishes at time t_b, no further job of τ_k or a task in $hp(\tau_k)$ is in the system right before t_b, and between t_a and t_b only jobs of τ_k or of jobs in $hp(\tau_k)$ are executed.

Please note that a busy interval for τ_k also ends at time t_b if a job of τ_k finishes at time t_b and one (or more) job of tasks in $hp(\tau_k)$ and/or of τ_k itself arrive at t_b, if no job of τ_k beside the one finishing at t_b is in the system right before t_b.

Considering constrained- and implicit-deadline task sets, a busy interval of τ_k can end due to different cases that are illustrated in Fig. 2. Let τ_1 be defined by $C_1^N = C_1^A = 1$, $T_1 = 2$, and $D_1 = 2$. Red boxes represent the execution of τ_1. For τ_2 assume $C_2^N = 1$, $C_2^A = 3$, $T_2 = 5$, and $D_2 = 5$ if it is not declared differently. Blue boxes with crosshatch patterns represent abnormal execution before the deadline of the related job, orange boxes with dot patterns represent abnormal execution after the deadline, and green boxes with slash patterns represent normal execution.

(1) Periodic tasks, implicit-deadline (Figure 2a): Busy intervals of τ_2 must always end with a job of τ_2 meeting its deadline. The first two jobs miss their deadline, the third job meets its deadline at 14.
(2) Sporadic tasks, implicit-deadlines (Figure 2b): In this case busy intervals can also end by the processor running idle with respect to $hep(\tau_2)$ as happening at time 6.
(3) Periodic tasks, constrained-deadline (Figure 2c): Let $C_2^A = 5$, $T_2 = 8, D_2 = 5$. Similar to the previous case, a busy interval for sporadic tasks with constrained deadlines can finish with a job meeting its deadline as happening at $t = 12$.

We partition the busy intervals of τ_k, depending on the number of jobs of τ_k that miss their deadlines:

Definition 4 (I_j Busy Interval of τ_k): A busy interval of τ_k is an I_j busy interval, if the first j jobs of τ_k in the busy interval miss their deadlines.

For this definition it does not matter, if the busy interval ends with a job of τ_k meeting its deadline or by the processor running idle with respect to tasks in $hep(\tau_k)$. Note that an I_0 busy interval does not contain any deadline misses. The probability of the occurrence for I_j is denoted as $\psi(I_j)$.

Theorem 1: For a given sequence of a sufficiently large number of jobs of τ_k, the jobs can be partitioned into busy intervals of τ_k with at most $\mathbb{J}$ jobs missing their deadline consecutively. All these busy intervals are independent from each other. Therefore, $\sum_{j=0}^{\mathbb{J}} \psi(I_j) = 1$.

Proof: As each busy interval either ends with a job of τ_k meeting its deadline or the processor running idle with respect to tasks in $hep(\tau_k)$, all the busy intervals are independent. If a job misses its deadline, it must be part of a busy interval of type I_j with at least one deadline miss, i.e., $j \geq 1$, with a known probability $\psi(I_j)$, since $D_k \leq T_k$. Otherwise, the probability that a job is meeting its deadline is $\psi(I_0)$. Since the intervals are independent, they are the probabilities for their occurrences. Therefore, $\sum_{j=0}^{\mathbb{J}} \psi(I_j) = 1$. ∎

V. Deadline Miss Rate

In this section, we first show how an upper bound on the expected deadline miss rate of task τ_k in a sporadic task set can be obtained based on the busy intervals of τ_k. While we explicitly analyze the miss rate for one task τ_k, the proposed approaches can be applied to each task in any given task sets.

According to Definition 2, the expected deadline miss rate $\mathbb{E}_k$ can be obtained by the following equation:

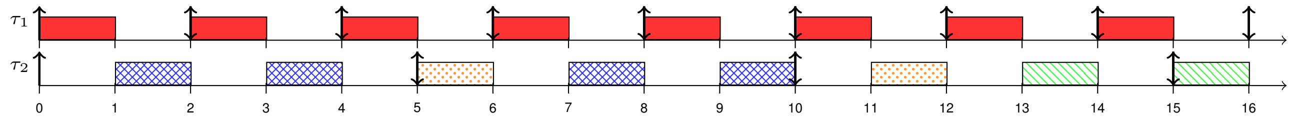
$$\mathbb{E}_k = \frac{\sum_{j=1}^{\mathbb{J}} \psi(I_j) \cdot j}{\sum_{j=1}^{\mathbb{J}} \psi(I_j) \cdot j + \psi(I_0)} \quad (1)$$

where $\psi(I_j)$ is the probability of the occurrence for I_j for task τ_k for any possible schedule. However, calculating the probability $\psi(I_j)$ precisely is an open problem. In this paper, we adopt an upper bound of $\psi(I_j)$ to over approximate Eq. (1).

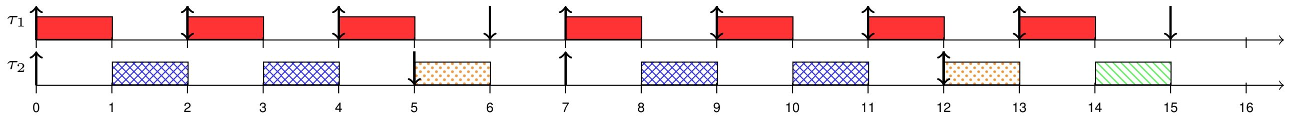
A. Upper Bound of $\psi(I_j)$

Let $\Phi_{k,j}$ be a safe upper bound on the probability that j (or more) consecutive jobs of τ_k miss their deadline. By definition, for $j \geq 1$, the probability $\psi(I_j)$ for an interval I_j with *exactly* j-consecutive deadline misses must be upper bounded by $\Phi_{k,j}$, i.e., $\psi(I_j) \leq \Phi_{k,j}$. Moreover, as the sequence of $\Phi_{k,j}$ is non-increasing for $j \geq 1$, $\Phi_{k,l}$ must be larger than or equal to the probability $\psi(I_j)$ for an interval I_j with *exactly* j-consecutive deadline misses if $l < j$. For instance, $\Phi_{k,1} \geq \psi(I_2)$.

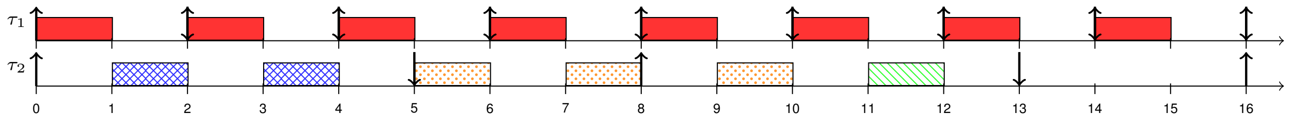
Several general approaches to calculate $\Phi_{k,j}$ are known from the literature. Job-level *Convolution-based* methods like in [16, 19] directly enumerate the WCET state space



(a) Periodic, implicit-deadline task sets: An I_2 interval of τ_2 with two consecutive deadline misses (at $t = 5$ and $t = 10$), followed by a job meeting its deadline at $t = 14$. At $t = 15$, a new busy interval of τ_2 starts.



(b) Sporadic, implicit-deadline task sets: Two I_1 intervals of τ_2, both ending with idle time at $t = 6$ and $t = 15$, respectively. While the first busy interval finishes with idle time, the second finishes with a job of τ_2 meeting its deadline.



(c) Periodic, constrained-deadline task sets: One I_1 interval ending at $t = 12$. It ends after the second job of τ_2 finishes at $t = 12$, meeting its deadline.

Figure 2: The possible scenarios for busy intervals. Red boxes represent the execution of τ_1. For τ_2, blue boxes with crosshatch patterns represent abnormal execution before the deadline of the related job, orange boxes with dot patterns represent abnormal execution after the deadline, and green boxes with slash patterns represent normal executions. The upward arrows represent the arrival time for each job, whereas the downward arrows represent the deadline for each job.

with the associated probabilities. Considering the jobs in non-decreasing order of arrival time, they convolute the current state of the system associated with a vector of possible states, i.e., possible total WCETs and related probability, with the current release jobs. By repeating this procedure until D_k is reached, all released jobs are convoluted, and the probability that the worst case response time is greater than D_k (at least one deadline miss) can be derived accordingly. In [24] a task-level convolution was proposed that evaluates the resulting deadline miss probability for a set of time points individually. Although these approaches focus on the probability of one deadline miss, i.e., $\Phi_{k,1}$, they can be extended to multiple deadline misses. For ℓ-consecutive deadline misses, instead of repeating until D_k, the procedure is repeated until $(\ell - 1)T_k + D_k$ is reached. Chen and Chen [8] provided an approach that can over-approximate the probability for ℓ-consecutive deadline misses and is based on the *Chernoff bound* and the *moment generating function* (*mgf*).

The approaches presented in [8, 16, 19, 24] all calculate the probability of deadline misses based on the probability that the workload S_t over a given interval of length t is larger than t. To be more precise, they evaluate if $P(S_t \geq t)$ for certain t values of interest, i.e., the deadline of τ_k and the release times of all jobs of higher priority tasks. If these probabilities are known for some or all of these values of interest, a safe upper bound on the probability for ℓ-consecutive deadline misses can be obtained by the following lemma:

Lemma 1: (Theorem 3 from Chen and Chen [8]) Given a set of constrained-deadline (or implicit-deadline) sporadic tasks Γ. Suppose that

$$\Phi^{\theta}_{k,w} = \min_{0<t\leq(w-1)T_k+D_k} P(S_t \geq t) \tag{2}$$

For notational brevity, let $\Phi_{k,0}$ be 1. The probability for ℓ-consecutive deadline misses of τ_k is upper bounded by

$$\Phi_{k,\ell} = \max\left\{\Phi^{\theta}_{k,w} \cdot \Phi_{k,\ell-w} | w \in \{1, 2, \dots \ell\}\right\} \tag{3}$$

Here we adopt the approach in [8] to demonstrate how to derive $P(S_t \geq t)$ and calculate Eq. (2). The approach in [8] uses the *mgf* [20] to express the probability distribution. For each task τ_i, its *mgf* with respect to a given real number s is

$$\text{mgf}_i(s) = \sum_{j=1}^{v_i} \exp(C_i^j \cdot\ s) \cdot\ P_i^j \tag{4}$$

Based on the assumption that all jobs are independent, the probability distribution over the sum of the execution times of these jobs can be defined as the multiplication of their *mgfs*

$$\text{mgf}_{hep(\tau_k)}(s) = \prod_{\tau_i \in hep(\tau_k)} (\text{mgf}_i(s))^{\rho_{i,t}} \tag{5}$$

where $\rho_{i,t}$ is the number of jobs from τ_i released in the interval from 0 to t. By definition, the workload S_t is the sum of the execution times from the jobs released by the tasks in $hep(\tau_k)$ from 0 to t. Hence, $P(S^t \geq t)$ can be calculated by

$$P(S^t \geq t) \leq \min_{s>0} \left(\frac{\text{mgf}_{hep(\tau_k)}(s)}{\exp(s \cdot t)} \right) \tag{6}$$

Eventually, testing over $(0, (w-1)T_k + D_k]$ with Eq.(6), Eq.(2) can be calculated accordingly.

Please note that while $\Phi_{k,\ell}$ as the upper bound of $\psi(I_j)$ can be calculated by all methods mentioned above, the job-level convolution-based analyses suffers from state explosion due to the large number of jobs and hence are not practicably applicable as shown in [8, 24]. Hence, either the task-level convolution presented by von der Brüggen et al. in [24] or the approach by Chen and Chen [8] should be applied.

B. Approximation of the Expected Miss Rate

To approximate Eq. (1), we assume that the safe bound on the probability for one deadline miss is greater than 0, i.e., $\Phi_{k,1} > 0$. Otherwise, the expected deadline miss rate is 0 trivially, i.e., $\mathbb{E}_k = 0$. A safe upper bound on the expected miss rate $\hat{\mathbb{E}_k}$ can be obtained by the following theorem:

Theorem 2 (Approximation of the Deadline Miss Rate): Suppose that we are given a schedule of a set of constrained-deadline (or implicit-deadline) sporadic tasks Γ where the largest number of consecutive deadline misses of τ_k in this schedule is $\mathbb{J}$. An upper bound on the expected deadline miss rate of task τ_k can be calculated as

$$\hat{\mathbb{E}_k} = \frac{1}{1 + \frac{1-\Phi_{k,1}}{\sum_{j=1}^{\mathbb{J}} \Phi_{k,j} \cdot j}} \tag{7}$$

where $\Phi_{k,1}$ and $\Phi_{k,j}$ are derived by Lemma 1.

Proof: By the above arguments for $\psi(I_j)$, we know $\psi(I_j) \leq \Phi_{k,j}$. For any possible schedule, the probability $\psi(I_0)$ that there is no job missing its deadline must be at least $1-\Phi_{k,1}$, i.e., $\psi(I_0) \geq 1-\Phi_{k,1}$, and $\sum_{j=1}^{\mathbb{J}} \psi(I_j) \cdot j > 0$, which means that

$$\frac{\psi(I_0)}{\sum_{j=1}^{\mathbb{J}} \psi(I_j) \cdot j} \geq \frac{1-\Phi_{k,1}}{\sum_{j=1}^{\mathbb{J}} \Phi_{k,j} \cdot j} \tag{8}$$

Therefore,

$$\begin{aligned} \mathbb{E}_k &= \frac{\sum_{j=1}^{\mathbb{J}} \psi(I_j) \cdot j}{\sum_{j=1}^{\mathbb{J}} \psi(I_j) \cdot j + \psi(I_0)} \\ &= \frac{1}{1 + \frac{\psi(I_0)}{\sum_{j=1}^{\mathbb{J}} \psi(I_j) \cdot j}} \leq \frac{1}{1 + \frac{1-\Phi_{k,1}}{\sum_{j=1}^{\mathbb{J}} \Phi_{k,j} \cdot j}} \\ &\leq \hat{\mathbb{E}_k} \end{aligned} \tag{9}$$

This concludes that $\mathbb{E}_k$ must be upper bounded by $\hat{\mathbb{E}_k}$, i.e., $\mathbb{E}_k \leq \hat{\mathbb{E}_k}$. ■

The following example illustrates how Eq. (1) and Eq. (7) can be used to calculate the expected miss rate.

Example 1: Suppose that the probabilities of I_j are given as: $\psi(I_0) = 0.99$, $\psi(I_1) = 0$, and $\psi(I_2) = 0.01$. With Eq. (1), we get the expected miss rate:

$$\frac{0.01 \cdot 2}{0.01 \cdot 2 + 0.99} = 0.0198$$

Let the safe bounds of them be given as follows: $\Phi_{k,1} = 0.05$, $\Phi_{k,2} = 0.02$, and $\Phi_{k,3} = 0$. With Eq. (7), a safe upper bound on the expected miss rate can be derived as:

$$\frac{1}{1 + \frac{0.95}{0.05+0.02\cdot 2}} = 0.0865$$

■

C. Threshold $\mathbb{J}'$ and Time Complexity

Since in Eq. (7) the maximum number of jobs with consecutive deadline misses $\mathbb{J}$ could be extremely large or even infinite, an additional approximation is required to bound the summation in Eq. (7). Let $S = \sum_{j=1}^{\infty} \Phi_{k,j} \cdot j$. We can use a threshold $\mathbb{J}'$ to simplify the procedure. Let $r_j = \frac{(j+1)\Phi_{k,j+1}}{j\Phi_{k,j}}$. Assume there is a $\mathbb{J}'$, such that $r_{\mathbb{J}'}$ is always larger than the other r_j, where $\mathbb{J}' < j$ and $0 < r_{\mathbb{J}'} < 1$. A safe upper bound on $\hat{\mathbb{E}_k}$ can be obtained by the following theorem:

Theorem 3: Suppose that we are given an index $\mathbb{J}'$ such that $r_{\mathbb{J}'}$ is always larger than any other r_j if $0 < \mathbb{J}' < j$. Let $\hat{S} = \sum_{j=1}^{(\mathbb{J}'-1)} j\Phi_{k,j} + \frac{\Phi_{k,\mathbb{J}'}}{1-r_{\mathbb{J}'}}$. An upper bound on the expected deadline miss rate of task τ_k can be calculated as

$$\hat{\mathbb{E}_k^*} = \frac{1}{1 + \frac{1-\Phi_{k,1}}{\hat{S}}} \tag{10}$$

Proof: Comparing to Eq. (7), Eq. (10) only replaces S with $\hat{S}$. Therefore we prove $\hat{S} \geq S$ as follows:

$$\begin{aligned} S &= \sum_{j=1}^{\infty} \Phi_{k,j} \cdot j \\ &= \sum_{j=1}^{(\mathbb{J}'-1)} j\Phi_{k,j} + \sum_{z=\mathbb{J}'}^{\infty} z\Phi_{k,z} \\ &= \sum_{j=1}^{(\mathbb{J}'-1)} j\Phi_{k,j} + \Phi_{k,\mathbb{J}'} + r_1\Phi_{k,\mathbb{J}'} + r_1 r_2 \Phi_{k,\mathbb{J}'} \ldots \\ &\leq \sum_{j=1}^{(\mathbb{J}'-1)} j\Phi_{k,j} + \sum_{z=0}^{\infty} (r_{\mathbb{J}'})^z \Phi_{k,\mathbb{J}'} \end{aligned} \tag{11}$$

As $\sum_{z=0}^{\infty} (r_{\mathbb{J}'})^z \Phi_{k,\mathbb{J}'}$ is an infinite series with a common ratio $r_{\mathbb{J}'}$, in which $0 < r_{\mathbb{J}'} < 1$, we can calculate Eq. (11) from the finite sum formula:

$$\hat{S} = \sum_{j=1}^{(\mathbb{J}'-1)} j\Phi_{k,j} + \frac{\Phi_{k,\mathbb{J}'}}{1-r_{\mathbb{J}'}}$$

Since $\hat{S} \geq S$, it follows directly that $\hat{\mathbb{E}_k^*} \geq \hat{\mathbb{E}_k}$. ■

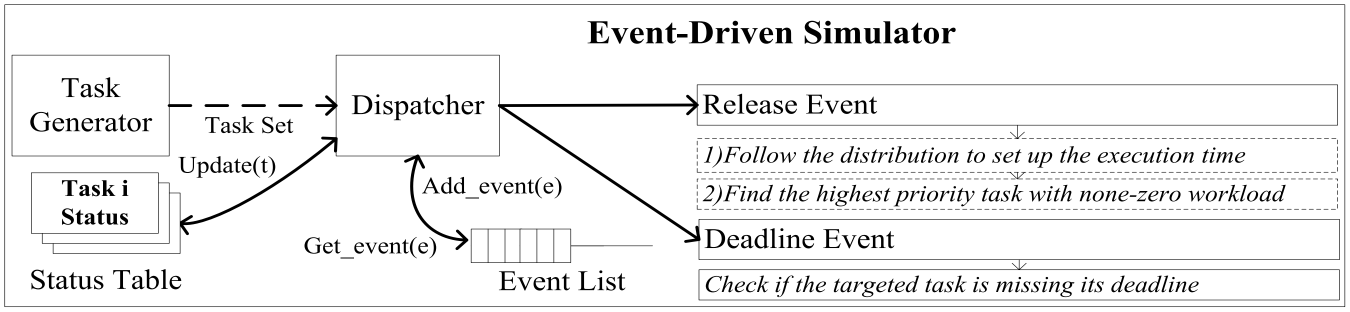

Event-Driven Simulator
Task Generator
Task Set
Dispatcher
Release Event
1)Follow the distribution to set up the execution time
2)Find the highest priority task with none-zero workload
Update(t)
Task i Status
Add_event(e)
Get_event(e)
Deadline Event
Check if the targeted task is missing its deadline
Status Table
Event List

Figure 3: Overview of the event-driven simulator released on [7].

The complexity of this approximation mainly comes from $\hat{S}$, which is dominated by the calculation of $j \cdot \Phi_{k,j}$ for $j = 1, 2, \dots, \mathbb{J}'$. According to Eq. (3), we need to calculate $\Phi^{\theta}_{k,w}$ in every iteration and use the derived $\Phi_{k,\ell-w}$ from the previous iterations. The space complexity to record all values $\Phi^{\theta}_{k,w}$ for $w = 1, 2, \dots, \mathbb{J}'$ is $O(\mathbb{J}')$. Suppose that the time complexity to determine if $P(S_t \geq t)$ for a given single time point t is $O(L)$. For each $\Phi^{\theta}_{k,w}$, the time complexity is $O(D_k \cdot L)$ with discretized time points t from 0 to D_k. Since $\Phi_{k,\ell}$ can be calculated by the derived $\Phi^{\theta}_{k,w}$ immediately, thus, the time complexity of $\hat{S}$ is $O(\mathbb{J}' \cdot D_k \cdot L)$ (if we only look at $\Phi_{k,\mathbb{J}'}$).

D. General Execution Time Distribution

In Section III-B we assume that each task has two distinct WCETs, i.e., C_i^A and C_i^N with their corresponding probability P_i^A and P_i^N. However, throughout this paper, this assumption is only used in the example in Section IV (as shown in Fig. 2) and the evaluation in Section VII. Therefore, there is in fact no impact on the proposed analyses if we relax this assumption since our method is agnostic to the method that is used to calculate the $\Phi_{k,\ell}$ values. We note, however, that the method chosen to calculate these values must be able to work with more general probability distributions to be applied, which, for instance, is the case for the methods proposed in [8, 24] that we consider in the evaluation.

VI. Event-Driven Simulator

To estimate the deadline miss rate empirically, we also implemented an event-driven simulator written in Python 2.7 to simulate the rate-monotonic scheduling policy. The complete scripts are available at [7]. An overview of the simulator is shown in Figure 3. For each task τ_i, there are only two type of events in the simulator, either `release` or `deadline`. A `release` event of τ_i adds its new workload to the entry of τ_i in the status table, and places a `deadline` event of τ_i into the event list. The `deadline` event of τ_i will check if the remaining workload of τ_i is zero in the status table (meet its deadline) or not (missing one deadline). The main components in the simulator are listed as follows:

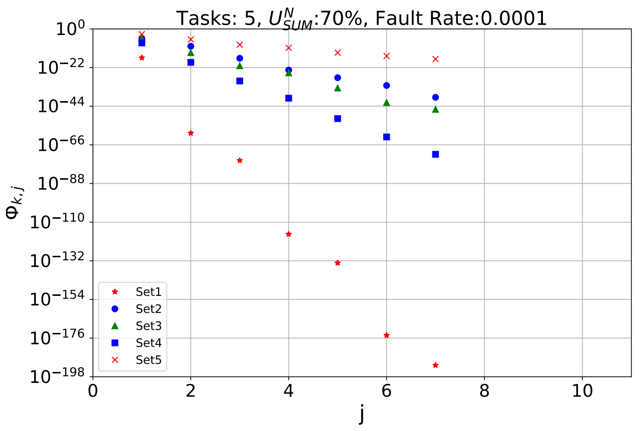
- **Task Generator:** By applying the well-known UUniFast method [4] and the suggestion from Davis et al. in [9], the task generator outputs a set of generated tasks under a given utilization value (see Section VII-A).
- **Dispatcher:** It checks if the number of released jobs from the targeted task (by default is the lowest priority task) is equal to the targeted number. If not, it continues to dispatch the next event from the event list.
- **Event List:** This linked list keeps track of the following events in the simulated task system. When a new event is inserted by another `release` event, the events in the list are sorted by their future occurring time.
- **Status Table:** It records the number of deadline misses, the number of released jobs, and the remaining workload for each unfinished job of a task.

According to the considered model, jobs are never aborted in the simulator. If a job misses its deadline, the remaining portion of execution time is still executed at the same priority level before the next job of the task can start executing. Whenever the dispatcher gets a new event, the workloads of the tasks in the status table are updated by the elapsed time from the previous event to the current event and the processor is assigned to the highest priority task under a fixed-priority scheduling policy (if any ready task is in the system; otherwise the processors runs idle until the next job is released).

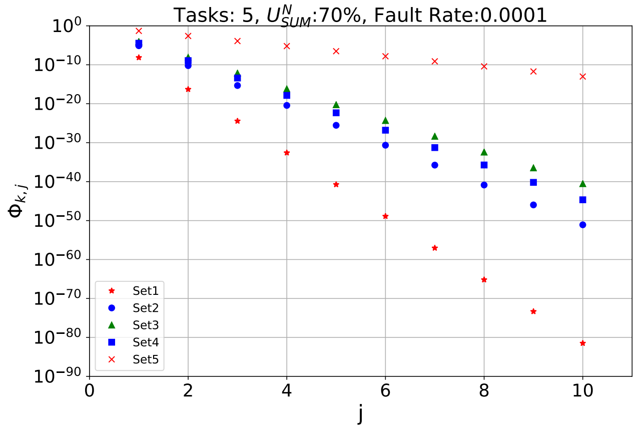
We implemented the task release to suit the evaluation performed in this paper by releasing tasks in two execution modes, either related to C_i^N with high probability P_i^N, or related to C_i^A with low probability P_i^A. This can be easily revised to fit any execution time distributions. We considered a preemptive fixed-priority scheduling policy in the dispatcher, i.e., the task with the highest priority with has non-zero workload will be executed. This dispatcher can also be extended to non-preemptive and dynamic-priority scheduling policies.

VII. Evaluation and Discussion

To evaluate the efficiency and the pessimism of our analysis, this section presents the derived bounds of the expected miss rates for different synthesized task sets. To analyze our



(a) $\Phi_{k,j}$ derived by CON method



(b) $\Phi_{k,j}$ derived by EMR method

Figure 4: Trends of $\Phi_{k,j}$ from five different task sets with respect to different j.

j	1	2	3	4	5
CON	0.02	0.36	2.58	11.15	36.26
EMR	0.91	3.50	8.45	18.50	39.33
j	6	7	8	9	10
CON	94.36	220.34	(Set 1:3276.27)	-	-
EMR	81.86	167.27	338.48	680.98	1366.56

Table I: Average time (sec) need for deriving $\Phi_{k,j}$.

approach, we used two different ways to determine $\Phi_{k,j}$, namely the analytical bounds from Chen and Chen [8][2], denoted as **EMR**, and the tighter task-level convolution-based approach presented by von der Brüggen et al. in [24], denoted as **CON**. Please note that the convolution-based approach in [19] leads to identical results as the task-level convolution-based method presented in [24] while it has (in general) a larger runtime than [24] and is therefore omitted. Moreover, we compared to the results derived from an event-driven simulator, introduced in Section VI, labeled as **SIM**.

A. *Simulation Setup*

We synthesized randomized implicit-deadline task sets with a given utilization value U^N_{sum} according to the UUniFast method [4] where $U^N_{\text{sum}} = \sum_{\tau_i \in \Gamma} U^N_i$ and $U^N_i = C^N_i / T_i$. To evaluate the impact of abnormal executions, we adopted time-demand analysis [14] to ensure the schedulability when all tasks are always executed normally, and discarded those task sets that were anyway not schedulable. To simulate a recovery mechanism, we set the error detection costs to 20% of the task execution time and assumed a complete re-execution if a fault is detected, i.e., C^A_i was set to $\frac{2.2}{1.2} \approx 1.83 \cdot C^N_i$ for all tasks $\tau_i \in \Gamma$ (similar to [23]). For each task set, the task periods were generated by a log-uniform distribution with two orders of magnitude, i.e., [1-100], according to the suggestion from Davis and Burns [9]. For each task, the normal execution time C^N_i was set to $U^N_i \cdot T_i$. Task sets with a cardinality of 2, 5 and 10 were considered. We used an identical fault rate P^A_i for all tasks. Due to space limitation, we only present the miss rates of the lowest-priority tasks.

For the configuration of the simulator, since we consider tasks with their specified minimum inter-arrival times (i.e., they are sporadic), it is unnecessary to release a higher-priority job if there is no unfinished job of a task under analysis. In order to fairly compare with our analytical approach considering the worst case, we enforced a worse release pattern in the simulator so that the estimated miss rate is closer to the worst case. Namely, if the lowest-priority task does not have any unfinished job at time t, a higher-priority task does not release any job even if its minimum inter-arrival time allows it to release a job at time t already. Instead, the next release of those higher-priority tasks is postponed to the point in time when the next job of τ_k is released.

Please note that our analyses should be carefully implemented, as the considered probabilistic values are very small. Due to the lack of precision in floating-point calculation, the terms in the commutative operations should be pre-sorted to avoid inconsistent results.

B. *Trends of $\Phi_{k,j}$ from CON and EMR*

Considering **CON** and **EMR**, we reported values of $\Phi_{k,j}$ for $j = 1, 2, \ldots, 10$. In all the generated task sets, the values of $\Phi_{k,j}$ decrease significantly with respect to j. The resulting values for 5 different task sets are presented in Figure 4. The average analysis time needed to gather $\Phi_{k,j}$ for these 5 task sets over different j by **CON** and **EMR** is presented in Table I. A timeout threshold of 30 minutes was used during the evaluation. Unfortunately, **CON** was not able to

[2]We adopt the original scripts from https://github.com/kuanhsunchen/EPST/ on March. 23 in 2018.

derive $\Phi_{k,j}$ for any task set with $j > 7$ before this threshold. When ignoring the threshold, **CON** needed around 55 mins to derive $\Phi_{k,8}$ for Set 1. This is due to the large number of jobs involved in the interval of analysis since the time complexity of **CON** is roughly the number of jobs of a task to the power of the number of tasks.

Generally, when j increases, $\Phi_{k,j}$ decreases drastically. When we consider the calculation of S, we can expect that the value contributed from $j \cdot \Phi_{k,j}$ for bigger j becomes negligible very soon. This observation supports our approximation in Section V-C. Since the derived upper bound of the expected miss rate is mainly dominated by those $\Phi_{k,j}$ with small j, such insignificant $\Phi_{k,j}$ values resulting from a large j and, hence, requiring significant analysis time can be sensibly ignored. We can observe that **EMR** needs a larger runtime for small j-values since the runtime in this case is dominated by finding a good internal parameter of the Chernoff bound, but the runtime increase with respect to j is not as significant as for **CON**. We set $\mathbb{J}'$ to 4 for $\hat{S}$ in the rest of evaluations to efficiently over-approximate the expected deadline miss rate $\hat{\mathbb{E}}_k$ for all the given task sets.

C. *Comparison among **SIM**, **CON** and **EMR***

To evaluate the efficiency of our analysis, we compared the derived upper bounds from **CON** and **EMR** to the estimated deadline miss rate from an event-driven simulator **SIM**. The cardinality of the task sets is 2, since deriving $\Phi_{k,j}$ using **SIM** to simulate the miss rates is really time consuming. The simulator stopped its simulation when two-million jobs from the lowest priority task finished their executions. We show the results for 20 different task sets where the expected miss rate derived from **SIM** was greater than 10^{-5} in Figure 5. The restriction regarding the displayed task sets is taken to increase the readability of the figure.

In Figure 5, the bounds derived from **CON** and **EMR** are all higher than the miss rates of **SIM**. This empirically shows that the derived upper bound from our analysis is safe. Although the bounds derived from **CON** in our setting are relatively close to the simulated miss rate from **SIM**, **CON** does not scale well with respect to the number of tasks in the system, especially when calculating $\hat{S}$. The bounds derived from **EMR** are generally greater than the estimated bound from **CON** by two orders of magnitude. However, it is more scalable than **CON** and **SIM**. Hence, this tradeoff has to be considered when applying our approach. If the error resulting from the pessimism of the Chernoff bounds is acceptable, **EMR** is still a good choice.

D. *Expected Miss Rate with More Tasks*

Since the variation of the derived miss rates is significant even under the same utilization setting and fault rate, we choose box plots to present the results, i.e., shwoing the medians (red lines), the interquartile range of the sample (the width of the boxes), the maximums (top lines), and the minimums (bottom lines). We recorded the miss rates of 100 synthesized task sets for each configuration. Unfortunately, **CON** could not obtain $\hat{\mathbb{E}}_k$ in an acceptable amount of time even with $\mathbb{J}' = 4$, i.e., 30 minutes per task set, whereas **EMR** could obtain the result for each task set in, on average, 2 minutes. Therefore, we only present the results $\hat{\mathbb{E}}^*_k$ while $\Phi_{k,j}$ are derived by **EMR**, i.e., Chen and Chen's method [8], for the cardinality with 10 tasks. As shown in Fig. 6, naturally the derived bounds are less if the fault rates are lower; the derived bounds are higher if U^N_{sum} is higher. The trends of results derived using different values of U^N_{SUM} are similar to the results presented in Fig. 6.

VIII. Conclusion

As soft real-time systems tolerate and allow certain deadline misses, the deadline miss rate is an important performance indicator to evaluate the proposed analyses, scheduling algorithms, etc. In this paper, we propose approaches to safely approximate the deadline miss rates. To the best of our knowledge, this is the first paper analyzing the expected deadline miss rate in general task and scheduling models.

Although we only show one applicable case in the evaluation, i.e., tasks with two distinct execution times in the evaluation, the analyses in this paper can be applied for any execution time distributions as long as $\Phi_{k,j}$ can be derived. In future work, we plan to mitigate the pessimism of our approximations, and extend the applicability of our analyses for more general real-time task models.

Acknowledgments

This work has been supported by Deutsche Forschungsgemeinschaft (DFG), as part of the priority program "Dependable Embedded Systems" - SPP1500 Get-SURE, and the Collaborative Research Center SFB 876 (http://sfb876.tu-dortmund.de/), subproject B2.

References

[1] Y. Abdeddaïm and D. Maxim. Probabilistic schedulability analysis for fixed priority mixed criticality real-time systems. In *Design Automation and Test in Europe (DATE)*, 2017.

[2] P. Axer and R. Ernst. Stochastic response-time guarantee for non-preemptive, fixed-priority scheduling under errors. In *Design Automation Conference (DAC)*, 2013.

[3] S. Ben-Amor, D. Maxim, and L. Cucu-Grosjean. Schedulability analysis of dependent probabilistic real-time tasks. In *24th International Conference on Real-Time Networks and Systems (RTNS)*, 2016.

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

[5] S. A. Brandt, S. Banachowski, C. Lin, and T. Bisson. Dynamic integrated scheduling of hard real-time, soft

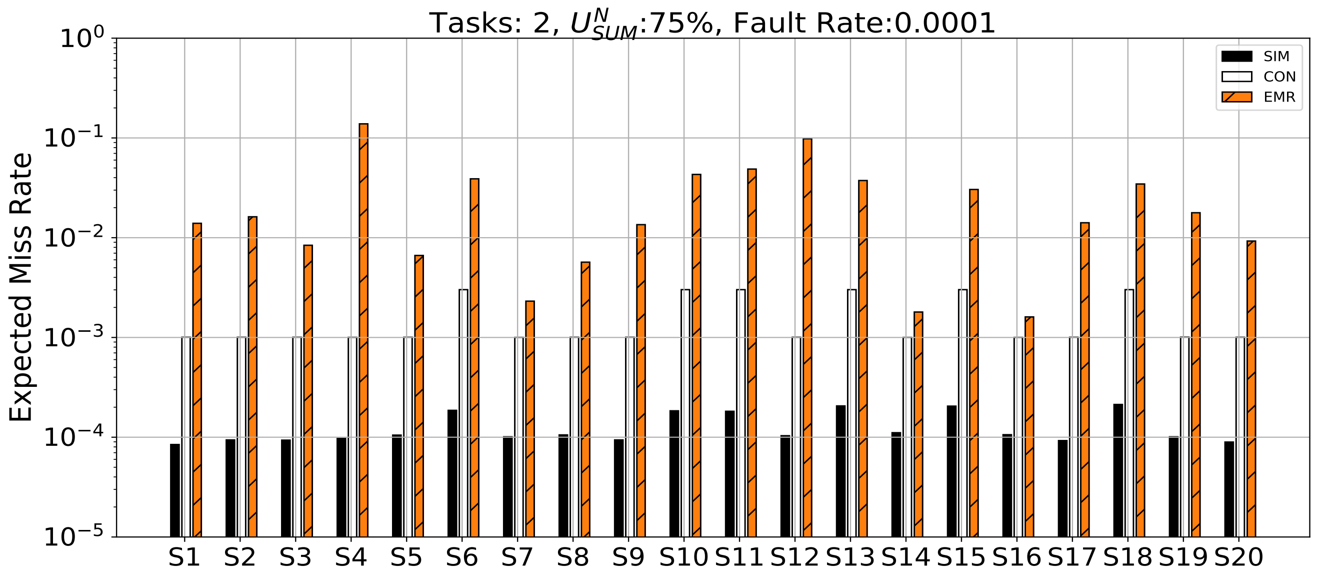
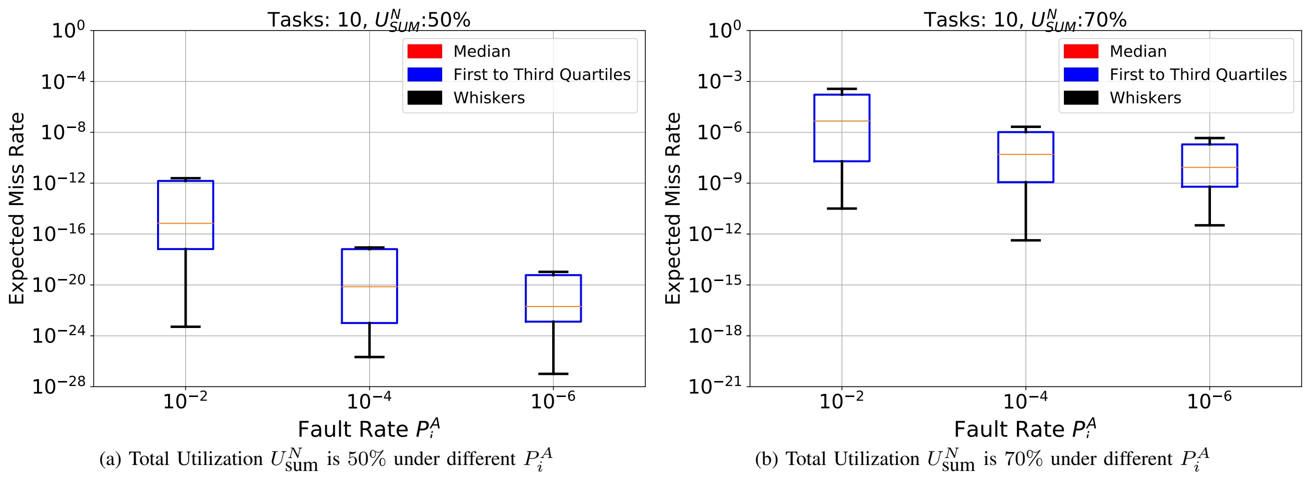


Figure 5: Comparison among SIM, CON, and EMR with 20 random task sets.



(a) Total Utilization U^N_{sum} is 50% under different P^A_i

(b) Total Utilization U^N_{sum} is 70% under different P^A_i

Figure 6: Expected Miss Rate $\hat{\mathbb{E}}_k$ derived from EMR for 10 tasks under different U^N_{sum} and different fault rates P^A_i.

real-time, and non-real-time processes. In *Real-Time Systems Symposium (RTSS)*, 2003.

[6] R. W. Butler and G. B. Finelli. The infeasibility of quantifying the reliability of life-critical real-time software. *IEEE Transactions on Software Engineering*, 19(1):3–12, Jan 1993.

[7] K.-H. Chen. Event-based Miss Rate Simulator. https://github.com/kuanhsunchen/MissRateSimulator/, 2018.

[8] K.-H. Chen and J.-J. Chen. Probabilistic schedulability tests for uniprocessor fixed-priority scheduling under soft errors. In *12th IEEE International Symposium on Industrial Embedded Systems*, 2017.

[9] R. I. Davis, A. Zabos, and A. Burns. Efficient exact schedulability tests for fixed priority real-time systems. *IEEE Trans. Computers*, 57(9):1261–1276, 2008.

[10] U. Devi and J. Anderson. Tardiness bounds under global EDF scheduling on a multiprocessor. In *Real-Time Systems Symposium (RTSS)*, 2005.

[11] J. L. Diaz, D. F. Garcia, K. Kim, C.-G. Lee, L. L. Bello, J. M. Lopez, S. L. Min, and O. Mirabella. Stochastic analysis of periodic real-time systems. In *23rd IEEE Real-Time Systems Symposium (RTSS)*, 2002.

[12] L. He, S. A. Jarvis, D. P. Spooner, H. Jiang, D. N. Dillenberger, and G. R. Nudd. Allocating non-real-time and soft real-time jobs in multiclusters. *IEEE Transactions on Parallel and Distributed Systems*, 17(2):99–112, 2006.

[13] B. Kao and H. Garcia-Molina. Subtask deadline assignment for complex distributed soft real-time tasks. In *14th International Conference on Distributed Com-*

puting Systems, 1994.

[14] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: exact characterization and average case behavior. In *Real Time Systems Symposium, Proceedings.*, pages 166–171, Dec 1989.

[15] C. Lin and S. A. Brandt. Improving soft real-time performance through better slack reclaiming. In *Real-Time Systems Symposium (RTSS)*, 2005.

[16] S. Manolache, P. Eles, and Z. Peng. Schedulability analysis of applications with stochastic task execution times. *ACM Trans. Embed. Comput. Syst.*, 3(4), 2004.

[17] S. Manolache, P. Eles, and Z. Peng. Task mapping and priority assignment for soft real-time applications under deadline miss ratio constraints. *ACM Trans. Embed. Comput. Syst.*, 2008.

[18] A. Masrur. A probabilistic scheduling framework for mixed-criticality systems. In *Design Automation Conference (DAC)*, 2016.

[19] D. Maxim and L. Cucu-Grosjean. Response time analysis for fixed-priority tasks with multiple probabilistic parameters. In *Real-Time Systems Symposium (RTSS), IEEE 34th*, pages 224–235, 2013.

[20] M. Mitzenmacher and E. Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[21] J. A. Stankovic, C. Lu, S. H. Son, and G. Tao. The case for feedback control real-time scheduling. In *Real-Time Systems, 1999. Proceedings of the 11th Euromicro Conference on*, 1999.

[22] B. Tanasa, U. D. Bordoloi, P. Eles, and Z. Peng. Probabilistic response time and joint analysis of periodic tasks. In *27th Euromicro Conference on Real-Time Systems*, pages 235–246, July 2015.

[23] G. v. d. Brüggen, K. H. Chen, W. H. Huang, and J. J. Chen. Systems with dynamic real-time guarantees in uncertain and faulty execution environments. In *IEEE Real-Time Systems Symposium (RTSS)*, 2016.

[24] G. v. d. Brüggen, N. Piatkowski, K.-H. Chen, J.-J. Chen, and K. Morik. Efficiently approximating the probability of deadline misses in real-time systems. In *2018 30th EUROMICRO Conference on Real-Time Systems (ECRTS)*, 2018.