Probabilistic Schedulability Tests for Uniprocessor Fixed-Priority Scheduling under Soft Errors

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

Citation: 10.1109/SIES.2017.7993392

BibTeX:

```
@inproceedings{DBLP:conf/sies/ChenC17,
  author    = {Kuan{-}Hsun Chen and
               Jian{-}Jia Chen},
  title     = {Probabilistic schedulability tests for uniprocessor fixed-priority
               scheduling under soft errors},
  booktitle = {12th {IEEE} International Symposium on Industrial Embedded Systems,
               {SIES} 2017, Toulouse, France, June 14-16, 2017},
  pages     = {1--8},
  publisher = {{IEEE}},
  year      = {2017},
  url       = {https://doi.org/10.1109/SIES.2017.7993392},
  doi       = {10.1109/SIES.2017.7993392},
}
```

Probabilistic Schedulability Tests for Uniprocessor Fixed-Priority Scheduling under Soft Errors

Kuan-Hsun Chen and Jian-Jia Chen
Department of Informatics, TU Dortmund University, Germany
Email: {kuan-hsun.chen, jian-jia.chen}@tu-dortmund.de

***Abstract*—Due to rising integrations, low voltage operations, and environmental influences such as electromagnetic interference and radiation, transient faults may cause soft errors and corrupt the execution state. Such soft errors can be recovered by applying fault-tolerant techniques. Therefore, the execution time of a job of a sporadic/periodic task may differ, depending upon the occurrence of soft errors and the applied error detection and recovery mechanisms. We model a periodic/sporadic real-time task under such a scenario by using two different worst-case execution times (WCETs), in which one is with the occurrence of soft errors and another is not. Based on a probabilistic soft-error model, the WCETs are hence with different probabilities. In this paper, we present efficient probabilistic schedulability tests that can be applied to verify the schedulability based on probabilistic arguments under fixed-priority scheduling on a uniprocessor system. We demonstrate how the Chernoff bounds can be used to calculate the task workloads based on their probabilistic WCETs. In addition, we further consider how to calculate the probability of ℓ-consecutive deadline misses of a task. The pessimism and the efficiency of our approaches are evaluated against the tighter and approximated convolution-based approaches, by running extensive evaluations under different soft-error rates. The evaluation results show that our approaches are effective to derive the probability of deadline misses and efficient with respect to the needed calculation time.**

I. Introduction

Due to rising integrations, low voltage operations, and environmental influences such as electromagnetic interference and radiation, transient faults may occur and flip a bit in the underlying hardware [3], which may corrupt the execution state or cause soft errors. Depending upon types and locations, a transient fault may severely affect the system behavior or even lead to system failures. Such transient faults can be resolved by adding additional circuits in the hardware or by applying software based fault tolerance techniques, e.g., Re-execution, Checkpoint-Based recovery, Dual Modular Redundancy (DMR), and Triple Modular Redundancy (TMR) [9, 12, 18]. However, the correctness of the system behavior depends not only on the functional correctness, but also on the timeliness. Even if all the transient faults can be handled properly, the additional computation incurs an execution time overhead in most of the cases if the additional circuit overhead is prohibited.

One natural assumption when applying fault-tolerant software techniques is that *the system functions normally most of time*, which aligns with the safety standards in the industry requiring low (or very low) probability of failure (e.g., due to deadline misses) such as IEC-61508 [13] and ISO-26262 [14]. Therefore, it is meaningful to use probabilistic worst-case execution times (WCETs) to model the execution of a task depending upon the occurrence of soft errors. This allows the system designer to provide probabilistic arguments based on the occurrence of error recovery. Otherwise, the system designer has to take the worst-case execution time (by assuming that the recovery always takes place) in the response time analysis, which can be very pessimistic.

To deal with such scenarios using probabilistic WCETs, probabilistic response-time analyses, e.g., [2, 11, 16], can be applied. Diaz et al. [11] developed a framework for calculating the deadline miss probability, but it only works for small examples as reported in [11] (e.g., 2 tasks with 7 jobs in the hyper-period), which is practically impossible to apply for generating the entire response time distribution. Axer et al. [2] proposed to evaluate the response-time distribution and iterate over the activations of job releases for non-preemptive fixed-priority scheduling. Maxim et al. [16] provided a probabilistic response time analysis by assuming probabilistic minimum inter-arrival times and probabilistic worst-case execution times on the fixed-priority scheduling policy. Tanasa et al. [19] developed a mathematical way to approximate the calculation of probabilistic response time distributions.

Overall, the aforementioned approaches are all based on convolution operations to calculate the probability density functions, indicating exponential time complexity. Naturally, they are computationally expensive to be applied when the number of tasks or jobs is large. Several approximation techniques like re-sampling [16], dynamic-programming with user-defined granularity, etc. may reduce the time complexity by introducing manual configurations. However, there is no fundamental analysis, to the best of our knowledge, to determine the required approximation needed in the above methods for balancing the trade-off between the accuracy and complexity. From our experiments, we note that the resulting error may be significant or the analysis is still not able to finish even with the approximation techniques (in Section VI). Recently, Xu et al. in [20] presented how to obtain the maximum number of deadline misses in a given time window. However, their method does not provide the probability of deadline misses.

In this paper, we alternatively provide efficient schedulability tests by using Chernoff bounds[1] and the moment-generating function (*mgf*) [17] (in Section IV) to calculate the probability of deadline misses without using any convolution. We notice that the convolution-based analyses are not necessary when our interest is a safe upper bound on the probability. Since our Chernoff-bound-based analyses are more efficient (at a price of less accuracy of the probability), we can further extend to handle more general cases for ℓ-consecutive deadline misses. To the best of our knowledge, our paper is the first work providing the upper bound on the probability of consecutive deadline misses of a specific task. Please note that, our approaches are not better in terms of accuracy of the probabilistic arguments, but they are essentially much faster than the convolution-based approaches and have better applicability. In the evaluation, we present how pessimistic and efficient the derived bound is, comparing to the tighter and approximated convolution-based approaches in [16].

Our Contributions: Instead of striving for a probabilistic response time analysis, in this paper, we intend to provide efficient and sufficient (probabilistic-based) schedulability tests under fixed-priority scheduling to obtain the probability of deadline misses of a given task τ_k. We further consider the probability of ℓ-consecutive deadline misses. Our contributions are as follows:

- Based on an assumption that the execution time of a job is independent from other jobs, we adopt the moment-generating functions and the Chernoff bounds as our backbone to safely bound the probability that the released workload in a specified time interval cannot finish in the given time interval in Section III.
- Based on the above backbone, we develop two probabilistic analyses to evaluate the upper bounds on the probability of deadline misses and ℓ-consecutive deadline misses for any positive integer ℓ in Section IV.
- We show the relationship between the upper bound on the probability of ℓ-consecutive deadline misses and the specified error rate by conducting extensive simulations based on synthesized task sets under different configurations. The pessimism and the efficiency of our approaches are also evaluated by comparing with the tighter and approximated convolution-based approaches [16] in Section VI.

II. System Model and Notation

In this section, we first review the task and scheduling models adopted in this paper. After that, the error handling model is introduced and the studied problem is defined.

A. Task and Scheduling Models

Given a set of n independent sporadic tasks $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n\}$ in a uniprocessor system, each task τ_i has a minimum inter-arrival time (or period) T_i, which specifies the minimum time between two consecutive job releases of τ_i, and a deadline D_i, which specifies the relative deadline of each job of task τ_i. That is, a job of task τ_i released at time t_a must be finished before or at time $t_a + D_i$ and the next job of task τ_i must be released at or after time $t_a + T_i$. Each task has two different worst case execution times $C_i^A \geq C_i^N$ depending on the occurrence of soft errors that are explained in detail in Section II-B. In this paper, we consider two types of task sets: 1) The relative deadline D_i of each task τ_i is equal to its period T_i, i.e., $D_i = T_i \; \forall \tau_i \in \Gamma$, so-called *implicit-deadline* task sets, or 2) $D_i \leq T_i \; \forall \tau_i \in \Gamma$, so-called *constrained-deadline* task sets.

Throughout this paper, we assume a preemptive fixed-priority scheduling policy where the priority of a task cannot be changed during runtime. This policy is widely used in the industrial practice and is supported in most real-time operating systems. The tasks in this paper are indexed from 1 to n; where τ_1 has the highest priority and τ_n has the lowest one.

B. Error Handling Model

To handle soft errors induced by transient faults, we assume that the tasks can be protected by certain fault tolerance techniques, so that soft errors may only affect the execution time without silent data corruption or even system crash. Two different worst case execution times (WCETs) are assumed for a task. In an abstract view, if there is no fault occurred during the execution of task τ_i, the execution is a *normal* execution with a smaller WCET denoted as C_i^N. If there is a fault detected, this job of task τ_i has a longer WCET denoted as C_i^A for potential error recovery for an *abnormal* execution, i.e., $C_i^A \geq C_i^N$. In this model, we assume that the fault detection is done at predefined checkpoints or the end of a job execution. This means, the overhead of the fault detection is part of C_i^N. These values can be specified depending upon the used fault tolerance techniques. We assume the occurrence of soft errors can be modeled by a given probability $\mathbb{P}_i^A$, which aligns with the probability of task τ_i executing in abnormal executions. We use the same assumption in [2] that the probability $\mathbb{P}_i^A$ is independent of previous errors and executions.

For the simplicity of presentation, we only consider two different WCETs for each task throughout the paper, but the proposed approaches are applicable for any general probabilistic distributions. We will explain how this assumption can be relaxed in Section V.

C. Probabilistic Deadline Misses

The proposed probabilistic analyses are designed to provide a probabilistic guarantee for a specific task τ_k that calculates the upper bound on the probability of deadline misses based on probabilistic WCETs. The upper bound probability of deadline misses is defined as follows:

Definition *The probability of deadline misses of task τ_k, denoted by Φ_k, is an upper bound on the probability that a job of task τ_k is not finished before its (relative) deadline D_k.*

In addition to the probability of deadline misses, we also consider the upper bound on the probability of ℓ-consecutive

[1]Chernoff bounds could give upper limits on the probability that a sum of random variables is greater or smaller than a given value.

deadline misses. We use Φ_k to $\Phi_{k,\ell}$ to represent the case for ℓ-consecutive deadline misses in Section IV-B.

III. Backbone of Our Analyses

To calculate the probability of deadline misses of task τ_k, we analyze the higher-priority workload released from 0 to t, for certain specified $t > 0$. Suppose that $hp(\tau_k)$ is the set of tasks with higher priority than τ_k and $hep(\tau_k)$ is $hp(\tau_k) \cup \{\tau_k\}$. For a task τ_i in $hp(\tau_k)$, suppose that $\rho_{i,t}$ jobs are released to interfere task τ_k up to time t, detailed in Section IV. We also define $\rho_{k,t}$ as the number of jobs of task τ_k in this analysis window.

The workload of the jobs released by the tasks in $hp(\tau_k)$ and task τ_k from 0 to t is the sum of $\rho_{k,t} + \sum_{\tau_i \in hp(\tau_k)} \rho_{i,t}$ job's execution time. In this paper, we use the moment generating function (*mgf*) of a random variable [17], which is an alternative specification of its probability distribution.

Since there are two alternative WCET values, i.e., $\{C_i^A, C_i^N\}$ for task τ_i in our task model, the *mgf* of the (worst-case) execution time of task τ_i with respect to a given real number s is

$$\mathrm{mgf}_i(s) = \exp(C_i^A \cdot\ s) \cdot\ \mathbb{P}_i^A + \exp(C_i^N \cdot\ s) \cdot\ (1 - \mathbb{P}_i^A) \quad (1)$$

In fact, the *mgf* can be used for any execution time distributions. Assume each task τ_i has v_i number of (but finite) possible values of execution time C_i^j, and each of them is associated to a probability $\mathbb{P}_i^j$. Eq.(1) can be generalized to

$$\mathrm{mgf}_i(s) = \sum_{j=1}^{v_i} \exp(C_i^j \cdot\ s) \cdot\ \mathbb{P}_i^j \quad (2)$$

For the simplicity of presentation, we only use Eq.(1) in the rest of the paper.

Since the execution times of the jobs under considerations are all independent, the probability distribution of the sum of the execution time of these jobs can be defined as the multiplication of their *mgfs*. Therefore, the distribution of the sum of the (worst-case) execution time of these jobs is

$$\mathrm{mgf}_{hep(\tau_k)}(s) = \prod_{\tau_i \in hep(\tau_k)} (\mathrm{mgf}_i(s))^{\rho_{i,t}}. \quad (3)$$

Suppose that S^t is the sum of the (worst-case) execution times of these jobs in $hep(\tau_k)$ from time 0 to time t. We are interested in deriving the probability for the case $S^t \geq t$. This can be derived by using the above *mgf*, and can be approximated by using the Chernoff bounds as follows:

Lemma 1: Suppose that S^t is the sum of the execution times of the $\rho_{k,t} + \sum_{\tau_i \in hp(\tau_k)} \rho_{i,t}$ jobs in $hep(\tau_k)$.

$$\mathbb{P}(S^t \geq t) \leq \min_{s>0} \left(\frac{\mathrm{mgf}_{hep(\tau_k)}(s)}{\exp(s \cdot t)} \right). \quad (4)$$

Proof From Lemma 2.9 in [6] and Pages 63-65 in [17], for a random variable X defined by a moment generating function $\mathrm{mgf}(s)$ for $s > 0$, the definition of Chernoff bounds is:

$$\mathbb{P}(X \geq t) \leq \mathrm{mgf}(s)/\exp(s \cdot t), \forall s > 0. \quad (5)$$

Therefore, a safe upper bound on the probability that $S^t \geq t$ under the moment generating function defined in Eq. (3) is equivalent to Eq. (4).

With Eq. (4), we can obtain the upper bound on the probability that the total released workload from $hep(\tau_k)$ is not able to finish at time point t. Although Chernoff bounds are pessimistic, it is the key of this paper to give the sufficient test in a quick manner.

IV. Probabilistic Tests

Here we first show how to obtain a probabilistic upper bound Φ_k for a specific task τ_k based on the condition in Eq. (4) and propose a k-point sufficient test. After that, we further consider how to obtain an upper bound on the probability of ℓ-consecutive deadline misses. Throughout this section, we explicitly focus our attention on the analyses of one task τ_k. Such analyses could be applied to each task in any given task sets.

A. Probability of Deadline Misses

Traditionally, if we are only interested in validating the schedulability for constrained-deadline sporadic task sets in uniprocessor systems in the worst case, the well-known *time-demand analysis* (TDA) [15] can be applied. That is, if

$$\exists\ t | 0 < t \leq D_k, \qquad C_k + \sum_{\tau_i \in hp(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq\ t, \quad (6)$$

then task τ_k is schedulable under the fixed-priority scheduling algorithm, where C_k and C_i in our model can be C_k^A and C_i^A, respectively. The following theorem shows that we can safely extend the above test in Eq. (6) to a probabilistic version and apply the Chernoff bounds in Eq. (4) to calculate a safe upper bound on the probability of deadline misses.

Theorem 1: Assume a given set of constrained-deadline (or implicit-deadline) sporadic tasks Γ.

1) If the condition in Eq. (6) holds, then the probability of deadline misses of task τ_k is 0.
2) Otherwise, the probability of deadline misses of task τ_k is upper bounded by Φ_k, defined as follows:

$$\Phi_k = \min_{0<t\leq D_k} \mathbb{P}(S^t \geq t), \quad (7)$$

where $\mathbb{P}(S^t \geq t)$ is derived by taking the right-hand side from Eq. (4) under $\rho_{i,t} = \left\lceil \frac{t}{T_i} \right\rceil$ for each task τ_i in $hp(\tau_k)$ and $\rho_{k,t} = 1$.

Proof The first assertion based on TDA holds based on the worst-case arguments [15]. We only prove the second assertion. We first prove why testing t in the range $(0, D_k]$ for τ_k is sufficient. Since the jobs of τ_k are only preempted by higher-priority jobs in $hp(\tau_k)$ and preempts any lower-priority jobs, we can safely remove any lower-priority jobs and only take $\tau_1, \ldots, \tau_k$ into consideration. Suppose a job of task τ_k is ready at time t' with completion time t_R, in which $t_R - t' > D_k$. Let t_{-1} be the latest instant before t_R, at which 1) either the processor idles at time t_{-1} or 2) all the

$t \in \mathbf{L}_k$	10	20	30	40	45	50	60	70	75
$\arg_{s>0}\min(\mathbb{P}(S^t \geq t))$	5.4999	5.4799	3.72	0.6214	0.6358	4.9155	0.6483	0.711	0.7216
$\min_{s>0}(\mathbb{P}(S^t \geq t))$	1.0	1.0	1.0	0.1041	0.05551	1.0	0.02921	0.00049	0.00024

TABLE I: Corresponding probabilities of deadline misses on all discretized points t in $\mathbf{L}_k$ gathered from Lemma 2.

jobs of task τ_k released *strictly* before t_{-1} have finished their executions. That is, from t_{-1} to t_R, the processor executes only the jobs of task τ_k and $hp(\tau_k)$ that are released after or at t_{-1}. Such a time point t_{-1} always exists, i.e., the starting time of the system.

Now, we remove all the jobs executed before t_{-1} from the schedule. The new schedule from t_{-1} to t_R is the same as the original schedule, in which only jobs arrived at or after t_{-1} are executed. It is possible that there are multiple jobs of task τ_k executed in the time interval $[t_{-1}, t_R)$. We consider two cases:

- **Case 1**, there is only one job of task τ_k executed in the time interval $[t_{-1}, t_R)$: We can move the release of the job of task τ_k from t' to t_{-1}. The response time of the job of task τ_k is not decreased.
- **Case 2**, there are at least two jobs of task τ_k executed in the time interval $[t_{-1}, t_R)$: By the definition that the schedule is busy for executing either task τ_k or $hp(\tau_k)$ in $[t_{-1}, t_R)$, the response time of the first job of task τ_k executed in this window must be greater than T_k. We can move the release time of this first job of task τ_k to t_{-1} as well. The response time of the first job of task τ_k in the time interval $[t_{-1}, t_R)$ is still greater than T_k.

In short, in both cases above, we can safely consider that task τ_k releases a job at time t_{-1}. In the first case, the deadline misses happen when the accumulated workload (sum of the requested execution time of the jobs released by τ_k and $hp(\tau_k)$) executed from t_{-1} to $t_{-1}+t$ is greater than t for any $0 < t \leq D_k$. In the second case, the deadline misses happen when the accumulated workload from t_{-1} to $t_{-1}+t$ is greater than t for any $0 < t \leq T_k$. By the assumption $D_k \leq T_k$, the probability that the accumulated workload executed from t_{-1} to $t_{-1}+t$ is greater than or equal to t for any $0 < t \leq D_k$ is a safe upper bound of the probability of deadline misses.

For notational brevity, let t_{-1} be 0. There are at most $\left\lceil \frac{t}{T_i} \right\rceil$ jobs of task τ_i and one job of task τ_k released from time 0 to time t for any $0 < t \leq D_k$. When a job of task τ_k misses its deadline, by the above analysis, we can safely take $\left\lceil \frac{t}{T_i} \right\rceil$ jobs of task τ_i and one job of task τ_k and evaluate the sum of their execution times up to time t. As a result the condition in Eq. (7) provides an upper bound on the probability of deadline misses of task τ_k. □

However, there is an infinite number of points in the interval $(0, D_k]$ in Eq. (7). The following lemma shows that it is sufficient to test only a pseudo-polynomial number of time points:

Lemma 2: Let $\mathbf{L}_k$ be a set of time interval lengths, where $\mathbf{L}_k = \{r \cdot T_i | \tau_i \in hp(\tau_k); r = 1, \ldots, \lfloor D_k/T_i \rfloor\} \cup \{D_k\}$. The upper-bounded probability Φ_k of deadline misses derived by using Eq. (7) is exactly the same as only testing the discretized points $t \in \mathbf{L}_k$, defined as follows:

$$\Phi_k = \min_{\{t \in \mathbf{L}_k\}} \mathbb{P}(S^t \geq t), \tag{8}$$

where $\mathbb{P}(S^t \geq t)$ is derived by taking the right-hand side from Eq. (4) under $\rho_{i,t} = \left\lceil \frac{t}{T_i} \right\rceil$ for each task τ_i in $hp(\tau_k)$ and $\rho_{k,t} = 1$.

Proof Suppose for contradiction that the minimum $\mathbb{P}(S^t \geq t)$ (by using Eq. (7)) happens when $t = t'$ and t' lies in an interval (α, β), where α and β are two consecutive discretized points in $\mathbf{L}_k$, i.e., $\nexists(\gamma \in \mathbf{L}_k$ and $\gamma \in (\alpha, \beta))$. More specifically, $\left\lceil \frac{t'}{T_i} \right\rceil$ is the same as $\left\lceil \frac{\beta}{T_i} \right\rceil$ for any task $\tau_i \in hep(\tau_k)$. Therefore, for each task τ_i in $hep(\tau_k)$, we know that $\rho_{i,t'}$ is also the same as $\rho_{i,\beta}$.

With this, $\text{mgf}_{hep(\tau_k)}(s)$ is the same when t is set to t' and β for any $s > 0$. Therefore, for any given $s > 0$, we have $\frac{\text{mgf}_{hep(\tau_k)}(s)}{exp(s \cdot t')} > \frac{\text{mgf}_{hep(\tau_k)}(s)}{exp(s \cdot \beta)}$ since $\beta > t'$, in which the contradiction is reached. □

To efficiently analyze the probability of deadline misses, by the definition in Eq. (8), we can select a few testing points in $\mathbf{L}_k$ and the minimum value among the probability in these points is still a safe upper bound on the probability of deadline misses. We here introduce a k-point probabilistic deadline-miss test to trade off the time complexity with the quality of delivered results, which is motivated by Chen et al. in [7] and Bini et al. in [5]. By following the rationale, we define k selected points by the $k-1$ higher-priority tasks and task τ_k. At each point t, we verify if the total released workload up to time t from $hep(\tau_k)$ can be finished. With Theorem 1, the proposed k-point probabilistic deadline-miss test is defined as follows:

Theorem 2: Given a set of constrained-deadline sporadic tasks Γ, the probability of deadline misses of task τ_k is upper bounded by $\hat{\Phi}_k$, defined as follows:

$$\hat{\Phi}_k = \min_{t \in \left\{\left\lfloor \frac{D_k}{T_1} \right\rfloor T_1, \left\lfloor \frac{D_k}{T_2} \right\rfloor T_2, \ldots, D_k\right\} \setminus \{0\}} \mathbb{P}(S^t \geq t), \tag{9}$$

where $\mathbb{P}(S^t \geq t)$ can be derived from Eq. (4) by setting $\rho_{i,t} = \left\lceil \frac{t}{T_i} \right\rceil$ for each task τ_i in $hp(\tau_k)$ and $\rho_{k,t} = 1$.

Proof Since these k selected points, i.e., $\left\lfloor \frac{D_k}{T_1} \right\rfloor T_1$, $\left\lfloor \frac{D_k}{T_2} \right\rfloor T_2$, $\ldots$, $\left\lfloor \frac{D_k}{T_{k-1}} \right\rfloor T_{k-1}, D_k$, lie in the range of $[0, D_k]$, thus, it is sufficient to only test those (up to) k selected points (by removing 0). It is clear that $\hat{\Phi}_k \geq \Phi_k$. □

The following example illustrates how Lemma 2 and Theorem 2 work for calculating the probability of deadline misses. Suppose a task set has three sporadic tasks:

- $\tau_1 : T_1 = D_1 = 10, C_1^N = 4, C_1^A = 6, \mathbb{P}_1^A = 10^{-5}$,
- $\tau_2 : T_2 = D_2 = 45, C_2^N = 10, C_2^A = 15, \mathbb{P}_2^A = 10^{-5}$,
- $\tau_3 : T_3 = D_3 = 75, C_3^N = 10, C_3^A = 30, \mathbb{P}_3^A = 10^{-6}$,

to be scheduled on a uniprocessor with the rate-monotonic (RM) fixed-priority scheduling policy. In this example, we evaluate the probability of deadline misses of task τ_3, i.e., $k = 3$. At first three time points are selected accordingly: $t \in \{45, 70, 75\}$. The upper bound probability $\mathbb{P}(S^t \geq 45)$ is at most 0.05551 when s is around 0.6358: $[(\exp(6s)\cdot 10^{-5}+\exp(4s)\cdot(1-10^{-5}))^4\times(\exp(15s)\cdot 10^{-5}+\exp(10s)\cdot(1-10^{-5}))\times(\exp(30s)\cdot 10^{-6}+\exp(10s)\cdot(1-10^{-6}))]/\exp(45s)$. For time point 70, the upper bound is 0.000492 when s is around 0.711. For time point 75, the upper bound is 0.00024 when s is around 0.721. Therefore, $\hat{\Phi}_k$ is set to 0.00024. We also provide the results gathered from Lemma 2 in Table I. In this example, by applying Eq. (4), we can observe that the minimum probability among all the time points t in $\mathbf{L}_k$ is 0.00024 while $t = 75$, which is the same as the delivered result from Theorem 2, i.e., $\Phi_k = \hat{\Phi}_k$ in this example.

B. *ℓ-Consecutive Deadline Misses*

After addressing the probability of deadline misses of task τ_k, now we consider how to handle more general cases for the upper bound on the probability of ℓ-consecutive deadline misses. For the rest of this section, we reform the notation of the probability of deadline misses from Φ_k to $\Phi_{k,\ell}$ for the probability of ℓ-consecutive deadline misses. While ℓ is 1, we define the probability $\Phi_{k,1}$ as Φ_k delivered by Theorem 1, i.e., $\Phi_{k,1} = \Phi_k$. The following theorem shows that we can extend the probabilistic analysis in Eq. (7) and recursively obtain a safe upper bound on the probability of ℓ-consecutive deadline misses:

Theorem 3: Given a set of constrained-deadline sporadic tasks Γ. Suppose that

$$\Phi^{\theta}_{k,w} = \min_{0<t\leq(w-1)T_k+D_k} \mathbb{P}(S^t \geq t), \tag{10}$$

where $\mathbb{P}(S^t \geq t)$ can be derived from Eq. (4) by setting $\rho_{i,t} = \left\lceil \frac{t}{T_i} \right\rceil$ for each task τ_i in $hep(\tau_k)$. That is, $\Phi^{\theta}_{k,w}$ is the upper bound on the probability of w-consecutive deadline misses when the processor executes at least w (consecutively released) jobs of task τ_k without any idling. For notational brevity, let $\Phi_{k,0}$ be 1. The probability of ℓ-consecutive deadline misses of task τ_k is upper bounded by $\Phi_{k,\ell}$, defined as follows:

$$\Phi_{k,\ell} = \max\left\{\Phi^{\theta}_{k,w} \cdot \Phi_{k,\ell-w} | w \in \{1, 2, \dots \ell\}\right\} \tag{11}$$

Proof We prove this theorem by constructing $\Phi_{k,j}$ from $j = 1, 2, \dots, \ell$ sequentially. When j is 1, the upper bound $\Phi_{k,1}$ is equal to Φ_k and can be derived by using Theorem 1 or 2. Therefore, suppose that $\Phi_{k,j}$ for $j \in \{1, 2, \dots, \ell-1\}$ is already calculated by the previous steps. For a preemptive fixed-priority schedule, removing tasks with priority lower than τ_k does not change the schedule of task τ_k. As a result, we only consider $hep(\tau_k)$ in the proof. To have ℓ-consecutive jobs of task τ_k with deadline misses, there must be at least ℓ consecutively released jobs of task τ_k missing t heir deadlines. Let J_ℓ be a job of task τ_k in which its previous $\ell - 1$ jobs, $J_1, J_2, \dots, J_{\ell-1}$, released by task τ_k all miss their deadlines.

Let t_j be the arrival time of job J_j released by task τ_k. Let t_R be the completion time of job J_ℓ. Since task τ_k is a sporadic task with a minimum inter-arrival time T_i, by definition, we have $t_\ell - t_1 \geq (\ell-1)T_k$ and $t_R - t_\ell > D_k$. That is, $t_R - t_1 > (\ell-1)T_k + D_k$. Let t_{-1} be the latest instant before t_R, at which either the processor idles at time t_{-1}, or all the jobs of task τ_k before t_{-1} have finished their executions. Suppose that there are w^* jobs of task τ_k released after or at time t_{-1}.

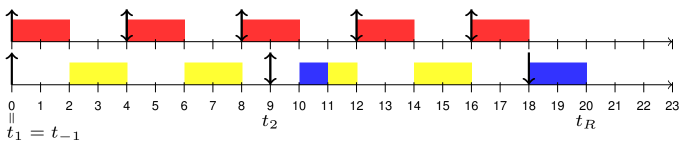
We consider two different cases in this interval $[t_{-1}, t_R)$:

1) **Case 1 - if** $t_1 \geq t_{-1}$: This implies that $w^* \geq \ell$ and the processor is busy from time t_{-1} to time t_R executing the jobs released at or after t_{-1}. Similar to the proof of Theorem 1, we can remove all the jobs executed before t_{-1} and set the release time of the first job of τ_k released in this interval to time t_{-1} in the schedule.[2] Similarly, we can also advance the subsequent jobs of τ_k to release at time $t_{-1} + T_k, t_{-1} + 2T_k, \dots$. This adjustment does not decrease the response times of these consecutively released jobs of task τ_k. Therefore, all of these w^* jobs of task τ_k still miss their deadlines. With a similar argument to the one made in the proof of Theorem 1, the processor is busy executing the *periodically* released workload from time t_{-1} to time $t_{-1} + (\ell-1)T_k + D_k \leq t_{-1} + (w^* - 1)T_k + D_k$. Hence, the upper bound on the probability of this case is $\Phi^{\theta}_{k,\ell}$.
2) **Case 2 - if** $t_1 < t_{-1}$: This implies that $w^* < \ell$ and the processor is busy from time t_{-1} to time t_R executing the jobs released at or after t_{-1}. Therefore, we know that from time t_1 to time t_{-1}, there must be at least $\ell - w^*$ consecutive jobs of task τ_k with deadline misses and probability upper bounded by $\Phi_{k,(\ell-w^*)}$. We now only have to evaluate the probability of the w^* consecutive deadline misses of task τ_k from time t_{-1} to time t_R, which is upper bounded by Φ^{θ}_{k,w^*}, as an identical scenario to Case 1. Therefore, the upper bound on the probability of this case is hence $\Phi^{\theta}_{k,w^*} \cdot \Phi_{k,(\ell-w^*)}$.

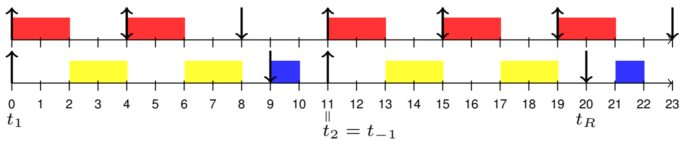
If w^* is known, one of these two cases defines the upper bound on the probability of ℓ-consecutive deadline misses of task τ_k. However, even though w^* is unknown, we can iterate all the possible values from 1 to ℓ. Therefore, the upper bound on the probability of ℓ-consecutive deadline misses can be found by Eq. (11). □

Fig. 1 illustrates the two cases in the proof of Theorem 3. Suppose that τ_2 is the targeted task τ_k in Theorem 3. As shown in Fig. 1a, the execution of the second job released at time 9 is pushed by the overrun of the first job, i.e., the first blue block.

[2]The first job of task τ_k released at or after t_{-1} may not be J_1.



(a) Case 1: $t_1 \geq t_{-1}$



(b) Case 2: $t_1 < t_2 = t_{-1}$

Fig. 1: Example of the two release scenarios in the proof of Theorem 3. In this example, τ_1 and τ_2 are implicit-deadline sporadic tasks with $C_1^A = 2, T_1 = D_1 = 4, C_2^A = 5, T_2 = D_2 = 9$. The blue marked blocks are overrunning executions.

Therefore, the analyzed window should cover the time interval $[t_1, t_R)$. Due to the space limitation, in Fig. 1b, there are only two jobs of τ_2 with an idle instant, but in fact the second job can be followed by the other $w^* - 1$ jobs consecutively without any idle instants. We can find that the analyzed schedule of the jobs finished before t_2 and the jobs released after t_2, can be individual.

With Theorem 3, we can obtain $\Phi_{k,\ell}$ for the upper bound on the probability of ℓ-consecutive deadline misses. To avoid testing all time points in the interval $(0, (w-1)T_k + D_k]$ in Eq. (10), we can again apply the same strategy as in Lemma 2 to generate a pseudo-polynomial number of time points $\mathbf{L}_k$ to test. That is, $\mathbf{L}_k = \{r \cdot T_i | \tau_i \in hp(\tau_k); r = 1, \dots, \lfloor((\ell-1)T_k + D_k)/T_i\rfloor\} \cup \{(w-1)T_k + D_k | w = 1, \dots, \ell\}$.

To efficiently analyze the probability $\Phi_{k,\ell}$ of ℓ-consecutive deadline misses, we propose to choose only up to $k \cdot \ell$ testing points to derive $\hat{\Phi}_{k,\ell}$, which can be similarly proved as Theorem 2, i.e., $\hat{\Phi}_{k,\ell} \geq \Phi_{k,\ell}$. That is, $\mathbf{L}_k = \{(w-1)T_k + D_k | w = 1, \dots, \ell\} \cup \{r \cdot T_i | \tau_i \in hp(\tau_k); r = \lfloor D_k/T_i \rfloor, \dots, \lfloor((\ell-1)T_k + D_k)/T_i\rfloor\}$.

V. Remark: General Distribution

With the assumption we made in Section II-B that all the tasks only have two different WCETs, we have presented how to use the Chernoff bounds and the *mgf* to derive the upper bound on the probability of ℓ-consecutive deadline misses. Generally, the proposed approaches in fact can be directly applied for any general execution time distributions. To release the assumption, for each task, we can generalize the distribution of execution time to a probabilistic worst case execution time model like in [16], in which each probabilistic execution time has a given corresponding probability. With respect to our approaches, the definition of the *mgf* should use Eq.(2) rather than Eq.(1), then all the theorems in this paper can be applied for any general execution time distributions.

VI. Result and Discussion

This section presents simulations with different synthesized task sets to analyze the performance of our proposed approaches with respect to the accuracy and the needed calculation times of the probabilistic analyses.

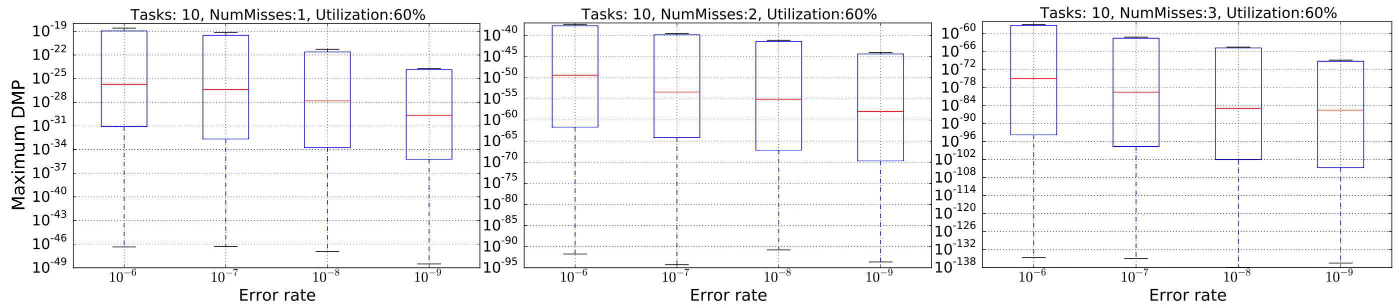
A. Experimental Setup

For the evaluation, we implement our efficient schedulability tests with Python 2.7 on Linux kernel 3.13.0. The adopted machine has an Intel Core i7-4770 CPU and 8GB DDR3 RAM. The complete scripts are all available at [8]. In the experiments, we synthesize random task sets with a given utilization value U^*, i.e., $U^* = \sum_{\tau_i \in \Gamma} U_i^N$ according to the UUniFast method [4], where U_i^N is defined as C_i^N / T_i. We follow the suggestion from Davis and Burns [10] to generate the task periods according to an log-uniform distribution with two orders of magnitude, i.e., $[1 - 100]$. With the generated utilization U_i^N, the normal execution time C_i^N is set to $U_i^N \cdot T_i$. The cardinalities of the task sets are: 10, 20, and 30 tasks. After observing that the results among these three cardinalities are similar, we only show the case for 10 tasks due to the space limitation. For each given error rate $\mathbb{P}_i^A$, we record 100 synthesized task sets. For each task set, we take the maximum deadline-miss probability (DMP) $\hat{\Phi}_{i,\ell}$ among all the tasks with the given ℓ, i.e., $\tau_i \in \Gamma$. We adopt Eq. (9) when $\ell = 1$ and adopt Eq. (11) when $\ell > 1$. To find the s with the minimal probability in Eq. (4), we use SciPy library [1]. Among these recorded 100 values for each error rate, we report the medians (red lines), the interquartile range of the sample (the width of the boxes), the maximums (top lines), and the minimums (bottom lines) with the box plots, where a base-10 log scale is used for the Y-axis. To consider the overhead of error recovery, i.e., one re-execution, we assume the error detection costs 20% of the task execution time and set C_i^A by $\frac{2.2}{1.2} \approx 1.83 \cdot C_i^N$ for all tasks $\tau_i \in \Gamma$. For the simplicity of presentation, Lemma 2 is called EPST and Theorem 2 is called EPST-K in the rest of this section.

B. Evaluation of Deadline-Miss Probability

Fig. 2a shows the relationship between the error rates and the maximum DMP among all the tasks when U^* is 60% by adopting EPST-K. The medians start to downgrade significantly when the error rate is 10^{-8}. However, the ranges between the maximums and minimums are slightly changed when the error rate goes down. Fig. 2b and Fig. 2c present the relationship for the probability of $\{2,3\}$-consecutive deadline misses. They are similar to Fig. 2a, the upper interquartiles are all close to the maximums. The interquartile ranges are changed significantly when ℓ becomes larger.

We also compare the results derived by EPST and EPST-K to evaluate the approximation by testing only k time points. However, the results are totally the same. That is, the probability $\hat{\Phi}_{i,\ell}$ of deadline misses delivered by EPST-K is always the same as $\Phi_{i,\ell}$ delivered by EPST. Although we know that testing a pseudo-polynomial number of time points may give us a tighter upper bound on the deadline-miss probability,



(a) DMP (probability of deadline misses)
(b) DMP of 2-consecutive deadline misses
(c) DMP of 3-consecutive deadline misses

Fig. 2: Maximum DMPs with $U^* = 60.0\%$, varying error rates and ℓ

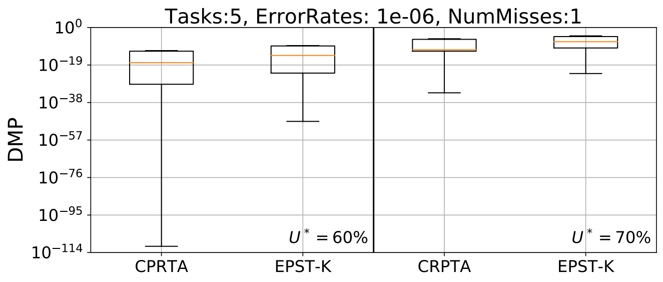


Fig. 3: DMPs under different approaches, varying U^*

our experiments empirically show that it may be sufficient to test only k-points derived by EPST-K to obtain the upper bound on the probability efficiently.

C. Chernoff-Bound-Based vs. Convolution-Based Approaches

In order to evaluate the pessimism and the efficiency, we compare our approaches, i.e., EPST and EPST-K, with the tighter convolution-based approach proposed in [16] without any approximation, called CPRTA in the paper. We used the released scripts in MATLAB and only changed the input of the simulation by using the task generator described in Section VI-A.[3] In this evaluation, the period of a task is uniformly distributed between [1, 50]. Comparing to typically using [1, 100], this is needed to reduce the number of iterations in CPRTA.

In Table II, we also present the analysis time of different analyses. We test over 100 task sets for each configuration and we set at most 10 minutes as the timeout threshold in each task sets. Unfortunately, CPRTA is not able to derive the DMP for $10, 20$ and 30 tasks. Without setting the timeout threshold, we also use 6 computers in our local cluster to derive the results by using CPRTA for 12 hours for task sets with 10 tasks. To the end, however, none of the CPRTA analyses is able to finish. The run time reported in [16] was 140 seconds in average for task sets with 16 tasks. However, the convolution-based approaches for testing the generated task sets in [16] were usually quite easy to finish. When we used the widely-accepted UUniFast method for generating the task sets, we note that the convolution takes much longer time to compute. Thus, such convolution-based approaches would be very time consuming and are not suitable for large task sets.

Moreover, we compare the derived DMPs of CPRTA and EPST-K under different U^* for the lowest priority task in each set. In Fig. 3, the results in the left-hand side are derived under $U^* = 60\%$, whereas the results in the right-hand side are derived under $U^* = 70\%$. As shown in Fig. 3, besides the extreme cases, our method is a bit pessimistic (the lower the tighter). However, if the pessimism of sufficient tests is acceptable especially under a large number of tasks, the proposed approaches can efficiently derive the upper bound on the probability of deadline misses instead of unnecessarily deriving the whole response time distribution.

D. CPRTA with Re-Sampling Approximation vs. EPST-K

As shown in the previous subsection, we can see that CPRTA without any approximation is very time consuming even with 10 tasks. In [16], the re-sampling technique by manually introducing a threshold of the valid number of data points was shown to improve the needed calculation time of CPRTA by orders of magnitudes faster. However, with the task sets generated by applying the UUniFast method, the results derived from the re-sampling technique may be worse than the results derived from EPST-K even for 10 tasks as shown in Fig. 4. Although for 10 tasks using re-sampling in CPRTA with a threshold set to 100 indeed reduces the calculation time, i.e., around 1 second in average, the derived results are all worse than our analysis in medians and the interquartile ranges of the sample as shown in Fig. 4. Especially when $U^* = 70\%$, the DMPs derived from CPRTA with re-sampling are really closed to 1. For 20 tasks and above, we can not obtain any result by using re-sampling with a threshold set to 100 within 6 hours. With threshold 1000, none of the CPRTA analyses is able to finish even with $U^* = 60\%$ in 6 hours. With threshold 50, the calculation time of CPRTA is extremely fast, but the results are all close to 1 even under utilization $U^* = 60\%$.

After all, we know that CPRTA with re-sampling could provide tighter results with a higher threshold but require much more time to calculate the probability. Conversely, CPRTA with re-sampling could need less time for calculation with

[3]The scripts were downloaded from https://who.rocq.inria.fr/Dorin.Maxim/ on Jan. 24 in 2017. The modified scripts and the input generated by UUniFast method can be found in [8].

Methods	Cardinality	5 tasks	10 tasks	20 tasks	30 tasks
CPRTA	Avg. Time (sec)	7.4812	-	-	-
	Successful Runs	98/100	0/100	0/100	0/100
EPST	Avg. Time (sec)	0.1406	0.4855	1.6738	2.7545
	Successful Runs	100/100	100/100	100/100	100 /100
EPST-K	Avg. Time (sec)	0.0418	0.1253	0.4760	0.7917
	Successful Runs	100/100	100/100	100/100	100/100

TABLE II: Analysis time need for 100 synthesized task sets per configuration.

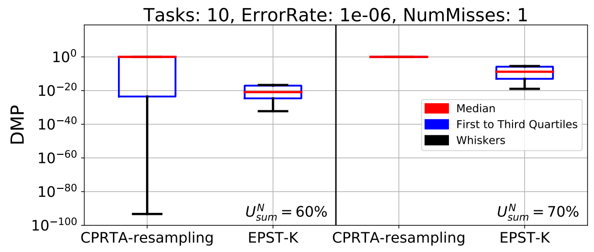


Fig. 4: Comparison between EPST-K and CPRTA by using the re-sampling method with a threshold set to 100, varying U^*

a lower threshold but provide looser results with respect to the probability of the deadline-miss rate. If using convolution-based approaches anyway cannot avoid to use approximations like the re-sampling technique to reduce the time complexity, how to properly approximate the convolution for the balance between the tightness and the calculation time is another considerable issue, which is the essential problem that needs to be solved for the convolution-based approached proposed in the literature.

VII. Conclusion

When transient faults occur the execution time of tasks can be prolonged due to the overhead of software based error recovery mechanisms. This paper provides a new trail that allows the system designer to provide probabilistic arguments for the probability of deadline misses based on the probability of those soft errors. An upper bound on the probability of $\ell-$consecutive deadline misses can be derived as well. Although Chernoff bounds are known to be pessimistic, it is efficient (with respect to time complexity) to evaluate the schedulability of a task instead of using convolution-based analyses unnecessarily.

Please note that, the proposed approaches can be directly applied for the other bounds in the literature. In future work, we plan to study correlated worst-case execution time probability distributions to handle soft errors that are dependent.

Our studied problem and result are aligned with the design requirement of safety-critical systems. For example, verifying if the probability as a threshold to reboot the system for resolving consecutive deadline misses is acceptable or not.

Acknowledgments

This paper is supported by DFG, as part of the Collaborative Research Center SFB876 (http://sfb876.tu-dortmund.de/) and the priority program "Dependable Embedded Systems" (SPP 1500 - spp1500.itec.kit.edu). The authors thank anonymous reviewers for their suggestions on improving this paper.

References

[1] Scipy. http://www.scipy.org/, 2016.

[2] P. Axer and R. Ernst. Stochastic response-time guarantee for non-preemptive, fixed-priority scheduling under errors. In *Design Automation Conference (DAC)*, pages 1–7, May 2013.

[3] R. C. Baumann. Radiation-induced soft errors in advanced semiconductor technologies. *IEEE Transactions on Device and Materials Reliability*, 5(3):305–316, Sept 2005.

[4] E. Bini and G. Buttazzo. Measuring the performance of schedulability tests. 2005.

[5] E. Bini and G. C. Buttazzo. Schedulability analysis of periodic fixed priority systems. *IEEE Transactions on Computers*, 53(11):1462–1473, Nov 2004.

[6] J. Bucklew. *Introduction to Rare Event Simulation*. Springer-Verlag, New York, NY, USA, 1st edition, 2004.

[7] J. J. Chen, W. H. Huang, and C. Liu. k2u: A general framework from k-point effective schedulability analysis to utilization-based tests. In *Real-Time Systems Symposium, 2015 IEEE*, pages 107–118, Dec 2015.

[8] K.-H. Chen. Efficient Probabilistic Schedulability Test. https://github.com/kuanhsunchen/EPST/, 2017.

[9] K. H. Chen, J. J. Chen, F. Kriebel, S. Rehman, M. Shafique, and J. Henkel. Task mapping for redundant multithreading in multi-cores with reliability and performance heterogeneity. *IEEE Transactions on Computers*, Nov 2016.

[10] R. Davis, A. Zabos, and A. Burns. Efficient exact schedulability tests for fixed priority real-time systems. *Computers, IEEE Transactions on*, 2008.

[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 *Real-Time Systems Symposium, 23rd IEEE*, pages 289–300, 2002.

[12] J. S. Hu, F. Li, V. Degalahal, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin. Compiler-directed instruction duplication for soft error detection. In *Design, Automation and Test in Europe*, March 2005.

[13] International Electrotechnical Commission (IEC). Functional safety of electrical / electronic / programmable electronic safety-related systems ed2.0. 2010.

[14] International Organization for Standardization (ISO). Iso/fdis26262: Road vehicles - functional safety. 2000.

[15] 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.

[16] 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.

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

[18] N. Oh, P. P. Shirvani, and E. J. McCluskey. Error detection by duplicated instructions in super-scalar processors. *IEEE Transactions on Reliability*, 51(1):63–75, Mar 2002.

[19] 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.

[20] W. Xu, Z. A. H. Hammadeh, A. Kröller, R. Ernst, and S. Quinton. Improved deadline miss models for real-time systems using typical worst-case analysis. In *27th Euromicro Conference on Real-Time Systems*, pages 247–256, 2015.