technische universität
dortmund

Critical Instant for Probabilistic Timing Guarantees: Refuted and Revisited

Kuan-Hsun Chen, Mario Günzel, Georg von der Brüggen, Jian-Jia Chen
University of Twente, Department of Computer Science, Enschede, The Netherlands
TU Dortmund University, Department of Computer Science, Dortmund, Germany

Citation:

BibTeX:

```
@inproceedings{22RTSS_ChenGBC,
  author={Kuan-Hsun Chen, Mario Günzel, Georg von der Brüggen, Jian-Jia Chen},
  booktitle={43rd IEEE Real-Time Systems Symposium (RTSS)},
  title={Critical Instant for Probabilistic Timing Guarantees: Refuted and Revisited},
  year={2022},
  volume={},
  number={},
  pages={},
  doi={}
}
```


computer
science 12

Critical Instant for Probabilistic Timing Guarantees: Refuted and Revisited

Kuan-Hsun Chen*‡, Mario Günzel§‡, Georg von der Brüggen§, and Jian-Jia Chen§
*University of Twente, The Netherlands, k.h.chen@utwente.nl
§TU Dortmund University, Germany, {mario.guenzel, georg.von-der-brueggen, jian-jia.chen}@tu-dortmund.de
‡Both authors contributed equally

***Abstract*—In soft real-time systems, tasks may occasionally miss their deadlines. This possibility has triggered research on probabilistic timing analysis for the execution time of a single program and probabilistic response time analysis of concurrently executed tasks. Under fixed-priority preemptive uniprocessor scheduling, it was shown that the classical critical instant theorem (for deriving the worst-case schedulability or response time) by Liu and Layland (in JACM 1973) can be applied to analyze the worst-case deadline failure probability (WCDFP) and the worst-case response time exceedance probability (WCRTEP).**

In this work, we present a counterexample for this result, showing that the WCDFP and WCRTEP derived by the classical critical instant theorem is unsound. We further provide two sound methods: one is to account for one additional *carry-in* job of a higher-priority task and another is to sample and *inflate* the execution time of certain jobs without adding one additional carry-in job. We show that these two methods do not dominate each other and, in the evaluation, apply them to two well-known approaches based on direct convolution and Chernoff bounds.

***Index Terms*—Real-Time Systems, Deadline Failure Probability, Critical Instant, Probabilistic Response Time Analysis**

I. Introduction

A classical, hard real-time analysis aims to determine whether tasks fulfill their timing constraints under all circumstances. The seminal work by Liu and Layland [22] provides fundamental knowledge to ensure worst-case timeliness when scheduling periodic real-time tasks on a uniprocessor system. A *periodic task* τ_i is an infinite sequence of task instances, called *jobs*, where two consecutive jobs of a task are released with a period T_i (i.e., the time interval length between the release time of two consecutive jobs is always T_i), all jobs of a task have the same *relative* deadline $D_i = T_i$ (i.e., the *absolute* deadline of a job arriving at time t is $t + D_i$), and each job has the same worst-case execution time (WCET) C_i.

One important observation by Liu and Layland [22] for (task-level) fixed-priority scheduling is the critical instant theorem: "*A critical instant for any task occurs whenever the task is requested simultaneously with requests for all higher priority tasks*". As this statement was incomplete, it was noted in the review paper by Sha et al. [33] that: "*A critical instant for a task is a release time for which the response time is maximized (or exceeds the deadline, in the case where the system is overloaded enough that response times grow without bound)*". As the critical instant only explains the beginning of the job releases but not the complete time interval of interest, Liu and Layland also defines the critical time zone for a task, which is the time interval between a critical instant and the end of the response to the corresponding request of the task.

The critical instant (and the critical time zone) theorem is further extended to cope with the sporadic real-time task model [29], in which two consecutive jobs of a task are separated by a specified minimum inter-arrival time. As reported in an empirical study [1] for industrial real-time systems, periodic and sporadic task activations (with minimum inter-arrival time constraints) are widely used in industry practice; specifically, in 82% and 47%, respectively, of the investigated systems.

With the critical instant theorem, the worst-case response time analysis and the feasibility of the fixed-priority schedule can be validated by simulating the worst-case job release pattern while utilizing the worst-case execution time. Alternatively, the time demand analysis (TDA) [18], [20] can be utilized by applying a fixed-point iteration test.

The critical-instant theorem has been widely used in research results. Some of the extensions of the critical instant theorem are correct (e.g., the level-i busy window concept in [21]) and some are unfortunately incorrect (e.g., the extensions to self-suspending tasks in [19], [28]).

While the critical instant is usually applied to ensure timeliness in a worst-case scenario (i.e., meeting the deadline under all circumstances), many embedded real-time systems can still function well even with occasional (bounded) deadline misses. Hence, providing quantification of deadline misses is of importance in practice. Specifically, safety standards such as IEC-61508 [16] and ISO-26262 [17] specify a (very) low failure probability but not necessarily a failure probability of zero. In contrast to the large body of research results regarding hard real-time systems, formal arguments and proofs for properties related to soft real-time systems with probabilistic, statistical, or deterministic guarantees remain an open research area even for simple settings.

In this regard, probabilistic timing analysis has been explored in the literature. For a probabilistic analysis, instead of having a scalar worst-case execution time C_i of a task τ_i, the execution time of a job of τ_i is specified by a random variable, describing the impact of hardware effects (e.g. caches, branch prediction, pipelines, etc.) as well as different input values and paths taken in the software on the execution time. More variances and information can be found in the survey by Davis and Cucu-Grosjean [12].

The usage of the critical instant theorem for probabilistic

timing analysis can be traced back to Tia et al. [34] in 1995 for a probabilistic time-demand analysis (PTDA). They assumed that the analyzed task set is released at the critical instant, which leads to "*an upper bound of the processor-time demand of j-th job of τ_i and all higher priority tasks only when the deadline and maximum computation time of every task are less than its period*" [34]. In 1999, Gardner and Liu noted that the PTDA is only valid if the worst-case utilization of the task set is not more than 1, since the worst-case pattern of releases might not follow the critical instant (the concept was called an *in-phase busy interval*). They explored this aspect by a stochastic time-demand analysis (STDA) and simulated different phases. However, they did not find any cases where the probability that jobs miss their deadlines was higher for random phasing than for the simultaneous release under the critical instant.

In 2002, Diaz et al. [13] noted that the worst-case scenario for a job in the first busy period following synchronous release (i.e., the classical critical instant theorem for periodic tasks) in PTDA [34] and SDTA [15] may lead to an underestimation of the response time distributions when the worst-case utilization of the task set exceeds 1. The main issue is the *backlog* at the end of each hyperperiod (i.e., the least common multiple of the periods of the periodic tasks). The analysis by Diaz et al. [13], refined later by Lopéz et al. [23] in 2008, considers the scenario when backlog may exist.

One way to avoid having backlog is to restart the system when there is a deadline miss or abort the job which misses its deadline after its deadline. Under this assumption, in 2013, Maxim and Cucu-Grosjean [25] presented a probabilistic response time analysis, proved the critical instant theorem from the probabilistic perspectives, and further extended the critical instant theorem to deal with tasks with inter-arrival times and relative deadlines also described by random variables.

Since then, the critical instant theorem under the probabilistic setup by Maxim and Cucu-Grosjean [25] has been widely used in the literature [26], [31], [32]. Due to an essential complication in probabilistic response-time analysis (i.e., convolution over multiple random variables), several techniques have been developed to tackle issues of intractability through various means, e.g., down-sampling [24], [25], [27], [30], concentration inequalities [8], [9], [35], task-level convolution [35], and Monte-Carlo simulation [6].

Our Contributions: This paper revisits the critical instant for the probabilistic timing analysis of sporadic real-time task sets, formally defined as the worst-case response time exceedance probability (WCRTEP) in Definition 2 and the worst-case deadline failure probability (WCDFP) in Definition 4. Our contributions are as follows:

- We present a counterexample in Section III, demonstrating that calculating the WCRTEP and WCDFP based on the critical instant theorem for fixed-priority scheduling can be *optimistic* even without backlog. Therefore, the critical instant theorem is unsound in the probabilistic setting, even when jobs that miss their deadline are aborted or when the system is restarted after a deadline miss. We also explain why the proof of the critical instant theorem for a static worst-case response time cannot be applied under the probabilistic setting.
- We propose two methods to derive the WCRTEP and WCDFP for sporadic real-time task systems with probabilistic execution times in Section IV. One method is to account for one additional *carry-in* job of a higher-priority task, whilst another is to *inflate* the execution of certain higher-priority jobs with a sampling process. We further provide examples to illustrate that these two methods do not dominate each other.
- In Section V, we discuss the impact of the unsound critical instant theorem for the probabilistic setting in the literature and provide suggestions on how these results may be revised.
- With numerical simulations, we compare the WCDFP derived from the proposed methods, realized by the direct convolution approach and the Chernoff bound approach, and evaluate their effectiveness under different experimental settings in Section VI.

II. System Model and Notation

We consider a set of n independent sporadic real-time tasks $\mathbb{T} = \{\tau_1, \ldots, \tau_n\}$ in a uniprocessor system. Each task $\tau_i \in \mathbb{T}$ is modeled by a tuple (C_i, D_i, T_i), where D_i is the relative deadline of τ_i and T_i is its minimum inter-arrival time. Each task releases an infinite number of successive task instances, called jobs, and $\tau_{i,j}$ denotes the j-th job of task τ_i. A job released at time t must finish not later than $t+D_i$, and the next job can only be released at or after $t+T_i$. We consider implicit-deadline task sets, i.e., $D_i = T_i \ \forall \tau_i \in \mathbb{T}$, and constrained-deadline task sets, i.e., $D_i \leq T_i \ \forall \tau_i \in \mathbb{T}$.

C_i is a random variable to describe the possible execution time of task τ_i, following a discrete distribution with h distinct but finite values, i.e., $h \geq 1$. The execution times of the jobs $\tau_{i,j}$, $j \in \mathbb{N}=\{1, 2, 3, \ldots\}$ are described by $C_{i,j}$, $j \in \mathbb{N}$ which are independent copies of C_i. We assume that each job is executed with exactly one of these h distinct time units, and the sum of their probabilities is 100%. We denote by $\mathbb{P}(A = x)$ the probability that a random variable A is equal to x. For instance, $\mathbb{P}(C_{i,j} = 5)$ stands for the probability that the execution time of j-th job of task τ_i is equal to 5. In addition, we assume that the jobs' execution times are *independent and identically distributed* (iid) from each other (i.e., their joint probability is equal to the product of their probabilities); an assumption common in the literature, see, e.g., [8], [9], [12], [25], [35].

We assume a preemptive fixed-priority scheduling policy, where each task has a unique fixed priority. That is, the jobs of task τ_i have the priority specified to task τ_i. At a given time t, for any two jobs of two distinct tasks τ_i and τ_k that are eligible to execute, the execution of τ_i precedes the execution of τ_k if $i < k$. We denote by $hp(\tau_i)$ the set of tasks with higher priority than τ_i. Moreover, *we assume that once a job misses its deadline, it is immediately aborted.*

Let $R_{k,j}$ be the response time distribution for the j-th job of τ_k. If a deadline is missed, then under that sample the

response time distribution is $R_{k,j} = \infty$ as the job never finishes. We are interested in the worst-case behavior of the probabilistic response time and deadline failure probability. As it may seem at the first glance strange to combine the worst-case phrase with the probabilistic argument, the following definitions provide concrete statements:

Definition 1 (Job-Level Response Time Exceedance Probability)**.** The response time exceedance probability of job $\tau_{k,j}$ is the probability that the response time of the j-th job of task τ_k is greater than t:

$$\mathbb{P}(R_{k,j} > t) \tag{1}$$

Definition 2 (Worst-Case Response Time Exceedance Probability)**.** The worst-case response time exceedance probability (WCRTEP) of task τ_k is an upper bound on the probability that the response time of a job of τ_k is greater than t, i.e.,

$$\sup_{j \in \mathbb{N}} \{\mathbb{P}(R_{k,j} > t)\}, \tag{2}$$

where sup indicates the supremum within set theory.

We are often not interested in the complete response time distribution but in the probability that the response time exceeds the relative deadline. That is, we are interested in the probability that a job misses its deadline.

Definition 3 (Job-Level Probability of Deadline Misses)**.** The deadline failure probability (DFP) of job $\tau_{k,j}$ is the probability that the j-th job of task τ_k misses its relative deadline D_k:

$$\mathbb{P}(R_{k,j} > D_k) \tag{3}$$

Definition 4 (Worst-Case Deadline Failure Probability)**.** The worst-case deadline failure probability (WCDFP) of task τ_k is an upper bound on the probability that a job of τ_k misses its relative deadline D_k:

$$\sup_{j \in \mathbb{N}} \{\mathbb{P}(R_{k,j} > D_k)\} \tag{4}$$

We note that this WCDFP definition is identical to the one by Davis and Cucu-Grosjean in their survey paper [12]. In contrast, they used the term probabilistic worst-case response time (pWCRT) to denote the WCRTEP defined here.

III. Counterexample for the Critical Instant

In this section, we provide a counterexample to demonstrate that calculating the WCDFP or WCRTEP based on the critical instant theorem for fixed-priority scheduling is *too optimistic* even without backlog. Towards this, we first review the critical instant theorem and its proof when the objective is an upper bound on the worst-case response time. Then, we present the counterexample and explain why the proof of the critical instant theorem cannot be applied when the objective is the WCRTEP or WCDFP.

A. Critical Instant for Worst-Case Response Time

The critical instant theorem [22] has been widely adopted as a backbone in worst-case response time analyses for periodic and sporadic real-time tasks. For completeness, we give a short summary and explain the correctness proof when determining the worst-case response time for periodic or sporadic tasks.

Definition 5 (Critical Instant, reworded from [22], [29])**.** Given a sporadic real-time task system, the *critical instant* of a task τ_k under uniprocessor preemptive fixed-priority scheduling occurs at the release of a job of τ_k when

1) every higher-priority task in $hp(\tau_k)$ releases a job simultaneously with the job of τ_k,
2) all subsequent jobs of the higher-priority tasks are released as early as possible by respecting their minimum inter-arrival times, i.e., periodically, and
3) every job is executed with its worst-case execution time.

Definition 6. (Critical Time Zone, reworded from [22]) With Definition 5, the *critical time zone* of task τ_k is the time interval between a critical instant and the finishing time of the job of τ_k released at the critical instant.

Theorem 7 (Worst-Case Response Time under the Critical Instant)**.** *Under uniprocessor fixed-priority preemptive scheduling, if the interval length of the critical time zone of task τ_k is no more than T_k, then the response time of this job of τ_k (i.e., the length of the critical time zone) is the worst-case response time of τ_k; otherwise, the worst-case response time of τ_k is greater than T_k.*

Although the statement is correct, the original proof of the critical instant theorem by Liu and Layland [22] was incomplete. Several patches are available, e.g., [2]. Here, we sketch the proof as it is used later to explain why the critical instant cannot be extended to derive WCDFP and WCRTEP.

Proof. Suppose that σ is the fixed-priority preemptive schedule of a set of jobs generated by the set $\mathbb{T}$ of sporadic tasks. In the schedule σ, if there is a job of task τ_k whose response time is greater than T_k, let job $\tau_{k,\ell}$ be the first job of them. Otherwise, let $\tau_{k,\ell}$ be any arbitrary job of τ_k. We denote the release time of job $\tau_{k,\ell}$ as $r_{k,\ell}$ and the finishing time of $\tau_{k,\ell}$ as $f_{k,\ell}$.

[Interval Extension]: Let t_0 be the earliest instant prior to $r_{k,\ell}$, i.e., $t_0 \leq r_{k,\ell}$, such that the processor only executes jobs generated by the higher-priority tasks in $hp(\tau_k)$ from t_0 to $r_{k,\ell}$ in the schedule σ. We now remove all jobs released before t_0. Let σ' be the resulting fixed-priority schedule of the remaining jobs. As the removal of such jobs has no impact on the schedule σ between t_0 and $f_{k,\ell}$, the two schedules σ and σ' are identical from t_0 to $f_{k,\ell}$.

[Release Time Modification]: According to the above definition, task τ_k is not executed from t_0 to $r_{k,\ell}$. Moving the release time of $\tau_{k,\ell}$ to t_0 does not change the schedule σ' but the response time of $\tau_{k,\ell}$ is increased by $r_{k,\ell} - t_0 \geq 0$.

[Simultaneous Release and Periodic Interference]: Since there is no job released prior to t_0 in the schedule σ', in order

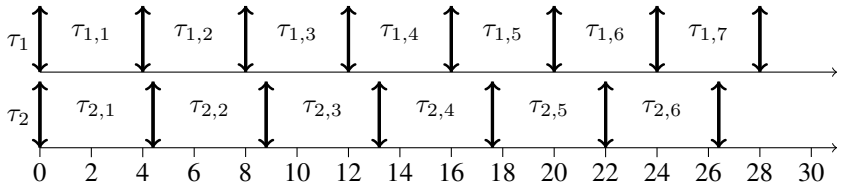


Fig. 1: Release pattern of the counterexample task set, where both tasks release synchronously at time 0.

to achieve the maximum interference, the jobs of $\tau_i \in hp(\tau_k)$ should be released as early as possible, starting from t_0. This implies that the first job of $\tau_i \in hp(\tau_k)$ should be released at time t_0, followed by subsequent jobs released periodically. Furthermore, every job runs for its worst-case execution time. That is, the critical instant in Definition 5 is at time t_0. Let σ'' be the resulting schedule and let $f''_{k,\ell}$ be the finishing time of $\tau_{k,\ell}$ in the schedule σ''. It can be proven that $f''_{k,\ell} \geq f_{k,\ell}$.

[Worst-Case Response Time]: Therefore, the response time of the job $\tau_{k,\ell}$ becomes $f''_{k,\ell} - t_0 \geq f_{k,\ell} - r_{k,\ell}$. This leads to the conclusion stated in Theorem 7. □

B. Critical Instant for WCDFP/WCRTEP & Counterexample

By following the critical instant theorem, Theorem 1 from [25] claims that the worst-case response time distribution of any job of a task occurs *for the first job* if the task set follows synchronous release. In the following, we provide a concrete counterexample to invalidate this statement. Note that implicit deadlines are assumed in [25]. For completeness, we first show their original theorem here:

Theorem 8 (From [25, Theorem 1])**.** *We consider a task system of n tasks with τ_i described by probabilistic C_i and T_i, $\forall i \in \{1, 2, \ldots, n\}$. The set is ordered according to the priorities of the tasks and the system is scheduled preemptively on a single processor. The response time distribution $R_{i,1}$ of the first job of task τ_i is greater than the response time distribution $R_{i,j}$ of any j-th job of task τ_i, $\forall i \in \{1, 2, \ldots, n\}$.*

It is assumed in [25] that all tasks are synchronous, i.e., every task releases its first job at time 0, following the critical instant theorem. That is, it is claimed that the WCRTEP is observed for $j = 1$ since $\mathbb{P}(R_{i,1} > t) \geq \mathbb{P}(R_{i,j} > t)$ for all $t \in \mathbb{R}$. Similarly, Theorem 1 from [8] states that the WCDFP of task τ_k can be derived under the critical instant.

Theorem 9 (Reworded from [8, Theorem 1])**.** *Suppose that S_t is the sum of the execution times of one job of τ_k and $\left\lceil \frac{t}{T_i} \right\rceil$ jobs of each higher priority task $\tau_i \in hp(\tau_k)$, i.e.,*

$$S_t := C_{k,1} + \sum_{\tau_i \in hp(\tau_k)} \sum_{q=1}^{\left\lceil \frac{t}{T_i} \right\rceil} C_{i,q}. \tag{5}$$

If the time-demand analysis (TDA) succeeds using the worst-case execution time of each task, then the probability of deadline misses of task τ_k is 0. Otherwise, the probability of deadline misses of task τ_k is upper bounded by

$$\inf_{0<t\leq D_k} \mathbb{P}(S_t \geq t), \tag{6}$$

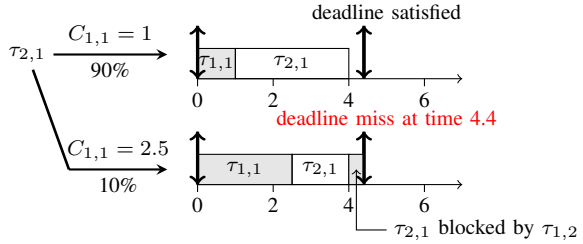


Fig. 2: State space for job $\tau_{2,1}$. Note that, since job $\tau_{2,1}$ misses its deadline at 4.4, it is aborted in the schedule below.

where inf *indicates the infimum within set theory.*

The statements in Theorem 8 and Theorem 9 are seemingly correct by simply applying the sketched proof of Theorem 7 considering probabilistic execution time. However, the analysis in the critical time zone does not consider the execution time distribution because only the worst-case execution time is needed for analyzing the worst-case response time.

The key issue is whether the interval extension from $r_{k,\ell}$ to t_0 in the proof of Theorem 7 may change the response time distribution of $\tau_{k,\ell}$. Considering the execution time distribution of the higher-priority jobs, the interval extension from $r_{k,\ell}$ to t_0 is not deterministic and is related to the probability of the execution times of the higher-priority tasks. Therefore, ignoring the probability of the feasibility of the interval extension may result in incorrect quantification of response time distribution.

This observation leads to the following counterexample:

Counterexample 10. Consider the periodic task set with two tasks τ_1 and τ_2, simultaneously released at time 0:

- $T_1 = 4$, $\mathbb{P}(C_{1,j} = 1) = 0.9$, $\mathbb{P}(C_{1,j} = 2.5) = 0.1$
- $T_2 = 4.4$, $\mathbb{P}(C_{1,j} = 3) = 1.0$

for all $j \in \mathbb{N}$. The release pattern of that task set is presented in Figure 1. In the following, we show that the probability $\mathbb{P}(R_{2,6} > t)$ is higher than the probability $\mathbb{P}(R_{2,1} > t)$, for some t with $0 < t \leq T_2$. That is, the first job of task τ_2 does not always have a greater response time distribution than the other jobs of τ_2. Consider $t = 4.4$.

For job $\tau_{2,1}$ (as shown in Figure 2): If $\tau_{1,1}$ has an execution time of 1, then $\tau_{2,1}$ finishes at time $1 + 3 = 4$. If $\tau_{1,1}$ has an execution time of 2.5, then $\tau_{2,1}$ does not finish its execution at time 4.4. Therefore, we obtain

$$\mathbb{P}(R_{2,1} > 4.4) = \mathbb{P}(C_{1,1} = 2.5) = 0.1 \tag{7}$$

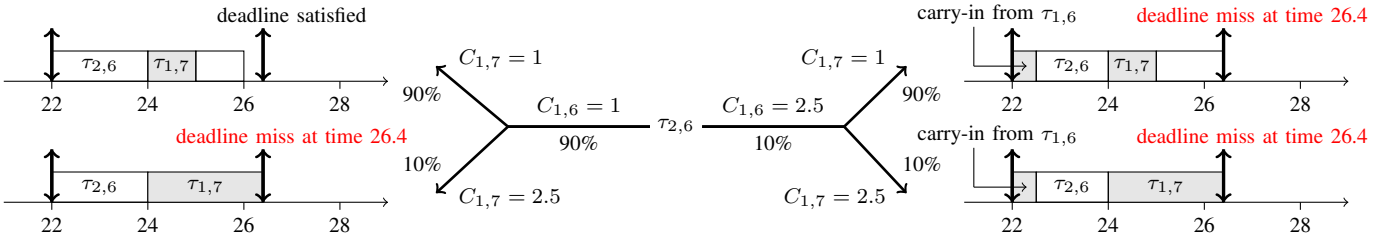


Fig. 3: State space for job $\tau_{2,6}$: For 3 out of 4 possible schedules in the time interval $[22, 26.4]$ job $\tau_{2,6}$ misses its deadline.

For job $\tau_{2,6}$ (as shown in Figure 3): If $\tau_{1,6}$ has an execution time of 1, it contributes no interference to $\tau_{2,6}$. In this case, the response time of $\tau_{2,6}$ is only larger than 4.4 if $\tau_{1,7}$ executes for 2.5 time units. If $\tau_{1,6}$ has an execution time of 2.5, then it contributes 0.5 time units as additional interference for $\tau_{2,6}$. In this case, the response time for $\tau_{2,6}$ is larger than 4.4 if $\tau_{1,7}$ executes for 1 time unit or for 2.5 time units. Hence, $R_{2,6} > 4.4$ if either $\tau_{1,6}$ or $\tau_{1,7}$ or both have an execution time of 2.5. Therefore, the probability that the response time of $\tau_{2,6}$ is larger than 4.4 is:

$$\mathbb{P}(R_{2,6} > 4.4) \tag{8}$$
$$= \mathbb{P}(C_{1,6} = 2.5) + \mathbb{P}(C_{1,6} = 1) \cdot \mathbb{P}(C_{1,7} = 2.5) \tag{9}$$
$$= 0.1 + 0.9 \cdot 0.1 = 0.19 \tag{10}$$

We get $\mathbb{P}(R_{2,1} > 4.4) = 0.1 < 0.19 = \mathbb{P}(R_{2,6} > 4.4)$, which contradicts Theorem 8. The counterexample invalidates Theorem 9 as well if we assume that τ_2 has an implicit deadline (i.e., $D_i = T_i = 4.4$), since according to the counterexample $\tau_{2,6}$ has a larger DFP than $\tau_{2,1}$ which contradicts that the WCDFP can be observed for the first job of τ_2. □

Consequence: With probabilistic execution times, the synchronous release of all tasks does not necessarily generate the maximum interference and is thus not always a critical instant. Therefore, Theorem 8 (i.e., Theorem 1 from [25]) and Theorem 9 (i.e., Theorem 1 from [8]) may result in an unsound WCDFP, as well as unsound WCRTEP.

C. Detailing the Misconception

The proof in [8] follows a similar proof structure as in the worst-case analysis presented in the proof of Theorem 7, whereas the proof in [25] directly refers to the classical critical instant theorem. Essentially, to prove that the worst case is to synchronously release one job from every higher-priority task together with the job of task τ_k under analysis, *interval extension* is needed. However, as demonstrated in Figure 3, the extension from $r_{k,\ell}$ (namely, 22) to t_0 (namely, 20) is also a probabilistic distribution function, depending on the execution time of the higher-priority jobs. It is therefore unsound to simply extend the interval of interest without considering the change of the probabilistic distribution in this regard.

IV. Safe Bounds for WCDFP and WCRTEP

In this section, we provide two methods to derive a sound WCDFP and WCRTEP. We focus on WCRTEP in our presentation, as WCDFP is a special form of it. The first method in Section IV-A is based on the inclusion of one additional job of a higher-priority task in the analysis window, typically called *carry-in*. This method has been widely used when the critical instant theorem does not work, e.g., for multiprocessor global scheduling [3] and self-suspending task systems [7]. The second method in Section IV-B inflates the execution time of the jobs of higher priority tasks to account for the uncertainty of the analysis interval extension under different samples. We demonstrate that these two methods do not dominate each other by providing two concrete task sets, one where carry-in is better and one where inflation is better.

A. Method 1: Carry-in

It is sufficient to consider *one additional* job of each higher-priority task $\tau_i \in hp(\tau_k)$ (in addition to the critical instant) for the WCRTEP analysis of task τ_k.

Theorem 11. *Consider a system of constrained-deadline sporadic real-time tasks $\mathbb{T}$ where a job is directly aborted when missing its deadline. The WCRTEP of task $\tau_k \in \mathbb{T}$ for a response time target R_k with $0 < R_k \leq T_k$ under uniprocessor preemptive fixed-priority scheduling is upper bounded by*

$$\inf_{0<t\leq R_k} \mathbb{P}(S_t > t), \tag{11}$$

where S_t is a random variable which provides the sum of the execution times of one job of τ_k and $\left\lceil \frac{t+D_i}{T_i} \right\rceil$ jobs of a higher-priority task τ_i for each task τ_i in $hp(\tau_k)$, i.e.,

$$S_t := C_{k,1} + \sum_{\tau_i \in hp(\tau_k)} \sum_{q=1}^{\left\lceil \frac{t+D_i}{T_i} \right\rceil} C_{i,q}. \tag{12}$$

Proof. By Definition 2, the WCRTEP for R_k is given by $\sup_{j\in\mathbb{N}}\{\mathbb{P}(R_{k,j} > R_k)\}$. In the following we show that $\mathbb{P}(R_{k,j} > R_k) \leq \inf_{0<t\leq R_k} \mathbb{P}(S_t > t)$ for all $\tau_{k,j}, j \in \mathbb{N}$ and for all possible release patterns. To achieve this, we consider one arbitrary job $\tau_{k,\ell}$ of τ_k and one arbitrary but fixed release pattern, and show that $\mathbb{P}(R_{k,\ell} > R_k) \leq \inf_{0<t\leq R_k} \mathbb{P}(S_t > t)$ under this release pattern.

[Equivalent Formulation]: For each higher priority task τ_i in $hp(\tau_k)$ we define by $X_i(a, b)$ the amount of time that τ_i is executed on the processor in the interval $[a, b)$. The following equivalence holds: $R_{k,\ell} > R_k$ if and only if $R_{k,\ell} > t$ for all $t \in (0, R_k]$. Hence, $\mathbb{P}(R_{k,\ell} > R_k) = \inf_{0<t\leq R_k} \mathbb{P}(R_{k,\ell} > t)$. If the job $\tau_{k,\ell}$ is not finished by time $r_{k,\ell} + t$, then from $r_{k,\ell}$ to $r_{k,\ell} + t$ the processor either executes $\tau_{k,\ell}$ or higher-priority

jobs of the tasks in $hp(\tau_k)$ due to preemptive fixed-priority scheduling. Therefore, we know that $R_{k,\ell} > t$ if and only if $\tau_{k,\ell}$ cannot be executed for $C_{k,\ell}$ time units in the interval $[r_{k,\ell}, r_{k,\ell}+t)$, i.e., $C_{k,\ell} + \sum_{\tau_i \in hp(\tau_k)} X_i(r_{k,\ell}, r_{k,\ell}+t) > t$. We conclude the equivalent formulation for $\mathbb{P}(R_{k,\ell} > R_k)$:

$$\mathbb{P}(R_{k,\ell} > R_k) \tag{13}$$

$$= \inf_{0<t\leq R_k} \mathbb{P}(C_{k,\ell} + \sum_{\tau_i \in hp(\tau_k)} X_i(r_{k,\ell}, r_{k,\ell}+t) > t) \tag{14}$$

[Quantification of Interference]: In this paragraph, we provide an upper bound on the interference $X_i(r_{k,\ell}, r_{k,\ell}+t)$. Any job of τ_i that is released before $r_{k,\ell} - D_i$ is either aborted or finished at time $r_{k,\ell}$. Therefore, only jobs of τ_i that are released in $[r_{k,\ell}-D_i, r_{k,\ell}+t)$ can be executed in $[r_{k,\ell}, r_{k,\ell}+t)$. The number of jobs of a sporadic task τ_i released inside an interval of length $t+D_i$ is at most $\left\lceil \frac{t+D_i}{T_i} \right\rceil$. Therefore, at most $\left\lceil \frac{t+D_i}{T_i} \right\rceil$ many jobs of τ_i can be executed in $[r_{k,\ell}, r_{k,\ell}+t)$.

Let τ_{i,j_i} be the first job of τ_i released at or after $r_{k,\ell} - D_i$.[1] Then only the jobs $\tau_{i,j_i}, \ldots, \tau_{i,j_i-1+\left\lceil \frac{t+D_i}{T_i} \right\rceil}$ of τ_i can be executed in $[r_{k,\ell}, r_{k,\ell}+t)$. This means that

$$X_i(r_{k,\ell}, r_{k,\ell}+t) \leq \sum_{q=j_i}^{j_i-1+\left\lceil \frac{t+D_i}{T_i} \right\rceil} C_{i,q}. \tag{15}$$

We conclude that $\mathbb{P}(R_{k,\ell} > R_k)$ is upper bounded by

$$\inf_{0<t\leq R_k} \mathbb{P}\left(C_{k,\ell} + \sum_{\tau_i \in hp(\tau_k)} \sum_{q=j_i}^{j_i-1+\left\lceil \frac{t+D_i}{T_i} \right\rceil} C_{i,q} > t \right). \tag{16}$$

[Exploit iid]: Since the random variable $C_{k,\ell}$ has the same probability distribution as $C_{k,1}$, we can replace $C_{k,\ell}$ by $C_{k,1}$ in Equation (16). Moreover, $C_{i,j_i}, \ldots, C_{i,j_i-1+\left\lceil \frac{t+D_i}{T_i} \right\rceil}$ ($C_{i,1}, \ldots, C_{i,\left\lceil \frac{t+D_i}{T_i} \right\rceil}$, respectively) are iid with the same probability distribution as C_i. Therefore, $\sum_{q=j_i}^{j_i-1+\left\lceil \frac{t+D_i}{T_i} \right\rceil} C_{i,q}$ and $\sum_{q=1}^{\left\lceil \frac{t+D_i}{T_i} \right\rceil} C_{i,q}$ have the same probability distribution for all $\tau_i \in hp(\tau_k)$. Hence, the random variables $C_{k,\ell} + \sum_{\tau_i \in hp(\tau_k)} \sum_{q=j_i}^{j_i-1+\left\lceil \frac{t+D_i}{T_i} \right\rceil} C_{i,q}$ and $S_t = C_{k,1} + \sum_{\tau_i \in hp(\tau_k)} \sum_{q=1}^{\left\lceil \frac{t+D_i}{T_i} \right\rceil} C_{i,q}$ have the same probability distribution, and $\mathbb{P}(C_{k,\ell} + \sum_{\tau_i \in hp(\tau_k)} \sum_{q=j_i}^{j_i-1+\left\lceil \frac{t+D_i}{T_i} \right\rceil} C_{i,q} > t)$ coincides with $\mathbb{P}(S_t > t)$. We apply this to Equation (16):

$$\mathbb{P}(R_{k,\ell} > R_k) \leq \inf_{0<t\leq R_k} \mathbb{P}(S_t > t) \tag{17}$$

[Closing remarks]: We have shown that Equation (17) holds for an arbitrary release pattern and therefore it holds for all possible release patterns. We can apply $\sup_{\ell \in \mathbb{N}}$ to obtain the statement from the theorem. □

[1]Please note that j_i is only dependent on the release pattern and not on the probabilistic behavior.

Corollary 12. *Under the same setup as in Theorem 11, the WCDFP of task τ_k under uniprocessor preemptive fixed-priority scheduling is at most*

$$\inf_{0<t\leq D_k} \mathbb{P}(S_t > t). \tag{18}$$

B. Method 2: Inflation

The second method in this section quantifies the interference of the higher-priority tasks by considering only $\left\lceil \frac{t}{T_i} \right\rceil$ jobs of task τ_i. However, the counterexample in Section III shows that including $\left\lceil \frac{t}{T_i} \right\rceil$ random variables of C_i is unsound and that some additional jobs have to be considered. Our method conquers this issue by considering the *inflation* of the execution time of some jobs in the random variables to ensure that the WCRTEP or the WCDFP is correctly bounded.

The idea behind inflation is as follows: Suppose that we have to consider λ_i^t jobs of τ_i in the analysis window (to be defined later). We randomly sample λ_i^t random variables of C_i but only the *largest* $\left\lceil \frac{t}{T_i} \right\rceil$ of them are considered in the response time analysis. More formally,

Definition 13. For any positive integers a and b with $a \leq b$, a-job execution time inflation of τ_i out of b jobs is a sampling process which first *samples* b random variables of the execution time of task τ_i and then *selects* the a *largest* values among the samples as the inflated execution time. We denote by $SAI(\tau_i, a, b)$ (named after *sample and inflate*) the random variable which provides the sum of the a selected samples. The random variable can be formulated as:

$$SAI(\tau_i, a, b) = \max \left\{ \sum_{c \in S} c \;\middle|\; S \subseteq \{C_{i,1}, \ldots, C_{i,b}\}, |S| = a \right\} \tag{19}$$

The following theorem shows that it is sound to sample $\left\lceil \frac{t+\sum_{\tau_\xi \in hp(\tau_k) \setminus hp(\tau_i)} D_\xi}{T_i} \right\rceil$ jobs and inflate $\left\lceil \frac{t}{T_i} \right\rceil$ jobs of a higher-priority task τ_i when analyzing the WCRTEP of τ_k.

Theorem 14. *Consider a system of constrained-deadline sporadic real-time tasks where a job is directly aborted when missing its deadline. The WCRTEP of task τ_k for a response time $0 < R_k \leq T_k$ under uniprocessor preemptive fixed-priority scheduling is at most*

$$\inf_{0<t\leq R_k} \mathbb{P}\left(C_k + \sum_{\tau_i \in hp(\tau_k)} SAI\left(\tau_i, \left\lceil \frac{t}{T_i} \right\rceil, \lambda_i^t\right) > t \right), \tag{20}$$

where λ_i^t is $\left\lceil \frac{t+\sum_{\tau_\xi \in hp(\tau_k) \setminus hp(\tau_i)} D_\xi}{T_i} \right\rceil$.

Proof. The proof of this theorem is very similar to the proof of Theorem 11 by further exploiting the concept of *Interval Extension* and *Release Time Modification* used in the proof of the critical instant theorem in Theorem 7. Under the probabilistic setting, after the interval extension and release time modification, we show how to safely quantify the interference by using the concept of sample and inflate.

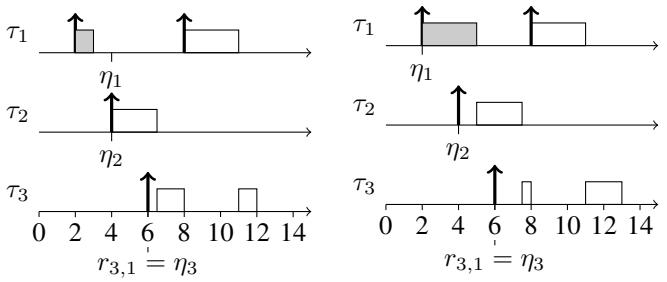


Fig. 4: Interval extension in the proof of Theorem 14 for two different execution scenarios (left: $C_{1,1} = 1$, right: $C_{1,1} = 3$). We observe that the changes in the random variable η_1 are dependent on the sample under analysis.

By Definition 2, the WCRTEP for R_k is given by $\sup_{j\in\mathbb{N}}\{\mathbb{P}(R_{k,j} > R_k)\}$. In the following we show that $\mathbb{P}(R_{k,j} > R_k)$ is upper bounded by (20) for all $\tau_{k,j}, j \in \mathbb{N}$ and for all possible release patterns. To achieve this, we consider one arbitrary job $\tau_{k,\ell}$ of τ_k and one arbitrary but fixed release pattern, and show that $\mathbb{P}(R_{k,\ell} > R_k)$ is upper bounded by (20) under that release pattern.

[Equivalent Formulation]: As in the proof of Theorem 11, the probability for $R_{k,\ell} > R_k$ is equivalently formulated by

$$\mathbb{P}(R_{k,\ell} > R_k) = \inf_{0<t\le R_k} \mathbb{P}\Big(C_{k,\ell} + \sum_{\tau_i\in hp(\tau_k)} X_i(r_{k,\ell}, r_{k,\ell}+t) > t\Big), \tag{21}$$

where $X_i(a,b)$ is the amount of execution time of jobs of τ_i in the interval $[a,b)$.

[Interval Extension]: Similar to the proof of the worst-case response time in Theorem 7, we extend the analysis interval to the left side, i.e., we define η_1 such that all jobs released before η_1 are not relevant for the analysis. We construct the time points $\eta_1, \dots, \eta_k$ in an iterative manner, starting from $i = k-1, k-2, \dots, 1$. Let η_k be $r_{k,\ell}$. After η_{i+1} is determined, we define η_i as follows:

$$\eta_i := \min(\eta_{i+1}, r_{i,\phi_i}), \tag{22}$$

where τ_{i,ϕ_i} the first job of τ_i that is executed after η_{i+1}, i.e.,

$$\phi_i := \min\{j \in \mathbb{N} \,|\, \tau_{i,j} \text{ executed after } \eta_{i+1}\}. \tag{23}$$

In Figure 4 the interval extension is presented for a task set with three tasks. Note that $\eta_1, \eta_2, \dots, \eta_{k-1}$ are random variables that depend on the execution behavior. With this definition of $\eta_1, \dots, \eta_k$ the following properties hold:

- P1 During $[\eta_1, \eta_i)$ the processor is busy executing jobs of $hp(\tau_i)$. (That is because if $\eta_{i-1} < \eta_i$, then the processor executes jobs of $hp(\tau_i)$ during $[\eta_{i-1}, \eta_i)$.)
- P2 None of the jobs of $\tau_i \in hp(\tau_k)$ released before η_i is executed after η_i. (If $\eta_i = \eta_{i+1}$ then there are by construction no jobs of τ_i released before η_{i+1} and executed after η_{i+1}. If $\eta_i = r_{i,\phi_i}$, then under the constrained deadline setup all previous jobs of τ_i have their deadline at η_i or earlier. Hence, they must be finished or aborted by η_i.)
- P3 $\eta_i + D_i > \eta_{i+1}$ due to the assumption in the system that a job is aborted after its deadline miss.

Please note that none of the jobs of $\tau_i \in hp(\tau_k)$ released before η_i is executed after η_1. (Otherwise, they have to finish by η_i because of Property P2. However, the jobs of τ_i cannot be executed in $[\eta_1, \eta_i)$ because of Property P1.) Therefore, jobs released before η_1 are not relevant for the analysis.

[Release Time Modification]: In the following we move the release of $\tau_{k,\ell}$ to η_1 for the analysis, i.e., we show that

$$\sum_{\tau_i\in hp(\tau_k)} X_i(r_{k,\ell}, r_{k,\ell}+t) \le \sum_{\tau_i\in hp(\tau_k)} X_i(\eta_1, \eta_1+t). \tag{24}$$

According to Property P1, during $[\eta_1, r_{k,\ell}) = [\eta_1, \eta_k)$ the processor is busy executing jobs of tasks of $hp(\tau_k)$. Therefore, $\sum_{\tau_i\in hp(\tau_k)} X_i(\eta_1, \eta_k) = \eta_k - \eta_1$ holds. Moreover, $\eta_k - \eta_1 \ge \sum_{\tau_i\in hp(\tau_k)} X_i(\eta_1+t, \eta_k+t)$ since the processor can only execute jobs for at most $\eta_k - \eta_1$ time units in the interval $[\eta_1 + t, \eta_k + t)$. Hence,

$$\sum_{\tau_i\in hp(\tau_k)} X_i(\eta_1, \eta_k) = \eta_k - \eta_1 \ge \sum_{\tau_i\in hp(\tau_k)} X_i(\eta_1+t, \eta_k+t) \tag{25}$$

holds. Since by Equation (25) $\sum_{\tau_i\in hp(\tau_k)} X_i(\eta_1, \eta_k) - \sum_{\tau_i\in hp(\tau_k)} X_i(\eta_1+t, \eta_k+t) \ge 0$, we obtain the following:

$$\sum_{\tau_i\in hp(\tau_k)} X_i(r_{k,\ell}, r_{k,\ell}+t) = \sum_{\tau_i\in hp(\tau_k)} X_i(\eta_k, \eta_k+t) \tag{26}$$

$$\le \left(\sum_{\tau_i\in hp(\tau_k)} X_i(\eta_k, \eta_k+t) + \sum_{\tau_i\in hp(\tau_k)} X_i(\eta_1, \eta_k) - \sum_{\tau_i\in hp(\tau_k)} X_i(\eta_1+t, \eta_k+t) \right) \tag{27}$$

$$= \sum_{\tau_i\in hp(\tau_k)} X_i(\eta_k, \eta_k+t) + X_i(\eta_1, \eta_k) - X_i(\eta_1+t, \eta_k+t) \tag{28}$$

$$= \sum_{\tau_i\in hp(\tau_k)} X_i(\eta_1, \eta_k+t) - X_i(\eta_1+t, \eta_k+t) \tag{29}$$

$$= \sum_{\tau_i\in hp(\tau_k)} X_i(\eta_1, \eta_1+t) \tag{30}$$

Note that the conversion from (28) to (29) and from (29) to (30) are both due to the fact that $X_i(a,b) + X_i(b,c) = X_i(a,c)$ for all real numbers $a \le b \le c$. This proves Equation (24).

[Quantification of Interference]: In the following we quantify the interference from higher priority task $\tau_i \in hp(\tau_k)$, which is $X_i(\eta_1, \eta_1+t)$. For all $\tau_i \in hp(\tau_k)$, during $[\eta_1, \eta_i)$ only jobs of $hp(\tau_i)$ are executed by Property P1. Therefore, $X_i(\eta_1, \eta_i) = 0$ and

$$X_i(\eta_1, \eta_1+t) \le X_i(\eta_1, \eta_i+t) = X_i(\eta_i, \eta_i+t). \tag{31}$$

By Property P2 only those jobs of τ_i that are released in $[\eta_i, \eta_i+t)$ can be executed in $[\eta_i, \eta_i+t)$. These are at most $\left\lceil \frac{t}{T_i} \right\rceil$ jobs. However, since η_i is a random variable dependent

on the execution behavior, it is not clear *which* jobs have to be included in the analysis.[2]

In the following we derive a set $\mathbb{J}^i$ of all jobs that may be released in $[\eta_i, \eta_i + t)$ under any execution behavior, i.e., the set is only dependent on the release pattern. By Property P3, η_i is lower bounded by $r_{k,\ell} - \sum_{\tau_\xi \in hp(\tau_k) \setminus hp(\tau_i)} D_\xi$. Hence, the interval $[\eta_i, \eta_i + t)$ is a subset of the interval $[r_{k,\ell} - \sum_{\tau_\xi \in hp(\tau_k) \setminus hp(\tau_i)} D_\xi, r_{k,\ell} + t)$. Let τ_{i,j_i} be the first job of τ_i that is released at or after $r_{k,\ell} - \sum_{\tau_\xi \in hp(\tau_k) \setminus hp(\tau_i)} D_\xi$. Please note that j_i is only dependent on the release pattern and independent from the execution behavior. Since at most $\lambda_i^t = \left\lceil \frac{t + \sum_{\tau_\xi \in hp(\tau_k) \setminus hp(\tau_i)} D_\xi}{T_i} \right\rceil$ jobs of τ_i are released in $[r_{k,\ell} - \sum_{\tau_\xi \in hp(\tau_k) \setminus hp(\tau_i)} D_\xi, r_{k,\ell} + t)$, only the jobs

$$\mathbb{J}^i = \left\{ \tau_{i,j_i}, \ldots, \tau_{i,j_i - 1 + \lambda_i^t} \right\} \tag{32}$$

can be released in $[r_{k,\ell} - \sum_{\tau_\xi \in hp(\tau_k) \setminus hp(\tau_i)} D_\xi, r_{k,\ell} + t)$. Hence, only jobs of $\mathbb{J}^i$ may be released in $[\eta_i, \eta_i + t)$.

As discussed earlier, by Property P2 only jobs of τ_i that are released in $[\eta_i, \eta_i + t)$ can be executed in $[\eta_i, \eta_i + t)$, i.e., at most $\left\lceil \frac{t}{T_i} \right\rceil$ jobs. Hence, only the execution time of $\left\lceil \frac{t}{T_i} \right\rceil$ jobs in $\mathbb{J}^i$ has to be considered for the interference. We conclude

$$X_i(\eta_i, \eta_i + t) \leq MAX_i^t \tag{33}$$

where MAX_i^t is the random variable that returns the sum of the $\left\lceil \frac{t}{T_i} \right\rceil$ maximal execution times of the jobs in $\mathbb{J}^i$. After the quantification step, we obtain that

$$\mathbb{P}(R_{k,\ell} > R_k) \leq \inf_{0 < t \leq R_k} \mathbb{P}\Big(C_{k,\ell} + \sum_{\tau_i \in hp(\tau_k)} MAX_i^t \Big), \tag{34}$$

which is a combination of Equations (21), (24), (31) and (33).

[Exploit iid]: Since the random variable $C_{k,\ell}$ has the same probability distribution as $C_{k,1}$, we can replace $C_{k,\ell}$ by $C_{k,1}$ in Equation (34). Moreover, $C_{i,j_i}, \ldots, C_{i,j_i - 1 + \lambda_i^t}$ ($C_{i,1}, \ldots, C_{i,\lambda_i^t}$, respectively) are iid with the same probability distribution as C_i. Therefore, MAX_i^t and $SAI(\tau_i, \left\lceil \frac{t}{T_i} \right\rceil, \lambda_i^t)$ have the same probability distribution for all $\tau_i \in hp(\tau_k)$. We conclude that $C_{k,\ell} + \sum_{\tau_i \in hp(\tau_k)} MAX_i^t$ and $C_{k,1} + \sum_{\tau_i \in hp(\tau_k)} SAI(\tau_i, \left\lceil \frac{t}{T_i} \right\rceil, \lambda_i^t)$ have the same probability distribution, and therefore $\mathbb{P}(C_{k,\ell} + \sum_{\tau_i \in hp(\tau_k)} MAX_i^t) = \mathbb{P}(C_{k,1} + \sum_{\tau_i \in hp(\tau_k)} SAI(\tau_i, \left\lceil \frac{t}{T_i} \right\rceil, \lambda_i^t))$ holds. Applying this to Equation (34) yields

$$\mathbb{P}(R_{k,\ell} > R_k) \leq \inf_{0 < t \leq R_k} \mathbb{P}\left(C_k + \sum_{\tau_i \in hp(\tau_k)} SAI\left(\tau_i, \left\lceil \frac{t}{T_i} \right\rceil, \lambda_i^t\right) > t \right). \tag{35}$$

[Closing remarks]: We have shown that Equation (35) holds for an arbitrary release pattern and therefore it holds for all possible release patterns. We can apply $\sup_{\ell \in \mathbb{N}}$ to obtain the statement from the theorem. □

[2]Please note that in the proof of Theorem 11 the jobs in the analysis, i.e., $\tau_{i,j_i}, \ldots, \tau_{i,j_i - 1 + \left\lceil \frac{t + D_i}{T_i} \right\rceil}$, can be determined directly from the release pattern. In this proof, the release time modification, i.e., moving the analysis interval to η_i, is dependent on the execution behavior.

Corollary 15. *Under the same setup of Theorem 14, the WCDFP of task τ_k under uniprocessor preemptive fixed-priority scheduling is at most*

$$\inf_{0 < t \leq D_k} \mathbb{P}\left(C_k + \sum_{\tau_i \in hp(\tau_k)} SAI\left(\tau_i, \left\lceil \frac{t}{T_i} \right\rceil, \lambda_i^t\right) > t \right). \tag{36}$$

C. Carry-In versus Inflation

We provide two examples showing that the carry-in and the inflation method do not dominate each other.

Case 1 – inflation outperforms carry-in: We re-examine the counterexample in Section III. In particular, we consider the task set $\mathbb{T} = \{\tau_1, \tau_2\}$ with two tasks described by:

- $T_1 = D_1 = 4$, $\mathbb{P}(C_1 = 1) = 0.9$, $\mathbb{P}(C_1 = 2.5) = 0.1$
- $T_2 = D_2 = 4.4$, $\mathbb{P}(C_2 = 3) = 1.0$

Task $\tau_k := \tau_2$ is under analysis. It is shown in Counterexample 10 that the WCDFP for τ_2 is at least 0.19.

First, we compute an upper bound on the WCDFP by applying the *carry-in* method. By Corollary 12, the WCDFP of τ_2 is upper bounded by

$$\inf_{0 < t \leq D_2} \mathbb{P}(S_t > t), \tag{37}$$

where S_t is the sum of the execution time of one job of τ_2 and $\left\lceil \frac{t + D_1}{T_1} \right\rceil$ jobs of the higher priority task τ_1. For all $t \in (0, D_2] = (0, 4.4]$, we have $\left\lceil \frac{t + D_1}{T_1} \right\rceil = \left\lceil \frac{t+4}{4} \right\rceil \geq 2$. Since the execution time of every job of τ_2 is 3 and the execution time of jobs of τ_1 is at least 1, we obtain $S_t \geq 5 > t$ for all $t \in (0, D_2]$. Hence, $\inf_{0 < t \leq D_2} \mathbb{P}(S_t > t) = 1.0$ and the carry-in method states that the WCDFP of τ_2 is at most 1.0.

In this case, the *inflation* method provides a tighter result. By Corollary 15, the WCDFP of τ_2 is upper bounded by

$$\inf_{0 < t \leq D_2} \mathbb{P}\left(C_2 + SAI(\tau_1, \left\lceil \frac{t}{T_1} \right\rceil, \lambda_1^t) > t \right), \tag{38}$$

where $\lambda_1^t = \left\lceil \frac{t + D_1}{T_1} \right\rceil$. The values of $\left\lceil \frac{t}{T_1} \right\rceil$ and λ_1^t are dependent on t as follows:

$$\left\lceil \frac{t}{T_1} \right\rceil = \begin{cases} 1 & t \in (0, 4] \\ 2 & t \in (4, 4.4] \end{cases} \qquad \lambda_1^t = \begin{cases} 2 & t \in (0, 4] \\ 3 & t \in (4, 4.4] \end{cases} \tag{39}$$

i.e., checking Equation (38) for $t = 4$ and $t = 4.4$ is sufficient.

For $t = 4$, the random variable $SAI\left(\tau_1, \left\lceil \frac{t}{T_1} \right\rceil, \lambda_1^t\right)$ is $SAI(\tau_1, 1, 2)$ which returns 1 if both sampled jobs have execution of 1, i.e., with probability $0.9 \cdot 0.9 = 0.81$, and 2.5 with probability $1 - 0.81 = 0.19$. If $SAI(\tau_1, 1, 2)$ returns 1, then $C_2 + SAI(\tau_1, 1, 2)$ returns 4 which is $\leq t = 4$. If $SAI(\tau_1, 1, 2)$ returns 2.5, then $C_2 + SAI(\tau_1, 1, 2)$ returns 5.5 which is $> t = 4$. Hence, the probability $\mathbb{P}(C_2 + SAI(\tau_1, \left\lceil \frac{t}{T_1} \right\rceil, \lambda_1^t) > t)$ for $t = 4$ is 0.19.

For $t = 4.4$, $SAI(\tau_1, \left\lceil \frac{t}{T_1} \right\rceil, \lambda_1^t)$ is $SAI(\tau_1, 2, 3)$ which returns at least 2. Therefore, $C_2 + SAI(\tau_1, 1, 2)$ returns at

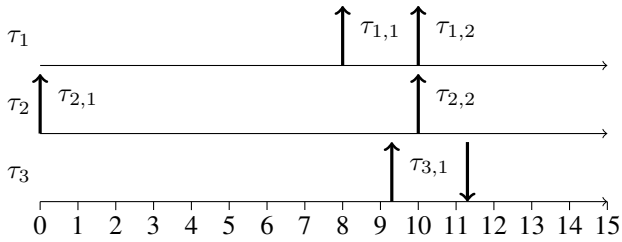


Fig. 5: Exemplary release pattern of the task set where the carry-in method outperforms the inflation method. Release times of the jobs are as follows: $r_{1,1} = 8$, $r_{1,2} = 10$, $r_{2,1} = 0$, $r_{2,2} = 10$, and $r_{3,1} = 9.3$.

least 5 which is $> t = 4.4$ in all cases. Thus, the probability $\mathbb{P}(C_2 + SAI(\tau_1, \left\lceil \frac{t}{T_1} \right\rceil, \lambda_1^t) > t)$ for $t = 4.4$ is 1.0.

The WCDFP from the inflation method is at most 0.19, which is the lower bound from the counterexample, i.e., the inflation method is exact for this example.

In fact, with the following theorem, Theorem 14 is always better than Theorem 11 when considering only two tasks.

Theorem 16. *When there are only two tasks in* $\mathbb{T}$*,*

$$\mathbb{P}\left(C_{2,1} + SAI\left(\tau_1, \left\lceil \frac{t}{T_1} \right\rceil, \left\lceil \frac{t+D_1}{T_1} \right\rceil\right) > t\right) \leq \mathbb{P}\left(S_t > t\right) \tag{40}$$

for all t *with* $0 < t \leq D_2$*. That is, Theorem 14 dominates Theorem 11 when there are only two tasks in* $\mathbb{T}$*.*

Proof. By definition, since $\lambda_1^t = \left\lceil \frac{t+D_1}{T_1} \right\rceil \geq \left\lceil \frac{t}{T_1} \right\rceil$, we have

$$\begin{aligned}
&\mathbb{P}\left(C_{2,1} + SAI\left(\tau_1, \left\lceil \frac{t}{T_1} \right\rceil, \left\lceil \frac{t+D_1}{T_1} \right\rceil\right) > t\right) \\
&\quad \leq \mathbb{P}\left(C_{2,1} + SAI\left(\tau_1, \left\lceil \frac{t+D_1}{T_1} \right\rceil, \left\lceil \frac{t+D_1}{T_1} \right\rceil\right) > t\right) \\
&\quad = \mathbb{P}\Big(C_{2,1} + \sum_{q=1}^{\left\lceil \frac{t+D_1}{T_1} \right\rceil} C_{1,q} > t\Big) = \mathbb{P}(S_t > t),
\end{aligned}$$

which leads to the condition in Equation (40). □

Case 2 – carry-in outperforms inflation: We consider the scenario with 3 tasks $\mathbb{T} = \{\tau_1, \tau_2, \tau_3\}$ described by:

- $T_1 = D_1 = 2$, $\mathbb{P}(C_1 = 0.2) = 0.9$, $\mathbb{P}(C_1 = 2) = 0.1$
- $T_2 = D_2 = 10$, $\mathbb{P}(C_2 = 0.2) = 0.9$, $\mathbb{P}(C_2 = 10) = 0.1$
- $T_3 = D_3 = 2$, $\mathbb{P}(C_3 = 1) = 1.0$

Task $\tau_k := \tau_3$ is under analysis. For a WCDFP lower bound we consider the release pattern shown in Figure 5. Under this release pattern, job $\tau_{3,1}$ has a deadline miss if and only if at least one of the jobs $\tau_{1,1}$, $\tau_{1,2}$, $\tau_{2,1}$, and $\tau_{2,2}$ executes with the worst-case execution time. The WCDFP of τ_3 is therefore lower bounded by $1 - 0.9 \cdot 0.9 \cdot 0.9 \cdot 0.9 = 0.3439$.

First, we compute an upper bound on the WCDFP by applying the *carry-in* method. By Corollary 12, the WCDFP of τ_3 is upper bounded by $\inf_{0<t\leq D_3} \mathbb{P}(S_t > t)$, where S_t is the sum of the execution time of one job of τ_3, $\left\lceil \frac{t+D_2}{T_2} \right\rceil$ jobs of the higher priority task τ_2, and $\left\lceil \frac{t+D_1}{T_1} \right\rceil$ jobs of the higher priority task τ_1. For all $t \in (0, D_3] = (0, 2]$, $\left\lceil \frac{t+D_2}{T_2} \right\rceil = \left\lceil \frac{t+10}{10} \right\rceil = 2$ and $\left\lceil \frac{t+D_1}{T_1} \right\rceil = \left\lceil \frac{t+2}{2} \right\rceil = 2$. Therefore, S_t returns the sum of C_3, two samples of C_2, and two samples of C_1. We only get $S_t \leq 2$ if both jobs of C_2 and both jobs of C_1 execute the smaller execution time. This case occurs with probability 0.9^4. Hence, $\inf_{0<t\leq D_3} \mathbb{P}(S_t > t) = 1 - 0.9^4$, i.e., the upper bound on the WCDFP of τ_3 obtained by the carry-in method is exact.

Second, for the *inflation* method, by Corollary 15 the WCDFP of τ_3 is upper bounded by

$$\inf_{0<t\leq D_3} \mathbb{P}\left(\begin{array}{c} C_{3,1} + SAI\left(\tau_2, \left\lceil \frac{t}{T_2} \right\rceil, \lambda_2^t\right) \\ +SAI\left(\tau_1, \left\lceil \frac{t}{T_1} \right\rceil, \lambda_1^t\right) > t \end{array}\right), \tag{41}$$

where

$$\lambda_i^t = \left\lceil \frac{t + \sum_{\tau_\xi \in hp(\tau_k) \setminus hp(\tau_i)} D_\xi}{T_i} \right\rceil \text{ for } i = 1, 2.$$

For all $t \in (0, D_3] = (0, 2]$, we have $\left\lceil \frac{t}{T_2} \right\rceil = \left\lceil \frac{t}{10} \right\rceil = 1$, $\left\lceil \frac{t}{T_1} \right\rceil = \left\lceil \frac{t}{2} \right\rceil = 1$, and

$$\begin{aligned}
\lambda_2^t &= \left\lceil \frac{t + D_2}{T_2} \right\rceil = \left\lceil \frac{t + 10}{10} \right\rceil = 2, \\
\lambda_1^t &= \left\lceil \frac{t + D_2 + D_1}{T_1} \right\rceil = \left\lceil \frac{t + 10 + 2}{2} \right\rceil = 7.
\end{aligned}$$

Therefore, the probability from Equation (41) can be formulated as $\mathbb{P}(C_{3,1} + SAI(\tau_2, 1, 2) + SAI(\tau_1, 1, 7) > 2)$. The random variable $SAI(\tau_2, 1, 2)$ returns the maximum of 2 samples of C_2, and the random variable $SAI(\tau_1, 1, 7)$ returns the maximum of 7 samples of C_1. Therefore, $SAI(\tau_2, 1, 2)$ is 0.2 if both samples of C_2 are 0.2, i.e., with probability 0.9^2, and 2 otherwise. Moreover, $SAI(\tau_1, 1, 7)$ is 0.2 if all 7 samples of C_1 are 0.2, i.e., with probability 0.9^7, and 10 otherwise. The random variable $C_{3,1} + SAI(\tau_2, 1, 2) + SAI(\tau_1, 1, 7)$ is ≤ 2, if and only if $SAI(\tau_2, 1, 2) = 0.2$ and $SAI(\tau_1, 1, 7) = 0.2$, i.e., with probability 0.9^9. Hence, the WCDFP of τ_3 is upper bounded by $\mathbb{P}(C_{3,1} + SAI(\tau_2, 1, 2) + SAI(\tau_1, 1, 7) > 2) = 1 - 0.9^9$ which is around 0.61. For this case, the upper bound on the WCDFP obtained by the inflation method is almost twice as large as the WCDFP obtained by the carry-in method.

D. SAI for Bernoulli Distributed Execution Time

In the following, we formulate a closed form for $SAI(\tau_i, a, b)$ for natural number $a \leq b$ if the execution time of τ_i is Bernoulli distributed, i.e.,

$$C_i = \begin{cases} c_1 & \text{with probability } p \\ c_2 & \text{with probability } 1 - p \end{cases} \tag{42}$$

for real values $c_1 > c_2$. Please note that any execution time distribution can be over-approximated by a Bernoulli distributed random variable. In such a case, the closed form for SAI is an over-approximation as well and can still be

utilized for the computation of the inflation-based method in Section IV-B.

Since C_i follows a Bernoulli distribution, the sequence of b samples of C_i is binomial distributed, i.e., for all $\xi \in \{0, \dots, b\}$ the probability that exactly ξ of the b samples of C_i are c_1 is $\binom{b}{\xi} p^{\xi}(1-p)^{b-\xi}$. Hence, SAI follows the probability distribution given by:

$$SAI(\tau_i, a, b) = \begin{cases} 0 \cdot c_1 + a \cdot c_2 & \text{with prob. } \binom{b}{0} p^0 (1-p)^b \\ 1 \cdot c_1 + (a-1) \cdot c_2 & \text{with prob. } \binom{b}{1} p^1 (1-p)^{b-1} \\ \quad \vdots & \\ (a-1) \cdot c_1 + 1 \cdot c_2 & \text{with prob. } \binom{b}{a-1} p^{a-1} (1-p)^{b-a+1} \\ a \cdot c_1 + 0 \cdot c_2 & \text{with the remaining prob.} \end{cases} \tag{43}$$

V. Influence on Related Work

Probabilistic response time analysis suffers from high computational complexity, as it is comprised of two inherently difficult problems:

1) How to efficiently bound the Response Time Exceedance Probability (RTEP) or the Deadline Failure Probability (DFP) of a specific job for a given release pattern (since a direct calculation via job-level convolution is intractable as time and space complexity are exponential with respect to the number of jobs in the pattern).
2) How to safely reduce the number of jobs/release patterns that must be considered in the analysis, as otherwise at least all jobs in the hyperperiod must be considered (see [36] for more detailed discussions).

Therefore, for an efficient probabilistic response time or deadline failure probability analysis, solutions for both problems must be provided.

The results by Maxim and Cucu-Grosjean [25] and Chen and Chen [8] both provide two main contributions, one with respect to each of the problems: (1) a technique to efficiently bound the RTEP (DFP, respectively) for a job under analysis considering a specific release pattern, and (2) a specific release pattern based on the critical instant that is claimed to provide the worst-case among all jobs of a specific task under static-priority scheduling (i.e., to calculate the WCRTEP or the WCDFP). Although their concluded critical instant is unsound, the efficient calculations for a specific release pattern remains correct in both cases. Hence, results that directly build on the proposed unsound critical instant are affected, while their contributions to solutions for improving tractability remain unaffected. In the remainder of this section, we discuss the effect (or lack thereof) for relevant publications in the literature.

A. Direct Adoption of the Unsound Critical Instant

While the number of considered release patterns must be reduced to allow efficient computation, the derived analysis might become unsound, since the response time distribution under the unsound critical instant is not necessarily the maximum.

For sensor networks, a series of results by Ren et al. analyzes the response time distribution, and Theorem 3.1 from [31] as well as Theorem 1 from [32] are affected. Their analysis assumes that the unsound critical instant from [25] is the worst-case scenario for bounding the probability of response time. They derived a simulation-based analysis and resolved the intractability issue by abstracting it as an additional probabilistic function. However, the abstraction (e.g., Def 4.3 and 4.4 in [31]) still relies on the assumption of the unsound critical instant to limit the state space.

For mixed criticality systems [4], Maxim et al. [26] adapted the probabilistic response time analysis derived in [25] as a backbone (i.e., Equations (4)-(6) in [25]) to build up several analyses for tasks with different criticality in different types of scheduling schemes. The influence of the unsound critical instant propagates by the direct adaption without further modifications. Therefore, the corresponding probabilistic response time analysis and the schedulability test in [26] are unsound.

The unsound critical instant was adopted by Chen et al. [10] to efficiently calculate the deadline miss rate, assuming that a job is never aborted after any deadline miss. Their approach partitions the schedule into busy intervals and analyzes the probabilities of individual cases. They utilized the unsound critical instant as their analysis backbone, and, therefore, concluded an unsound worst-case miss rate analysis. Our analyses in this paper can unfortunately not be adopted to fix their approach, as we assume that jobs are aborted right after missing their deadlines, whilst *their problem definition disallows such a treatment*.

B. Techniques for Efficient Calculation

When considering a specific job under a given release pattern, a direct computation through naïve convolution is intractable. The reason is that the time and space complexity is exponential in the number of jobs released in the considered interval, which can easily be hundreds or thousands of jobs for practical systems. Hence, several distinct approaches have been proposed to mitigate or even avoid the issues (by trading faster calculation for pessimism), e.g., down-sampling approaches [24], [25], [27], [30], concentration inequalities [8], [9], [35], task-level convolution [35], and the Monte Carlo response time analysis [6].

When calculating the probabilistic response time of a specific job using convolution with down-sampling [25], [27], [30] or the Monte Carlo response time analysis [6] the unsound critical instant is merely adopted as a specific evaluation scenario. Although the considered scenario is not the worst-case scenario, the technical contribution hence remains unaffected.

A similar misconception can be found in Theorem 9 (i.e., Theorem 1 from [8]), where the accumulated workload of higher-priority tasks also overlooks potential carry-in jobs. The analytical upper bounds by Chen and Chen [8], von der Brüggen et al. [35], and Chen et al. [9] as well as the task-

level convolution by von der Brüggen et al. [35] utilize the same concept of accumulated workload.

The efficient calculation techniques discussed in this subsection can still directly be applied to upper-bound the probabilistic response time distribution (for [6], [25], [30]) or the WCRTEP (for [8], [9], [35]) by replacing the adoption of the unsound critical instant theorem in these results with the sound analyses in Theorem 11 or Theorem 14 provided in this work.

VI. Evaluation

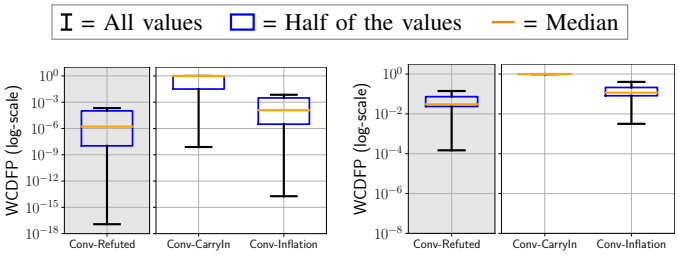
In this section, we show the results for our experimental evaluation of the effectiveness of the two bounds proposed in Section IV. The focus of our evaluation was to compare the derived WCDFP based on the two over-approximations, i.e., Corollary 12 and 15, and to determine the difference to the refuted analysis, considering synthesized task sets.

We generated implicit-deadline task sets, where the utilization values of the individual tasks were determined by using the UUniFast method [5] for a given utilization $U_{sum} = \sum_{\tau_i \in \mathbb{T}} U_i$ and a given number of tasks. We followed the suggestion from Emberson et al. [14] to generate the periods of those tasks according to a log-uniform distribution.

Similar to the evaluation in [35], we considered tasks with two distinct execution times and corresponding probabilities in the evaluation: a normal execution $\{C_i^N, P_i^N\}$ and an abnormal execution $\{C_i^A, P_i^A\}$, where $C_i^N = U_i \cdot T_i$, $C_i^A = 1.83 \cdot C_i^N \ \forall \tau_i \in \mathbb{T}$, $P_i^A = 0.025$, and $P_i^N = 1 - P_i^A$. For each configuration, we evaluated 100 task sets. For the simplicity of presentation, we focus on the resulting WCDFP of the lowest-priority task under the rate-monotonic scheduling policy. The evaluations were deployed on a server equipped with AMD EPYC 7742 running Linux. In the evaluations, we implemented the proposed bounds, i.e., Corollary 12 and 15, in two manners: 1) with direct convolution, i.e., **Conv-CarryIn** and **Conv-Inflation**; 2) with the Chernoff Bound approach [9], i.e., **CB-CarryIn** and **CB-Inflation**. We also implemented both refuted analyses in Theorem 8 and Theorem 9, with direct convolution as **Conv-Refuted** and the Chernoff bound as **CB-Refuted**, respectively.

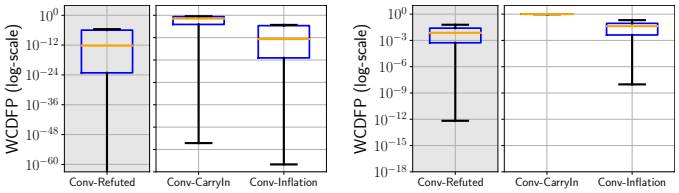
A. Results for Direct Convolution

Figure 6 and Figure 7 show the results of **Conv-Refuted**, **Conv-CarryIn** and **Conv-Inflation** for $U_{sum} = 80\%$ and $U_{sum} = 60\%$. As the derived WCDFP from **Conv-CarryIn** or **Conv-Inflation** is a sound upper bound, a smaller value means a tighter bound, i.e., the lower the better. We evaluated the impact of the task period range, drawing periods from [1,10] (in Figure 6) and [1, 100] (in Figure 7). Due to the high complexity of direct convolution, for each utilization U_{sum}, we considered 5 tasks in $\mathbb{T}$. In general, **Conv-Refuted** derives lower WCDFP than the two sound bounds, and **Conv-Inflation** outperforms **Conv-CarryIn**. When $U_{sum} = 80\%$, **Conv-CarryIn** always returned a WCDFP of 1. Although **Conv-CarryIn** and **Conv-Inflation** do not dominate each other, as discussed in Section IV, we did not observe any case in our evaluation where **Conv-CarryIn** could outperform **Conv-Inflation**.



(a) $U_{sum} = 60\%$ (b) $U_{sum} = 80\%$

Fig. 6: The WCDFP (displayed in log-scale) for 5 tasks with varying total utilization, determined by direct convolution. The task periods were randomly drawn from [1, 10].



(a) $U_{sum} = 60\%$ (b) $U_{sum} = 80\%$

Fig. 7: The WCDFP (displayed in log-scale) for 5 tasks with varying total utilization, determined by direct convolution. The task periods were randomly drawn from [1, 100].

B. Results for the Chernoff Bound Approach

The Chernoff bound approach is the state-of-the-art in terms of approximation quality among the analytical approaches [8], [9], [35]. While it offers no quantitative guarantee for its approximation loss, empirical results in [9] showed that the loss compared to the task-level convolution approach can be reasonably small (under the unsound critical instant theorem).

First, we show the results of **Conv-Refuted**, **Conv-CarryIn**, **Conv-Inflation**, **CB-Refuted**, **CB-CarryIn**, and **CB-Inflation** in Figure 8 for $U_{sum} = 60\%$ and 5 tasks in $\mathbb{T}$. Overall, **CB-CarryIn** performs worse than **Conv-CarryIn**, whereas **CB-Inflation** performs much worse than **Conv-Inflation**. During the evaluation, we observed arbitrary disturbances even if the optimized version of the Chernoff bound approach [9] was applied. In general, the derived WCDFP under the Chernoff bound was much higher than under the direct convolution.

We further show the derived WCDFP when $U_{sum} = 45\%$ and the numbers of tasks in $\mathbb{T}$ is changed. **CB-Refuted** is omitted in the figures, as it is close to 0 most of the time regardless of the number of tasks in this setup. Distinct from the direct convolution, as shown in Figures 9 and 10, **CB-Inflation** usually performs worse than **CB-CarryIn**, regardless of the period range. The same trend can be observed when the number of tasks in $\mathbb{T}$ was increased up to 25, as shown in Figure 11, where the derived WCDFP from **CB-Inflation** is always 1. For 2 tasks in $\mathbb{T}$, in a few cases **CB-CarryIn** derived a higher WCDFP than **CB-Inflation**. The derived WCDFP of

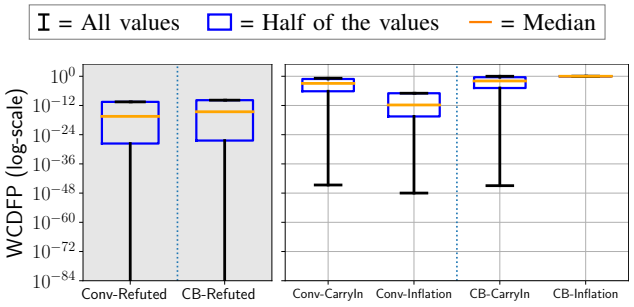
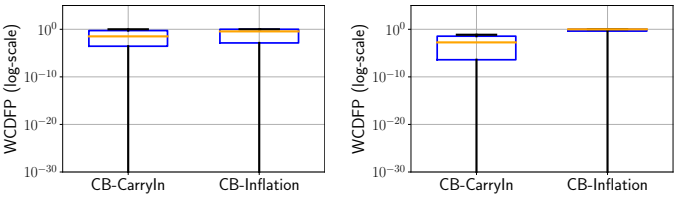


Fig. 8: The WCDFP (displayed in log-scale) for $U_{sum} = 60\%$ and 5 tasks, determined by direct convolution and the Chernoff Bounds. The task periods were randomly drawn from [1, 100].



(a) 2 tasks in $\mathbb{T}$ (b) 5 tasks in $\mathbb{T}$

Fig. 9: The WCDFP (displayed in log-scale) for $U_{sum} = 45\%$ and varying number of tasks, determined by the Chernoff Bounds. The task periods were randomly drawn from [1, 10].

CB-Inflation is still (at least close to) 1 when the number of tasks in $\mathbb{T}$ is more than 2.

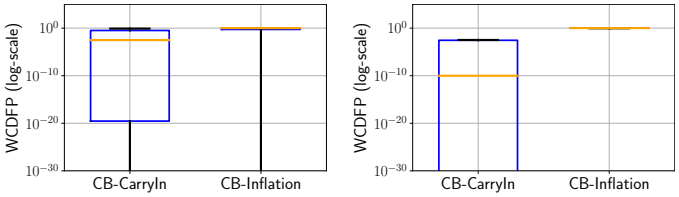
C. Discussion

Overall, the WCDFP derived from both refuted analyses differs largely from the WCDFP derived under the two sound bounds provided in this work. Considering these results, we suggest to first analyze with **CB-CarryIn** and if a sufficiently low deadline miss probability cannot be guaranteed, **Conv-Inflation** should be successively applied. Since both sound bounds do not dominate each other, applying **Conv-CarryIn** might still be able to reach a sound but lower WCDFP.

Since direct convolution suffers from a high runtime complexity, methods with different tradeoffs between runtime and precision, e.g., task-level convolution [35] and the Monte Carlo response time analysis [6], should also be considered if Chernoff Bounds cannot provide a sufficiently low bound.

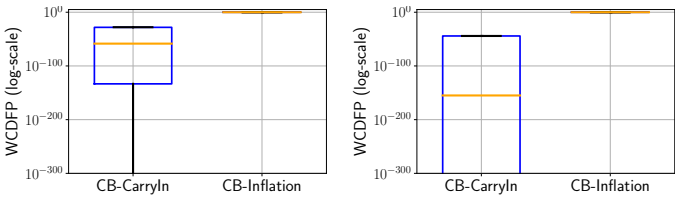
VII. Conclusion

This paper revisits the critical instant for the worst-case response time exceedance probability (WCRTEP) and the worst-case deadline failure probability (WCDFP). A counterexample is provided which shows that the existing critical instant theorem for the WCRTEP [25] and the WCDFP [8] is unsound. Two methods are proposed to safely bound the WCRTEP and WCDFP: one is based on the carry-in approach, and another is based on the sample and inflation approach. It is shown that these two methods do not dominate each other. Evaluation considering the direct convolution approach and the Chernoff bound approach [9] shows that these two approaches



(a) 2 tasks in $\mathbb{T}$ (b) 5 tasks in $\mathbb{T}$

Fig. 10: The WCDFP (displayed in log-scale) for $U_{sum} = 45\%$ and varying number of tasks, determined by the Chernoff Bounds. The task periods were randomly drawn from [1, 100].



(a) 15 tasks in $\mathbb{T}$ (b) 25 tasks in $\mathbb{T}$

Fig. 11: The WCDFP (displayed in log-scale) for $U_{sum} = 45\%$ and varying number of tasks, determined by the Chernoff Bounds. The task periods were randomly drawn from [1, 100].

have different performance in terms of WCDFP, and that the WCDFP from the refuted analyses differs significantly from the results for the two sound methods.

This paper focuses on the scenario that the execution time of a task is a random variable, assuming that the inter-arrival time of a task is not a random variable. The results presented in [25] also consider that both the execution time of a job and the inter-arrival time of two consecutive jobs are random variables. The counterexample in this paper also invalidates the applicability of the unsound critical instant theorem for WCRTEP and WCDFP for this scenario, since periodic releases can be described as a degenerated random variable (i.e., the inter-arrival time is exactly the period with probability 100%). However, if the inter-arrival time of two consecutive jobs is a random variable but the execution time of a job is not a random variable, as studied in [11], our counterexample in this paper does not invalidate the critical instant theorem for such a scenario. Please note that the proposed methods are not proven to handle scenarios where the inter-arrival time of two consecutive jobs is also a random variable. Whether they can be extended or modified to such a scenario should be further explored.

The results presented in this paper are only valid for constrained-deadline task systems based on the assumptions that a) the jobs' execution times are independent and identically distributed (iid) from each other, and b) a job is immediately aborted once it misses its deadline. These are important properties required to achieve sound analyses in Theorems 11 and 14. Deviations from the above conditions are interesting open problems, which may require non-trivial revisions of our analyses or completely new analytical flows.

Acknowledgment

This work has been supported by Deutsche Forschungsgemeinschaft (DFG), as part of Sus-Aware (Project No. 398602212). This result is part of a project (PropRT) that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 865170).

References

[1] B. Akesson, M. Nasri, G. Nelissen, S. Altmeyer, and R. I. Davis. An empirical survey-based study into industry practice in real-time systems. In *IEEE Real-Time Systems Symposium, RTSS*, pages 3–11. IEEE, 2020.

[2] T. Baker. Rate monotone scheduling. http://www.cs.fsu.edu/~baker/realtime/restricted/notes/rmscheduling.html.

[3] T. P. Baker. Multiprocessor EDF and deadline monotonic schedulability analysis. In *IEEE Real-Time Systems Symposium*, pages 120–129, 2003.

[4] S. Baruah, A. Burns, and R. Davis. Response-time analysis for mixed criticality systems. In *2011 IEEE 32nd Real-Time Systems Symposium*, pages 34–43, 2011.

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

[6] S. Bozhko, G. von der Brüggen, and B. B. Brandenburg. Monte carlo response-time analysis. In *2021 IEEE Real-Time Systems Symposium (RTSS)*, pages 342–355, 2021.

[7] J.-J. Chen, G. Nelissen, and W.-H. Huang. A unifying response time analysis framework for dynamic self-suspending tasks. In *Euromicro Conference on Real-Time Systems (ECRTS)*, 2016.

[8] K.-H. Chen and J.-J. Chen. Probabilistic schedulability tests for uniprocessor fixed-priority scheduling under soft errors. In *2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 1–8, 2017.

[9] K.-H. Chen, N. Ueter, G. von der Brüggen, and J.-J. Chen. Efficient computation of deadline-miss probability and potential pitfalls. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 896–901, 2019.

[10] K.-H. Chen, G. von der Brüggen, and J.-J. Chen. Analysis of deadline miss rates for uniprocessor fixed-priority scheduling. In *24th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA*, pages 168–178, 2018.

[11] L. Cucu and E. Tovar. A framework for the response time analysis of fixed-priority tasks with stochastic inter-arrival times. *SIGBED Rev.*, 3(1):7–12, 2006.

[12] R. I. Davis and L. Cucu-Grosjean. A survey of probabilistic schedulability analysis techniques for real-time systems. *Leibniz Trans. Embed. Syst.*, 6(1):04:1–04:53, 2019.

[13] 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.

[14] P. Emberson, R. Stafford, and R. I. Davis. Techniques for the synthesis of multiprocessor tasksets. In *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010)*, pages 6–11, 2010.

[15] M. K. Gardner and J. W.-S. Liu. Analyzing stochastic fixed-priority real-time systems. In *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, TACAS '99, page 44–58, Berlin, Heidelberg, 1999. Springer-Verlag.

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

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

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

[19] K. Lakshmanan and R. Rajkumar. Scheduling self-suspending real-time tasks with rate-monotonic priorities. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 3–12, 2010.

[20] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: exact characterization and average case behavior. In *RTSS*, pages 166–171, 1989.

[21] J. P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *RTSS*, pages 201–209, 1990.

[22] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[23] J. M. López, J. L. Díaz, J. Entrialgo, and D. F. García. Stochastic analysis of real-time systems under preemptive priority-driven scheduling. *Real Time Syst.*, 40(2):180–207, 2008.

[24] F. Marković, A. V. Papadopoulos, and T. Nolte. On the Convolution Efficiency for Probabilistic Analysis of Real-Time Systems. In *33rd Euromicro Conference on Real-Time Systems (ECRTS 2021)*, pages 16:1–16:22, 2021.

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

[26] D. Maxim, R. I. Davis, L. Cucu-Grosjean, and A. Easwaran. Probabilistic analysis for mixed criticality systems using fixed priority preemptive scheduling. In *Proceedings of the 25th International Conference on Real-Time Networks and Systems (RTNS)*, page 237–246, 2017.

[27] D. Maxim, M. Houston, L. Santinelli, G. Bernat, R. I. Davis, and L. Cucu-Grosjean. Re-sampling for statistical timing analysis of real-time systems. In *Proceedings of the 20th International Conference on Real-Time and Network Systems*, page 111–120, New York, NY, USA, 2012. Association for Computing Machinery.

[28] L. Ming. Scheduling of the inter-dependent messages in real-time communication. In *Proc. of the First International Workshop on Real-Time Computing Systems and Applications*, 1994.

[29] A. K. Mok. Fundamental design problems of distributed systems for the hard-real-time environment. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1983.

[30] K. S. Refaat and P.-E. Hladik. Efficient stochastic analysis of real-time systems via random sampling. In *2010 22nd Euromicro Conference on Real-Time Systems*, pages 175–183, 2010.

[31] J. Ren, G. Wu, X. Li, P. Pirozmand, and M. S. Obaidat. Probabilistic response-time analysis for real-time systems in body area sensor networks. *Int. J. Commun. Syst.*, 28(16):2145–2166, nov 2015.

[32] J. Ren, Z. Xu, C. Yu, C. Lin, G. Wu, and G. Tan. Execution allowance based fixed priority scheduling for probabilistic real-time systems. *Journal of Systems and Software*, 152:120–133, 2019.

[33] L. Sha, T. F. Abdelzaher, K. Årzén, A. Cervin, T. P. Baker, A. Burns, G. C. Buttazzo, M. Caccamo, J. P. Lehoczky, and A. K. Mok. Real time scheduling theory: A historical perspective. *Real Time Syst.*, 28(2-3):101–155, 2004.

[34] T.-S. Tia, Z. Deng, M. Shankar, M. F. Storch, J. Sun, L.-C. Wu, and J. W.-S. Liu. Probabilistic performance guarantee for real-time tasks with varying computation times. In *1st IEEE Real-Time Technology and Applications Symposium*, pages 164–173, 1995.

[35] G. von der Brüggen, N. Piatkowski, K.-H. Chen, J.-J. Chen, and K. Morik. Efficiently Approximating the Probability of Deadline Misses in Real-Time Systems. In *30th Euromicro Conference on Real-Time Systems (ECRTS 2018)*, volume 106, pages 6:1–6:22, 2018.

[36] G. von der Brüggen, N. Piatkowski, K.-H. Chen, J.-J. Chen, K. Morik, and B. B. Brandenburg. Efficiently approximating the worst-case deadline failure probability under EDF. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 214–226, 2021.