Scheduling Constrained-Deadline Tasks in Precise Mixed-Criticality Systems on a Varying-Speed Processor

Tianning She
t_s374@txstate.edu
Texas State University
USA

Zhishan Guo
zsguo@ucf.edu
University of Central Florida
USA

Kecheng Yang
yangk@txstate.edu
Texas State University
USA

ABSTRACT

Real-time systems usually require guarantees in all possible scenarios including the worst case. As a result, when each system parameter is specified by a single estimate, significant pessimism is inevitably introduced. To mitigate such pessimism, mixed-criticality (MC) design has been proposed, where a single system parameter is provided multiple estimates. Most existing work on MC scheduling is directed to (fully or partially) sacrifice low-critical workloads in the event of high-critical workloads overrunning their normal-case estimate. Recently, another approach called precise MC scheduling has been investigated, where no low-critical workload is sacrificed in such situation but the speed of the processor is boosted to accommodate the extra execution requirement by high-critical workloads.

Prior work on precise MC scheduling has focused on implicit-deadline tasks only. In this work, we extend the efforts in precise MC scheduling to constrained-deadline tasks by developing demand-based schedulability analysis in place of the utilization-based ones in prior work. This new analysis also enables more flexible virtual-deadline settings. The synthetic experiments have shown that significant schedulability improvements are achieved by this new analysis and by the flexibility in setting virtual deadlines.

CCS CONCEPTS

• **Computer systems organization** → *Real-time systems*; *Embedded systems.*

KEYWORDS

constrained deadlines, mixed-criticality systems, precise scheduling, varying-speed processor, virtual deadlines.

ACM Reference Format:
Tianning She, Zhishan Guo, and Kecheng Yang. 2022. Scheduling Constrained-Deadline Tasks in Precise Mixed-Criticality Systems on a Varying-Speed Processor. In *Proceedings of the 30th International Conference on Real-Time Networks and Systems (RTNS '22), June 7–8, 2022, Paris, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3534879.3534897

RTNS '22, June 7–8, 2022, Paris, France

ACM ISBN 978-1-4503-9650-9/22/06…$15.00
https://doi.org/10.1145/3534879.3534897

1 INTRODUCTION

Safety-critical systems often requires stringent worst-case correctness guarantees for their temporal behaviors. They rely on pessimistic worst-case estimates of system execution behaviors (parameters) and thus lead to huge resource inefficiency in normal (non-worst-case) scenarios. Mixed-criticality (MC) design has been proposed to mitigate such pessimism by adopting multiple estimates to a single system parameter, allowing the system to execute in multiple modes—each of which has its own correctness criteria, or provides service guarantee according to corresponding estimates associated with system modes.

Although MC design and scheduling were originally proposed for different levels of pessimism and confidence for offline (pre-runtime) verification and certification, there also has been attempts and believes that it may also be suitable in handling runtime robustness requirements, which is orthogonal to *a priori* verification. As suggested in [2, 8], in this context, robustness means that system performance will degrade gracefully (if at all) when its behavior does not conform to the models that were assumed during verification. An example of degradation would be to compromise less important aspects of system functionalities. However, in extending the verification-oriented MC models and algorithms to handling runtime robustness requirements, there is still criticism of MC scheduling. The key argument is that low-critical workloads are not non-critical but are of certain criticality level; therefore, their execution guarantees should not be compromised even in the event of more important tasks demanding more resources.

In addressing this issue, the precise-MC model has been proposed [11]. Under the precise-MC model, upon a mode switch, no (low- or high-critical) task will be sacrificed either fully or partially. Instead, the additional execution requirements demanded by high-critical tasks are supposed to be compensated by boosting the platform capability, such as increasing the processor speed. Since it was introduced, existing work on the precise-MC model has been focused on *implicit-deadline* tasks only and therefore often yields *utilization-based* schedulability tests. Although system execution modes are triggered by overrun of hi-critical tasks under both Vestal-MC and precise-MC, the non-sacrificing setting under precise-MC makes the problem completely different. We do not know the computation complexity of the schedulability problem for precise-MC yet. Also, even if we could adapt Vestal-MC [33] schedulers such as the virtual-deadline based ones to precise-MC, the original schedulability analysis does not apply to precise-MC setting. New schedulability tests and analysis techniques must be developed from first principles, as demonstrated in [11].

In this paper, focusing on *constrained-deadline* tasks, we develop new *demand-based* analysis that significantly improves the schedulability, compared to directly adapting the existing methods for implicit deadlines to constrained deadline by treating density as utilization.

Contributions. To the best of our knowledge, this is the first work to address constrained-deadline tasks for the precise-MC model. In this work, we

- formalize the schedulability problem for precise-MC tasks with constrained deadlines on a varying-speed processor;
- present a virtual-deadline based algorithm for this problem, called EDF-VD-FLX, which extends EDF-VD in [11] by relaxing the "common factor" restriction in virtual deadlines;
- derive a sufficient schedulability test for EDF-VD-FLX by demand-based analysis, where new techniques are developed and presented to overcome new challenges in the context of the precise-MC model; and
- conduct schedulability experiments by synthetic workloads to evaluate the improvements made by the new analysis only and by the new analysis together with the flexibility in setting virtual deadlines, respectively.

Organization. The rest of the paper is organized as follows. Section 2 specifies the system model and assumptions to respect in this paper. Section 3 presents algorithm EDF-VD-FLX and its associated schedulability test. Section 4 proves the schedulability test by showing the demand-based analysis. Section 5 reports our schedulability results in synthetic experiments. Section 6 reviews related work, and Section 7 concludes the paper.

2 SYSTEM MODEL

We denote a set of n sporadic MC tasks by $\mathcal{T} = \{\tau_1, \tau_2, \cdots, \tau_n\}$ and denote a task by a 4-tuple as $\tau_i = (T_i, C_i^L, C_i^H, D_i)$. Each task τ_i releases a (potentially infinite) sequence of *jobs* with a minimum release separation of T_i time units, and every job has an absolute deadline D_i time units after its release. In this paper, we only consider constrained-deadline tasks, *i.e.*, $\forall i, D_i \leq T_i$. The worst-case execution requirement of task τ_i, defined by the worst-case execution time on a unit-speed processor, is estimated a two criticality levels: a low-criticality estimate C_i^L and a high-criticality estimate C_i^H, where it is assumed that $C_i^L \leq C_i^H, \forall i$. Besides, C_i^L (respectively, C_i^H) is also the execution requirement budgets of task τ_i in the L (respectively, H)-mode, to be described later. In particular, $C_i^L < C_i^H$ indicates that τ_i is a high-criticality task that may trigger a system mode switch, whereas $C_i^L = C_i^H$ indicates that τ_i is a low-criticality task that cannot trigger any system mode switch. In this paper, we say "HI-task" for "high-criticality task" and "LO-task" for "low-criticality task" for short. Similarly, we also use "HI-job(s)" to mean "job(s) of HI-task(s)" and use "LO-job(s)" to mean "job(s) of LO-task(s)." Let the j^{th} job of task τ_i be denoted as $J_{i,j}$.

In this paper, we assume that all job release times, all absolute actual deadlines, and all absolute virtual deadlines (to be introduced in the next section) must be at integer time instants, but we allow the amount of work and the execution of jobs to be continuous. In other words, $\forall i$, T_i, D_i, and D_i' (to be introduced in the next section) must be integers, but (L- or H-) C_i and C_i/ρ (where ρ is the degraded speed to be introduced in the next paragraph) are not necessarily to be integers. Also, the mode switch might happen at a non-integer time instant. Moreover, in this paper, we assume that the preemption and migration overheads *e.g.*, due to memory interference are negligible. (Equivalently, we assume such overheads are pessimistically taken into account in the execution requirements estimates.)

Varying-speed processor and mode switch. We consider the problem of scheduling the set of tasks τ on a single energy-conserving processor that can operate at a *degraded* or *full* speed. The processor begins with a degraded speed $\rho < 1.0$, which indicates that any workload being executed under this speed for t time units is equivalent to that under a unit-speed processor for $\rho \times t$ time units[1]. During runtime, the amount of workload completed for each job is being monitored. If any job $J_{i,j}$ has cumulatively executed for a workload of C_i^L under degraded processing speed ρ (*i.e.*, has received a cumulative actual execution *time* of C_i^L/ρ units) but still requires further execution, the system is immediately notified, and the processor starts to perform its full speed 1.0 from that instance. We call this moment as the time instant of *mode switch*, from the L-mode (where the processor speed is ρ) to the H-mode (where the processor speed becomes 1.0). In the H-mode, once the processor becomes idle, the system immediately switches back to the L-mode.

Note that, in contrast to the majority of existing works on MC scheduling, *no* task is *entirely or partially dropped* upon a mode switch, and every job meets its absolute deadline in any system mode. The difference between the two execution requirement budgets upon mode switch, *i.e.*, $C_i^H - C_i^L$, is supposed to be compensated by the processor recovering to its full speed. Furthermore, any job $J_{i,j}$ that has cumulatively executed for a workload of C_i^H yet still not completed, is considered as *erroneous* and would be terminated. That is, only HI-tasks, for which $C_i^L < C_i^H$, could trigger a mode switch.

We denote the utilization of a task τ_i in L- and H-modes, respectively, by

$$u_i^L = \frac{C_i^L}{T_i} \quad \text{and} \quad u_i^H = \frac{C_i^H}{T_i}.$$

We further denote the total of all tasks in L- and H-modes, respectively, by

$$U^L = \sum_{\tau_i \in \mathcal{T}} u_i^L \quad \text{and} \quad U^H = \sum_{\tau_i \in \mathcal{T}} u_i^H.$$

We also denote the set of all LO-tasks by $\mathcal{T}_{\text{LO}}$ and denote the set of all HI-tasks by $\mathcal{T}_{\text{HI}}$. Then, we also have following per-task-set-per-mode total utilizations:

$$U_{\text{LO}}^L = \sum_{\tau_i \in \mathcal{T}_{\text{LO}}} u_i^L, \quad U_{\text{HI}}^L = \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} u_i^L,$$
$$U_{\text{LO}}^H = \sum_{\tau_i \in \mathcal{T}_{\text{LO}}} u_i^H, \quad \text{and} \quad U_{\text{HI}}^H = \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} u_i^H.$$

It is also clear that $U^L = U_{\text{LO}}^L + U_{\text{HI}}^L$ and $U^H = U_{\text{LO}}^H + U_{\text{HI}}^H$. Since $C_i^L = C_i^H$ holds for every LO-task, it also holds $u_i^L = u_i^H$ for such task. Therefore, it is also clear that $U_{\text{LO}}^L = U_{\text{LO}}^H$.

[1] Such linear relationship is *safe*, as a lowered processor frequency may not cause the execution time to grow much.

Schedulability in this paper. We address the problem of scheduling the MC tasks on a varying-speed uniprocessor. A system is said *schedulable* under a given algorithm if and only if it is guaranteed that all deadlines are met in *all* scenarios with the following *additional requirements* on processor execution speed:

- the processor must only operate at its energy-conserving speed ρ if *all* jobs finish within their C_i^L budget;
- the processor may operate at full speed 1.0 if *any* HI-job executes beyond its C_i^L budget (yet finishes within its C_i^H budget).

Also, a system is said *feasible* if and only if there exists an algorithm under which this system is schedulable.

3 ALGORITHM EDF-VD-FLX

In this section, we introduce the scheduling algorithm, EDF-VD-FLX, on which we will be focused in this paper. EDF-VD-FLX indeed follows the conventional way of assigning virtual deadlines to address MC problems. The suffix "FLX" (which stands for "flexible") is to distinguish with the EDF-VD in [11], where the ratio between relative virtual deadline and relative actual deadline must be the same for all HI-tasks. EDF-VD-FLX relaxes this restriction on virtual-deadline settings by allowing each HI-task to set its own relative virtual deadline independently. As shown in Section 4, the schedulability test to be presented in this paper can indeed take this flexibility into account. Furthermore, the experiments in Section 5 shows that even following the same virtual-deadline setting as that in [11] (*i.e.*, same virtual-to-actual ratio for every HI-task), the advances in the proposed schedulability analysis also gain a significant schedulability margin compared to adopting the existing analysis in [11].

Virtual deadlines. Under EDF-VD-FLX, each (HI- or LO-) task τ_i is assigned one more parameter D_i', which is its *relative virtual deadline*; and each (HI- or LO-) job $J_{i,j}$ is assigned one more parameter $d_{i,j}'$, which is its *absolute virtual deadline* and is set at $d_{i,j}' = a_{i,j} + D_i'$, where $a_{i,j}$ is the arrival (release) time of job $J_{i,j}$. Furthermore, for every LO-task τ_ℓ, we set $D_\ell' = D_\ell$; for every HI-task τ_h, the setting of D_h' can be arbitrary as long as $0 \leq D_h' \leq D_h$. As we will see later, the concrete virtual-deadline setting for HI-tasks is taken as an input to the schedulability test, and in Section 5, we will present and evaluate certain specific such settings.

Runtime scheduling policies and mode switches. In the runtime, EDF-VD-FLX has two *modes* to select ready jobs to execute on the processor. In the L-mode, the ready (HI- or LO-) job with the earliest absolute *virtual* deadline is always selected to execute; by contrast, in the H-mode, the ready (HI- or LO-) job with the earliest absolute *actual* deadline is always selected to execute. The system (and EDF-VD-FLX) always starts with the L-mode, where the processor also operates at its energy-saving speed ρ. Upon a time instant where some HI-job $J_{i,j}$ has cumulatively executed a workload of C_i^L but has not finished yet, the mode is immediately switched to the H-mode and the processor thereafter operates at its full speed 1.0 as well. On the other hand, upon a time instant where the processor becomes idle, the mode is immediately switched back to the L-mode and the processor reduces its speed to ρ.

Schedulability test. For any given virtual deadline setting under EDF-VD-FLX, we present the following schedulability test, for which the detailed analysis and proofs will be presented next in Section 4. Also, we will discuss and evaluate a few methods and heuristics to set the virtual deadlines in Section 5.

Please note that this schedulability test requires that $U^L < \rho$ and $U^H < 1$. Since it is trivial that $U^L \leq \rho$ and $U^H \leq 1$ are necessary for the feasibility. This requirement barely eliminates the cases where $U^L = \rho$ and $U^H = 1$ and similar requirements often exist in demand-based schedulability analysis, such as the classic one from [9].

Given that $U^L < \rho$ and $U^H < 1$, we claim that a system follows the models in Section 2 is schedulable under EDF-VD-FLX, if both of the following **(A)** and **(B)** are true.

(A) $\forall \ell \in \mathbb{Z}^+$ such that $\ell < K$,

$$\sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i'}{T_i} \right\rfloor + 1 \right) C_i^L \leq \rho \ell,$$

where

$$K = \frac{U^L}{\rho - U^L} \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i'\};$$

(B) $\forall \ell, \ell' \in \mathbb{Z}^+$ such that $\ell' \leq \ell < K'$

$$\sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i}{T_i} \right\rfloor + 1 \right) C_i^L + \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} \left(\left\lfloor \frac{\ell' + D_i' - D_i}{T_i} \right\rfloor + 1 \right) (C_i^H - C_i^L) \leq (\ell - \ell')\rho + \ell'$$

where

$$K' = \frac{U^L}{\min\{\rho - U^L, 1 - U^H\}} \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i\} + \frac{U^H - U^L}{\min\{\rho - U^L, 1 - U^H\}} \cdot \max_{\tau_i \in \mathcal{T}_{\text{HI}}} \{T_i + D_i' - D_i\}.$$

Note that this schedulability test runs in pseudo-polynomial time for any task system where the L-mode total utilization U^L is upper bounded by some constant $c_1 < \rho$ and the H-mode total utilization U^H is upper bounded by some constant $c_2 < 1$.

4 SCHEDULABILITY ANALYSIS

In this section, we present our schedulability analysis under EDF-VD-FLX and prove a sufficient schedulability test, which is the one as claimed in the prior section.

In particular, the two parts of the schedulability test, **(A)** and **(B)**, are to be proved in the following Sections 4.1 and 4.2, respectively. The analysis and proofs in Section 4.1 are somewhat straight application and adaption of the classic demand based schedulability analysis from [9]. Readers who are familiar with this kind of analysis might be intuitively convinced by looking at **(A)** itself and skip the proof; nonetheless, we provide the full proof here in Section 4.1 for completeness and showing all details and modifications. On the other hand, the analysis and proofs in Section 4.2 do require new techniques to achieve a meaningful result as **(B)**.

4.1 Schedulability in the L-mode

The following lemma proves a sufficient condition under which it is guaranteed that all virtual deadlines in the L-mode must be met, although it potentially requires to verify the inequality for an infinite number of values of ℓ.

Lemma 4.1. *All virtual deadlines in the L-mode must be met, if*

$$\forall \ell \in \mathbb{Z}^+, \sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i'}{T_i} \right\rfloor + 1 \right) C_i^L \leq \rho \ell$$

Proof. We prove the contrapositive, *i.e.*, if there is a missed virtual deadline in the L-mode, then

$$\exists \ell \in \mathbb{Z}^+, \sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i'}{T_i} \right\rfloor + 1 \right) C_i^L > \rho \ell.$$

We let t_d denote the first missed virtual deadline in the L-mode and let t_0 denote the latest idle[2] time instant at or before t_d. Then, the processor is busy during time interval $(t_0, t_d]$ and provides a supply of $\rho(t_d - t_0)$ for executing jobs with virtual deadlines at or before t_d. On the other hand, each (LO- or HI-) task τ_i can only have at most $\left(\left\lfloor \frac{t_d - t_0 - D_i'}{T_i} \right\rfloor + 1 \right)$ jobs with virtual deadlines at or before t_d that contribute to the demand in time interval $(t_0, t_d]$. This yields a total demand of

$$\sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{t_d - t_0 - D_i'}{T_i} \right\rfloor + 1 \right) \cdot C_i^L.$$

Then, missing the virtual deadline at t_d implies that

$$\sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{t_d - t_0 - D_i'}{T_i} \right\rfloor + 1 \right) \cdot C_i^L > \rho(t_d - t_0).$$

Letting $\ell = t_d - t_0$, it becomes

$$\sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i'}{T_i} \right\rfloor + 1 \right) \cdot C_i^L > \rho \ell.$$

By their definition, it is clear that $t_0 < t_d$. Also, because t_0 and t_d are a release time and an absolute virtual deadline, both of which must be integers by our system model assumptions, ℓ must be an integer as well. Therefore, $\ell \in \mathbb{Z}^+$, and the lemma follows. ∎

Then, the following lemma shows that, in fact, only a finite set of values of the ℓ in Lemma 4.1 is needed to verify.

Lemma 4.2. *If*

$$\forall \ell \in \mathbb{Z}^+ \text{ such that } \ell < \frac{U^L}{\rho - U^L} \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i'\}, \sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i'}{T_i} \right\rfloor + 1 \right) C_i^L \leq \rho \ell,$$

then

$$\forall \ell \in \mathbb{Z}^+, \sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i'}{T_i} \right\rfloor + 1 \right) C_i^L \leq \rho \ell$$

Proof. We prove the contrapositive, *i.e.*, if

$$\exists \ell \in \mathbb{Z}^+, \sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i'}{T_i} \right\rfloor + 1 \right) C_i^L > \rho \ell, \tag{1}$$

then

$$\exists \ell \in \mathbb{Z}^+ \text{ such that } \ell < \frac{U^L}{\rho - U^L} \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i'\}, \sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i'}{T_i} \right\rfloor + 1 \right) C_i^L > \rho \ell. \tag{2}$$

We let ℓ^* denotes such an ℓ that satisfies (1), *i.e.*, $\ell^* \in \mathbb{Z}^+$ and

$$\sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell^* - D_i'}{T_i} \right\rfloor + 1 \right) C_i^L > \rho \ell^*.$$

So, it follows that

$$\sum_{\tau_i \in \mathcal{T}} \left(\left(\frac{\ell^* - D_i'}{T_i} \right) + 1 \right) C_i^L > \rho \ell^*,$$

which is

$$\sum_{\tau_i \in \mathcal{T}} \left(\ell^* - D_i' + T_i \right) \frac{C_i^L}{T_i} > \rho \ell^*.$$

Therefore,

$$\left(\ell^* + \max_{\tau_i \in \mathcal{T}} \{T_i - D_i'\} \right) \cdot \sum_{\tau_i \in \mathcal{T}} \frac{C_i^L}{T_i} > \rho \ell^*,$$

which is

$$\left(\ell^* + \max_{\tau_i \in \mathcal{T}} \{T_i - D_i'\} \right) U^L > \rho \ell^*.$$

By rearranging, it yields

$$\ell^* < \frac{U^L}{\rho - U^L} \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i'\}.$$

That is, ℓ^* is such an integer that satisfies (2). In other words, (2) is true and the lemma follows. ∎

Combining Lemmas 4.1 and 4.2, we can conclude the following theorem, which is part **(A)** of our schedulability test.

Theorem 4.3. *All virtual deadlines in the L-mode must be met, if*

$$\forall \ell \in \mathbb{Z}^+ \text{ such that } \ell < K, \quad \sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i'}{T_i} \right\rfloor + 1 \right) C_i^L \leq \rho \ell, \tag{3}$$

$$\text{where } K = \frac{U^L}{\rho - U^L} \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i'\}.$$

4.2 Schedulability in the H-mode

Theorem 4.3 above has shown that (3), which is part **(A)** of our schedulability test, is sufficient to ensure all virtual deadlines to be met in the L-mode. To determine the schedulability for a given system, we will always examine **(A)** for the schedulability in the L-mode; and only if it is true, we will then continue to examine **(B)** for the schedulability in the H-mode. Therefore, in the following lemmas and theorem that establish part **(B)** of our schedulability test, it is assumed that **(A)** is true, *i.e.*, all virtual deadlines in the L-mode are already guaranteed to be met.

Based on this assumption, the following lemma proves a sufficient condition for ensuring all actual deadlines in the H-mode to be met, although it potentially requires to verify the inequality for an infinite number of values of ℓ and ℓ'.

[2]Executing a job with a virtual deadline after t_d is considered as idle.

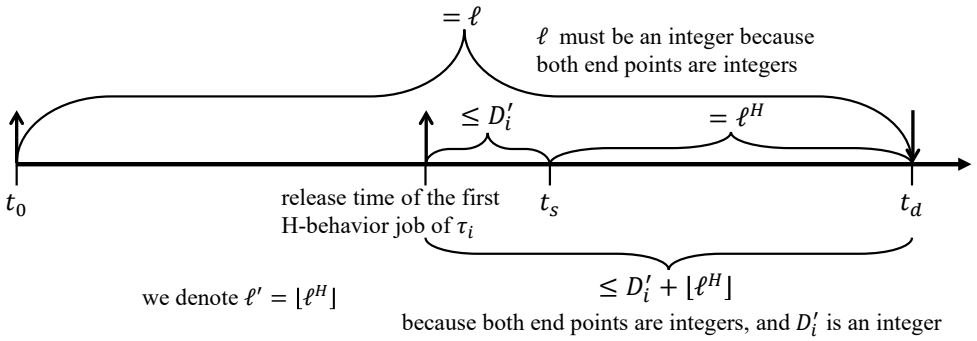


Figure 1: An illustration of the time instants and intervals in the proof of Lemma 4.4.

Lemma 4.4. *All actual deadlines in the H-mode must be met if* $\forall \ell, \ell' \in \mathbb{Z}^+$ *such that* $\ell' \leq \ell$, *the following inequality* (4) *holds.*

$$\sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i}{T_i} \right\rfloor + 1 \right) C_i^L + \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} \left(\left\lfloor \frac{\ell' + D_i' - D_i}{T_i} \right\rfloor + 1 \right) (C_i^H - C_i^L) \leq (\ell - \ell')\rho + \ell' \quad (4)$$

Proof. We prove the contrapositive, *i.e.*, if there is a missed actual deadline in the H-mode, then

$$\exists \ell, \ell' \in \mathbb{Z}^+ \text{ such that } \ell' \leq \ell, \text{ and}$$
$$\sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i}{T_i} \right\rfloor + 1 \right) C_i^L + \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} \left(\left\lfloor \frac{\ell' + D_i' - D_i}{T_i} \right\rfloor + 1 \right) (C_i^H - C_i^L) > (\ell - \ell')\rho + \ell' \quad (5)$$

We let t_d denote the first missed actual deadline in the H-mode and let t_0 denote the latest idle[3] time instant at or before t_d. Then, at time t_0, the system must be in the L-mode, because under EDF-VD-FLX, the system would switch back to the L-mode at an idle time instant even if it was in the H-mode before. Also, it is clear that some job is released at time t_0 and the entire time interval $(t_0, t_d]$ must be busy; otherwise, t_0 would be a later time instant. Thus, within time interval $(t_0, t_d]$, there must exist and only exist one mode switch from the L-mode to the H-mode, and we denote the mode-switch time instant by t_s. Figure 1 depicts an illustration of these time instants. We let ℓ denote the length of this busy time interval of interest, *i.e.*, $\ell = t_d - t_0$, and let ℓ^H denote the length of the H-mode sub-interval, *i.e.*, $\ell^H = t_d - t_s$.

We then examine a necessary condition for the deadline at t_d to be missed. Namely, the *demand* must exceed the *supply* for the time interval of interest, *i.e.*, $[t_0, t_d]$, where the demand is defined as the total work from all jobs that both are released and have an *actual* deadline within this time interval and supply is defined as the total amount of work the processor can complete within this busy time interval.

For each (LO- or HI-) task τ_i, it can release at most $\left(\left\lfloor \frac{\ell - D_i}{T_i} \right\rfloor + 1 \right)$ jobs that both are released and have an actual deadline within time interval $[t_0, t_d]$. We count a C_i^L towards the demand due to each of such job. Then, it upper bounds the work from LO-jobs and L-behaviour HI-jobs (that do not overrun their C_i^L), and partially covers the work from H-behaviour HI-jobs (that do overrun their C_i^L). This part of work can be summarized as

$$W_1 = \sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i}{T_i} \right\rfloor + 1 \right) C_i^L.$$

Note that, t_0 must be the released time of some job and t_d must be an actual absolute deadline of some job, so both t_0 and t_d must be integer time instants by our system model assumptions. Thus, $\ell = t_d - t_0$ must be an integer.

Then, rest of the work that contribute to the demand are all from the work beyond C_i^L of H-behaviour HI-jobs. For each HI-task τ_i, its first H-behaviour job cannot be released earlier than $t_s - D_i'$. Otherwise, the mode switch must have already been triggered before t_s. This is because, due to the assumption of meeting all virtual deadlines in the L-mode, any job must have completed C_i^L amount of work by its virtual deadline, which is before t_s for a job released before $t_s - D_i'$; and a H-behaviour HI-job that completes C_i^L amount of work but has not finished yet would trigger the mode switch immediately. In other words, the length of the H-behaviour *window* for each HI-task τ_i, *i.e.*, the time interval from the release of its first H-behaviour job to t_d, is at most $t_d - (t_s - D_i') = \ell^H + D_i'$. Note that, this time window begins from some job's release (which must be an integer time instant) and ends at some job's absolute deadline (which also must be an integer time instant), so the length of this time window must be an integer as well. Therefore, the length of the H-behaviour window for each HI-task τ_i is at most $\lfloor \ell^H \rfloor + D_i'$, given that the relative deadline D_i' must be an integer. (ℓ^H might not be an integer because the mode-switch instant t_s might not be an integer time instant as we have the processor speed varying and assume the execution is continuous.)

Therefore, letting $\ell' = \lfloor \ell^H \rfloor$, each HI-task τ_i can release at most $\left(\left\lfloor \frac{\ell' + D_i' - D_i}{T_i} \right\rfloor + 1 \right)$ H-behaviour jobs that have an actual deadline at or before t_d. Note that a certain part of the work from these H-behaviour jobs—an amount of C_i^L for each—has already been counted in W_1. Therefore, the additional work towards the demand due to these H-behaviour jobs is upper bounded by

$$W_2 = \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} \left(\left\lfloor \frac{\ell' + D_i' - D_i}{T_i} \right\rfloor + 1 \right) (C_i^H - C_i^L).$$

[3]Executing a job with an actual deadline after t_d is considered as idle.

Thus, the total demand is at most $W_1 + W_2$. On the other hand, the supply within $[t_0, t_d]$ is $(\ell - \ell^H)\rho + \ell^H \geq (\ell - \ell')\rho + \ell'$, where the inequality is because $\ell' = \lfloor \ell^H \rfloor \leq \ell^H$ by its definition and $\rho < 1$ as the degraded speed. Therefore,

$$W_1 + W_2 > (\ell - \ell')\rho + \ell'$$

is necessary for the demand exceeding the supply for time interval $[t_0, t_d]$, which is necessary for missing an actual deadline at t_d. That is,

$$\sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i}{T_i} \right\rfloor + 1 \right) C_i^L + \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} \left(\left\lfloor \frac{\ell' + D_i' - D_i}{T_i} \right\rfloor + 1 \right) (C_i^H - C_i^L) > (\ell - \ell')\rho + \ell'$$

is necessary for missing the deadline at t_d.

Earlier in this proof, we have already discussed and shown that ℓ must be an integer, and ℓ' also must be an integer by its definition that $\ell' = \lfloor \ell^H \rfloor$. Thus, (5) is a necessary condition for missing actual deadline in the H-mode, and the lemma follows. ∎

Next, the following lemma shows that, to apply Lemma 4.4, it is sufficient to just examine the possible values of ℓ and ℓ' uppper-bounded by K'.

Lemma 4.5. *If $\forall \ell, \ell' \in \mathbb{Z}^+$ such that $\ell' \leq \ell < K'$, (4) is true, then $\forall \ell, \ell' \in \mathbb{Z}^+$ such that $\ell' \leq \ell$, (4) is true, where K' is defined as follows.*

$$K' = \frac{U^L}{\min\{\rho - U^L, 1 - U^H\}} \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i\} + \frac{U^H - U^L}{\min\{\rho - U^L, 1 - U^H\}} \cdot \max_{\tau_i \in \mathcal{T}_{\text{HI}}} \{T_i + D_i' - D_i\}$$

Proof. We prove the contrapositive, *i.e.*, if $\exists \ell, \ell' \in \mathbb{Z}^+$ such that $\ell' \leq \ell$, (4) is false, then $\exists \ell, \ell' \in \mathbb{Z}^+$ such that $\ell' \leq \ell < K'$, (4) is false. We let ℓ^* and ℓ'^* denote an instance of such ℓ and ℓ' that make (4) be false, *i.e.*, $\ell^*, \ell'^* \in \mathbb{Z}^+$ such that $\ell'^* \leq \ell^*$ and

$$\sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell^* - D_i}{T_i} \right\rfloor + 1 \right) C_i^L + \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} \left(\left\lfloor \frac{\ell'^* + D_i' - D_i}{T_i} \right\rfloor + 1 \right) (C_i^H - C_i^L) > (\ell^* - \ell'^*)\rho + \ell'^*. \quad (6)$$

Then, our goal is to show that $\ell^* < K'$.

By (6), it follows that

$$\sum_{\tau_i \in \mathcal{T}} \left(\left(\frac{\ell^* - D_i}{T_i} \right) + 1 \right) C_i^L + \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} \left(\left(\frac{\ell'^* + D_i' - D_i}{T_i} \right) + 1 \right) (C_i^H - C_i^L) > (\ell^* - \ell'^*)\rho + \ell'^*,$$

which is

$$\sum_{\tau_i \in \mathcal{T}} \left(\ell^* + T_i - D_i \right) \frac{C_i^L}{T_i} + \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} \left(\ell'^* + T_i + D_i' - D_i \right) \frac{C_i^H - C_i^L}{T_i} > (\ell^* - \ell'^*)\rho + \ell'^*.$$

Therefore,

$$\left(\ell^* + \max_{\tau_i \in \mathcal{T}} \{T_i - D_i\} \right) \sum_{\tau_i \in \mathcal{T}} \frac{C_i^L}{T_i} + \left(\ell'^* + \max_{\tau_i \in \mathcal{T}_{\text{HI}}} \{T_i + D_i' - D_i\} \right) \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} \frac{C_i^H - C_i^L}{T_i} > (\ell^* - \ell'^*)\rho + \ell'^*,$$

which is

$$\left(\ell^* + \max_{\tau_i \in \mathcal{T}} \{T_i - D_i\} \right) U^L + \left(\ell'^* + \max_{\tau_i \in \mathcal{T}_{\text{HI}}} \{T_i + D_i' - D_i\} \right) (U_{\text{HI}}^H - U_{\text{HI}}^L) > (\ell^* - \ell'^*)\rho + \ell'^*.$$

Because LO-tasks have the same utilization in both L- and H-modes, $U_{\text{HI}}^H - U_{\text{HI}}^L = (U_{\text{HI}}^H + U_{\text{LO}}^H) - (U_{\text{HI}}^L + U_{\text{LO}}^L) = U^H - U^L$, by which we have

$$\left(\ell^* + \max_{\tau_i \in \mathcal{T}} \{T_i - D_i\} \right) U^L + \left(\ell'^* + \max_{\tau_i \in \mathcal{T}_{\text{HI}}} \{T_i + D_i' - D_i\} \right) (U^H - U^L) > (\ell^* - \ell'^*)\rho + \ell'^*.$$

By rearranging,

$$(\rho - U^L)\ell^* + (1 - \rho - U^H + U^L)\ell'^* < U^L \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i\} + (U^H - U^L) \cdot \max_{\tau_i \in \mathcal{T}_{\text{HI}}} \{T_i + D_i' - D_i\} \quad (7)$$

By the supposition that $\ell^*, \ell'^* \in \mathbb{Z}^+$ such that $\ell' \leq \ell$, we can discuss the two exhaustive cases of $(1 - \rho - U^H + U^L)$ as follows:

If $1 - \rho - U^H + U^L \geq 0$, (7) implies

$$(\rho - U^L)\ell^* < U^L \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i\} + (U^H - U^L) \cdot \max_{\tau_i \in \mathcal{T}_{\text{HI}}} \{T_i + D_i' - D_i\}.$$

By rearranging, in this case,

$$\ell^* < \frac{U^L \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i\} + (U^H - U^L) \cdot \max_{\tau_i \in \mathcal{T}_{\text{HI}}} \{T_i + D_i' - D_i\}}{\rho - U^L}.$$

If $1 - \rho - U^H + U^L < 0$, since $\ell'^* < \ell^*$, (7) implies

$$(\rho - U^L)\ell^* + (1 - \rho - U^H + U^L)\ell^* < U^L \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i\} + (U^H - U^L) \cdot \max_{\tau_i \in \mathcal{T}_{\text{HI}}} \{T_i + D_i' - D_i\},$$

which is

$$(1 - U^H)\ell^* < U^L \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i\} + (U^H - U^L) \cdot \max_{\tau_i \in \mathcal{T}_{\text{HI}}} \{T_i + D_i' - D_i\}.$$

By rearranging, in this case,

$$\ell^* < \frac{U^L \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i\} + (U^H - U^L) \cdot \max_{\tau_i \in \mathcal{T}_{\text{HI}}} \{T_i + D_i' - D_i\}}{1 - U^H}.$$

Thus, we can conclude that, in either case of $(1 - \rho - U^H + U^L)$, it holds that

$$\ell^* < \frac{U^L \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i\} + (U^H - U^L) \cdot \max_{\tau_i \in \mathcal{T}_{\text{HI}}} \{T_i + D_i' - D_i\}}{\min\{\rho - U^L, 1 - U^H\}},$$

where the right-hand side is indeed K'. So, the lemma follows. ∎

Combining Lemmas 4.4 and 4.5 and recalling that the H-mode analysis is based on the assumption that the L-mode schedulability has been established first, the following theorem directly follows.

Theorem 4.6. *Given that all virtual deadlines in the L-mode are met, all actual deadlines in the H-mode must be met, if* $\forall \ell, \ell' \in \mathbb{Z}^+$ *such that* $\ell' \leq \ell < K'$,

$$\sum_{\tau_i \in \mathcal{T}} \left(\left\lfloor \frac{\ell - D_i}{T_i} \right\rfloor + 1 \right) C_i^L + \sum_{\tau_i \in \mathcal{T}_{\text{HI}}} \left(\left\lfloor \frac{\ell' + D_i' - D_i}{T_i} \right\rfloor + 1 \right) (C_i^H - C_i^L) \leq (\ell - \ell')\rho + \ell'$$

where

$$K' = \frac{U^L}{\min\{\rho - U^L, 1 - U^H\}} \cdot \max_{\tau_i \in \mathcal{T}} \{T_i - D_i\} + \frac{U^H - U^L}{\min\{\rho - U^L, 1 - U^H\}} \cdot \max_{\tau_i \in \mathcal{T}_{\text{HI}}} \{T_i + D_i' - D_i\}.$$

Theorems 4.3 and 4.6 together have established that **(A)** and **(B)** are a sufficient schedulability test, as claimed in Section 3.

5 EXPERIMENTAL EVALUATION

In this section, we conduct schedulability experiments by synthetic workloads to evaluate the effectiveness of our proposed new schedulability analysis as well as the flexible virtual-deadline settings it enables.

Schemes to compare. Given the focus on the precise scheduling of MC tasks with constrained deadlines, we specifically compare the following three schemes for schedulability analysis and setting virtual deadlines.

- **(S1)** Directly adopt EDF-VD and its schedulability analysis in [11] with minimum modifications. Since implicit-deadline tasks are assumed in [11] where utilization-based test was derived, we just need to use the density (*i.e.*, C_i/D_i) in place of every corresponding utilization. Then, the results in [11] can be applied. This simple adoption is the base line to compare against in our experiments.
- **(S2)** While applying the new schedulability analysis in this paper, we keep the setting of virtual deadlines (almost) the same as **(S1)** (*i.e.*, the same as [11]). In other words, we also calculate a common factor x for shrinking virtual deadlines by
$$x = \frac{\sum_{\tau_i \in \mathcal{T}_{\text{HI}}} \frac{C_i^L}{D_i}}{\rho - \sum_{\tau_i \in \mathcal{T}_{\text{LO}}} \frac{C_i^L}{D_i}},$$
and then assign the relative virtual deadline for each HI-task τ_i by $D_i' = \lceil x \cdot D_i \rceil$. Note that, we would require the ceilings here for deriving D_i' because the schedulability analysis presented in this paper require all virtual deadlines to be integers.
- **(S3)** While applying the new schedulability analysis in this paper, we also use a more "personalized" way to set virtual deadlines for each individual HI-task. Specifically, for each HI-task τ_i, we calculate a separate factor x_i by $x_i = C_i^L/C_i^H$ and then derive its relative virtual deadline by $D_i' = \lceil x_i \cdot D_i \rceil$.

To compare the above **(S1)**, **(S2)**, and **(S3)**, we conduct the schedulability experiments on randomly generated MC tasks with constrained deadlines.

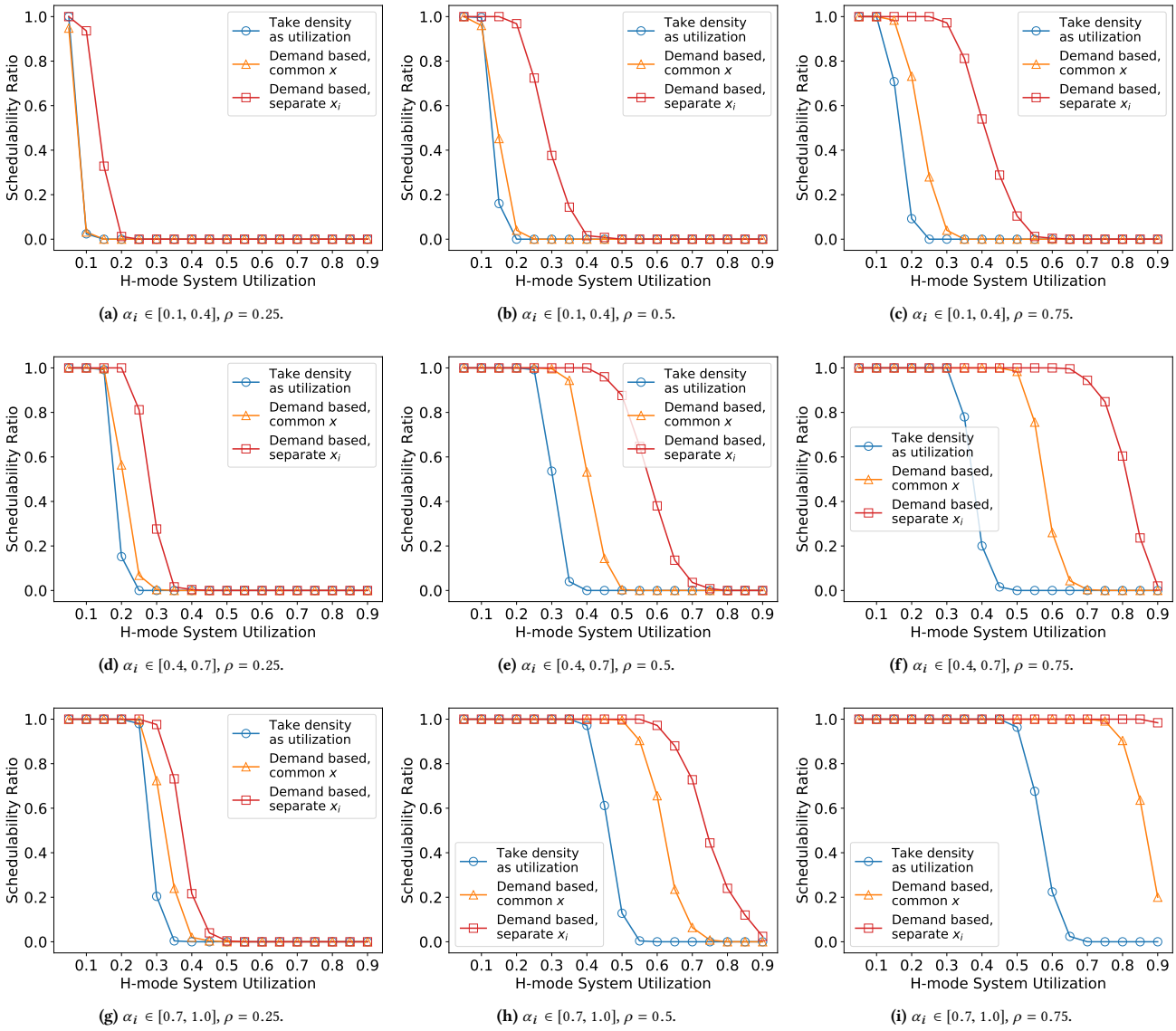
Task generation. For each given H-mode total utilization, we generated 500 task sets and we report the percentage of these task sets that are deemed schedulable under **(S1)**, **(S2)**, and **(S3)**, respectively. For each task set, we generate 20 MC tasks as follows. Given H-mode total utilization, we first generate the H-mode utilization for each task by UUnifastDiscard [16]. Then, each task is selected to be a HI-task with probability P (*i.e.*, to be a LO-task with probability $1 - P$). In our experiments reported in this paper, we have set $P = 0.75$. If a task τ_i has been selected to be a LO-task, then its L-mode utilization u_i^L must equal to its H-mode utilization u_i^H. By contrast, u_i^L is randomly and uniformly chosen from $[0.2 \times u_i^H, 0.8 \times u_i^H]$ for each HI-task. Moreover, the period T_i for each (LO- or HI-) task τ_i is randomly generated from a log-uniform distribution with the range $[10, 100]$. We also set a factor α_i such that $0 \leq \alpha_i \leq 1$ to represent how constrained the relative deadlines are expected to be for each task τ_i. Specifically, the relative deadline D_i is set by $D_i = \lceil C_i^H + (T_i - C_i^H) \times \alpha_i \rceil$. Note that the ceilings are used to ensure that the relative deadlines are integers, which is required by the proposed schedulability analysis in this paper, and that $D_i \geq C_i^H$ is necessarily required for feasibility.

Results. In our experiments, we consider three different ranges, $[0.1, 0.4]$, $[0.4, 0.7]$, $[0.7, 1.0]$ for uniformly selecting α_i, and we also consider three different settings, 0.25, 0.50, and 0.75, for the degraded processor speed ρ. Therefore, we have $3 \times 3 = 9$ combinations of the setting for α_i and ρ to conduct the schedulability experiments. The results have been summarized and plotted in the nine sub-figures in Figure 2. In the legends, "Take density as utilization" stands for results under scheme **(S1)**, "Demand based, common x" stands for results under scheme **(S2)**, and "Demand based, separate x_i" stands for results under scheme **(S3)**.

Observations. Comparing each pair of the blue, circle plot and the orange, triangle plot in all sub-figures in Figure 2, we can see that the proposed demand-based schedulability analysis significantly out performs the utilization-based on from prior work [11], even if the virtual-deadline setting remains the same. Furthermore, comparing each pair of the orange, triangle plot and the red, square plot in all sub-figures in Figure 2, we can see that the individual, personalized virtual-deadline setting is able to further improve the schedulability also significantly. Note that this more flexible setting of virtual deadlines is enabled by the demand-based analysis proposed in this paper and was not supported by the analysis in [11]. We have also summarized the total number of schedulable task sets (graphically, the "area" between the plot and the x-axis) for the three plots that represent **(S1)**, **(S2)**, and **(S3)**, respectively. In this metrics, **(S2)** is 1.38 times of **(S1)**, and **(S3)** is 1.86 times of **(S1)**.

6 RELATED WORK

Since MC model is originally introduced by Vestal [33], several versions of MC model has been proposed in the real-time systems research community. A comprehensive review of updated models and results is presented in [13]. Traditionally, LO-tasks were fully dropped for sufficient budget to HI-tasks. [1, 6, 14, 15] However,



(a) $\alpha_i \in [0.1, 0.4]$, $\rho = 0.25$. **(b)** $\alpha_i \in [0.1, 0.4]$, $\rho = 0.5$. **(c)** $\alpha_i \in [0.1, 0.4]$, $\rho = 0.75$.

(d) $\alpha_i \in [0.4, 0.7]$, $\rho = 0.25$. **(e)** $\alpha_i \in [0.4, 0.7]$, $\rho = 0.5$. **(f)** $\alpha_i \in [0.4, 0.7]$, $\rho = 0.75$.

(g) $\alpha_i \in [0.7, 1.0]$, $\rho = 0.25$. **(h)** $\alpha_i \in [0.7, 1.0]$, $\rho = 0.5$. **(i)** $\alpha_i \in [0.7, 1.0]$, $\rho = 0.75$.

Figure 2: Schedulability experiment results. Schedulability ratio is defined as the ratio of the number of schedulable task sets and the number of randomly generated task sets for each given H-mode system utilization.

Such sacrifice are criticized by Ernst et al. [17]. Dropping all LO-tasks could not be practical. Recently, more practical model was presented by [12], known as imprecise MC model(MC), providing degraded service for LO-tasks, where the execution time of LO-tasks is reduced in the event of a mode-switch rather than dropped. Several subsequent works of imprecise scheduling providing degrade service [3, 12, 22, 24, 25, 27] either in the form of reduced execution window, increased period, or dropping some jobs.

The schedulability analysis of the IMC model has been investigated for fixed-priority scheduling and EDF-VD in [12] and [28], respectively. A generalization of the Vestal model is considered in [3], where the less critical functionalities are not fully dropped even in H-mode. A fluid model-based scheduling algorithm, called MC-Fluid, is presented in [26] for MC tasks on multiprocessors. In the MC-Fluid scheduling, each task may receive a fraction of a processor and have a constant execution rate. All tasks may progress at specific rates simultaneously. A simplified variant of MC-Fluid, MCF, is proposed by [5]. It has a speedup bound no worse than 1.33, improved from 1.618 for a dual-criticality system. For the adaptive MC- Weakly Hard model, a response time-based schedulability analysis was proposed in [18] that guarantees a minimum service for LO-tasks in the event of a mode switch.

Although such degraded service is better than no service for LO-tasks, it is not acceptable for certain applications as pointed out in [17]. To address this shortcoming, the precise MC scheduling [11] was proposed which provide full service for low-critical tasks even

at the mode-switch. The problem of precise MC scheduling was investigated on varying-speed uniprocessor [11, 34] and multiprocessors [31]. Reserving processors by precise MC scheduling on multiprocessors is proposed in [30], where a part of processors are reserved in L-mode for extra workloads in H-mode but the speed of processors is constant in both modes.

Non-functional requirements such as energy consumption and its relationship to the operating frequency of the processors were considered in non-mixed-crticality systems [10, 20, 21, 32]. and mix-criticality systems [7, 19, 23]. [23] proposed the energy minimization by reducing operating frequency using the DVFS technique. The frequency of the processor can be later changed to higher by the DVFS technique when needed, such as the mode switch. The advantage of reducing the energy consumption of the system by throttling speed of processors during run-time was also concluded in [23]. A natural extension to multiprocessors was presented in [4, 29].

7 CONCLUSION

In this work, we addressed the precise scheduling of MC tasks with constrained deadlines on a varying-speed processor. We presented a virtual-deadline based algorithm for this problem, called EDF-VD-FLX, which extends EDF-VD in prior work [11] by relaxing the "common factor" restriction in setting virtual deadlines. By analyzing the demand carefully, we derived a sufficient schedulability test for EDF-VD-FLX. As demonstrated by synthetic experiments, both the new schedulability analysis and the flexibility in setting virtual deadlines are able to significantly improve the schedulability.

ACKNOWLEDGMENTS

This work was supported in part by NSF grants CNS-1850851, CNS-2104181, a start-up grant from University of Central Florida, and start-up and REP grants from Texas State University.

REFERENCES

[1] Sanjoy Baruah, Vincenzo Bonifaci, Gianlorenzo D'angelo, Haohan Li, Alberto Marchetti-Spaccamela, Suzanne Van Der Ster, and Leen Stougie. 2015. Preemptive uniprocessor scheduling of mixed-criticality sporadic task systems. *Journal of the ACM (JACM)* 62, 2 (2015), 14.

[2] Sanjoy Baruah and Alan Burns. 2019. Incorporating robustness and resilience into mixed-criticality scheduling theory. In *2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*. IEEE, 155–162.

[3] Sanjoy Baruah, Alan Burns, and Zhishan Guo. 2016. Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors. In *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE, 131–138.

[4] Sanjoy Baruah, Bipasa Chattopadhyay, Haohan Li, and Insik Shin. 2014. Mixed-criticality scheduling on multiprocessors. *Real-Time Systems* 50, 1 (2014), 142–177.

[5] Sanjoy Baruah, Arvind Easwaran, and Zhishan Guo. 2015. MC-Fluid: simplified and optimally quantified. In *2015 IEEE Real-Time Systems Symposium*. 327–337.

[6] Sanjoy Baruah and Zhishan Guo. 2014. Scheduling mixed-criticality implicit-deadline sporadic task systems upon a varying-speed processor. In *Proceedings of the 35th Real-Time Systems Symposium (RTSS), IEEE*. IEEE, 31–40.

[7] Sanjoy Baruah and Zhishan Guo. 2014. Scheduling Mixed-Criticality Implicit-Deadline Sporadic Task Systems upon a Varying-Speed Processor. In *2014 IEEE Real-Time Systems Symposium*. 31–40.

[8] Sanjoy K Baruah. 2018. Mixed-Criticality Scheduling Theory: Scope, Promise, and Limitations. *IEEE Des. Test* 35, 2 (2018), 31–37.

[9] Sanjoy K Baruah, Aloysius K Mok, and Louis E Rosier. 1990. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings 11th Real-Time Systems Symposium*. IEEE, 182–190.

[10] Ashikahmed Bhuiyan, Zhishan Guo, Abusayeed Saifullah, Nan Guan, and Haoyi Xiong. 2018. Energy-efficient real-time scheduling of DAG tasks. *ACM Transactions on Embedded Computing Systems (TECS)* 17, 5 (2018), 84.

[11] Ashikahmed Bhuiyan, Sai Sruti, Zhishan Guo, and Kecheng Yang. 2019. Precise scheduling of mixed-criticality tasks by varying processor speed. In *Proceedings of the 27th International Conference on Real-Time Networks and Systems*. 123–132.

[12] Alan Burns and Sanjoy Baruah. 2013. Towards a more practical model for mixed criticality systems. In *Workshop on Mixed-Criticality Systems*.

[13] Alan Burns and Robert Ian Davis. 2022. Mixed Criticality Systems-A Review:(February 2022). (2022).

[14] Arvind Easwaran. 2013. Demand-based scheduling of mixed-criticality sporadic tasks on one processor. In *Proceedings of the 34th Real-Time Systems Symposium (RTSS), IEEE*. IEEE, 78–87.

[15] Pontus Ekberg and Wang Yi. 2014. Bounding and shaping the demand of generalized mixed-criticality sporadic task systems. *Real-time systems* 50, 1 (2014), 48–86.

[16] Paul Emberson, Roger Stafford, and Robert I Davis. 2010. Techniques for the synthesis of multiprocessor tasksets. In *proceedings 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*. 6–11.

[17] Rolf Ernst and Marco Di Natale. 2016. Mixed Criticality Systems - A History of Misconceptions? *IEEE Design & Test* 33, 5 (2016), 65–74.

[18] Oliver Gettings, Sophie Quinton, and Robert I Davis. 2015. Mixed criticality systems with weakly-hard constraints. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*. ACM, 237–246.

[19] Zhishan Guo and Sanjoy Baruah. 2015. The concurrent consideration of uncertainty in WCETs and processor speeds in mixed-criticality systems. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*. 247–256.

[20] Zhishan Guo, Ashikahmed Bhuiyan, Di Liu, Aamir Khan, Abusayeed Saifullah, and Nan Guan. 2019. Energy-Efficient Real-Time Scheduling of DAGs on Clustered Multi-Core Platforms. In *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 156–168.

[21] Zhishan Guo, Ashikahmed Bhuiyan, Abusayeed Saifullah, Nan Guan, and Haoyi Xiong. 2017. Energy-efficient multi-core scheduling for real-time DAG tasks. In *29th Euromicro conference on real-time systems (ECRTS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[22] Zhishan Guo, Kecheng Yang, Sudharsan Vaidhun, Samsil Arefin, Sajal K Das, and Haoyi Xiong. 2018. Uniprocessor Mixed-Criticality Scheduling with Graceful Degradation by Completion Rate. In *2018 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 373–383.

[23] Pengcheng Huang, Pratyush Kumar, Georgia Giannopoulou, and Lothar Thiele. 2014. Energy efficient dvfs scheduling for mixed-criticality systems. In *Proceedings of the 14th International Conference on Embedded Software, ACM*. ACM, 11.

[24] Pengcheng Huang, Pratyush Kumar, Georgia Giannopoulou, and Lothar Thiele. 2015. Run and be safe: Mixed-criticality scheduling with temporary processor speedup. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*. IEEE, 1329–1334.

[25] Mathieu Jan, Lilia Zaourar, and Maurice Pitel. 2013. Maximizing the execution rate of low criticality tasks in mixed criticality system. *Proc. WMC, RTSS* (2013), 43–48.

[26] Jaewoo Lee, Kieu-My Phan, Xiaozhe Gu, Jiyeon Lee, Arvind Easwaran, Insik Shin, and Insup Lee. 2014. Mc-fluid: Fluid model-based mixed-criticality scheduling on multiprocessors. In *2014 IEEE Real-Time Systems Symposium*. 41–52.

[27] Di Liu, Jelena Spasic, Nan Guan, Gang Chen, Songran Liu, Todor Stefanov, and Wang Yi. 2016. EDF-VD scheduling of mixed-criticality systems with degraded quality guarantees. In *2016 IEEE Real-Time Systems Symposium (RTSS)*. 35–46.

[28] Di Liu, Jelena Spasic, Nan Guan, Gang Chen, Songran Liu, Todor Stefanov, and Wang Yi. 2016. EDF-VD scheduling of mixed-criticality systems with degraded quality guarantees. In *Proceedings of the 37th Real-Time Systems Symposium (RTSS), 2016 IEEE*. 35–46.

[29] Sujay Narayana, Pengcheng Huang, Georgia Giannopoulou, Lothar Thiele, and R Venkatesha Prasad. 2016. Exploring energy saving for mixed-criticality systems on multi-cores. In *Proceedings of the 22nd Real-Time and Embedded Technology and Applications Symposium (RTAS), IEEE*. IEEE, 1–12.

[30] Tianning She, Zhishan Guo, Qijun Gu, and Kecheng Yang. 2021. Reserving Processors by Precise Scheduling of Mixed-Criticality Tasks. In *2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 103–108.

[31] Tianning She, Sudharsan Vaidhun, Qijun Gu, Sajal Das, Zhishan Guo, and Kecheng Yang. 2021. Precise scheduling of mixed-criticality tasks on varying-speed multiprocessors. In *29th International Conference on Real-Time Networks and Systems*. 134–143.

[32] Saad Zia Sheikh and Muhammad Adeel Pasha. 2018. Energy-Efficient Multicore Scheduling for Hard Real-Time Systems: A Survey. *ACM Transactions on Embedded Computing Systems (TECS)* 17, 6 (2018), 94.

[33] S. Vestal. 2007. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proceedings of the 28th IEEE Real-Time Systems Symposium (RTSS)*.

[34] Kecheng Yang, Ashikahmed Bhuiyan, and Zhishan Guo. 2020. F2VD: Fluid Rates to Virtual Deadlines for Precise Mixed-Criticality Scheduling on a Varying-Speed Processor. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–9.