Mixed-Criticality Scheduling with Limited HI-Criticality Behaviors

Zhishan Guo, Luca Santinelli, Kecheng Yang

Mixed-Criticality Scheduling with Limited HI-Criticality Behaviors*

Zhishan Guo[1,2], Luca Santinelli[3], and Kecheng Yang[4,5]

[1] University of Central Florida, USA
[2] Missouri University of Science and Technology, USA
[3] ONERA - Toulouse, France
[4] Texas State University, USA
[5] University of North Carolina at Chapel Hill, USA
guozh@mst.edu luca.santinelli@onera.fr yangk@cs.unc.edu

Abstract. Due to size, weight, and power considerations, there is an emerging trend in real-time embedded systems design towards implementing functionalities of different levels of importance upon a shared platform, or implementing Mixed-Criticality (MC) systems. Much existing work on MC scheduling focuses on the classic Vestal model, where upon a mode switch, it is pessimistically assumed that all tasks may simultaneously exceed their less pessimistic execution time estimations, or LO-WCETs. In this paper, a less pessimistic MC model is proposed for system designers to specify the maximum number of tasks that may simultaneously exceed their LO-WCETs. The applicability and schedulability of the classic EDF-VD scheduler under this newly proposed model are studied, and a new schedulability test is presented. Experiments demonstrate that, by applying the proposed model and new schedulability test, significantly better schedulability can be achieved.

1 Introduction and Motivation

The Worst-Case Execution Time (WCET) abstraction models the execution behavior of real-time tasks. Given a piece of code to execute upon a specified platform, the WCET is an upper bound to the time duration needed to finish the execution of a single invocation of that piece of code. Unfortunately, even when severe restrictions are placed upon the structure of the code e.g., known loop bounds, it is still difficult to determine the exact WCET. Furthermore, the occurrence of the WCET is usually extremely unlikely, unless under highly pathological circumstances such as faults.

In order to utilize the significant gap between the actual running time and the WCET, it has been proposed to implement functionalities of different degrees of importance (or criticalities) upon a shared platform. Under such design, for each of the more important tasks, a less pessimistic execution time estimation is

* Supported by NSF grants CNS 1755965, CNS 1563845, and CNS 1717589, ARO grant W911NF-17-1-0294, and fundings from General Motors and Center for Advancing Faculty Excellence of the University of Missouri System.

also provisioned in addition to the most pessimistic WCET. When the more important tasks actually complete by these less pessimistic estimations, less important tasks are allowed to execute as well, so that processor capacities are not wasted. In contrast, in occasional situations where the more important tasks execute beyond their less pessimistic estimations, the less important tasks may be dropped. In order to validate systems under this design approach, Mixed-Criticality (MC) scheduling techniques are needed.

Prior research on MC scheduling (see [7] for an up-to-date review) focused on the Vestal model [15], which assigns multiple WCET estimations for each individual task. Typically, in the two-criticality-level case, each task is designated as being of either higher (HI) or lower (LO) criticality. Two WCETs are specified for each HI-criticality task: a LO-WCET and a larger HI-WCET which could be larger than the LO-WCET by several orders of magnitude. One WCET is specified for each LO-criticality task: the LO-WCET.

The Vestal model defines two system *modes*, each associated with different guarantees. In the normal mode, every HI-criticality task completes its execution by its LO-WCET, and each LO-criticality task should be guaranteed to execute up to its LO-WCET as well. On the other hand, whenever any HI-criticality task does not signal its completion after exhausting its LO-WCET, a system mode switch will be triggered; in the new mode, all of the LO-criticality tasks are dropped in order to guarantee every HI-criticality task to execute up to its HI-WCET. In this traditional MC Scheduling model, *all* HI-criticality tasks may *simultaneously* exceed their LO-WCETs, requiring executions up to their HI-WCETs in the new mode.

Motivation. However, in some cases, this assumption about the execution of HI-criticality tasks in the classic Vestal model could be too pessimistic. Indeed, having all HI-criticality tasks simultaneously exceeding their LO-WCETs could be non-representative of many real-world real-time embedded systems.

The following two pieces of codes shown in Figure 1 is a toy example to illustrate this motivation in more details.

```
task Anti_Frozen {
  f1();
  t = temperature();
  if (t < 0) {
    f2();
  }
  if (t < -20) {
    f3();
  }
}
```

```
task Over_Heat {
  g1();
  t = temperature();
  if (t > 50) {
    g2();
  }
  if (t > 80) {
    g3();
  }
}
```

Fig. 1. Two example tasks in a safety critical system.

Let us assume both tasks are HI-criticality tasks as they perform some important safety features of the system, dealing with either frozen (task `Anti_Frozen`) or over-heating (task `Over_Heat`) situation. Under normal circumstances, actions in `f2()` and `g2()` are more than enough to bring the ambient temperature (around the platform) back to normal range (0 to 50), such that `f3()` and `g3()` will not need to be executed. As a result, we may assign LO-WCET of task `Anti_Frozen` as the maximum time to execute `f1()` and `f2()`; HI-WCET of task `Anti_Frozen` as the maximum time to execute `f1()`, `f2()`, and `f3()`; LO-WCET of task `Over_Heat` as the maximum time to execute `g1()` and `g2()`; and HI-WCET of task `Over_Heat` as the maximum time to execute `g1()`, `g2()`, and `g3()`.

A straightforward observation is that, even under extreme situations, only *one* of the above two tasks will need to execute their final *if* branches; i.e., there will not be any time instant that both tasks require executions of their HI-WCETs simultaneously. As a result, any analysis following the Vestal model is over pessimistic in this example, as it will need to take the impossible case into consideration, where both tasks exceed their LO-WCETs at the same time.

Note that, when all HI-criticality tasks simultaneously exceed their LO-WCET due to certain system degradation or failure, it is computationally more efficient to characterize such behaviors with the MC varying speed model [5, 6, 9–11], which better represents the uncertainties arising from the executing speed of the platform, rather than with Vestal model by using multiple estimations of WCETs (due to the NP-hardness[1]).

Contribution. In this work, we propose a new MC system model to cope with more realistic assumptions for real-time embedded systems. The proposed model is more general than the existing well-studied Vestal model in the sense that it allows a system designer to specify the number of HI-criticality tasks that can exceed their LO-WCET simultaneously. We then analyze how this additional specification could impact the schedulability and develop an MC scheduler for this new model. We finally conduct schedulability experiments and compare the results from our scheduler and a classic MC scheduler, namely EDF-VD. The advantages from having only subsets of HI-criticality tasks exceeding their LO-WCET thresholds simultaneously are validated by these experimental results.

Organization. Section 2 describes the proposed MC system model. Section 3 adapts an existing scheduler for the problem, and proves its correctness. Section 4 evaluates the performance of the proposed scheduler under various parameter settings, and compares it with an existing MC task scheduler. Section 5 concludes the work and points out some future directions.

2 Model and Definitions

Mixed-Criticality Tasks. A MC periodic task set τ is specified as a finite collection of MC periodic tasks, each of which generates an unbounded number of MC jobs. Each task τ_i has a period, T_i, modeling the time separation between

two consecutive jobs of this task, and each job of τ_i has to complete its execution by D_i time units. In this paper, the tasks are assumed to have implicit deadlines, i.e., $D_i = T_i$. The integer time model is also assumed—all task periods are non-negative integers and all job arrivals occur at integer time instants.

We consider a uniprocessor system where all tasks execute on and share the single processor, while the scheduler determines how it is shared.

A task exhibits LO-criticality behavior if *all* of its jobs complete execution by its LO-WCET. In contrast, a task is in HI-criticality behavior if *any* of its jobs requires an execution longer than its LO-WCET, but no more than its HI-WCET. Any other behavior is considered *erroneous*.

A HI-criticality task τ_i can be specified by $\tau_i = ([c_i(\text{LO}), c_i(\text{HI})], T_i)$. T_i is the period and the relative deadline of task τ_i; $[c_i(\text{LO}), c_i(\text{HI})]$ is the tuple of WCET estimations, $c_i(\text{LO})$ for the LO-WCET and $c_i(\text{HI})$ for the HI-WCET, where $c_i(\text{LO}) \leq c_i(\text{HI})$.

A LO-criticality task τ_j is represented with two parameters $\tau_j = (c_j(\text{LO}), T_j)$. T_j is the period and the deadline of the task and $c_j(\text{LO})$ characterizes the LO-criticality mode worst-case execution time. For LO-criticality tasks only the LO-criticality behavior is possible.

The two WCETs specified for each HI-criticality task τ_i may come from timing analysis tools with different levels of pessimism:

- $c_i(\text{LO})$, which is determined by a less pessimistic timing analysis tool (or with less guarantees of being the worst-case for any possible execution condition) — a HI-criticality task may require an execution length of more than $c(\text{LO})$; and
- $c_i(\text{HI})$, which is sometimes larger than the LO-WCET by several orders of magnitude — it may be determined by a more conservative timing analysis, and it presents the worst-case execution time for any possible execution condition the task may experience.

The *utilizations* of tasks are defined for HI- and LO-criticality tasks respectively. Each HI-criticality task has two associated utilizations—one in each mode, whereas each LO-criticality task has only one associated utilization as follows:

- $U_{\text{HI}}^{\text{HI}}(\tau_i) = c_i(\text{HI})/T_i$ - HI-criticality task utilization in HI-criticality mode;
- $U_{\text{HI}}^{\text{LO}}(\tau_i) = c_i(\text{LO})/T_i$ - HI-criticality task utilization in LO-criticality mode;
- $U_{\text{LO}}^{\text{LO}}(\tau_i) = c_i(\text{LO})/T_i$ - LO-criticality utilization.

Mixed-Criticality Systems. An MC system is defined to run under two possible *modes*: a normal mode (LO-criticality mode) where every job completes upon executing for no more than its LO-WCET and a HI-criticality mode where some HI-criticality job executes for more than its LO-WCET but imperatively completes upon execution for no more than its HI-WCET.

The system mode will be switched from LO-criticality mode to HI-criticality mode if *any* HI-criticality task has exhausted its LO-WCET but has not completed. Only HI-criticality tasks are guaranteed to be met their deadlines under HI-criticality mode.

Definition 1 (MC Task Instance) *A MC task instance I is composed of an MC task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$, where both* HI*-criticality tasks and* LO*-criticality tasks may be in τ. n_{HI} denotes the number of* HI*-criticality tasks in τ, and $n_{\text{HI}} \leq n$. Each* HI*-criticality task τ_i is represented as $\tau_i = ([c_i(\text{LO}), c_i(\text{HI})], T_i)$, while each* LO*-criticality tasks τ_j is represented as $\tau_j = (c_j(\text{LO}), T_j)$.*

The notion of *utilization difference* for HI-criticality tasks is defined as follows.

Definition 2 (Utilization Difference) *The utilization difference of a* HI*-criticality task τ_i is defined by*

$$\delta_i = \frac{c_i(\text{HI}) - c_i(\text{LO})}{T_i}. \tag{1}$$

We assume that the tasks are indexed by criticality—from HI-criticality ones to LO-criticality ones; and HI-criticality tasks are indexed by utilization difference—the larger the utilization difference the lower the index, and utilization difference ties are broken arbitrarily. That is, the HI-criticality tasks are indexed $1, 2, \ldots, n_{\text{HI}}$, and $\delta_i \geq \delta_j$ for any $1 \leq i \leq j \leq n_{\text{HI}}$.

Then, the per mode utilizations of either criticality task set are defined:

$$U_{\text{HI}}^{\text{HI}} = \sum_{i=1}^{n_{\text{HI}}} c_i(\text{HI})/T_i; \tag{2}$$

$$U_{\text{HI}}^{\text{LO}} = \sum_{i=1}^{n_{\text{HI}}} c_i(\text{LO})/T_i; \tag{3}$$

$$U_{\text{LO}}^{\text{LO}} = \sum_{i=n_{\text{HI}}+1}^{n} c_i(\text{LO})/T_i. \tag{4}$$

Mixed-Critical Scheduling. The MC scheduling objective is to determine a run-time scheduling strategy which ensures that: i) *all* jobs of *all* tasks complete by their deadlines if no job exceeds its LO-WCET; ii) *all* jobs of tasks designated as being of HI-criticality continue to complete by their deadlines (although the LO-criticality jobs may not) if *any* HI-criticality job requires execution for more than its LO-WCET (but no larger than its HI-WCET) to complete.

Limited HI-Criticality Behaviors. As motivated in Sec. 1, in some systems, it could be reasonable to assume that only a limited number N of HI-criticality tasks that may exceed their LO-WCET and reach their HI-WCET simultaneously, where $N \leq n_{\text{HI}}$. In contrast, existing MC analysis usually makes the most pessimistic assumption that all of the n_{HI} HI-criticality tasks may execute beyond their LO-WCET and reach its HI-WCET simultaneously. Even if this could actually happen, it can also be viewed as a special case ($N = n_{\text{HI}}$) under the new MC model we propose in this paper By saying *simultaneously* (or "at the same time"), we mean within any time window of length $\overline{T} = \max_i\{T_i\}$ [6]. That

[6] When considering a certain time window of length $\overline{T}$, any task τ_i with a partially overlapping scheduling window that experience HI-criticality behavior counts (al-

is, *at most* N HI-*criticality tasks can require an execution time larger than their* $c_i(\text{LO})$ *within any time window of length* $\overline{T}$. Again, please note that the Vestal model is a special case of our model, by assigning $N = n_{\text{HI}}$.

Determine N**.** In this paper, we generally assume that the parameter N is a parameter given offline, instead of to be determined online by the scheduler. That is, how to determine N is not the focus of this paper, and we mainly focus on the problem of how to schedule the tasks with a valid schedulability test when N is given as an input parameter. Nonetheless, for the sake of inspiring future work, we also briefly discuss a couple of potential sources for where the N parameter could come from.

First, it could come from physical constraints in the systems. Different set of HI-criticality tasks may be triggered to perform their HI-criticality behaviors by different physical measurements. Such difference may be significant enough so that they cannot have simultaneous impacts on the system.

Second, it could come from contradicting logic control flows in the code. When the code of tasks has branches, which branch is chosen to execute may depend on some global variables. Different task might have the same global variables in their code, and the same global variables control the branch choices in multiple tasks. As a result, it could be logically impossible for some HI-criticality tasks to take their worst branch choices simultaneously. That is, they cannot have their HI-criticality behaviors to have simultaneous impacts on the system.

Third, it could also come from probabilistic analysis if the WCETs of HI-criticality tasks are *independent* [8]. In this approach, the probability of multiple HI-criticality tasks performing HI-criticality behaviors could be calculated as a product of multiple (hopefully small) probabilities for each individual task to perform its HI-criticality behavior. When this product is sufficiently small, the simultaneous HI-criticality behaviors of these tasks could be probabilistically deemed *impossible*.[7] This setting was also considered in [12, 14], which more focuses on the various detailed combinations of tasks that may not perform their HI-criticality behaviors. Therefore, a somewhat complicated scheduling approach was studied there. In this paper, we mainly focused on the maximum *number* of such tasks only, and therefore enable the applicability of the relatively simple scheduler, EDF-VD.

3 EDF-VD Schedulability Analysis

In this section, we review a commonly used and adapted MC scheduler, namely EDF-VD [2], which was proposed for the classic Vestal model. We will refine the original analysis of EDF-VD to cope with our less pessimistic assumptions, and derive a new schedulability test for EDF-VD under the new model proposed in this paper.

---

though it may be already finished by the beginning of the period of interest, or it did not start executing by the end of the period of interest).

[7] Or equivalently, even if it does happen, it is viewed as erroneous, and the system design does not take care of it.

EDF-VD. Similar to the classic EDF scheduler, EDF-VD is a deadline-based, dynamic-priority scheduler. In contrast to EDF, EDF-VD assigns virtual deadlines, which are earlier than the actual deadlines, to HI-criticality jobs. In the runtime, their priorities are determined by their virtual deadlines in the LO-criticality mode; upon a mode switch, their priorities are changed back to their actual deadlines in the HI-criticality mode. Intuitively, the virtual deadlines in the LO-criticality mode provide the room for the HI-criticality tasks to still meet their actual deadlines in the HI-criticality mode, when they occasionally overrun their LO-WCETs.

Let τ denote the MC implicit-deadline sporadic task system that is to be scheduled on a preemptive uniprocessor. Prior to run-time, EDF-VD performs a schedulability test to determine whether τ can be correctly scheduled by it or not. If τ is deemed schedulable, then an additional parameter x is computed for setting virtual deadlines to HI-criticality tasks. Each virtual relative deadline T_i' can be calculated by "shrinking" the actual relative deadline T_i by the scaling factor x.

Next, we describe a schedulability test for EDF-VD under the proposed new model and prove its correctness. Note that, when $N = n_{\text{HI}}$, this schedulability test reduces to the one for the classic Vestal model in [2].

Schedulability test. First, given an MC task instance, the parameter x is calculated as follows:

$$x \leftarrow \frac{U_{\text{HI}}^{\text{LO}}}{1 - U_{\text{LO}}^{\text{LO}}}. \tag{5}$$

By Theorem 1 (to be presented later), this assignment of x will be able to guarantee the schedulability under LO-criticality mode.

Then, the schedulability under HI-criticality mode can also be guaranteed if the following inequality holds:

$$xU_{\text{LO}}^{\text{LO}} + U_{\text{HI}}^{\text{LO}} + \sum_{i=1}^{N} \delta_i \leq 1. \tag{6}$$

That is, given an MC task instance, the schedulability test needs to check whether Inequality (6) is satisfied.

The schedulability test returns success if Inequality (6) is satisfied, and failure otherwise.

Upon success, EDF-VD assigns virtual deadline parameters for all HI-criticality tasks as follows:

$$T_i' \leftarrow x \cdot T_i. \tag{7}$$

Correctness proof. The correctness proof of the above schedulability test contains two parts: (i) all deadlines being met under LO-mode (Theorem 1) and (ii) HI-criticality deadlines under HI-mode (Theorem 2).

Theorem 1 *Under EDF-VD, all tasks meet their deadlines in* LO*-mode (where all jobs complete upon receiving execution time up to their* LO*-WCETs) if*

$$x \geq \frac{U_{\mathrm{HI}}^{\mathrm{LO}}}{1 - U_{\mathrm{LO}}^{\mathrm{LO}}}. \tag{8}$$

Proof: By the density test in [13], $U_{\mathrm{LO}}^{\mathrm{LO}} + U_{\mathrm{HI}}^{\mathrm{LO}}/x \leq 1$ is sufficient to ensure that EDF-VD successfully schedules all LO-criticality behaviors of τ. Theorem follows by rearranging this inequality. □

Lemma 1 *For any period of length t, total demand by* HI*-criticality tasks can not exceed $(U_{\mathrm{HI}}^{\mathrm{LO}} + \sum_{i=1}^{N} \delta_i)t$.*

Proof: It is assumed that HI-criticality tasks are ordered (decreasingly) according to their δ_i values. Consider the scenario that tasks $\tau_1, ..., \tau_N$ requires for executions more than its c_i(LO), than it is obvious that the total demand by HI-criticality tasks can not exceed $(U_{\mathrm{HI}}^{\mathrm{LO}} + \sum_{i=1}^{N} \delta_i)t$.

We prove by contradiction. Assume there is another scenario with total demand larger than the above mentioned case. We can always identify the difference between this new release pattern with the one we have – by "replacing" one job that is released by one of the tasks from $\tau_1, ..., \tau_N$ by a job released by some task other than $\tau_1, ..., \tau_N$, one at a time. We can not directly add any task since we have reached the maximum number (N) of tasks that can require demands higher than their LO-WCETs. However, since tasks are ordered by their δ_i values decreasingly, the demand of new tasks in the period of interest (between the release and the deadline of the job being replaced) cannot exceed the one created by the original job. Therefore, such "swaps" will always result into a decreasing of the total demand, which contradicts our assumption. □

Theorem 2 *Under EDF-VD, all* HI*-criticality tasks meet their deadlines in* HI*-mode if Inequality* (6) *holds. In the* HI*-mode, some but no more than N* HI*-criticality job(s) have not completed upon receiving execution time up to their* LO*-WCETs but will complete upon receiving execution time up to their* HI*-WCETs.*

Proof: It is assumed that the reader is familiar with the correctness proof for EDF-VD in [2], so we will skip many parts of the proof that will look identical. We also adopt all notations there: t_f as the first HI-criticality deadline that is missed, 0 as the last idle instant before t_f, $t^* < t_f$ as the mode switch point, η_i denote the amount of execution over the interval $[0, t_f)$ that is needed by jobs generated by task τ_i. a_1 as the release time of the job with the earliest release time amongst all those that execute in $[t^*, t_f)$, and η_i.

The proofs of Facts 1 and 2 remain unchanged due to the minimal set assumption and the same strategy used under LO mode. Regarding Fact 3, here we calculate the maximum total HI-criticality demand over $[0, t_f)$ instead, and then sum the cumulative demand of all the tasks over $[0, t_f)$.

From Lemma 1 we know that during interval $[a_1, t_f)$, the total HI-criticality demand will not exceed $(t_f - a_1)(U_{\text{HI}}^{\text{LO}} + \sum_{i=1}^{N} \delta_i)$. As a result, we have the following upper bound for cumulative demand of all HI-criticality tasks over $[0, t_f)$:

$$\sum_{\chi_i=\text{HI}} \eta_i \leq \frac{a_1}{x} U_{\text{HI}}^{\text{LO}} + (t_f - a_1)(U_{\text{HI}}^{\text{LO}} + \sum_{i=1}^{N} \delta_i). \tag{9}$$

From the infeasibility of the instance (due to deadline miss at t_f), we have

$$a_1 + (t_f - a_1)(xU_{\text{LO}}^{\text{LO}} + U_{\text{HI}}^{\text{LO}} + \sum_{i=1}^{N} \delta_i) > t_f \tag{10}$$

$$\Leftrightarrow \quad (t_f - a_1)(xU_{\text{LO}}^{\text{LO}} + U_{\text{HI}}^{\text{LO}} + \sum_{i=1}^{N} \delta_i) > t_f - a_1 \tag{11}$$

$$\Leftrightarrow \quad xU_{\text{LO}}^{\text{LO}} + U_{\text{HI}}^{\text{LO}} + \sum_{i=1}^{N} \delta_i > 1 \tag{12}$$

The contrapositive is exactly Inequality (6), which is sufficient to ensure HI-criticality schedulability by EDF-VD. □

Runtime behavior. During runtime, if a LO-criticality job of task τ_i arrives at time-instant t_a, then the priority of this job is determined by its deadline $t_a + T_i$, whereas its priority will be determined by its virtual deadline $t_a + T_i'$ if it is a HI-criticality job. If any HI-criticality job executes for a duration exceeding its LO-WCET without signaling completion, the scheduler immediately discards all LO-criticality jobs[8] and executes HI-criticality HI-criticality tasks according to EDF order with their *actual* (instead of virtual) deadlines. Moreover, *idleness* always serves as the trigger to LO-criticality mode of the system.

Additional discussions. Under the MC scheduling approach, LO-criticality jobs will be dropped in the HI-criticality mode, and any HI-criticality job overrunning its LO-WCET will trigger the mode switch. With the proposed model, this dropping may not be necessary. The following inequality should be examined before the system starts any execution:

$$U_{\text{LO}}^{\text{LO}} + U_{\text{HI}}^{\text{LO}} + \sum_{i=1}^{N} \delta_i \leq 1. \tag{13}$$

If Inequality (13) is true, then actually no mode switch nor virtual deadline is needed. The system can be scheduled by ordinary preemptive EDF scheduler and all deadlines will be met. This result directly follows from Lemma 1. If Inequality (13) is false, we then apply the MC scheduling techniques described earlier in this section, and examine Inequality (6) to verify the schedulability.

[8] An efficient implementation of such a run-time dispatcher may be obtained using the technique described in [2, Sec. V-A], to have runtime that is logarithmic in the number of tasks.

4 Experimental Evaluation

In Section 2, we have proposed a new MC system model that specifies the maximum number of tasks N that can simultaneously experience HI-criticality behaviors within *any* time window of length $\max_i\{T_i\}$. With this additional information in the model comparing to the classic Vestal model, we are expecting a "better" schedulability result for EDF-VD under the new model.

In this section, we conduct schedulability experiments to evaluate the effectiveness of the proposed model against the classic Vestal model. Various per-mode utilizations as well as N's are considered in our experiments. The MC task instances in our experiments are generated by the MC task generator described in [3], which has passed artifact evaluation.

In each set of our experiments, the average normalized utilization [4] of the generated task set range from 0.5 to 1 with increasing at step size 0.05. For every average utilization, 1000 task sets are generated and the acceptance ratio indicates how many of them passed the corresponding schedulability test (and thus can be scheduled correctly).

Figures 2, 3, and 4 demonstrate the effectiveness of the new model along with the corresponding EDF-VD schedulability test under various settings of numbers of HI-criticality tasks (16, 32, and 64) and sizes of N (i.e., number of HI-criticality tasks that can simultaneously exceed LO-WCETs. It is natral that the acceptance ratios will drop when system is more heavily loaded (with higher utilization). However, we notice that our methods maintains relatively higher acceptance ratio even when normalized utilization gets close to 1.

These results also show that, if less pessimistic assumptions (about the N) can be made, the schedulability can be significantly increased. We do not notice much different in the trends when total number of HI-criticality tasks varies.

5 Conclusion

This paper extends the classic Vestal model for MC scheduling by allowing system designers to specify an additional parameter, representing the maximum number of HI-criticality tasks that may simultaneously exceed their LO-WCETs during runtime. By simultaneously, we mean within any sliding time window of length less than or equal to the maximum period among all tasks. The well-known scheduler, namely EDF-VD, has been studied under the proposed model, and a new schedulability test has been proposed and analyzed. Schedulability experiments have demonstrated that by applying the proposed model in place of the classic Vestal model, significant schedulability improvements can be achieved.

For future work, we would like to consider fixed-priority schedulers under the proposed model, in addition to the deadline-based scheduler, EDF-VD, we considered in this paper. The results may also be extended (at a measurable cost) into multi-processor and/or multi-criticality-level cases.

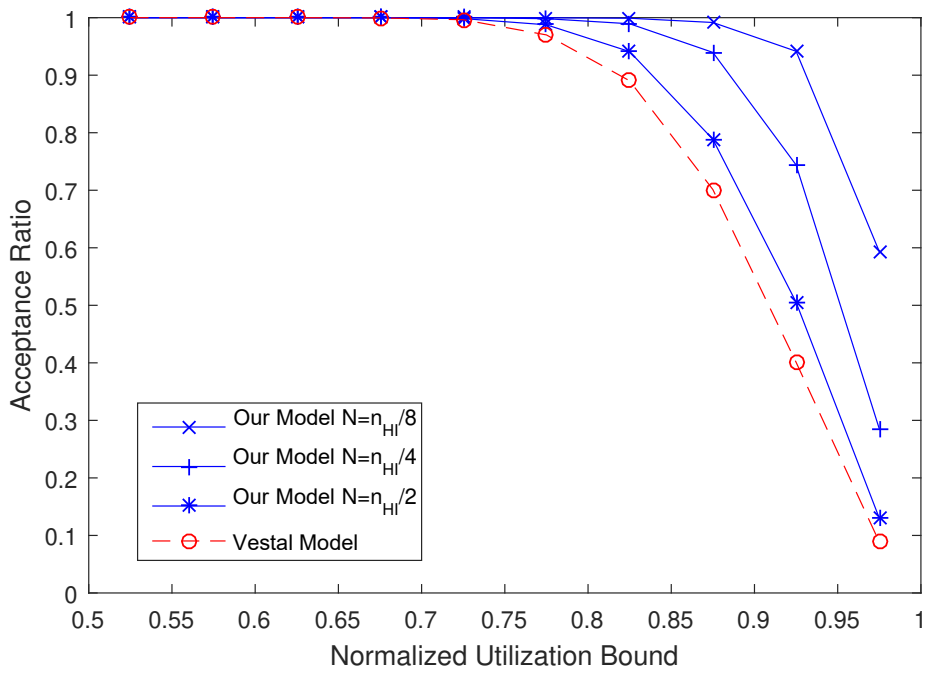

Fig. 2. Schedulability ratio comparison of our proposed model and the classic Vestal model under various N's, with $n_{\mathrm{HI}} = 16$.

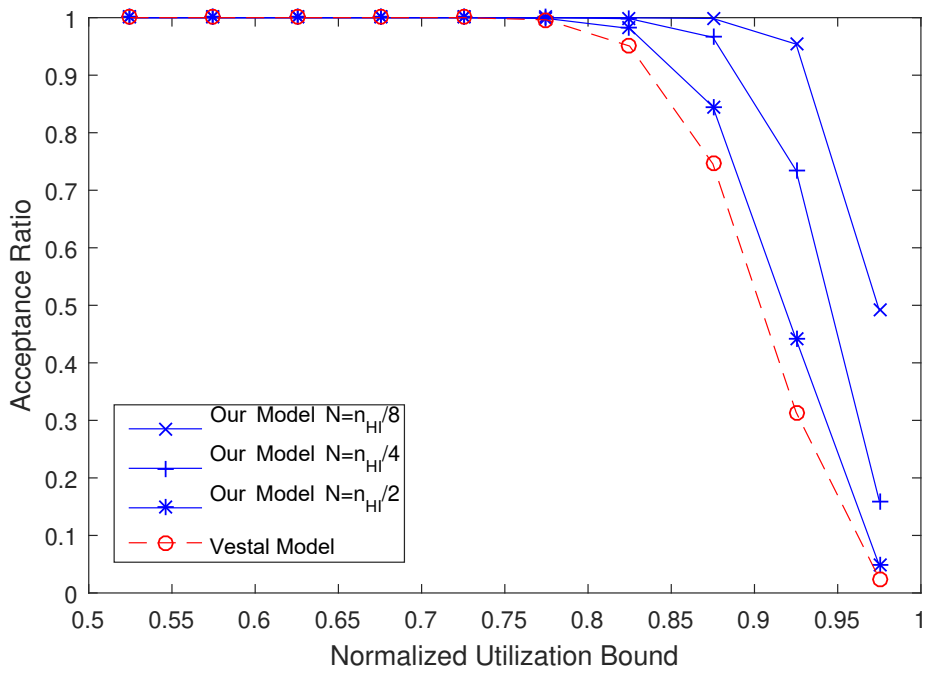

Fig. 3. Schedulability ratio comparison of our proposed model and the classic Vestal model under various N's, with $n_{\mathrm{HI}} = 32$.

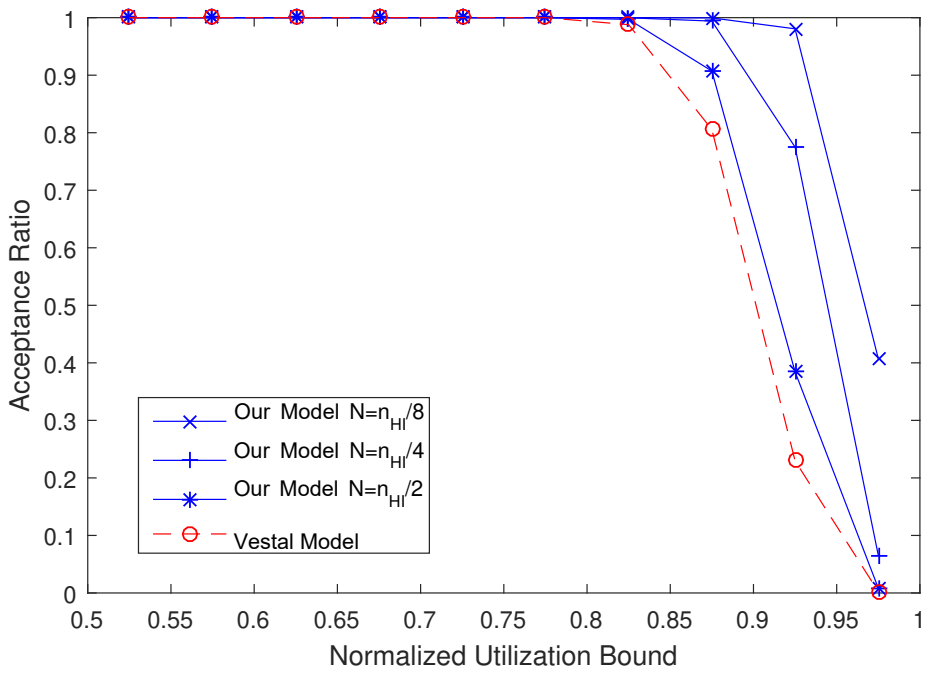

Fig. 4. Schedulability ratio comparison of our proposed model and the classic Vestal model under various N's, with $n_{\mathrm{HI}} = 64$.

References

1. S. Baruah. Mixed criticality schedulability analysis is highly intractable. `http://www.cs.unc.edu/~baruah/Submitted/02cxty.pdf`, 2008.
2. S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie. The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems. In *the 24th Euromicro Conference on Real-Time Systems (ECRTS'12)*, 2012.
3. S. Baruah, A. Burns, and Z. Guo. Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors. In *the 28th Euromicro Conference on Real-Time Systems (ECRTS'16)*, 2016.
4. S. Baruah, A. Easwaran, and Z. Guo. MC-Fluid: simplified and optimally quantified. In *Proceedings of the 36th IEEE Real-Time Systems Symposium (RTSS'15)*, 2015.
5. S. Baruah and Z. Guo. Mixed-criticality scheduling upon varying-speed processors. In *the 34th IEEE Real-Time Systems Symposium (RTSS'13)*, 2013.
6. S. Baruah and Z. Guo. Scheduling mixed-criticality implicit-deadline sporadic task systems upon a varying-speed processor. In *Proceedings of the 35th IEEE Real-Time Systems Symposium (RTSS'14)*, 2014.
7. A. Burns and R. Davis. Mixed-criticality systems: A review. `http://www-users.cs.york.ac.uk/~burns/review.pdf`, 2016.
8. L. Cucu-Grosjean. Independence - a misunderstood property of and for (probabilistic) real-time systems. Invited paper to the 60th birthday of A. Burns, 2013.
9. Z. Guo and S. Baruah. Mixed-criticality scheduling upon varying-speed multiprocessors. *Leibniz Transactions on Embedded Systems*, 1(2):3:1–3:19, 2014.
10. Z. Guo and S. Baruah. Mixed-criticality scheduling upon varying-speed multiprocessors. In *Proceedings of the 12th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC'14)*, 2014.
11. Z. Guo and S. Baruah. The concurrent consideration of uncertainty in WCETs and processor speeds in mixed-criticality systems. In *the 23rd International Conference on Real-Time and Network Systems (RTNS'15)*, 2015.
12. Z. Guo, L. Santinalli, and K. Yang. EDF schedulability analysis on mixed-criticality systems with permitted failure probability. In *Proceedings of the 21st IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'15)*, 2015.
13. J. Liu. *Real-Time Systems*. Prentice-Hall, 2000.
14. L. Santinalli and Z. Guo. A sensitivity analysis for mixed criticality: Trading criticality with computational resource. In *Proceedings of IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA'18)*, 2018.
15. S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *the 28th IEEE Real-Time Systems Symposium (RTSS'07)*, 2007.