Uniprocessor Mixed-Criticality Scheduling with Graceful Degradation by Completion Rate

Zhishan Guo[1] Kecheng Yang[2] Sudharsan Vaidhun[1] Samsil Arefin[3] Sajal K. Das[3] Haoyi Xiong[4]
[1]Department of Electric and Computer Engineering, University of Central Florida
[2]Department of Computer Science, Texas State University
[3]Department of Computer Science, Missouri University of Science and Technology
[4]Baidu Inc., Beijing, China

***Abstract*—The scheduling of mixed-criticality (MC) systems with graceful degradation is considered, where LO-criticality tasks are guaranteed some service in HI mode in the form of minimum cumulative completion rates. First, we present an easy-to-implement admission-control procedure to determine which LO-criticality jobs to complete in HI mode. Then, we propose a demand-bound-function-based MC schedulability test that runs in pseudo-polynomial time for such systems under EDF-VD scheduling, wherein two virtual deadline setting heuristics are considered. Furthermore, we discuss a mechanism for the system to switch back from HI to LO mode and quantify the maximum time duration such recovery process would take. Finally, we show the effectiveness of our proposed method by experimental evaluation in comparison to state-of-the-art MC schedulers.**

***Index Terms*—graceful degradation, mixed criticality, uniprocessor, EDF-GVD**

I. Introduction and Motivation

There is an emerging trend in real-time system's design and implementation towards mixed-criticality (MC), where functionalities of different degrees of importance (or criticalities) are implemented upon a shared platform. Under such design, the less important tasks are expected to utilize the computing resources only when important tasks complete earlier than their desired worst cases, such that processor capacities are well utilized. In the meanwhile, when tasks of higher importance execute beyond their estimated common case running time (occasionally), the less important tasks may be dropped. This is what has been broadly called *Mixed-Criticality (MC)* scheduling in the real-time systems community.

Prior research on MC scheduling mostly focused upon the Vestal model [32], which defines the correctness as follows: *all deadlines will be met under normal circumstances, while if some more important tasks overrun, a mode switch is triggered and only* HI*-critical deadline will be met.* Specifically, under the two-criticality-level case, each task is designated as being of either higher (HI) or lower (LO) criticality. Two worst-case execution time (WCET) estimations are specified for each HI-criticality task: a LO-WCET and a HI-WCET. The HI-WCET is larger than the LO-WCET (potentially by several orders of magnitude). For each LO-criticality task, only one WCET is specified. Whenever there is one HI-criticality task that does not signal its completion after exhausting its LO-WCET, a *system-wide mode switch* will be triggered; *all* HI-criticality tasks may *simultaneously* exceed their LO-WCETs requiring executions up to their HI-WCETs under the new mode. A LO-criticality task exceeding its WCET is considered an erroneous condition and the task instance is terminated on such occurrence. Please refer to [13] for an up-to-date thorough review and refer to Section II for formal definitions.

Contact information: zhishan.guo@ucf.edu, yangk@txstate.edu.

This system-mode based MC model has been proved very successful in identifying some of the core challenges that arise in resource-efficient scheduling of real-time systems, and spawning a large body of research to tackle some of these challenges. However, this model has met with some criticism from systems engineers and researchers [12] that it does not match their expectations in an important aspect: LO*-criticality functionalities are* not *non-critical — they should be guaranteed with some* degree of service*, regardless of* HI*-criticality tasks' behaviors.*

Graceful Degradation. The term *graceful degradation* of any system refers to the ability to maintain limited functionality even when a large portion of it is destroyed or rendered inoperative. With graceful degradation, LO-criticality tasks are designed to receive certain amount of service (instead of fully being abandoned) upon a mode switch of an MC system. A wide range of applications can benefit from the system design incorporating graceful degradation. For example, a networked control system can run stably while skipping a limited number of computation tasks (for control signal processing) within a certain period [11]. Moreover, it is possible to guarantee the stability of a continuous closed-loop system with an optimal disturbance rejection performance via completing a minimal fraction of some computations (tasks) on time [26].

In the real-time systems community, some existing work in MC scheduling considers the concept of graceful degradation and proposes effective scheduling techniques, such as fluid-based scheduling [5], utilization-based earliest deadline first with virtual deadlines (EDF-VD) [25], probabilistic model-based scheduling [20], putting restriction on the asymptotic rate-based correctness notation [28] [1], and involving mixed-criticality weakly hard constraints [17].

However, most of the existing works have at least one of the following two issues:

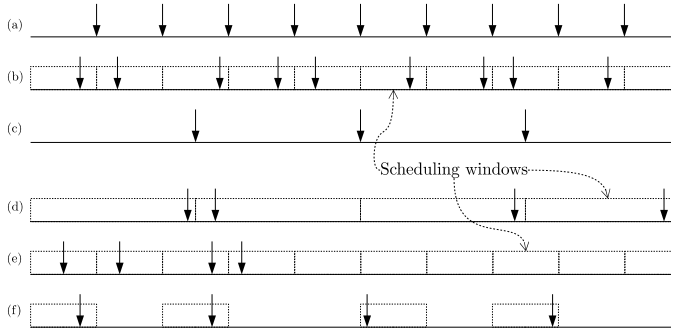


Fig. 1: (a) Desired control updates in LO mode; (b) A possible schedule during LO mode; (c) Desired control updates in HI mode; (d) A possible schedule in HI mode using utilization based schedulers; (e) A possible schedule in HI mode using schedulers with asymptotic bound on the degraded rate; (f) A possible schedule in HI mode using our proposed scheduler with guaranteed lower bound on degradation.

1) Utilization based schedulers reduce LO-criticality utilization upon a mode switch. On one hand, a shortened execution time for each task may lead to unfinished execution (malfunction), since LO-criticality tasks' execution time is already optimistically estimated (with tight margin). On the other hand, a longer period may results in the loss of timeliness. For instance, delayed responses or calculations will not only result in degraded control performance, but may also lead to data corruption and system malfunction due to unsynchronized communication in a sensor-based distributed control system.

2) Asymptotic bounds on admission rates do not guarantee the service rate in any fine granularity: both 1-out-of-4 and $10k$-out-of-$40k$ are considered as 0.25 service rate in the long term. However, a feasible schedule under the latter condition may drop thousands of consecutive jobs of the same task, which could lead to severe malfunction of certain parts of a system.

To further drive the point, let us consider a control task in a cyber-physical system that provides control updates at the rate of $100\ Hz$ for a smooth and stable control of the system as shown in Figure 1(a). Even with a periodic task model, depending on the scheduler, there is a jitter introduced in the control update rate. A possible control update sequence with jitter is shown in Figure 1(b). If the control task is of LO-criticality, then to maintain a stable control when there occurs a mode switch, let the required minimum control update rate be $25\ Hz$ and the expected update sequence is shown in Figure 1(c). With the utilization based schedulers, increasing the period will worsen the jitter as well as the update rate. In other words, instead of the LO-mode requirement of a control update every $40\ ms$, the schedule can result in an update sequence with $\approx 80\ ms$ between two consecutive updates in the worst case as seen in Figure 1(d) between jobs 2 and 3. Similarly, with the asymptotic bounds on the update rate guaranteed, there could be a pattern with consecutive updates every $10\ ms$ for 50 updates and then no update for the next 50 instances. We address these issues in the existing methodologies by guaranteeing a lower bound on the degradation of the LO-criticality tasks. Figure 1(f) shows the resulting task window where the control updates can occur for a scheduler with a guaranteed upper bound on the cumulative dropped task instances. Although the rate is not exactly the expected rate, the jitter has a tighter bound and there is a pattern in the job sequence.

Another prominent application which requires guaranteed bounds on degradation is communication networks. In a constrained shared bandwidth communication where real-time and nonreal-time messages require bandwidth access, nonreal-time messages can be degraded to maintain a lower but non-zero bandwidth access to handle more end users. Such a degradation may lead to loss of quality in communication for the end users, which when unbounded causes disconnection [22]. In such applications, a rate control mechanism [24] provides the minimum rate that has to be guaranteed for an accepted task. As seen in the above mentioned applications, the quality of control in a cyber-physical system or the quality of experience in a communication network is affected by degradation. There are techniques in those application that have a bound on the maximum degradation that can be tolerated and our proposed algorithm guarantees those bounds while scheduling.

In this paper, we target the scheduling of *dual-criticality systems with graceful degradation upon a uniprocessor*. Specifically, we make the following *key contributions* in this paper:

We propose a new MC system model by re-defining the HI-criticality mode, where LO-criticality tasks can be guaranteed with a graceful degradation of service specified by the system designers.

We introduce a completion rate r_i to Vestal's MC task model to better fit the need from system designers in the network and control communities: upon a system-wide mode switch, for any $N \in \mathbb{Z}^+$, at least $\lceil r_i \cdot N \rceil$ out of N new consecutive job releases (by task τ_i) will receive full execution within their original scheduling window. This model provides a much stronger guarantee than several existing works (that shrink execution time or only guarantee asymtotic rates) and is applicable to many senarios (as mentioned in Sec II).

We propose an easy-to-implement admission-control procedure for LO-criticality tasks upon mode switch to guarantee the rate r_i for any task τ_i. Moreover, if a rate r_i can be expressed in the fraction form $r_i = m_i/k_i$, then our procedure further guarantees that **after mode switch occurs**, at least m_i jobs will receive full execution out of **any** k_i consecutive jobs of a LO-criticality task τ_i.

We adapt the earliest-deadline-first (EDF) with virtual deadlines scheduler, and proposed a pseudo-polynomial time schedulability test for the MC system under new model, based on MC demand-bound function (DBF) analysis.

We include a backward mode switch mechanism (from HI mode back to LO mode), then prove the maximum length of period (upper bound) to bring the system back to LO mode safely, under different scenarios of execution patterns via the proposed scheduler. To our knowledge, this is the first work that provided the theoretical analysis for upper bound of mode switch waiting time. We conduct simulation experiments to

verify the theoretical results and demonstrate the effectiveness of our algorithm comparing to existing MC schedulers.

The rest of this paper is organized as follows. Section II presents the system model and formally defines our problem. Section III proposes a new admission-control protocol for LO-criticality tasks under HI mode. Section IV presents our proposed algorithm earliest deadline first with graceful virtual deadline (EDF-GVD) and derives its associated schedulability test, while Section V further proves the upper bound for the period EDF-GVD would take to trigger a mode switch back to LO mode. Section VI implements EDF-GVD and compares the proposed method with other state-of-the-art MC schedulers. Section VII discusses related work and Section VIII concludes the paper.

II. System Model

In this paper, we consider a workload model consisting of independent mixed-criticality (MC) constrained-deadline sporadic tasks. We assume that each task generates an infinite number of MC jobs.

Traditional dual-criticality MC task model considers two different WCET estimations, C^{LO} and C^{HI}, for each HI-criticality task. Under LO mode, all the tasks finish execution upon receiving a cumulative execution of length C^{LO}. When a HI-criticality task takes longer time than C^{LO}, the system is switched to HI mode and all LO-criticality tasks are dropped. HI-criticality tasks are guaranteed to receive an execution length up to their HI-WCETs under HI mode.

In this paper, in addition to the existing MC task model, we introduce a minimum cumulative admission rate r_i for each LO-criticality task τ_i. The value of r_i for a specific task τ_i denotes that, at HI mode, the *completion rate* of that task must be at least r_i. That is, for the first N jobs released by τ_i after the mode switch (to the HI mode), at least $\lceil r_i \cdot N \rceil$ number of jobs should be completed.

Note that we separately model a task τ_i according to its criticality level $\chi_i \in \{\text{LO}, \text{HI}\}$, such that the whole task set τ can be separated into the LO-criticality task set $\tau_{\text{LO}} = \{\tau_i | \chi_i = \text{LO}\}$ and the HI-criticality task set $\tau_{\text{HI}} = \{\tau_i | \chi_i = \text{HI}\}$. Each LO-criticality task τ_i can be characterized as:

$$\tau_i = \{C_i, T_i, D_i, r_i, \chi_i\}\,,\ \forall \tau_i \in \tau_{\text{LO}},$$

while each HI-criticality task τ_i can be characterized as:

$$\tau_i = \{C_i^{\text{LO}}, C_i^{\text{HI}}, T_i, D_i, \chi_i\}\,,\ \forall \tau_i \in \tau_{\text{HI}}.$$

That is, each LO-criticality task τ_i has only a single WCET estimation C_i plus a required completion rate r_i in HI mode. In contrast, each HI-criticality task τ_i has two WCET estimations – a less pessimistic estimation C_i^{LO} and a more pessimistic estimation C_i^{HI}– and every HI-criticality job must complete by its deadline in both LO and HI modes.

The r_i value may vary for each LO-criticality task – when they are all zero, our problem is identical to the traditional sporadic MC task scheduling one; while if they take value 1, the system becomes non-mixed-criticality as no jobs can be dropped (HImode schedulability will dominate). As a result, the system model described in this paper is not only a *generalization of many existing models*, but also *bridges mixed-criticality* and *non-mixed-criticality scheduling*. We provide the system designer the full flexibility to require different levels of guarantees for LO-criticality tasks under HI mode.

Correctness Criteria. An MC scheduling is correct if both the following properties are satisfied:

- Under LO mode, each task τ_i signals its completion upon receiving an execution length no more than C_i^{LO}, and all jobs complete on or before their deadlines.
- Under HI mode, each HI-criticality task τ_i may receive an execution up to C_i^{HI}, while LO-criticality jobs will nevertheless be executed at a minimum cumulative admission rate r_i. In other words, starting from the last HI mode switch point, out of any $N \in \mathbb{Z}^+$ consecutive jobs released by any LO-criticality task τ_i, $\lceil r_i \cdot N \rceil$ jobs *are guaranteed to receive full execution (between their release time and the deadlines)*.

The schedulability analysis of such task sets faces two key challenges: (1) the correctness definition requires the completion rate of a LO-criticality task *never drops below r_i upon a mode switch*. In Section III, we identify a proper admission-control mechanism such that the condition is satisfied. (2) The m-out-of-k scheduling problem arising in our research is a strong guarantee, which cannot be ensured by existing work on graceful degradation [5] [25] with shrinking WCETs.

III. Admission Control Under HI Mode

In the proposed model, every LO-criticality task τ_i has a minimum cumulative admission rate r_i upon mode switch. Due to HI-criticality tasks' overruns, not all jobs of LO-criticality tasks can be executed. The dropping of LO-criticality jobs need to respect the minimum cumulative admission rate r_i; i.e., for any task τ_i, if $\lceil r_i \cdot N \rceil$ out of the first N consecutive jobs after the mode switch are executed ($\forall N \in \mathbb{Z}^+$), then r_i is satisfied. To accomplish this goal we need to design an admission control protocol, where in HI mode, the execution of LO-criticality jobs is controlled dynamically.

Algorithm 1: Admission Control Procedure

```
Data: (r_i) values for all LO-criticality tasks
foreach τ_i ∈ τ_LO do
    a_i = 0;                    // Number of executed jobs
end
while true do
    if j_b ∈ τ_i released then
        // j_b is b^th job of τ_i
        if (a_i < b × r_i) then
            // schedule j_b using EDF
            a_i = a_i + 1;
        end
        else
            // The job j_b is dropped
        end
    end
end
```

Admission Control Algorithm. The admission control procedure for LO-criticality tasks under HI mode is presented in

Algorithm 1. As per our correctness criteria, to satisfy r_i for any task τ_i, $\lceil r_i \cdot N \rceil$ out of the **first** N **consecutive jobs after** the mode switch are required to be executed. To ensure this, the algorithm controls the LO-criticality job admissions in a way, such that the admission rate (the proportion of admitted jobs over all released jobs) never drops below the cumulative admission rate r_i. The algorithm keeps track of the total number of executed jobs (a_i) while a newly released job j_b is scheduled for execution only if the condition $a_i < b \times r_i$ satisfies; otherwise, the job is dropped.

We now show how the admission control scheme works with an example of several tasks.

Example III.1. *Consider a task set τ with the requirements of minimum cumulative admission rates as given in Table I, where $S(r_i)$ denotes the maximum number of consecutive drops to a certain task (which is a function of r_i).*

TABLE I: Sample completion rates with associated admission patterns and maximum job acceptance separation.

Task ID	r_i	Job Admission	$S(r_i)$
τ_1	0.4	$\{10100\}^\infty$	2
τ_2	0.625	$\{11011010\}^\infty$	1
τ_3	$1/\sqrt{2}$	$11101101110\cdots$	1

In Table I, the job admission pattern is shown by a series of 1's and 0's where 1's represent the admitted jobs while 0's represent the dropped ones. When running task τ_1, its cumulative admission rate (the proportion of admitted jobs over all released jobs) pattern is $1, \frac{1}{2}, \frac{2}{3}, \frac{1}{2}, \frac{2}{5}, \frac{1}{2}, \frac{3}{7}, \frac{1}{2}, \frac{4}{9}, \frac{2}{5}, \cdots$, which never drops below $r_i = 0.4$. The same claims can be made for τ_2 and τ_3. The admission control is dynamically determined by checking the condition $a < b \cdot r_3$ for all tasks.

Note that, when r_i is a rational number (e.g., r_1 and r_2), the admission control follows a repeated pattern, however, for irrational r_i values (e.g., r_3), there will not be any repetition. Also, the maximum number of consecutive drops can be calculated from r_i which might be critical for many applications.

Remark 1. *One important aspect of our admission-control protocol is that we minimize the maximum number of consecutive drops $S(r_i)$ while maintaining the minimum cumulative admission rate r_i. The following equation reveals the relationship between $S(r_i)$ and r_i:*

$$S(r_i) = \left\lceil \frac{1}{r_i} \right\rceil - 1. \tag{1}$$

Remark 2. *When r_i is a rational number for task τ_i, it can be represented in a fractional form (m_i/k_i), where m_i and k_i are co-prime integer values. In this case, it is guaranteed that for any consecutive k_i jobs released by task τ_i, at least m_i of them will be admitted.*

Remark 3. *Upon a mode switch (to HI mode), for first of any $N \in \mathbb{Z}^+$ consecutive job releases of any LO-criticality task τ_i, the admitted number of jobs equals to $\lceil r_i \cdot N \rceil$. Note that the guarantee is based on N, the number of released jobs [1] so far after mode switch[2]. The model fits both periodic and sporadic release patterns.*

IV. Scheduler and Schedulability Analysis

With the admission control procedure described in Section III, the minimum cumulative admission rate requirements are transformed into a certain execution pattern, which can be described with constrained-deadline sporadic MC task model, for each LO-criticality task upon a mode switch. In this section, we proposed a novel algorithm named EDF-GVD (Graceful-Virtual-Deadline). Compared to existing similar approaches like EDF-VD [3], our algorithm provides graceful degradation to LO-criticality tasks under HI mode. To guarantee a graceful degradation, new virtual deadline setting mechanisms and associated schedulability test needs to be derived. As pointed out in Sections I and II, our graceful degradation definition follows the needs for many applications in network communication and control systems, and provides stronger guarantees than existing works on graceful degradation (with shrunk WCET or extended period), and thus dominates them. As a result, the standard MC sporadic task schedulability analysis (that are utilization based) does not apply here.

In this section, we first describe our EDF-GVD algorithm and propose two virtual deadline setting mechanisms, then present the schedulability test with time complexity analysis and correctness guarantees.

A. Algorithm EDF-GVD

Let $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$ denote the targeted MC constrained-deadline sporadic task set, to be scheduled on a uniprocessor platform.

Virtual Deadline Settings. Setting the virtual deadlines is usually a search problem in the EDF-VD family. We first provided a simple linear-time setting heuristic as a starting point. Next, we provided a refined binary search that approximates to the optimal uniformed setting to any degree.

To calculate the virtual deadlines D_i^v for HI-criticality tasks we first use a simple yet efficient heuristic such that the per-mode utilizations are the same for each task, i.e.,

$$D_i^v = \frac{C_i^{\text{LO}}}{C_i^{\text{HI}}} D_i. \tag{2}$$

Since such heuristic may not lead to an *optimal* virtual deadline setting, we propose a second scheme for setting virtual deadlines:

$$D_i^v = q \cdot D_i, \ \ s.t. \ \forall i \ \chi_i = \text{HI}. \tag{3}$$

where $q \in (0, 1]$ is the common virtual deadline scaling factor for the whole set, to be determined using Algorithm 2.

[1] Note that the guarantee is based on the first N jobs instead of any N jobs, or else by taking $N = 1$, we are not allowed to drop any job.

[2] The LO-criticality job during mode switch is always dropped for minimizing schedulability interferences between two modes. Such single drop typically do not affect system performance hugely, and the proposed appraoch guarantees that its seccessor is always admitted and executed correctly; i.e., all patterns begins with 1.

Algorithm 2: Virtual Deadline Setting

Data: Task set τ, Accuracy ϵ
$\Delta = 0.5$ //step size;
$q = 0.5$ //virtual deadline shrinking parameter;
while $\Delta \geq \epsilon$ **do**
  $\Delta = \Delta/2$;
  foreach $\tau_i \in \tau_{HI}$ **do**
    $D_i^v = q \cdot D_i$;
  end
  $C_A = Check$(Condition (A) in Sec IV-B);
  $C_B = Check$(Condition (B) in Sec IV-B);
  if $C_A = true$ && $C_B = true$ **then**
    Return q;
  end
  else if $C_A = true$ && $C_B = false$ **then**
    $q = q - \Delta$;
  end
  else if $C_A = false$ && $C_B = true$ **then**
    $q = q + \Delta$;
  end
  else
    Return failure; //no q can be found
  end
  Return -1; //still possible with a smaller ϵ
end

In short, Algorithm 2 performs a binary search for q over the range $(0, 1]$, *to any desired degree of accuracy* $\epsilon > 0$, to determine a proper value such that both LO and HI mode correctness can be satisfied. In each round, the algorithm first decreases the step size by half and checks the schedulability conditions (based on DBF, introduced in Section IV-B) for each mode. If both conditions are satisfied, we declare a victory; if neither is satisfied, we claim a failure; if the schedulability under HI (LO) mode is violated, we reduce (increase) q by the current step size and move to the next round. With the proper q value that Algorithm 2 returns, we set virtual deadlines of HI-criticality tasks according to Equation (3).

Time complexity. As the schedulability test for any q (mentioned in Sec IV-B) takes pseudo-polynomial time, the whole process mentioned above will take pseudo-polynomial time (to a desired degree of accuracy, say 2^{-10}).

Run-time behavior. After computing the feasible virtual deadlines D_i^v, run-time scheduling for all tasks is done in an EDF manner. Specifically, under the LO mode, jobs of each HI-criticality task τ_i are assigned a virtual deadline of D_i^v after their releases, while relative deadline of a LO-criticality job remains D_i. If some HI-criticality job does not signal its completion upon receiving a cumulative execution time of its LO-criticality WCET, the system mode is switched to HI and *current unfinished* LO-*critcality jobs are dropped*. Future releases of LO-criticality jobs are handled based on the admission control protocol described in Section III, while all HI-criticality jobs will be admitted. All jobs will be scheduled with EDF within their actual deadline D_i in HI mode.

Example IV.1. *Consider the MC system consisting of three tasks shown in Table II.*

TABLE II: An MC set with minimum degradation execution rates.

Task ID	C_i (C_i^{LO})	C_i^{HI}	T_i	D_i	χ_i	r_i
τ_1	1	3	6	6	HI	-
τ_2	1	-	3	3	LO	0.5
τ_3	2	-	6	4	LO	0.4

The task set is EDF-schedulable under LO *mode as the system density (i.e.,* $\sum_i C_i/D_i = 1/6 + 1/3 + 2/4$*) is* 1*. However, it is not EDF schedulable under* HI *mode without dropping any job since the overall system utilization (i.e.,* $\sum_i C_i/T_i = 3/6 + 1/3 + 2/6 = 7/6$*) becomes greater than* 1*. Fig. 2 shows that the task set is EDF-GVD schedulable with a virtual deadline setting for* τ_1 *as* $D_1^v = 4$*. Under* HI *mode, selected jobs of both* LO*-criticality tasks (*τ_2 *and* τ_3*) receive full amount of execution, with minimum cumulative admission rates at* 0.5 *and* 0.4 *respectively.*

Under LO *mode, the* HI*-criticality task* τ_1 *is scheduled using virtual deadline* $D_1^v = 4$*. At time* $t = 10$*, the second job of* τ_1 *has executed for a length of* $C_1^{\text{LO}} = 1$ *and did not signal its completion, hence the system is switched to* HI *mode. At the same time, the pending (*4^{th}*)* LO*-criticality job of* τ_2 *is dropped. After the mode switch, the* HI*-crticality task* τ_1 *is scheduled using EDF with* $C_1^{\text{HI}} = 3$ *and deadline* $D_1 = 6$*, while the* LO*-criticality jobs are scheduled based on the admission control procedure described in Section III (where un-admitted jobs are marked 'dropped' in the figure). The system is ready for a potential switch back to* LO *mode when the processor is idle and the last jobs of all* HI*-criticality tasks are completed within its* C^{LO} *amount of time (denoted by dashed green lines). However, if the last job of any* HI*-criticality task executes more than its* C^{LO}*, there cannot be any backward mode switch even if the system is idle (denoted by dashed blue lines)*

B. DBF Based Schedulability Analysis

Given a virtual deadline setting from Sec. IV-A, we perform a schedulability test in this section to check whether both LO and HI mode correctness can be guaranteed. We begin with the *demand bound function* (DBF), which has been demonstrated to be a successful approach to analyze the schedulability of both ordinary [9] and mixed-criticality [15] real-time systems.

The function $\text{dbf}(\tau_i, \ell)$ denotes the maximum execution demand of task τ_i during *any* time interval of length ℓ, where the *demand* is calculated by the cumulative execution requirement of all jobs of τ_i that have both release times and deadlines in that interval. Lemma 1 (from [9]) presents the classical necessary and sufficient schedulability test for non-mixed-criticality tasks with EDF.

Lemma 1. *A constrained-deadline task set* τ *can be successfully scheduled by EDF on a uniprocessor, if and only if*

$$\sum_{\tau_i \in \tau} \text{dbf}(\tau_i, \ell) \leq \ell, \ \forall \ell \geq 0.$$

Ekberg and Yi [15] proposed to analyze the DBF in HI and LO modes separately, and derived a schedulability test for mixed-criticality tasks. In our context, we further

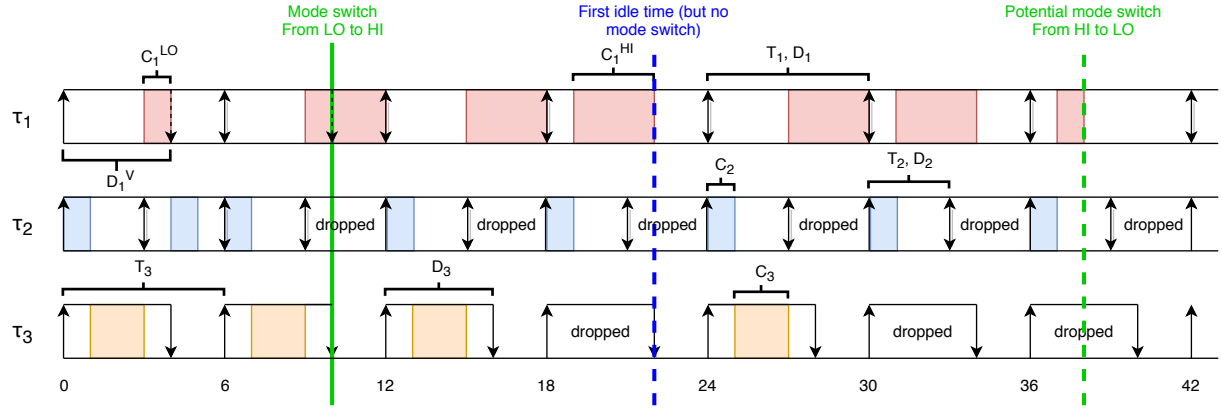


Fig. 2: EDF-GVD scheduling of the task set provided in Example IV.1.

breakdown the consideration of the DBF function of a task τ_i into the following four cases.

- $\text{dbf}^{\text{LO}}_{\text{LO}}(\tau_i, \ell)$ — the demand bound of a LO-criticality task τ_i in LO mode for any time interval of length ℓ.
- $\text{dbf}^{\text{LO}}_{\text{HI}}(\tau_i, \ell)$ — the demand bound of a HI-criticality task τ_i in LO mode for any time interval of length ℓ.
- $\text{dbf}^{\text{HI}}_{\text{LO}}(\tau_i, \ell)$ — the demand bound of a LO-criticality task τ_i in HI mode for any time interval of length ℓ.
- $\text{dbf}^{\text{HI}}_{\text{HI}}(\tau_i, \ell)$ — the demand bound of a HI-criticality task τ_i in HI mode for any time interval of length ℓ.

Carry-Over Job. A carry-over job is one that is released before the mode switch and has a deadline after the mode switch. It is clear that carry-over jobs do not affect the calculation of $\text{dbf}^{\text{LO}}_{\text{LO}}(\tau_i, \ell)$ and $\text{dbf}^{\text{LO}}_{\text{HI}}(\tau_i, \ell)$. Furthermore, according to EDF-GVD, carry-over jobs of LO-criticality tasks will be dropped, so they are not considered in the computation of $\text{dbf}^{\text{HI}}_{\text{LO}}(\tau_i, \ell)$ either. In the calculation of $\text{dbf}^{\text{HI}}_{\text{HI}}(\tau_i, \ell)$, the demand of a carry-over job can be considered as a special job that releases at the time of mode switch and has an execution time equal to the remaining execution requirement that has not been completed before the mode switch.

As a result, Lemma 1 can be extended to the following schedulability test in a straightforward way:

Theorem 1. *A mixed-criticality system τ can be successfully scheduled by EDF-GVD on a uniprocessor if both of the following conditions hold.*

$$\textit{(A):} \sum_{\tau_i \in \tau_{\text{LO}}} \text{dbf}^{\text{LO}}_{\text{LO}}(\tau_i, \ell) + \sum_{\tau_i \in \tau_{\text{HI}}} \text{dbf}^{\text{LO}}_{\text{HI}}(\tau_i, \ell) \leq \ell,\ \forall \ell \geq 0.$$

$$\textit{(B):} \sum_{\tau_i \in \tau_{\text{LO}}} \text{dbf}^{\text{HI}}_{\text{LO}}(\tau_i, \ell) + \sum_{\tau_i \in \tau_{\text{HI}}} \text{dbf}^{\text{HI}}_{\text{HI}}(\tau_i, \ell) \leq \ell,\ \forall \ell \geq 0.$$

To apply the above schedulability test, we only need to compute the DBF with respect to each task.

Compute $\text{dbf}^{\text{LO}}_{\text{LO}}(\tau_i, \ell)$ and $\text{dbf}^{\text{LO}}_{\text{HI}}(\tau_i, \ell)$. In LO mode, the DBF for both LO- and HI-criticality tasks is calculated as that for traditional non-mixed-criticality tasks in [9]. That is,

$$\forall \tau_i \in \tau_{\text{LO}},\ \text{dbf}^{\text{LO}}_{\text{LO}}(\tau_i, \ell) = \left(\left\lfloor \frac{\ell - D_i}{T_i} \right\rfloor + 1 \right) C_i,$$

$$\forall \tau_i \in \tau_{\text{HI}},\ \text{dbf}^{\text{LO}}_{\text{HI}}(\tau_i, \ell) = \left(\left\lfloor \frac{\ell - D_i^v}{T_i} \right\rfloor + 1 \right) C_i^{\text{LO}}.$$

Compute $\text{dbf}^{\text{HI}}_{\text{LO}}(\tau_i, \ell)$. Because the carry-over job of a LO-criticaliy task will be dropped upon the mode switch to HI mode, we only need to consider subsequent jobs that are released after the mode switch. Let $\text{Num}_i(x)$ denote the *maximum* number of jobs that are selected to execute, among *any* x consecutive jobs of a LO-criticality task τ_i in HI mode. The DBF of a LO-criticality task in HI mode can be computed as:

$$\forall \tau_i \in \tau_{\text{LO}},\ \text{dbf}^{\text{HI}}_{\text{LO}}(\tau_i, \ell) = C_i \cdot \text{Num}_i \left(\left\lfloor \frac{\ell - D_i}{T_i} \right\rfloor + 1 \right). \quad (4)$$

The following lemma shows how to derive $\text{Num}_i(x)$.

Lemma 2. *$\forall x \geq 0, \text{Num}_i(x) = \lceil r_i \cdot x \rceil$ under the admission control procedure described in Sec. III.*

Proof. Let $\text{num}(p, q)$ denote the number of jobs that are admitted to execute among the p^{th} to the q^{th} jobs of τ_i after the system switched to HI mode. By the admission control policy, the number of jobs of τ_i that are admitted to execute among the *first* x jobs is $\lceil r_i \cdot x \rceil$. That is, $\text{num}(1, x) = \lceil r_i \cdot x \rceil$. Next, we only need to show that $\forall y \geq 1, \text{num}(y+1, y+x) \leq \lceil r_i \cdot x \rceil$.

In order to show that, we suppose the contrary that $\exists y \geq 1$, $(y+1, y+x) \geq \lceil r_i \cdot x \rceil + 1$, and derive a contradiction. Let the $(y+z)^{\text{th}}$ job $(2 \leq z \leq x)$ be the $(\lceil r_i \cdot x \rceil + 1)^{\text{th}}$ admitted job among these x jobs, i.e., $\text{num}(y+1, y+z) = \lceil r_i \cdot x \rceil + 1$ and $\text{num}(y+1, y+z-1) = \lceil r_i \cdot x \rceil$. Moreover, by the admission control policy, $\text{num}(1, y) = \lceil r_i \cdot y \rceil$. Therefore,

$$\begin{aligned} \text{num}(1, y+z-1) &= \text{num}(1, y) + \text{num}(y+1, y+z-1) \\ &= \lceil r_i \cdot y \rceil + \lceil r_i \cdot x \rceil \\ &\geq r_i \cdot (y+x) \\ &\geq r_i \cdot (y+z). \end{aligned}$$

Thus, by the admission control policy, the $(y+z)^{\text{th}}$ job would not have been selected to execute, which contradicts the definition of z that the $(y+z)^{\text{th}}$ job is admitted to execute. □

Compute $\text{dbf}^{\text{HI}}_{\text{HI}}(\tau_i, \ell)$. The DBF of a HI-criticality task in HI mode can be computed in the similar way as in [15]. Let

full(τ_i, ℓ) denote the demand of τ_i during any time interval of length ℓ in HI mode, when counting C_i^{HI} for all jobs of τ_i including one potential carry-over job. Let done(τ_i, ℓ) denote the *minimum* amount of work of the carry-over job of τ_i that must have finished in LO mode. Then, the DBF of a HI-criticality task in HI mode can be computed by:

$$\forall \tau_i \in \tau_{\text{HI}},\ \text{dbf}_{\text{HI}}^{\text{HI}}(\tau_i, \ell) = \text{full}(\tau_i, \ell) - \text{done}(\tau_i, \ell). \quad (5)$$

According to [15], full(τ_i, ℓ) and done(τ_i, ℓ) can be computed as follows.

$$\text{full}(\tau_i, \ell) = \left(\left\lfloor \frac{\ell - (D_i - D_i^v)}{T_i} \right\rfloor + 1 \right) C_i^{\text{HI}}, \quad (6)$$

$$\text{done}(\tau_i, \ell) = \begin{cases} \max\{0, C_i^{\text{LO}} - \rho + D_i - D_i^v\} \\ \qquad \text{if } D_i > \rho \geq D_i - D_i^v, \\ 0, \quad \text{otherwise;} \end{cases} \quad (7)$$

where $\rho = \ell \mod T_i$.

Time Complexity of Schedulability Test. In Conditions (A) and (B), it is stated that "$\forall \ell$" needs to be assessed, which implies an unbounded number of dbf computations of different values of ℓ. We next show that, in general, only a limited number of ℓ values need to be considered, which yields a schedulability test with pseudo-polynomial time complexity.

Lemma 3. *Let c be a constant such that $c < 1$ and $\sum_{\tau_i \in \tau_{\text{LO}}}(C_i/T_i) + \sum_{\tau_i \in \tau_{\text{HI}}}(C_i^{\text{LO}}/T_i) \leq c$. Then, Condition (A) is true for $\forall \ell \geq 0$, if it is true for all ℓ such that*

$$\ell < \frac{c}{1-c} \cdot \max\{ \max_{\tau_i \in \tau_{\text{LO}}} \{T_i - D_i\}, \max_{\tau_i \in \tau_{\text{HI}}} \{T_i - D_i^v\} \}.$$

Proof. Because in LO mode, both HI- and LO-criticality tasks behave as conventional sporadic tasks, this lemma directly follows from [9]. □

Lemma 4. *Let c_1 and c_2 be two constants such that $\sum_{\tau_i \in \tau_{\text{LO}}}(r_i \cdot C_i/T_i) \leq c_1$, $\sum_{\tau_i \in \tau_{\text{HI}}}(C_i^{\text{HI}}/T_i) \leq c_2$, and $c_1 + c_2 < 1$. Then, Condition (B) is true for $\forall \ell \geq 0$, if it is true for all ℓ such that*

$$\ell < \frac{c_1 \cdot \max_{\tau_i \in \tau_{\text{LO}} \wedge r_i > 0} \{T_i - D_i + \frac{T_i}{r_i}\} + c_2 \cdot \max_{\tau_i \in \tau_{\text{HI}}} \{T_i - D_i + D_i^v\}}{1 - c_1 - c_2}.$$

Proof. Let ℓ_0 denote the length of a time interval where Condition (B) is not true. That is,

$$\sum_{\tau_i \in \tau_{\text{LO}}} \text{dbf}_{\text{LO}}^{\text{HI}}(\tau_i, \ell_0) + \sum_{\tau_i \in \tau_{\text{HI}}} \text{dbf}_{\text{HI}}^{\text{HI}}(\tau_i, \ell_0) > \ell_0. \quad (8)$$

By applying Eq. (4) and Lemma 2,

$$\begin{aligned} &\sum_{\tau_i \in \tau_{\text{LO}}} \text{dbf}_{\text{LO}}^{\text{HI}}(\tau_i, \ell_0) = \sum_{\tau_i \in \tau_{\text{LO}} \wedge r_i > 0} \text{dbf}_{\text{LO}}^{\text{HI}}(\tau_i, \ell_0) \\ &= \sum_{\tau_i \in \tau_{\text{LO}} \wedge r_i > 0} \left(C_i \cdot \left\lceil r_i \cdot \left(\left\lfloor \frac{\ell_0 - D_i}{T_i} \right\rfloor + 1 \right) \right\rceil \right) \\ &< \sum_{\tau_i \in \tau_{\text{LO}} \wedge r_i > 0} \left(C_i \cdot \left(r_i \cdot \left(\frac{\ell_0 - D_i}{T_i} + 1 \right) + 1 \right) \right) \\ &= \sum_{\tau_i \in \tau_{\text{LO}} \wedge r_i > 0} \left(r_i \cdot \frac{C_i}{T_i} \cdot \left(\ell_0 + T_i - D_i + \frac{T_i}{r_i} \right) \right) \\ &\leq \sum_{\tau_i \in \tau_{\text{LO}} \wedge r_i > 0} \left(r_i \cdot \frac{C_i}{T_i} \right) \cdot (\ell_0 + \max_{\tau_i \in \tau_{\text{LO}} \wedge r_i > 0} \{T_i - D_i + \frac{T_i}{r_i}\}) \\ &\leq c_1 \cdot (\ell_0 + \max_{\tau_i \in \tau_{\text{LO}} \wedge r_i > 0} \{T_i - D_i + \frac{T_i}{r_i}\}). \end{aligned} \quad (9)$$

By applying Eqs. (5), (6), and (7),

$$\begin{aligned} &\sum_{\tau_i \in \tau_{\text{HI}}} \text{dbf}_{\text{LO}}^{\text{HI}}(\tau_i, \ell_0) = \sum_{\tau_i \in \tau_{\text{HI}}} (\text{full}(\tau_i, \ell_0) - \text{done}(\tau_i, \ell_0)) \\ &\leq \sum_{\tau_i \in \tau_{\text{HI}}} \text{full}(\tau_i, \ell_0) \\ &= \sum_{\tau_i \in \tau_{\text{HI}}} \left(\left\lfloor \frac{\ell_0 - (D_i - D_i^v)}{T_i} \right\rfloor + 1 \right) C_i^{\text{HI}} \\ &\leq \sum_{\tau_i \in \tau_{\text{HI}}} \left(\frac{\ell_0 - (D_i - D_i^v)}{T_i} + 1 \right) C_i^{\text{HI}} \\ &= \sum_{\tau_i \in \tau_{\text{HI}}} \left(\frac{C_i^{\text{HI}}}{T_i} (\ell_0 + T_i - D_i + D_i^v) \right) \\ &\leq \sum_{\tau_i \in \tau_{\text{HI}}} \left(\frac{C_i^{\text{HI}}}{T_i} \right) \cdot (\ell_0 + \max_{\tau_i \in \tau_{\text{HI}}} \{T_i - D_i + D_i^v\}) \\ &\leq c_2 \cdot (\ell_0 + \max_{\tau_i \in \tau_{\text{HI}}} \{T_i - D_i + D_i^v\}). \end{aligned} \quad (10)$$

By applying Eqs. (8), (9), and (10),

$$\ell_0 < \frac{c_1 \cdot \max_{\tau_i \in \tau_{\text{LO}} \wedge r_i > 0} \{T_i - D_i + \frac{T_i}{r_i}\} + c_2 \cdot \max_{\tau_i \in \tau_{\text{HI}}} \{T_i - D_i + D_i^v\}}{1 - c_1 - c_2}.$$

Thus, the stated lemma follows. □

V. Mode Switch in Both Directions

A key difference between traditional Vestal model and the proposed MC model is the graceful degradation. Vestal model drops all LO-criticality tasks under HI mode since it originally stands from satisfy certification and system verification view (or, at least is interpreted as so). It is extremely unlikely that there will be a mode switch during run-time [2]. For this reason, the mode switch part from HI mode to LO mode is often skipped. However, in our model, the LO-WCET estimations are no longer real WCET. Instead, they are optimistic estimations and are designed to be violated from time to time. Thus, certain level of service to LO-criticality tasks are guaranteed, even when certain violations of estimations occur. As a result, we believe that *an MC scheduler with*

graceful degradation should take HI-*to*-LO *mode switch into consideration.*

As for the **mode switch mechanism**, we extend the existing work on MC scheduling and propose a more realistic backward mode switch behavior: Whenever some HI-criticality task overruns, the system will switch to HI mode immediately. The system is again switched back to LO-criticality mode when: (i) the processor *idles* in HI-criticality mode, and (ii) the last instance of each HI-criticality task is finished within its LO-WCET, where all further LO-criticality releases are accepted and all deadlines are met.

Remark 4. *The proposed scheme guarantees that mode switch will not occur very frequently under the assumption that consecutive jobs of any task should very likely experience either all* LO-*criticality or all* HI-*criticality behavior; i.e., either they all finish within C^{LO}, or all require more than that. When the assumption does not hold, we consider the system unsuitable for our model and recommend the system designers to re-evaluate the choice of C^{LO} values. Nevertheless, under such extreme scenarios where the disadvantage of our model is maximized, we still guarantee full correctness to all* HI-*criticality tasks under all modes; i.e., our model (and algorithm) extends the Vestal model (and EDF-VD for non-graceful degradation) with no loss.*

The idle time instant for a backward mode switch can be mathematically bounded, however, we cannot predict whether the last jobs of all HI-criticality tasks will be finished within their LO-WCET. In this section, we show that an idle time instant must appear after a bounded time duration in HI mode, which probably can trigger the system to switch back to LO mode if the last HI-criticality jobs are finished within their LO-WCET. In particular, we provide two bounds on time durations between: (1) the absolute deadline of the latest HI-criticality job that overruns its LO-WCET in this HI mode and the first idle time instant after that (L_1); (2) the latest instant that the system mode switched to HI and the first idle time instant after that (L_2). The first bound (L_1) applies to the scenarios where, after some HI-criticality jobs overrun, remaining jobs execute for at most their LO-WCETs. In contrast, the second bound (L_2) applies to any scenario, in particular when HI-criticality jobs frequently overrun their LO-WCETs. Overall, the first idle time instant must occur by the two bounds, whichever comes first. Intuitively, the first bound is proposed to provide a guaranteed earlier idle time instant when overrunning LO-WCET is a rare event. However, as mentioned before, these bounds provide one precondition of the backward mode switch while the other precondition (HI-jobs finishing within their LO-WCET) is fully dependent on the runtime behavior of the system.

In the following theorems, we suppose

$$\sum_{\tau_i\in\tau_{\mathrm{HI}}}\frac{C_i^{\mathrm{LO}}}{T_i}+\sum_{\tau_i\in\tau_{\mathrm{LO}}}\frac{r_i\cdot C_i}{T_i}\le\sum_{\tau_i\in\tau_{\mathrm{HI}}}\frac{C_i^{\mathrm{HI}}}{T_i}+\sum_{\tau_i\in\tau_{\mathrm{LO}}}\frac{r_i\cdot C_i}{T_i}<1,$$

which is, in fact, required for our schedulability test.

Theorem 2. *Let t_d denote the absolute deadline of the latest* HI-*criticality job that overruns its* LO-*WCET in this* HI *mode. Then, there must be an idle time instant at or before t_d+L_1, where*

$$L_1=\frac{\sum_{\tau_i\in\tau_{\mathrm{HI}}}(2C_i^{\mathrm{LO}})+\sum_{\tau_i\in\tau_{\mathrm{LO}}}(C_i+2r_i\cdot C_i)}{1-\sum_{\tau_i\in\tau_{\mathrm{HI}}}(C_i^{\mathrm{LO}}/T_i)-\sum_{\tau_i\in\tau_{\mathrm{LO}}}(r_i\cdot C_i/T_i)}.$$

Proof. Suppose that the theorem is not true, i.e., $[t_d,t_d+L_1)$ is a busy time interval. Then the total *work* done within $[t_d,t_d+L_1)$ is L_1. Note that the term *work* is different from the term *demand* earlier in this paper by that *work* includes all executions in this time interval regardless of the releases and deadlines of the jobs of these executions.

Letting W_{HI} and W_{LO} denote the total work by HI- and LO-criticality tasks in the time interval $[t_d,t_d+L_1)$, we have

$$W_{\mathrm{HI}}+W_{\mathrm{LO}}=L_1. \tag{11}$$

Because the tasks we are considering have constrained deadlines, the execution of each job must be within its own period. Therefore, for each task τ_i, at most $(\lfloor L_1/T_i\rfloor+2)$ jobs can potentially contribute to the work in the time interval $[t_s,t_s+L_1)$, where at most $\lfloor L_1/T_i\rfloor$ integral periods are in the middle and at most one partial period on each side. The HI-criticality jobs executed during the time interval $[t_d,t_d+L_1)$ do not execute for their HI-WCET. So,

$$\begin{aligned}W_{\mathrm{HI}}&\le\sum_{\tau_i\in\tau_{\mathrm{HI}}}\left(\left\lfloor\frac{L_1}{T_i}\right\rfloor+2\right)C_i^{\mathrm{LO}}\\&\le\sum_{\tau_i\in\tau_{\mathrm{HI}}}\left(\frac{L_1}{T_i}+2\right)C_i^{\mathrm{LO}}\\&=L_1\cdot\sum_{\tau_i\in\tau_{\mathrm{HI}}}\frac{C_i^{\mathrm{LO}}}{T_i}+\sum_{\tau_i\in\tau_{\mathrm{HI}}}(2C_i^{\mathrm{LO}}).\end{aligned} \tag{12}$$

For each LO-criticality tasks τ_i, by Lemma 2, at most $\lceil r_i\cdot(\lfloor L_2/T_i\rfloor+2)\rceil$ jobs of $(\lfloor L_2/T_i\rfloor+2)$ potential contributing jobs can be selected to execute. Therefore,

$$\begin{aligned}W_{\mathrm{LO}}&\le\sum_{\tau_i\in\tau_{\mathrm{LO}}}\left\lceil r_i\cdot\left(\left\lfloor\frac{L_1}{T_i}\right\rfloor+2\right)\right\rceil C_i\\&<\sum_{\tau_i\in\tau_{\mathrm{LO}}}\left(r_i\cdot\left(\left\lfloor\frac{L_1}{T_i}\right\rfloor+2\right)+1\right)C_i\\&\le\sum_{\tau_i\in\tau_{\mathrm{LO}}}\left(r_i\cdot\left(\frac{L_1}{T_i}+2\right)+1\right)C_i\\&=L_1\cdot\sum_{\tau_i\in\tau_{\mathrm{LO}}}\left(\frac{r_i\cdot C_i}{T_i}\right)+\sum_{\tau_i\in\tau_{\mathrm{LO}}}(C_i+2r_i\cdot C_i).\end{aligned} \tag{13}$$

By applying Expressions (11), (12), and (13),

$$L_1\cdot\sum_{\tau_i\in\tau_{\mathrm{HI}}}\frac{C_i^{\mathrm{LO}}}{T_i}+\sum_{\tau_i\in\tau_{\mathrm{HI}}}(2C_i^{\mathrm{LO}})+L_1\cdot\sum_{\tau_i\in\tau_{\mathrm{LO}}}\left(\frac{r_i\cdot C_i}{T_i}\right)+\sum_{\tau_i\in\tau_{\mathrm{LO}}}(C_i+2r_i\cdot C_i)>L_1.$$

That is,

$$L_1 < \frac{\sum_{\tau_i \in \tau_{\text{HI}}} (2C_i^{\text{LO}}) + \sum_{\tau_i \in \tau_{\text{LO}}} (C_i + 2r_i \cdot C_i)}{1 - \sum_{\tau_i \in \tau_{\text{HI}}} (C_i^{\text{LO}}/T_i) - \sum_{\tau_i \in \tau_{\text{LO}}} (r_i \cdot C_i/T_i)},$$

which contradicts the definition of L_1 in the theorem statement. Thus, this theorem follows. □

VI. Experimental Study

In this section, we present the experimental evaluation of the performance of EDF-GVD compared to the existing state-of-the-art algorithms. The baseline algorithms include: EDF-VD [3] (no guarantee to LO tasks), EDF-VD with (m, k) guarantee (the same guarantee to LO, under pessimistic density based analysis) [3], and EDF-VD with shrunk C^{LO} [25] (provides weaker guarantees to LO tasks). We compare with the EDF-VD family due to its similarity, simplicity, and speedup-optimality for uniprocessor MC task scheduling.

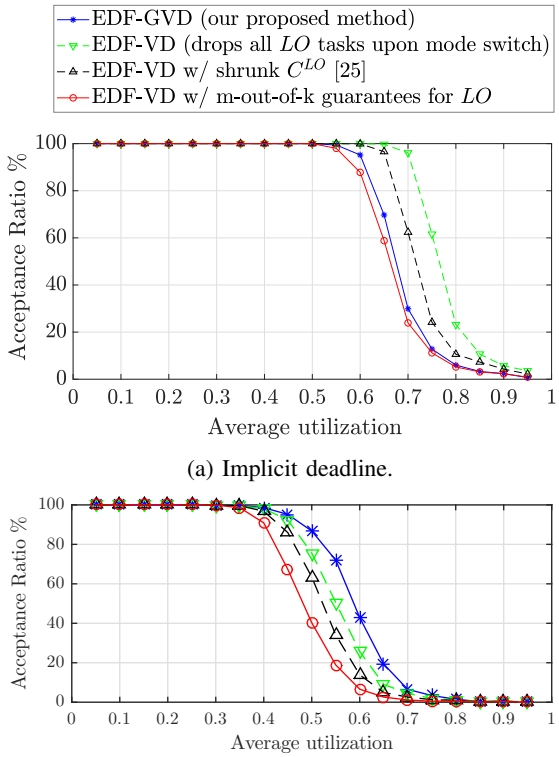
Workload generation. MC workload is generated with similar approach to [15], based on the following parameters (here U stands for uniform distribution):

- $P_{\text{HI}} = \{0.05, 0.25, 0.5, 0, 75, 0.95\}$: An individual task's probability of being HI-criticality. The criticality level χ_i is decided based on this parameter.
- $C_i^{\text{LO}} \sim U[1, 10]$: The values of C_i^{LO} (and C_i) are uniformly generated from this range.
- $R_{\text{HI}} = \{1, 2, 4, 8, 16\}$: It denotes the upper bound of the ratio of C_i^{HI} to C_i^{LO}; if $\chi_i = \text{HI}$, then the value of C_i^{HI} is uniformly generated from the range $[C_i^{\text{LO}}, R_{\text{HI}} \cdot C_i^{\text{LO}}]$.
- $T_{max} = 200$: The value of period T_i is uniformly generated from the range $[C_i(\chi_i), T_{max}]$.
- $minDR \sim U[0.1, 0.9]$: This value sets the lower bound of the ratio between deadlines and periods. After generating $minDR$ uniformly from the given range, the deadline is calculated by $D_i = \alpha \cdot T_i$, where α is uniformly generated from the range $[minDR, 1]$.

To generate the tasks of each task set τ, we use a predetermined value of *target average utilization* U^*. After generating each task we calculate the *average utilization* $U_{\text{AVG}}(\tau)$ for that specific task set by taking the average of U_{LO} and U_{HI}. Our goal is to get an average utilization near the value of U^*. For this purpose, we find an average utilization with the value in the range $[U^*_{\text{MIN}}, U^*_{\text{MAX}}]$, where $U^*_{\text{MIN}} = U^* - 0.005$ and $U^*_{\text{MAX}} = U^* + 0.005$.

We keep on adding the generated tasks to τ as long as $U_{\text{AVG}}(\tau) < U^*_{\text{MIN}}$. If at any point we get a $U_{\text{AVG}}(\tau)$ that is greater than U^*_{MAX}, then we immediately discard the whole task set and continue to make attempts. If $U_{\text{AVG}}(\tau)$ is within the range, then we have a candidate task set – it can still be discarded if all the tasks of τ have the same criticality level, $U_{\text{LO}}(\tau) > 0.99$, or $U_{\text{HI}}(\tau) > 0.99$.

As all other three algorithms target implicit-deadline MC task models, hence for those algorithms, we use C_i/D_i as the utilization of task τ_i instead of C_i/T_i. Also, for EDF-VD with shrunk C_i^{LO} , as the requirement in HI mode for LO-criticality task is $C_i^{\text{LO}} \geq C_i^{\text{HI}}$, we set $C_i^{\text{HI}} = r_i \cdot C_i^{\text{LO}}$. The results of our experiments are presented in Fig. 3.



(a) Implicit deadline.

(b) Constrained deadline with $minDR = 0.5$.

Fig. 3: Task set acceptance ratio under varying average utilization (U_{AVG}) settings ($P_{\text{HI}} = 0.5$, $R_{\text{HI}} = 4$).

TABLE III: Breakdown of virtual deadline selections, mode switch upper bounds, and task set sizes (AR refers to Acceptance Ratio).

Utilization	0.4	0.5	0.6	0.7	0.8	0.9
AR (simple D^v) %	91.6	65.7	19.3	2.7	0.3	0.1
AR (full) %	99.3	86.2	44.9	7.1	0.7	0.3
Avg. L_1/T_{max}	0.45	0.61	0.78	0.77	0.69	0.39
Avg. size of task set	6.00	6.99	7.93	9.12	10.74	12.01

With increasing *Average Utilization* from 0.05 to 0.95 at a step size of 0.05, we generate 1000 tasks for each case. The *acceptance ratios* (i.e., fraction of schedulable task sets) of each algorithm, for each case are reported in Fig. 3, where Fig. 3(a) is for implicit-deadline task sets and Fig. 3(b) is for constrained deadlines task sets.

Observation. For constrained-deadline tasks, Algorithm EDF-GVD clearly outperforms the other algorithms, as they are designed for implicit-deadline task scheduling. As overall utilization is above 0.5, the proposed algorithm EDF-GVD delivers better acceptance ratio than existing algorithms. Note that another reason for existing method to perform poorly is that they are not designed for graceful degradation: in order existing unitization based schedulability test to be correct for the new model, we have to replace periods with deadlines in order to adapt, which involves additional pessimism.

For implicit-deadline task sets, EDF-GVD still performs better than EDF-VD with the same guarantees. Note that the

outperforming two lines (black and green) make *weaker/no guarantee* to LO tasks upon mode switch and thus their acceptance ratio can be higher than EDF-GVD. The plots are a measurement of how much schedulability is needed to sacrifice in order to achieve higher guarantees for LO tasks.

We present the virtual deadline setting mechanism in Section III. The first two rows of Table III report the percentage of accepted task sets under each utilization setting – the first row using the easy virtual deadline setting following Equation (2) and the second row following the time-consuming binary search given in Algorithm 2. It is clear that the simple heuristic can only identify a small portion of all EDF-GVD schedulable tasks, especially when utilization is high.

Section IV-C presents the bound for system to switch back to LO mode (when idle occurs), which is reported in the middle row of Table III. The last row of Table III demonstrates the average size of (number of tasks in) a task set.

VII. Related Works

Since Vestal's first proposal [32] of mixed-criticality task scheduling problem, much work has been conduced to schedule MC tasks efficiently in both uniprocessor and multiprocessor environments [13] [3] [30] [7] [8] [19] [6]. While these algorithms provide guarantees for the completion of HI-criticality tasks quite efficiently, they provide no service guarantee to LO-criticality tasks upon a mode switch.

To overcome the above shortcomings and to provide a minimum level of service to LO-criticality tasks several different approaches have been proposed over time. Baruah et al. [27] demonstrated alternative models which allow LO tasks to continue to execute even after a criticality mode change. They consider assigning a lower priority level to the LO-criticality tasks. However, this model does not provide a minimum level of guarantee (i.e., all LO-criticality tasks can still miss the deadlines). Santy et al. [29] proposed a different approach where LO-criticality jobs get some service until the HI-criticality jobs are guaranteed to have enough execution time. Static analysis does not apply since the available slack time cannot be calculated beforehand. Elastic task model (also known as task stretching) has been used in several works [31] [23], where LO-criticality jobs receive degraded service with dynamically enlarging periods and deadlines.

In [16], Fleming and Burns introduced the notion of 'importance', where they focused on maintaining the operation of LO-criticality tasks under the HI-criticality mode by deciding which tasks are suspended first.

While the aforementioned works focus on best-effort algorithms, efforts are also conducted to provide some guarantee for LO-criticality tasks under HI mode. Baruah, Burns, and Guo [5] proposed a speedup-optimal scheduling algorithm using the fluid-based technique where they provided a generalization to Vestal model. A degraded (but non-zero) level of service is guaranteed under all non-erroneous behavior of the system. Liu et al. [25] proposed an utilization-based schedulability test under EDF-VD scheduling to guarantee some service to LO-criticality tasks in HI-critical mode. In [20], Guo et al. proposed a different approach where they incorporate failure probability information into the mixed criticality task model and derived the corresponding EDF schedulability analysis.

The concept of graceful degradation can be traced back to the 1990's. Hamdaoui and Ramanathan [21] propose (m, k)-firm model for streams where at least m deadlines out of consecutive k must be met. However, no scheduling guarantee is provided. Later, Bernat et al. [10] introduced weakly-hard real-time systems model which can tolerate a pre-defined degree of missed deadlines with four constraints. Similarly, Saha et al. [28] considered the scheduling of control systems task with an expected asymptotic success rate. Getting et al. [17] proposed an MC scheduling technique with graceful degradation by incorporating weakly-hard constraints into Adaptive Mixed Criticality (AMC) [4], where they considered skipping s amount of jobs to ensure $(m - s)$ out of m deadlines. In [14], Chwa et al. explored the trade-off between enlarging periods and consecutive task drops to guarantee schedulability while maintaining stability in a cyber-physical system. A new cyber-physical system task model is proposed, but the mixed-critical tasks are not considered.

A dynamic approach is proposed by Gu and Easwaran [18] for calculating a minimum guaranteed budget allocation for LO-criticality tasks under HI-critical mode, by delaying the mode-switch. In this model, LO-criticality budgets for individual HI-criticality applications are determined at runtime instead of the one with static analysis.

Note that part of our proposed system model (under HI mode) follows from the weakly-hard degradation model in [10] (for non-MC systems), which dominates other graceful degradation definitions such as the *upper-bound-rate-based* and the *m-out-of-any-k* one.

VIII. Conclusion and future work

In this paper, we considered the problem of providing service guarantee to LO-criticality tasks under HI mode for uniprocessor MC scheduling (with two levels). Different from prior work where such guarantee is interpreted as partially executing every LO-criticality job, our work aims at fully executing a subset of LO-criticality jobs. In this context, we developed an admission control procedure, a virtual-deadline based scheduling algorithm along with a DBF-based schedulability test, and an analysis about switching back to LO mode (from HI mode). Experimental results demonstrated that our proposed techniques are able to effectively address the problem.

This work proposed two approaches in the homothetic computation of virtual deadlines. A better schedulability ratio may be possible if such constraint is removed, at the cost of significantly increasing the complexity and search space, which is left as future work. The extension to include more than two criticality levels may not be trivial: virtual deadline settings for multiple criticality levels can be more challenging, while the carry-over workload calculation can be quite complicated under back-and-forth mode switching.

ACKNOWLEDGEMENT

The authors are grateful to the input from Prof. Sanjoy Baruah at Washington University at St Louis.

This work is partially supported by NSF grants (CNS-1837472, CNS-1850851, CNS-1545050, and CCF-1725755).

REFERENCES

[1] S. Baruah. A scheduling model inspired by control theory. In *Proceedings of the 6th International Real-Time Scheduling Open Problems Seminar*, 2015.

[2] S. Baruah. Mixed-criticality scheduling theory: Scope, promise, and limitations. *IEEE Design and Test*, 35(2):31–37, 2018.

[3] 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 *Proceedings of the 24th Euromicro Conference on Real-Time Systems*, 2012.

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

[5] S. Baruah, A. Burns, and Z. Guo. Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors. In *Proceedings of the 28th Euromicro Conference on Real-Time Systems*, 2016.

[6] S. Baruah, A. Easwaran, and Z. Guo. MC-Fluid: simplified and optimally quantified. In *Proceedings of the 36th IEEE Real-Time Systems Symposium*, 2015.

[7] S. Baruah and Z. Guo. Mixed-criticality scheduling upon varying-speed processors. In *Proceedings of the 34th IEEE Real-Time Systems Symposium (RTSS'13)*, 2013.

[8] 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.

[9] S. Baruah, A. Mok, and L. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the 11th IEEE Real-Time Systems Symposium*, 1990.

[10] G. Bernat, A. Burns, and A. Liamosi. Weakly hard real-time systems. *IEEE Transactions on Computers*, 50(4):308–321, 2001.

[11] M. Branicky, S. Phillips, and W. Zhang. Scheduling and feedback co-design for networked control systems. In *Proceedings of the 41st IEEE Conference on Decision and Control*, 2002.

[12] A. Burns. How to gracefully degrade. Keynote given at Dagstuhl Seminar 17131, 2017.

[13] A. Burns and R. Davis. Mixed-criticality systems: A review. Avaliable at http://www-users.cs.york.ac.uk/burns/review.pdf, 2017.

[14] H. S. Chwa, K. G. Shin, and J. Lee. Closing the gap between stability and schedulability: a new task model for Cyber-Physical Systems. In *Proceedings of the 24th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'18)*, 2018.

[15] P. Ekberg and W. Yi. Bounding and shaping the demand of generalized mixed-criticality sporadic task systems. *Real-Time Systems*, 50:48–86, 2014.

[16] T. Fleming and A. Burns. Incorporating the notion of importance into mixed criticality systems. In *Proceedings of the 2nd Workshop on Mixed Criticality Systems*, 2014.

[17] O. Gettings, S. Quinton, and R. I. Davis. Mixed criticality systems with weakly-hard constraints. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*, 2015.

[18] X. Gu and A. Easwaran. Dynamic budget management with service guarantees for mixed-criticality systems. In *Proceedings of the 37th IEEE Real-Time Systems Symposium*, 2016.

[19] 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.

[20] 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*, 2015.

[21] M. Hamdaoui and P. Ramanathan. A dynamic priority assignment technique for streams with (m, k)-firm deadlines. *IEEE Transactions on Computers*, 44(12):1443–1451, 1995.

[22] J. A. Hassan, M. Hassan, S. K. Das, and A. Ramer. Managing Quality of Experience for wireless VOIP using noncooperative games. *IEEE Journal on Selected Areas in Communications*, 30(7):1193–1204, 2012.

[23] M. Jan, L. Zaourar, and M. Pitel. Maximizing the execution rate of low-criticality tasks in mixed criticality systems. In *Proceedings of the 34th IEEE Real-Time Systems Symposium*, 2013.

[24] H. Lin, M. Chatterjee, S. K. Das, and K. Basu. ARC: An integrated admission and rate control framework for competitive wireless CDMA data networks using noncooperative games. *IEEE Transactions on Mobile Computing*, 4(3):243–258, 2005.

[25] D. Liu, J. Spasic, N. Guan, G. Chen, S. Liu, T. Stefanov, and W. Yi. EDF-VD scheduling of mixed-criticality systems with degraded quality guarantees. In *Proceedings of the 37th IEEE Real-Time Systems Symposium*, 2016.

[26] R. Majumdar, I. Saha, and M. Zamani. Performance-aware scheduler synthesis for control systems. In *Proceedings of the 9th ACM International Conference on Embedded Software*, 2011.

[27] S. Baruah and A. Burns. Towards a more practical model for mixed criticality systems. In *Proceedings of the Workshop on Mixed-Criticality Systems*, 2014.

[28] I. Saha, S. Baruah, and R. Majumdar. Dynamic Scheduling for Networked Control Systems. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, 2015.

[29] F. F. Santy, L. George, P. Thierry, and J. J. Goossens. Relaxing mixed-criticality scheduling strictness for task sets scheduled with FP. In *Proceedings of the 24th Euromicro Conference on Real-Time Systems*, 2012.

[30] D. Socci, P. Poplavko, S. Bensalem, and M. Bozga. Mixed critical earliest deadline first. In *Proceedings of the 25th Euromicro Conference on Real-Time Systems*, 2013.

[31] H. Su and D. Zhu. An elastic mixed-criticality task model and its scheduling algorithm. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition*, 2013.

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