Precise Scheduling of Mixed-Criticality Tasks by Varying Processor Speed

Ashikahmed Bhuiyan*
University of Central Florida

Zhishan Guo
University of Central Florida

Sai Sruti
Missouri University of Science & Technology

Kecheng Yang
Texas State University

ABSTRACT

In this paper, we extend the imprecise mixed-criticality (IMC) model to precise scheduling of tasks. We also integrate the IMC model with the dynamic voltage and frequency scaling (DVFS) technique to enable energy minimization. The challenge in precise scheduling of MC systems is to guarantee the timing correctness all tasks under both pessimistic and optimistic assumptions simultaneously. To our knowledge, this is the first work to address the integration of DVFS energy-conserving techniques with precise scheduling of all tasks of the MC model. We present utilization based schedulability tests and sufficient conditions for such systems under two well-known MC frameworks, EDF-VD and MCF. A quantitative study in the forms of speedup bound and approximation ratio are derived for the unified model. Empirical studies based on randomly generated sets are conducted to verify the theoretical results as well as the effectiveness of the proposed algorithms.

CCS CONCEPTS

• **Computer systems organization** → *Real-time system architecture.*

KEYWORDS

precise scheduling, mixed-criticality, varying-speed platform, speedup bound, approximation ratio.

ACM Reference Format:
Ashikahmed Bhuiyan*, Zhishan Guo, Sai Sruti, and Kecheng Yang. 2019. Precise Scheduling of Mixed-Criticality Tasks by Varying Processor Speed. In *27th International Conference on Real-Time Networks and Systems (RTNS 2019), November 6–8, 2019, Toulouse, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3356401.3356410

1 INTRODUCTION

There has been an exponential rise in the study of essential security systems with mixed-criticality implementation. Here, components assigned different levels of criticality are facilitated onto a common framework to enable minimization of energy and reduction of resource costs. The real-time systems community has extensively followed the model for representing such mixed-criticality workloads proposed by Vestal [33] over a decade ago. There have been multiple extensions of this model analyzing the aspects of scheduling and schedulability conditions under various platforms. These scheduling strategies are centered around: (i) guaranteeing resources to all the tasks under less pessimistic behaviors of the system and (ii) protecting more important (HI-criticality) tasks under more pessimistic behaviors, e.g., in the event of task overrun.

State-of-the art. Most of the current and existing literature focuses on the real-time facet of MC systems, adopts a popular workload model to specify these systems. The system starts in the normal mode where all the tasks are guaranteed execution w.r.t their less pessimistic worst-case execution times (WCET). However, in case of a task overrun, i.e., if a task of great importance exceeds its normal budget of WCET without signaling completion, the system switches to HI-criticality mode. In that case (i.e., task overrun), the more critical tasks are guaranteed execution while no guarantees are made to the less critical ones [3, 8]. Sometimes in HI-criticality mode, degraded services are provided to the less critical tasks, and the released resources are used to guarantee to meet HI-criticality task deadlines [11, 22]. More recently, the imprecise mixed-criticality system (IMC) model is being studied, which allows graceful degradation of LO-criticality tasks in HI-criticality mode [4, 11, 28]. IMC model embraces the concept of imprecise computing, i.e., upon mode-switch, each individual lo-criticality task can execute with inaccuracy in computing. Such allowance results in relatively short worst-case execution time (WCET), thus saving resources for the more critical tasks.

Varying-speed processor: context and motivation. To date, a significant amount of research [7, 8, 19, 23, 29] in mixed criticality systems has focused on the changing speeds of platforms on which MC systems are executed. For example, unpredictability and varying the speed of Commercial-Off-The-Shelf (COTS) processors during runtime [8] or intentionally varying the frequency of the processor to minimize energy. Mixed-criticality systems typically run on the battery-operated platforms with an increasing demand for drastically exaggerated computing. Hence, energy reduction for such systems is turning crucial. The dynamic voltage and frequency scaling (DVFS) feature have made this target (energy reduction) feasible. Several modern processors are equipped with the DVFS capability, where processor frequency is decreased at runtime to save energy. Huang et al. proposed the integration of dynamic voltage

*Authors are alphabetically ordered by last names. Contact author Zhishan Guo: zsguo@ucf.edu.

RTNS 2019, November 6–8, 2019, Toulouse, France

ACM ISBN 978-1-4503-7223-7/19/11…$15.00
https://doi.org/10.1145/3356401.3356410

and frequency scaling (DVFS) technique with the earliest deadline first with virtual deadlines (EDF-VD) scheduling scheme for dual-criticality systems [23] to enable energy minimization. Huang et al. established that increased speeds during overrun conditions are beneficial to minimize expected energy consumption of the system. This model is extended to accommodate multi-core processors, in which a trade-off is determined between both static and dynamic energy consumption in different operation modes [29].

All of the works mentioned above consider a stringent model in which all the LO-criticality tasks are dropped upon mode switch. Such Handling of the LO-criticality tasks is argumentative as in HI-criticality mode, all LO-criticality tasks are penalized and can result in failures in timing assumptions in HI-criticality tasks [12, 15]. Burns et al. [11] and Su et al. [32] exploits the elastic task model where LO-criticality tasks continue to execute with extended time-periods. This model generates accurate but delayed execution results. Imprecise computing was introduced in mixed-criticality systems in [4, 11, 28] to balance the safety and performance features. In this model, each LO-criticality task is also guaranteed to some (degraded) service after the system switches to the HI-criticality behavior. However,Ernst et al. [15] argues that the period and priority of a task are functional requirements and cannot be altered easily. Also, degrading services for the execution of LO-criticality tasks can result in performance or service loss. Pathan et al. [30] observed that in case of utilization slack during the execution of HI-criticality tasks in HI-criticality mode, all the LO-criticality tasks need not be penalized with degraded service. Considering the implicit-deadline IMC sporadic task, a scheduling technique was proposed by [30]. In this technique, some (if not all) of the LO-criticality tasks were provided with full service during the HI-criticality behavior as well.

This research. As discussed above, significant work on MC scheduling has considered some level of precision in scheduling LO-criticality tasks in a pessimistic condition. They also have considered speed scaling to minimize energy during run-time. Our work addresses the need to save energy in platforms supporting mixed-criticality applications. The idea behind this research is to entertain both accuracy and energy efficiency in real-time system applications. Numerous advanced processors support the dynamic voltage and frequency scaling (DVFS), where processor frequency can be diminished at run-time to conserve energy as demonstrated by the energy model in [23]. We adopt an off-line DVFS scheme to diminish energy utilization in the normal mode by choosing a minimum speed (≤ 1) for the processor while protecting mixed-criticality schedulability of the framework. In this paper, we aim to integrate the precision model in [30] and the varying-speed model in [23], such that precise computing to *all* tasks can be guaranteed in any mode. We also ensure that the processor can run in a (close to) optimal energy-conserving speed in normal mode. The unified model is used to schedule implicit-deadline sporadic tasks by the well-known EDF-VD [9] and MCF [6] scheduling policies. The main contributions of this paper are:

- This paper explores the aggregation of mixed-criticality scheduling with design options of recent computing platforms, i.e., combining precise computing of LO-criticality tasks on varying-speed processors.
- We present conditions to derive the minimum speed for the processor to execute in normal mode, while correctly scheduling all the tasks in each mode of operation.
- We propose a sufficient test for our precise-energy conserving model under EDF-VD (see Theorem 3.3) and prove a quantitative speedup bound and approximation ratio on the worst-case performance of EDF-VD.
- We further adopted a metric named approximation ratio, which compares the minimum possible degraded processor speed without speeding up upon the mode switch. We also proved the relationship between the approximation ratio of our algorithm and the per-level utilization of the input task.
- Based on MCF, a fluid scheduling framework for MC tasks, we proposed another approach for determining the lowest possible speed under normal mode, as well as the scheduling strategy. We prove its correctness and derived the utilization based approximation ratio for this approach.
- Experimental studies are conducted based on randomly generated synthetic tasks, which supports the theoretical findings as well as the effectiveness of the proposed algorithms.

Organization. The rest of the paper is organized as follows: In Section 2, the adopted model is elaborated in detail comprising of system behavior, varying-speed processors, and correctness specification. Section 3 consists of the literature of the EDF-VD scheduling algorithm and the revised schedulability conditions about the chosen model; we also determine the approximation ratio of the proposed approach. Section 4 describes the fluid-based scheduler, shows its correctness, and derives the approximation ratio of the approach. Section 5 reports our experimental results, while Sections 6 and 7 consist of the literature of previous works and the conclusion.

2 MODEL AND PROBLEM

The considered MC workload comprises of an implicit-deadline [1] sporadic task model where each task-set τ includes n tasks that are scheduled on a preemptive uniprocessor. Every task $\tau_i \in \tau$ may generate an unbounded number of MC jobs, with successive jobs being released at least T_i time units apart. Without loss of generality, we assume that all tasks in τ start at time 0.

MC instance. In this paper, we restrict our attention to dual-criticality task systems where the system has two criticality levels, $\chi_i \in \{\text{LO}, \text{HI}\}$. Each task $\tau_i \in \tau$ is characterized by 4-tuples = $\{T_i, \chi_i, C_i^L, C_i^H\}$, where T_i represents the minimum inter-arrival time between any two consecutive job releases (by the same task), $C_i^L, C_i^H \in \mathbb{R}_+$ are the WCET estimations, and $\chi_i \in \{\text{LO}, \text{HI}\}$ represents the criticality level of the task. Due to pessimistic impositions on assurance for HI-criticality tasks, for our model, we assume that $0 < C_i^L < C_i^H \leq T_i$ hold for HI tasks (τ_{HI}) and $0 < C_i^L = C_i^H \leq T_i$ hold for LO tasks (τ_{LO}).

Although the work by Esper et al. [16] and Ernst et al. [15] criticized the MC model regarding its applicability, we would like to point out that no criticism has been made to the original Vestal model, where different certification requirements lead to various WCET estimations. For run-time robustness, this work takes a

[1] In an implicit deadline task, task deadlines are equal to its inter-arrival period. Hence, these two terms are used interchangeably.

relatively *safe* assumption, such that no job is ever dropped and correctness's are guaranteed to all jobs under all circumstances (see problem definition towards the end of this section).

The per-mode utilization of each task τ_i are determined as follows:

$$\forall \tau_i, \quad u_i^L = \frac{C_i^L}{T_i};$$

$$\forall \tau_i, \quad u_i^H = \frac{C_i^H}{T_i}.$$

The total utilization for each mode of operation is represented as follows:

- Since we do not degrade services for LO-criticality tasks in HI-criticality mode, their utilization in both modes of operation remains the same, i.e.,

$$U_{\text{LO}}^L = U_{\text{LO}}^H = \sum_{\tau_i \in \tau_{\text{LO}}} u_i^L = \sum_{\tau_i \in \tau_{\text{LO}}} u_i^H.$$

- The total utilization for all HI-criticality tasks in LO- and HI-criticality modes can be represented as:

$$U_{\text{HI}}^L = \sum_{\tau_i \in \tau_{\text{HI}}} u_i^L \quad \text{and} \quad U_{\text{HI}}^H = \sum_{\tau_i \in \tau_{\text{HI}}} u_i^H.$$

Varying-speed processor. State of the art processors is manufactured with several advanced features, such as the DVFS scheme, where processor frequency can be diminished at run-time to conserve energy [23]. We examine off-line DVFS to reduce energy utilization in the LO-criticality mode by choosing a minimum speed for the processor while protecting mixed-criticality schedulability of the framework. The processor is characterized by a normal speed s (without loss of generality, $s \leftarrow 1$) and an energy-conserving speed ρ ($\rho \leq 1$). During LO-criticality mode, the processor is assumed to exhibit *energy conserving behavior* where its speed remains ρ. In the event of overrun, when the system switches to HI-criticality mode, the processor exhibits *normal behavior* where the speed of the processor is maximized ($\leftarrow 1$). Note that task overrun should be a rare event, hence the system mode switch (LO- to HI-criticality). So, the system is expected to run at the energy-conserving speed for most of the time. As a result, processor frequency-changing overhead is not considered in this paper.

System behavior. The behavioral semantics of the MC workload is as follows: a job released by task T_i may first execute for its LO-criticality WCET (C_i^L units of time). If all jobs indicate completion at after executing for their LO-criticality WCETs, the system is claimed to perform in LO-criticality mode. Else a system-wide mode switch is triggered, and the system exhibits HI-criticality behavior. Analogous to the traditional MC task-model behavior, instead of discarding all LO-criticality tasks in HI-criticality mode, our model *guarantees execution time to all tasks* according to their given WCETs during *both* modes of operation. We incorporate the DVFS technique to impose a minimum energy-conserving speed in the LO-criticality mode. Upon mode-switch, the processor immediately performs at normal speed (of 1). Since we do not know statically how long the system will overrun, we can safely assume that the system can recover when the processor is idle, i.e., when all arrived/active workloads are finished.

Assumption. The relationship between the speed and worst-case execution time (WCET) of the task is considered to be linear, e.g., reducing the execution speed by half will lead to doubling the execution time. Hence, a task with LO-criticality WCET of C_i (on a speed-1 processor) would take C_i/ρ time units to finish its execution on a processor of degraded speed ρ, while the same task will finish its HI-criticality WCET (i.e., C_i) after a system mode-switch at $C_i/1$ time units, on a processor with maximum speed 1.

In practice, this assumption should always hold since the speed of cache and memory access, I/O bus, etc. are not extensively affected by a change in the processor speed. We believe it is *safe* in terms of schedulability to consider a linear relationship, as illustrated in Figure 1. The actual C_i is a considerably accurate portrayal of the relationship between processor speed and execution time. We are considering a *pessimistic* upper bound on C_i (expected C_i from Figure 1) by assuming a linear relationship.

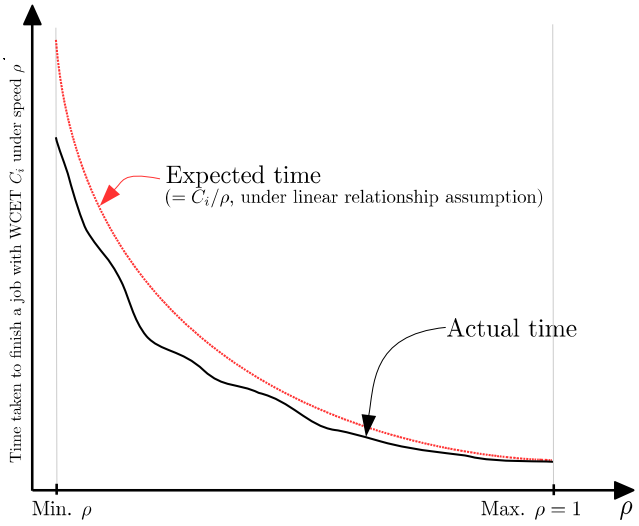

Figure 1: Relationship between variable-speed and execution time, where the expected WCET always caps the actual execution time. Where ρ represents the energy conserving speed.

Problem. It is widely accepted that it is rare for any task to exhibit HI-criticality behavior; i.e., not signaling completion after executing up-to the LO-criticality WCET. As a result, more attention should be placed on LO-criticality mode in terms of energy consumption. In this paper, *we seek to reduce energy utilization in the LO-criticality mode by minimizing the energy-conserving speed ρ for the processor, while protecting mixed-criticality correctness of the system.* We present a model that integrates precise scheduling of LO-criticality tasks and energy-minimization using DVFS techniques. The *correctness* of the system is mode based, defined as follows:

- During all LO-criticality behaviors of the system, the processor is down-scaled by the energy-conserving speed ρ. All jobs receive up to their LO-criticality WCET and meet their deadlines.

- In HI-criticality mode, where one or more HI-criticality tasks did not signal their completion upon receiving cumulative execution

budget of their LO-criticality WCET, the processor speed increases to 1, *where all jobs (both LO and HI) receive computation time up to their HI-criticality WCET* and still meet their deadlines.

3 EDF-VD AND ITS CORRECTNESS

In this section, we first present how the EDF-VD algorithm for traditional MC system model in [9] is modified to handle the focused case (where LO-criticality tasks can no longer be dropped). The proof of its correctness in both execution modes is included thereafter.

Traditional EDF-VD scheduling: Baruah et. al [9] proposed an adaptation to the Earliest Deadline First (EDF) algorithm to schedule dual-criticality implicit-deadline sporadic task systems on a unit-speed processor. In the situation of task overrun, resources have to be retained for the HI-criticality tasks, to guarantee that they can still meet their deadlines. In the EDF-VD algorithm, the reservation of resources for HI-criticality tasks is accomplished in the LO-criticality mode by artificially scaling down the deadlines of HI-criticality tasks. Such scaled virtual deadline settings enables HI-criticality tasks to budget enough resources to handle overrun while finishing their LO-criticality WCET within the LO-criticality mode. In order to address the resource demands on different levels of criticality:

- In LO-criticality mode, the EDF-VD algorithm adjusts the deadlines of all the HI-criticality tasks by a common factor x. This is done to retain budget for the HI-criticality tasks in HI-criticality mode by triggering an earlier mode-switch.
- In HI-criticality mode, the HI-criticality tasks are scheduled according to their original deadlines (using EDF) and all LO-criticality tasks are discarded (or executed with best effort in the background).

3.1 EDF-VD for Precise Energy-Conserving Model

In this section, we describe in detail the enhanced EDF-VD algorithm to compromise our precise energy-conserving model. As discussed earlier, we consider the DVFS technique to conserve energy in LO-criticality mode by slowing down the processor to speed ρ. Figure 2 represents a *modified* EDF-VD algorithm which determines if the task-set τ is schedulable. Then, the modified EDF-VD assigns virtual deadline $\widehat{T}_i$ for all HI-criticality tasks of the schedulable task-set. Note that, we describe how to derive the minimum ρ with any given MC set later in Theorem 3.4.

According to the algorithm in Figure 2, the scaling factor x is computed and $\widehat{T}_i$ values are assigned to all HI-criticality tasks as $\widehat{T}_i \leftarrow xT_i$. Note that $U^L_{\text{LO}} + U^L_{\text{HI}} \leq \rho$ is a necessary condition for schedulability under LO-criticality mode, $x \leq 1$ always holds for the proposed assignment. Theorem 3.1 demonstrates how parameter x is derived. Contradictory to the traditional MC model, instead of discarding all LO-criticality tasks in HI-criticality behaviors, our model schedules both LO- and HI-criticality tasks with their given WCETs at processor speed $s \leftarrow 1$. Note that, we did not consider any additional overhead (that may be introduced) for changing the speed from ρ to the maximum speed (i.e., $s \leftarrow 1$). This assumption does not have a profound effect for two reasons. First, mode-switching is a rare event, and hence the changing of the speed. Second, we change the speed only once per mode switch.

For a dual-criticality task-set $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$ to be scheduled by energy conserving preemptive processor with normal speed ρ and max speed 1:

- Scaling factor x is computed to determine virtual deadline of HI-criticality tasks:
$$x \leftarrow \frac{U^L_{\text{HI}}}{\rho - U^L_{\text{LO}}}$$
- If $U^L_{\text{LO}} + \frac{U^H_{\text{HI}}}{(1-x)} \leq 1$
then virtual-deadline $\widehat{T}_i \leftarrow xT_i$ for every HI task τ_i;
Else return failure.

Figure 2: Modified EDF-VD schedulability condition and scaling factor x

3.2 Correctness under LO-Criticality Mode

In the LO-criticality mode, LO-criticality (HI-criticality) tasks are guaranteed to receive time budgets equal to their LO-WCET values within their deadlines (virtual deadlines) at processor speed ρ. For all tasks being scheduled in LO-criticality mode, we establish the following theorem.

THEOREM 3.1. *The following condition is sufficient for guaranteeing that EDF-VD correctly schedules all the assignments in LO-criticality mode:*

$$x \geq \frac{U^L_{\text{HI}}}{\rho - U^L_{\text{LO}}} \tag{1}$$

PROOF. According to the EDF-VD algorithm, the virtual deadlines of all the HI-criticality tasks for our model are determined prior to run-time. That is, for all $\tau_i \in \tau_{\text{HI}}$, we determine the virtual deadline, $\widehat{T}_i$, where $\widehat{T}_i = xT_i$. Scaling down the period of each HI-criticality task by parameter x indicates increase in its utilization by a factor of x. If all the jobs execute for no more than LO-criticality WCETs (C^L_i), the utilization bound of EDF for implicit-deadline tasks which is equal to processor capacity [27]. We can therefore conclude that:

$$U^L_{\text{LO}} + \frac{U^L_{\text{HI}}}{x} \leq \rho$$

$$\implies x \geq \frac{U^L_{\text{HI}}}{\rho - U^L_{\text{LO}}}$$

is sufficient for guaranteeing that EDF-VD correctly schedules all the assignments in LO-criticality mode. ■

The smallest value of x such that Theorem 3.1 is satisfied is chosen by the EDF-VD algorithm:

$$x \leftarrow \frac{U^L_{\text{HI}}}{\rho - U^L_{\text{LO}}} \tag{2}$$

We now derive a sufficient condition to ensure that EDF-VD meets all deadlines in HI-criticality mode using obtained value of parameter x.

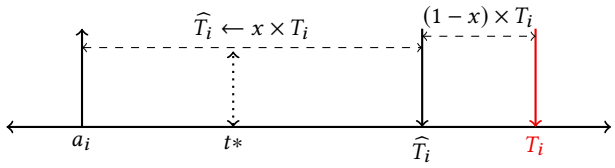

Figure 3: Relation between time instants under EDF-VD framework.

3.3 Correctness under HI-Criticality Mode

Similar to the classical model, if a HI-criticality task τ_i does not signal completion after C_i^L units of execution within its virtual deadline equal to $\widehat{T}_i$, the system exhibits HI-criticality behaviors and triggers a mode-switch. At a specific time instant t* during run-time, in the event that the scheduler identifies a HI-criticality task executing for a duration greater that its C_i^L without signaling completion, a system wide mode-switch is activated which indicates a need to perform the following:

- re-assignment of time period T_i of active HI-criticality tasks from $\widehat{T}_i$ (xT_i) to T_i.
- continue to execute LO-criticality tasks without discarding them (anomalous to the traditional MC-model).
- the speed of the processor increases from ρ to 1.

Theorem 3.2. *The following condition is sufficient for guaranteeing that EDF-VD correctly schedules all the assignments in* HI*-criticality mode:*

$$U_{\text{LO}}^L + \frac{U_{\text{HI}}^H}{(1-x)} \leq 1 \tag{3}$$

Proof. If there is an active HI-criticality task at mode-switch instant t*, the relative deadline of the HI-criticality task is adjusted to $\widehat{T}_i = xT_i$. The actual deadline is $T_i - xT_i = (1-x)T_i$ time units in the future. The relation is illustrated in Figure 3. Thus the utilization of HI-criticality tasks after time instant t* is upper bounded by $\frac{C_i^H}{(1-x)T_i}$.

Summing over all HI- and LO-criticality tasks according to the fact that EDF has a utilization bound equal to the processor capacity (which in our case is 1) we conclude that the following condition is sufficient for guaranteeing that EDF-VD correctly schedules all the assignments in the HI-criticality mode.:

$$\sum_{i|\chi_i=LO} \frac{C_i^L}{T_i} + \sum_{i|\chi_i=HI} \frac{C_i^H}{(1-x)T_i} \leq 1$$

$$U_{\text{LO}}^L + \frac{U_{\text{HI}}^H}{(1-x)} \leq 1$$

Note that we assume $U_{\text{HI}}^H > U_{\text{HI}}^L$ and thus deadline must be shrunk to create some resource capacity for the additional workload after mode switch, i.e., $x < 1$. ■

The upper bound of scaling parameter x can be determined as:

$$(3) \quad \Leftrightarrow (1-x)U_{\text{LO}}^L + U_{\text{HI}}^H \leq 1-x \tag{4}$$

$$\Leftrightarrow U_{\text{LO}}^L - xU_{\text{LO}}^L + U_{\text{HI}}^H \leq 1-x \tag{5}$$

$$\Leftrightarrow x(1-U_{\text{LO}}^L) \leq 1-(U_{\text{HI}}^H + U_{\text{LO}}^L) \tag{6}$$

$$\Leftrightarrow x \leq \frac{1-(U_{\text{HI}}^H+U_{\text{LO}}^L)}{(1-U_{\text{LO}}^L)} \tag{7}$$

We have thus justified the correctness of the EDF-VD scheduling algorithm. From Theorem 3.1, the value of x ensures the correctness of all LO-criticality behaviors, and Theorem 3.2 guarantees the correctness of all HI-criticality behaviors. We give the following sufficient condition for MC-schedulability by EDF-VD for our precise energy-conserving model.

Theorem 3.3. *If τ satisfies*

$$U_{\text{LO}}^L + min\left(U_{\text{HI}}^H, \frac{U_{\text{HI}}^L}{\left(1-\frac{U_{\text{HI}}^H}{1-U_{\text{LO}}^L}\right)}\right) \leq \rho \tag{8}$$

then it is schedulable by EDF-VD.

Proof. We consider two cases:

Case A: $U_{\text{LO}}^{\text{LO}} + U_{\text{HI}}^{\text{HI}} \leq \rho$, In this case, all the LO- and HI-criticality tasks are considered performing on processor with speed ρ which is worst-case reservation schedulable by EDF [2]. The total utilization of the tasks can be represented as:

$$\frac{U_{\text{LO}}^L}{\rho} + \frac{U_{\text{HI}}^H}{\rho} \leq 1$$

The task set can be scheduled by EDF without deadline scaling for HI-criticality tasks at an energy conserving speed.

Case B: $U_{\text{LO}}^L + U_{\text{HI}}^H \geq \rho$

For Condition (8) to hold, it must be the case that,

$$U_{\text{LO}}^L + \frac{U_{\text{HI}}^L}{(1-\frac{U_{\text{HI}}^H}{1-U_{\text{LO}}^L})} \leq \rho$$

$$\Rightarrow \frac{U_{\text{HI}}^L}{\frac{1-(U_{\text{LO}}^L+U_{\text{HI}}^H)}{1-U_{\text{LO}}^L}} \leq \rho - U_{\text{LO}}^L$$

$$\Rightarrow \frac{U_{\text{HI}}^L}{\rho - U_{\text{LO}}^L} \leq \frac{1-(U_{\text{LO}}^L+U_{\text{HI}}^H)}{1-U_{\text{LO}}^L}$$

$$\Rightarrow x \leq \frac{1-(U_{\text{LO}}^L+U_{\text{HI}}^H)}{1-U_{\text{LO}}^L}$$

$$\Rightarrow U_{\text{LO}}^L + \frac{U_{\text{HI}}^H}{(1-x)} \leq 1$$

which is the schedulability condition to correctly schedule all the tasks in HI-criticality mode. ■

By combining Theorem 3.1 and Theorem 3.2, we prove the following theorem.

Theorem 3.4. *Given a precise mixed criticality model task set, the minimum value of ρ for the task set to be schedulable by EDF-VD is:*

$$min\left(U_{\text{LO}}^L + U_{\text{HI}}^H,\ U_{\text{LO}}^L + \frac{U_{\text{HI}}^L(1-U_{\text{LO}}^L)}{1-(U_{\text{HI}}^H+U_{\text{LO}}^L)}\right) \tag{9}$$

only when,

$$U^L_{\text{LO}} + \frac{U^L_{\text{HI}}(1 - U^L_{\text{LO}})}{1 - (U^H_{\text{HI}} + U^L_{\text{LO}})} \leq 1$$

Proof. On combining Theorem 3.1 and 3.2, from Condition (1) and Condition (7) we can represent the range of the value of parameter x:

$$\frac{U^L_{\text{HI}}(\tau)}{\rho - U^L_{\text{LO}}(\tau)} \leq x \leq \frac{1 - (U^H_{\text{HI}} + U^L_{\text{LO}})}{(1 - U^L_{\text{LO}})}$$

Thus determining the minimum value of ρ as:

$$\frac{U^L_{\text{HI}}(\tau)}{\rho - U^L_{\text{LO}}(\tau)} \leq \frac{1 - (U^H_{\text{HI}} + U^L_{\text{LO}})}{(1 - U^L_{\text{LO}})}$$

$$\Rightarrow U^L_{\text{LO}} + \frac{U^L_{\text{HI}}(1 - U^L_{\text{LO}})}{1 - (U^H_{\text{HI}} + U^L_{\text{LO}})} \leq \rho$$

$$\Rightarrow U^L_{\text{LO}} + \frac{U^L_{\text{HI}}}{\frac{1 - (U^H_{\text{HI}} + U^L_{\text{LO}})}{1 - U^L_{\text{LO}}}} \leq \rho$$

which is the sufficient condition (refer to Theorem 3.3) to ensure that EDF-VD successfully schedules all the HI-criticality tasks in τ. ■

3.4 Approximation Ratio

It has been proven that MC-schedulability for dual-criticality recurrent task systems is NP-hard in the strong sense, thus adopting non-optimal algorithms (EDF-VD) is justified [1]. An instance is declared as MC-schedulable if any non clairvoyant on-line algorithm correctly schedules it.

Definition 3.5. An algorithm $\mathcal{A}$ has an **approximation ratio** of $\alpha \geq 1$ if and only if some non clairvoyant on-line algorithm can guarantee MC correctness with full speed of 1 and a energy conserving speed of ρ, Algorithm $\mathcal{A}$ guarantees MC correctness to the same set with a processor of normal speed 1 and energy conserving speed of $\alpha \times \rho$.

Theorem 3.6. *For the precise MC scheduling problem, algorithm EDF-VD has an approximation ratio no larger than*

$$1 + \frac{U^L_{\text{HI}}(1 - U^L_{\text{LO}})}{U^L_{\text{LO}}(1 - (U^H_{\text{HI}} + U^L_{\text{LO}}))} \tag{10}$$

Proof. We observe that any task-set τ that is correctly scheduled by a clairvoyant scheduler upon a processor with normal speed 1 and energy conserving speed $\alpha \times \rho$ must necessarily satisfy:

$$max\left(\frac{U^L_{\text{LO}}}{\alpha\rho} + \frac{U^L_{\text{HI}}}{\alpha\rho}, U^L_{\text{LO}} + U^H_{\text{HI}}\right) \leq 1 \tag{11}$$

Inequality (11) only indicates that the speed of the processor is increased by an approximation ratio α in the LO-criticality mode.

It is safe to assume that energy conserving speed is always $\leq$ normal speed. For an on-line algorithm A, to correctly claim that τ is MC-schedulable, we derive a bound (range) for the approximation ratio where $\alpha \times \rho \leq 1$. To generate a viable upper bound for approximation ratio, we have the following two cases:

Case 1: If $\rho \geq U^L_{\text{LO}} + min\big(U^H_{\text{HI}}, \frac{U^L_{\text{HI}}(1 - U^L_{\text{LO}})}{1 - (U^H_{\text{HI}} + U^L_{\text{LO}})}\big)$.

If this condition is true, we can claim that $\rho \geq \rho_{min}$. This is evident from Condition (9) of Theorem 3.4. If $\rho \geq \rho_{min}$, for any value of $\rho \leq 1$ the system will be schedulable. Thus for this case the maximum value of α to guarantee MC-correctness is 1.

Case 2: If $\rho \leq U^L_{\text{LO}} + min\big(U^H_{\text{HI}}, \frac{U^L_{\text{HI}}(1 - U^L_{\text{LO}})}{1 - (U^H_{\text{HI}} + U^L_{\text{LO}})}\big)$.

The maximum value of α for an on-line algorithm to correctly schedule a system can be given as:

$$\alpha \leq 1 + \frac{U^L_{\text{HI}}(1 - U^L_{\text{LO}})}{U^L_{\text{LO}}(1 - (U^H_{\text{HI}} + U^L_{\text{LO}}))}$$

We justify the reason for selecting such a bound below:

From Theorem 3.4 we have the minimum value of ρ_{min} as,

$$min\left(U^L_{\text{LO}} + U^H_{\text{HI}}\ ,\ U^L_{\text{LO}} + \frac{U^L_{\text{HI}}(1 - U^L_{\text{LO}})}{1 - (U^H_{\text{HI}} + U^L_{\text{LO}})}\right)$$

The schedulability condition is guaranteed if and only if $\rho \geq \rho_{min}$. In this case the value of α should be such that $\alpha\rho$ should satisfy the schedulability condition i.e., $\alpha \geq \rho_{min}/\rho$. Keeping this as a minimum bound required for α, we select a maximum bound for α as $\rho_{min}/U^{\text{LO}}_{\text{LO}}$. Since $0 \leq U^{\text{LO}}_{\text{LO}} < \rho \leq 1$, for this value of α, the schedulability condition will always hold. Thus we have

$$\alpha \leq 1 + \frac{U^L_{\text{HI}}(1 - U^L_{\text{LO}})}{U^L_{\text{LO}}(1 - (U^H_{\text{HI}} + U^L_{\text{LO}}))}$$

The inequality (10) does not represent the tightest upper bound, but an significantly improvement over the loose upper-bound of $1/\rho$. The performance of the algorithm with the derived value of α is demonstrated in Figure 8 in Section 8. ■

4 FLUID SCHEDULING

We now present another approach to tackle the same problem, which is based on fluid scheduling framework. In fluid scheduling, all the tasks receive a fraction of the processor and have a constant execution rate from their release to the deadline. To be a feasible schedule, the summation of the assigned executing speeds of all tasks should not exceed the capacity (or the speed) of the processor.

For each LO-criticality task, a minimum necessary execution speed of $\theta_i = u_i$ would be sufficient under both modes. While for a HI-criticality task, it would need a relatively larger speed under the normal mode (to create enough gap after the mode switch to handle the additional execution requirement), and even a larger speed after the mode switch. Such a relationship is demonstrated in Figure 4, where the blue character indicates LO-criticality task setting and the red character represents HI-criticality task settings.

Note that, MC-DP-Fair [26] can transform any fluid MC schedule into a schedule where each task is assigned with either zero or one processor at each time instant, i.e, a schedule applicable to the actual computing platforms. Hence in the remainder of this section, we will not handle the issue on constructing non-fluid schedules.

First, some simplified notations for the per-mode utilization of the whole task set τ:

$$U^L = \sum_i u^L_i;\ U^H = \sum_i u^H_i.$$

Each task τ_i is assigned a execution speed θ_i in the HI-criticality mode and is assigned $\lambda \cdot \theta_i$ in the LO-criticality mode where $0 < \lambda \leq 1$.

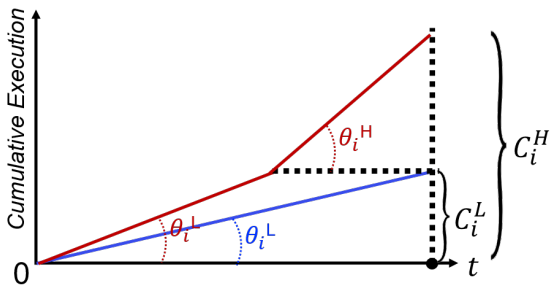

Figure 4: Relation between fluid execution speed and cumulative execution over time of a task under MCF framework.

In this section, we will show that defining λ and θ_i in the following manner can result in a schedule where all deadlines are guaranteed to meet in both the HI-criticality mode and the LO-criticality mode, provided a minimum speed λ is granted in the LO-criticality mode (and the full speed, i.e., the unit-speed 1.0, of the processor that would be enabled in the HI-criticality mode).

For a dual-criticality task-set $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$ to be scheduled by energy conserving preemptive processor:

- A system-wide parameter λ and per-task parameters θ_i are computed as:

$$\lambda = \frac{U^L}{1 + U^L - U^H}. \tag{12}$$

$$\forall i, \quad \theta_i = \frac{u_i^L}{\lambda} + u_i^H - u_i^L \tag{13}$$

- If the energy-conserving speed $\rho \geq \lambda$
  then each task τ_i is to be executed at speed $\lambda \cdot \theta_i$ in the LO-criticality mode and at speed θ_i in the HI-criticality mode;
  Else return failure.

Figure 5: Modified MCF speed assignments and schedulability condition

Note that, we have $0 < \lambda \leq 1$ (see Equation (12)) because $U^{\text{LO}} > 0$ and $U^{\text{HI}} \leq 1$. Also, the following Lemma shows that the total allocated execution speeds to tasks in the HI-mode match the full speed of the processor (i.e., 1.0).

Lemma 4.1. $\sum_i \theta_i = 1$.

Proof.

$$\begin{aligned}\sum_i \theta_i &= \frac{\sum_i u_i^L}{\lambda} + \sum_i u_i^H - \sum_i u_i^L \\ &= \frac{U^L}{\lambda} + U^H - U^L \\ &= 1 + U^L - U^H + U^H - U^L \\ &= 1\end{aligned}$$

■

This lemma plus the condition that $\rho \geq \lambda$ also directly implies that the total allocated execution speeds to tasks in the LO-criticality mode do not exceed the energy-conserving speed, i.e., $\sum_i(\lambda\theta_i) = \lambda \leq \rho$.

Lemma 4.1 indicates the proposed fluid executing speeds assignment does not exceed the processor's speed limit. Then, Theorem 4.2 shows that this speed assignment is sufficient to guarantee all deadlines to be met.

Theorem 4.2. *All deadlines are met in the fluid schedule where task τ_i's speed is $\lambda\theta_i$ under LO-criticality mode and θ_i under HI-criticality mode, where θ_i and λ values are assigned according to Equations* (12) *and* (13).

Proof. We prove this theorem by focusing on an arbitrary job $\tau_{i,j}$ in the systems and discuss three cases by where its release time and deadline are. We let $r_{i,j}$, and $d_{i,j}$ denote the release time and deadline of the job of interest $\tau_{i,j}$ and let t_s denotes the time instant when the mode switch was triggered. If there is no mode switch, t_s can be interpreted as $+\infty$. Then, we have the following three cases for the job of interest $\tau_{i,j}$.

Case 1: $r_{i,j} < d_{i,j} \leq t_s$**.** Because $r_{i,j} < d_{i,j} \leq t_s$, $\tau_{i,j}$ must complete its execution when receives C_i^{LO} execution; otherwise, a mode switch must be triggered at an earlier time than t_s. Also, in the LO-criticality mode, $\tau_{i,j}$ is being executed at speed $\lambda \cdot \theta_i = u_i^L + (u_i^H - u_i^L)\lambda \geq u_i^L$, because $u_i^H \geq u_i^L$. Therefore, $\tau_{i,j}$ must meet its deadline $d_{i,j} = r_{i,j} + T_i$.

Case 2: $t_s \leq r_{i,j} < d_{i,j}$**.** In the HI-criticality mode, $\tau_{i,j}$ is being executed at speed $\theta_i = \frac{u_i^L}{\lambda} + u_i^H - u_i^L \geq u_i^H$, because $0 < \lambda \leq 1$ implies $\frac{1}{\lambda} - 1 \geq 0$. Also, $\tau_{i,j}$ must complete its execution when receives C_i^H execution. Therefore, $\tau_{i,j}$ must meet its deadline $d_{i,j} = r_{i,j} + T_i$.

Case 3: $r_{i,j} < t_s < d_{i,j}$**.** In this case, $\tau_{i,j}$ is released in the LO-criticality mode but has a deadline in the HI-criticality mode. Therefore, $\tau_{i,j}$ may be executed during time interval $[r_{i,j}, t_s)$ at speed $\lambda \cdot \theta_i$ and during time interval $[t_s, d_{i,j})$ at speed θ_i.

Starting with an executing speed $\lambda \cdot \theta_i$, the job of interest $\tau_{i,j}$ must have done $\lambda \cdot \theta_i(t_s - r_{i,j})$ execution at time t_s, and will then being executed by speed θ_i. Therefore, letting f denote the time instant at which $\tau_{i,j}$ completes its execution, it must hold that

$$\begin{aligned} f &\leq t_s + \frac{C_i^H - \lambda \cdot \theta_i(t_s - r_{i,j})}{\theta_i} \\ &= (1-\lambda)t_s + \lambda \cdot r_{i,j} + \frac{C_i^H}{\theta_i}. \end{aligned} \tag{14}$$

On the other hand, it must hold that

$$\lambda \cdot \theta_i(t_s - r_{i,j}) \leq C_i^L; \tag{15}$$

otherwise, a mode switch would have been trigger at an earlier time than t_s. It is also clear that (15) implies

$$t_s \leq r_{i,j} + \frac{C_i^L}{\lambda \cdot \theta_i}. \tag{16}$$

Therefore, by (14), (15), and the fact that $\lambda \leq 1$,

$$\begin{aligned} f &\leq (1-\lambda)\left(r_{i,j} + \frac{C_i^L}{\lambda \cdot \theta_i}\right) + \lambda \cdot r_{i,j} + \frac{C_i^H}{\theta_i} \\ &= r_{i,j} + \frac{C_i^L}{\lambda \cdot \theta_i} + \frac{C_i^H}{\theta_i} - \frac{C_i^L}{\theta_i} \\ &= r_{i,j} + \frac{u_i^L \cdot T_i}{\lambda \cdot \theta_i} + \frac{u_i^H \cdot T_i}{\theta_i} - \frac{u_i^L \cdot T_i}{\theta_i} \\ &= r_{i,j} + \left(\frac{u_i^L}{\lambda} + u_i^H - u_i^L\right)\frac{T_i}{\theta_i} \\ &= r_{i,j} + T_i \\ &= d_{i,j}. \end{aligned}$$

That is, $\tau_{i,j}$ also must meet its deadline in Case 3.

Because the job of interest $\tau_{i,j}$ was chosen arbitrarily and the three cases exhausted all possibilities for $\tau_{i,j}$, all jobs must meet their deadlines, and the theorem follows. ■

Theorem 4.3. *The modified MCF algorithm in Figure 5 has an approximation ratio no greater than* $1/(1 + U^{\text{LO}} - U^{\text{HI}})$.

Proof. Let λ^* denote the minimal required degraded speed such that the system is schedulable under an optimal scheduler (potentially with clairvoyance) and λ is calculated by (12). Then, our goal is to upper bound λ/λ^*.

λ^* must be at least U^L so that the system is not over-utilized in the LO-criticality mode, i.e., $\lambda^* \geq U^L$. Then, it, by (12), directly leads to that $\lambda/\lambda^* \leq 1/(1 + U^L - U^H)$. ■

Note that MCF and EDF-VD are incomparable to each other; i.e., there exists an MC task set where MCF would return a smaller normal mode speed than EDF-VD, while there also existing an MC task set with the opposite relationship. Such a relationship is demonstrated via experiments in Section 5.

5 EXPERIMENTAL EVALUATION

We have conducted a progression of schedulability tests to assess the effectiveness of the EDF-VD scheduling technique to guarantee that MC implicit deadline sporadic task systems are correctly scheduled.

The experiments were conducted on a randomly generated task-set that were generated according to the workload generation model established by Guan et al. [18] with further modifications. The input specifications for our workload generation are as follows:

- U_{bound} is the desired upper bound of utilization of the system: $(U_{\text{LO}}^L(\tau) + U_{\text{HI}}^H(\tau))$
- The time period of a task is randomly chosen in the range $[T_{down}, T_{up}]$; $0 \leq T_{down} \leq T_{up}$.
- For each task, a value is randomly selected in the range $[U_{down}, U_{up}]$ and multiplied with task's period, to obtain execution time in the LO-mode; $0 \leq U_{down} \leq U_{up} \leq 1$.
- The ratio of HI-criticality WCET and LO-criticality WCET is drawn from the range $[Z_{down}, Z_{up}]$; $1 \leq Z_{down} \leq Z_{up}$.
- P: Probability that the chosen task is HI-critical; $0 \leq P \leq 1$

For the generation of a MC-workload from the combination of these parameter values, the task generation algorithm adds tasks to an empty set until the utilization bound is met iteratively. In our experiments, we determine the ratio of systems scheduled correctly against the system utilization U_{bound}. Simulations are carried out for different values of ρ. Although we cannot draw authoritative conclusions from the experiments as the random workload generator influences the results, we do make some interesting observations.

When the average utilization percentage is smaller than 0.5, the task system is always schedulable. This observation from Figure 6 matches our speed-up factor computation, and it demonstrates the ratio of systems scheduled correctly as a function of system utilization. For different values of ρ ranging from [0.5,0.9] and Utilization ranging from [0.4,1.0], Figure 6 reports the schedulability ratio for EDF-VD and the MCF (left and mid sub-figure respectively). We see that the system is completely schedulable when the average utilization is ≤ 0.5. Figure 6 (right) compares the performance of EDF-VD and the MCF (in terms of schedulability ratio) for different utilization values ranging from [0.4,1.0] and different ρ values ranging from [0.6,0.9]. This result reports that none of them is superior (inferior) over (to) the other. However, when the utilization is not more than 0.8, MCF performs better (or equal) than EDF-VD. Similarly, in Figure 7, the performance of the EDF-VD and MCF algorithm (left and middle sub-figure), and their performance comparison (right sub-figure) is demonstrated for a workload with different values of ρ and $[Z_{down}, Z_{up}] = [1, 8]$.

Figure 8 shows the ratio of correctly scheduled task sets with an energy-conserving speed of $\alpha \times \rho$ against system utilization The performance of the algorithm was determined again with an energy-conserving speed of $\alpha\rho$ and normal speed 1, where α is the approximation ratio. The maximum value of α was considered according to Condition (10) as proved in Subsection 3.4. It is interesting to observe that the maximum bound chosen for approximation ratio α is sufficient to guarantee MC correctness by an on-line non-clairvoyant algorithm.

6 RELATED WORKS

Significant work has concentrated on various versions of the MC model proposed by Vestal [33]. A thorough review of the various adaptations is reviewed in the survey by Burns et al. [12]. Several of these current works adopt a rigorous approach of dropping the LO-criticality jobs in HI-criticality mode [3, 8, 13, 14]. Burns et al. [11] were the first to address this issue by assigning time budgets to LO-priority tasks by switching their priority or degrading their services by extending the periods in HI-criticality mode [24, 25]. However, the work by Ernst et al. [15] has criticized these techniques mentioned above because of having minor setbacks and being not practical. Another approach is presented by Burns et al. [11], popularly known as the IMC model, where the execution time of LO-criticality tasks is diminished in the event of a mode-switch. The schedulability analysis of the IMC model has been studied for both fixed-priority scheduling and EDF-VD by Burns et al. [11] and Liu et al. [28] respectively. The work by Baruah et al. [5] considered a generalization of the Vestal model where the less critical functionalities are not entirely discarded even in the HI-criticality mode. Lee et al. [26] proposed the MC-Fluid (a fluid

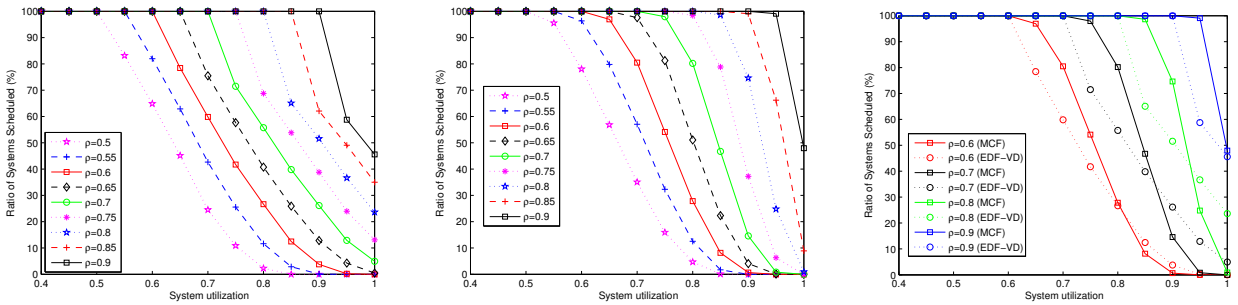

Figure 6: Example outcome of schedulability experiments of (left) EDF-VD, (mid) MCF, and (right) comparison between EDF-VD and the MCF. We change the value of ρ with other fixed parameters (i.e., $[U_{down}, U_{up}] = [0.02, 0.2]; [T_{down}, Tup] = [5, 50]; [Z_{down}, Z_{up}] = [1, 4]; P = 0.5$).

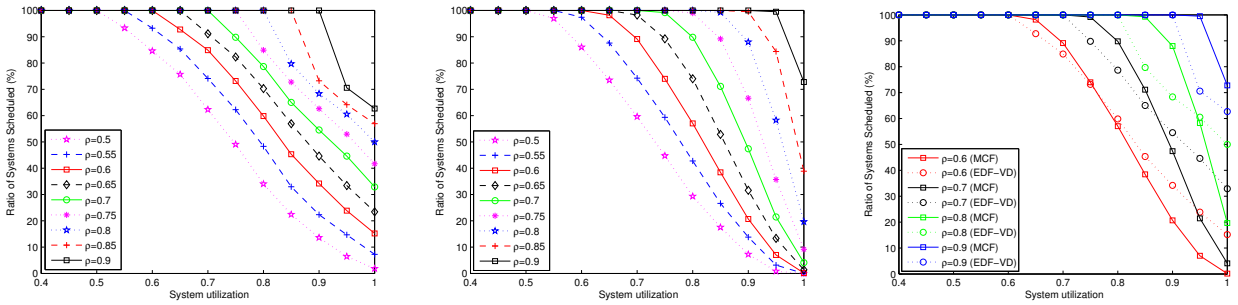

Figure 7: Example outcome of schedulability experiments of (left) EDF-VD, (mid) MCF, and (right) comparison between EDF-VD and the MCF. We change the value of ρ with other fixed parameters (i.e., $[U_{down}, U_{up}] = [0.02, 0.2]; [T_{down}, Tup] = [5, 50]; [Z_{down}, Z_{up}] = [1, 8]; P = 0.5$).

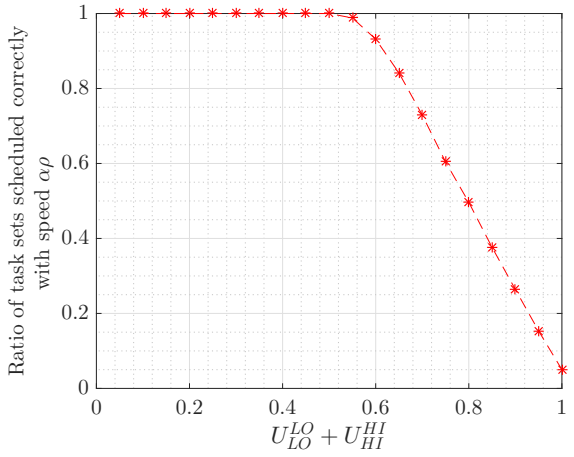

Figure 8: Performance of the EDF-VD algorithm under normal speed 1 and energy conserving speed $\alpha\rho$; with α value determined from Equation (10)

model-based scheduling algorithm) for the MC tasks in a multiprocessor platform. In the MC-Fluid model, for each task, a criticality dependent execution rate is determined. They also proposed MC-DP-Fair, which is an implementable version (on a real-hardware) of MC-Fluid. The work by Baruah et al. [6] derived MCF, which is a simplified variant of MC-Fluid. For a dual-criticality system, they proved that the MCF has a speedup bound no worse than 1.33. Finally, they improved the speedup bound for MC-Fluid from 1.618 to 1.33. Considering the adaptive MC- Weakly Hard model Gettings et al. [17] proposed a response time-based schedulability analysis. MC- Weakly Hard model guarantees a minimum service for LO-criticality tasks in case of a mode switch.

Energy minimization has also become a rising concern in the non-MC [10, 20, 21, 31] and the MC applications [23]. Huang et al. [23] have exploited the DVFS technique to address energy minimization issue in mixed-criticality systems by speeding up the processor speed during overrun. Huang et al. [23] also established that increased speeds during overrun conditions are beneficial to minimize expected energy consumption of the system. However, in their approach, all the LO-criticality tasks are penalized in HI-criticality mode. This model was extended by Narayana et al. [29] to accommodate multi-core processors, in which a trade-off is determined between both static and dynamic energy consumption in different operation modes (LO- and HI-criticality).

7 CONCLUSION

The conventional mixed-criticality model, despite its popularity, is controversial for penalizing all LO-criticality tasks in HI-criticality mode. Recent works throw light on overcoming this setback by partially (if not entirely) trying to accommodate LO-criticality tasks even under pessimistic behaviors. In this work, we develop an integrated model combining precise scheduling of LO-criticality tasks on energy conserving platforms that adopt the DVFS strategy. Two sufficient tests for this unified model under both EDF-VD and MCF scheduling framework are proposed. The sufficient test is evaluated theoretically with sound proofs and via schedulability experiments on randomly generated workloads. We provide results on calculating the approximation ratio to satisfy real-time requirements in the typical situations.

As the future work, we seek to derive and prove a tighter bound for the approximation ratio (if one exists). We also plan to work on counterexamples to show the minimum possible bound, and also explore schedulability conditions under a task-wise mode-switch, contrary to the system-wise mode switch adopted in this work. We also wish to conduct a simulation study on actual energy savings with on-board implementations.

ACKNOWLEDGMENTS

We would like to thank Prof. Sanjoy Baruah from Washington University in St Louis for sharing his valuable insights. This work is supported by NSF grant CNS-1850851, a start-up grant from the University of Central Florida, and start-up and REP grants from Texas State University.

REFERENCES

[1] Sanjoy Baruah and Kunal Agarwal. 2018. Intractability issues in mixed-criticality scheduling. In *Proceedings of the 30th EuroMicro Conference on Real-Time Systems (ECRTS), IEEE*. IEEE.

[2] Sanjoy Baruah, Vincenzo Bonifaci, Gianlorenzo DAngelo, Haohan Li, Alberto Marchetti-Spaccamela, Suzanne Van Der Ster, and Leen Stougie. 2012. The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems. In *Proceedings of the 24th Euromicro Conference on Real-Time Systems (ECRTS), IEEE*. IEEE, 145–154.

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

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

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

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

[7] Sanjoy Baruah and Zhishan Guo. 2013. Mixed-criticality scheduling upon varying-speed processors. In *2013 IEEE 34th Real-Time Systems Symposium*. IEEE, 68–77.

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

[9] Sanjoy K Baruah, Vincenzo Bonifaci, Gianlorenzo D-Angelo, Alberto Marchetti-Spaccamela, Suzanne Van Der Ster, and Leen Stougie. 2011. Mixed-criticality scheduling of sporadic task systems. In *European Symposium on Algorithms*. Springer, 555–566.

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

[11] Alan Burns and Sanjoy Baruah. 2013. Towards a more practical model for mixed criticality systems. In *Workshop on Mixed-Criticality Systems (colocated with RTSS)*.

[12] Alan Burns and Robert I Davis. 2017. A survey of research into mixed criticality systems. *ACM Computing Surveys (CSUR)* 50, 6 (2017), 82.

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

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

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

[16] Alexandre Esper, Geoffrey Nelissen, Vincent Nélis, and Eduardo Tovar. 2015. How realistic is the mixed-criticality real-time system model?. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*. ACM, 139–148.

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

[18] Nan Guan, Pontus Ekberg, Martin Stigge, and Wang Yi. 2013. Improving the scheduling of certifiable mixed-criticality sporadic task systems. *Technical Report 2013–008* (2013).

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

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

[21] Zhishan Guo, Ashikahmed Bhuiyan, Abusayeed Saifullah, Nan Guan, and Haoyi Xiong. 2017. Energy-efficient multi-core scheduling for real-time DAG tasks. (2017).

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

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

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

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

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

[27] Chung Laung Liu and James W Layland. 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)* 20, 1 (1973), 46–61.

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

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

[30] Risat Mahmud Pathan. 2017. Improving the Quality-of-Service for Scheduling Mixed-Criticality Systems on Multiprocessors. In *LIPIcs-Leibniz International Proceedings in Informatics*, Vol. 76. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

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

[32] Hang Su and Dakai Zhu. 2013. An elastic mixed-criticality task model and its scheduling algorithm. In *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 147–152.

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