MC-ADAPT: Adaptive Task Dropping in Mixed-Criticality Scheduling

ABSTRACT

Recent embedded systems are becoming integrated systems with components of different criticality. To tackle this, mixed-criticality systems aim to provide different assurance to components of different criticality while achieving efficient resource utilization. Many approaches have been proposed to execute more lower-criticality tasks without affecting the timeliness of higher-criticality tasks. Those previous approaches however have at least one of the two limitations; i) they penalize all lower-criticality tasks at once upon a certain situation, or ii) they make the decision how to penalize lower-criticality tasks at design time. As a consequence, they underutilize resources by imposing an excessive penalty on low-criticality tasks. Unlike those existing studies, we present a novel framework, called MC-ADAPT, that aims to minimally penalize lower-criticality tasks by fully reflecting the dynamically changing system behavior into adaptive decision making. Towards this, we propose a new scheduling algorithm and develop its runtime schedulability analysis capable of capturing the dynamic system state. Our proposed algorithm adaptively determines which task to drop based on the runtime analysis. To determine the quality of task dropping solution, we propose the speedup factor for task dropping while the conventional use of the speedup factor only evaluates MC scheduling algorithms in terms of the worst-case schedulability. We apply the speedup factor for a newly-defined task dropping problem that evaluates task dropping solution under different runtime scheduling scenarios. We derive that MC-ADAPT has a speedup factor of 1.619 for task drop. This implies that MC-ADAPT can behave the same as the optimal scheduling algorithm with optimal task dropping strategy does under any runtime scenario if the system is sped up by a factor of 1.619.

KEYWORDS

Mixed Criticality Scheduling, The Processor Speedup Factor

ACM Reference format:
. 2017. MC-ADAPT: Adaptive Task Dropping in Mixed-Criticality Scheduling. In *Proceedings of ACM EMSOFT conference, Seoul, Korea, Oct. 2017 (EMSOFT'17),* 11 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

One of the growing trends in safety-critical embedded systems is towards increasing complex systems with various applications of different importance or criticality. Their real-world examples are avionics systems [18] and automotive systems [1]. These systems are called *Mixed-Criticality (MC) systems*, which integrate multiple components with different criticality levels on a single computing platform. The main design goal for MC systems is to provide different levels of assurance to functionalities of different criticality levels, while achieving efficient resource utilization.

Since the seminal work of Vestal [22], a vast amount of work has been developed for scheduling of real-time MC systems (see [7] for a survey). In typical approaches, the system has two levels of criticality (high and low), and a high-criticality task comprises of two WCET (Worst-Case Execution Time) estimates with different levels of confidence. Each high-criticality task typically starts in the low-criticality mode during which it completes execution without exceeding a low-confidence WCET estimate. Upon overrunning the low-confidence WCET estimate, the task is considered to transit to the high-criticality mode during which it can execute up to its pessimistic (high-confidence) WCET estimate. A typical requirement for MC systems is that 1) all high-criticality tasks always satisfy their deadlines and 2) all low-criticality tasks meet deadlines as long as all high-criticality tasks remain in the low-criticality modes. A major challenge is to guarantee the MC-schedulability while at the same time improving resource utilization.

A vast majority of existing solutions [2, 3, 6, 9–11, 13, 15, 17, 20, 21] employ an assumption of *system-level mode switch* that when a task exhibits high-criticality behavior by violating its low-confidence WCET estimate, all the other high-criticality tasks also show high-criticality behavior simultaneously. Upon any task transiting to the high-criticality mode, those existing solutions commonly penalize all of the low-criticality tasks, i.e., either by dropping all [2, 3, 9, 13, 20] or degrading all the services offered to them [6, 10, 11, 15, 17, 21]. However, this is overly pessimistic, because not all high-criticality tasks necessarily violate their low-confidence WCET estimates at the same time. For example in automotive, it is a rare case that both adaptive cruise control (ACC) component and anti-lock braking system (ABS) component simultaneously violate the normal ranges of WCET estimates because the behavior of each component depends on different sensors (ACC depends on laser and radar sensors while ABS depends on friction and speed sensors).

Relaxing such an assumption, we consider *task-level mode switch*, where tasks can exhibit high-criticality behavior at different times, independently from each other. Under task-level mode switch, it is allowed that some high-criticality tasks execute in the high-criticality mode while others remain in the low-criticality mode. This makes it possible to penalize some of the low-criticality tasks selectively in the event of mode switch, rather than all of them unnecessarily. In the automotive example, various infotainment components such as head-up display can be maximally serviced even under deviation of some high-criticality tasks. Our goal is then to minimize the total number of low-criticality tasks to be dropped under task-level mode switch subject to the MC schedulability constraints. To achieve this, we seek to develop a new MC scheduling framework that dynamically determines which tasks to drop at runtime.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

EMSOFT'17, Oct. 2017, Seoul, Korea

© 2017 Copyright held by the owner/author(s).

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

Task dropping decision under task-level mode switch is challenging since the system state is dynamically changing in the sense that a set of tasks in the high-criticality (or low-criticality) mode as well as a set of tasks dropped (or active) change dynamically over time. In addition, when a high-criticality task requires additional resources due to its mode switch, the actual resources to be secured by dropping low-criticality tasks may vary depending on the current system state. There exist some recent studies [12, 14, 19] considering task dropping under task-level mode switch; however, all of them do not yet take the dynamic behavior of the system into full consideration since the tasks to be dropped are determined (and analyzed) at design time and then remain unchanged during runtime. Such a static decision has a considerable degree of pessimism, leading to unnecessary dropping of low-criticality tasks. This is because existing solutions cannot fully capture dynamic system states and incorporate them efficiently into decision making of task dropping at runtime.

In this paper, we focus on the following research questions to address these challenges.

- Q1. How can we analyze the impact of dynamic system state changes on the MC schedulability at runtime?
- Q2. How can we make adaptive decision on task dropping without sacrifice in the MC schedulability?
- Q3. How can we evaluate the quality of task dropping solution for MC scheduling algorithms?

This paper presents *MC-ADAPT*, which is an adaptive MC scheduling framework that makes online task dropping decision according to dynamic system states under task-level mode switch. In particular, to address Q1, we develop a run-time schedulability analysis capable of capturing dynamic system states, which serves as a basis of online task dropping decision. Our run-time analysis is efficient in the sense that it is sufficient to consider only the current system state without tracking the previous history of all system state changes when deciding which tasks to drop.

To address Q2, we design a new scheduling algorithm, called EDF-AD, by extending EDF-VD to support adaptive task dropping under task-level mode switch. EDF-AD utilizes the proposed run-time schedulability analysis to find a minimal set of low-criticality tasks to be dropped, so as to secure the additional resources requested by a mode-transiting task at the current system state. We found that a straightforward extension of EDF-VD leads to schedulability loss compared to EDF-VD. We develop another scheduling algorithm, called EDF-AD-E, which identifies the subset of tasks triggering schedulability loss and isolates them from other tasks. However, improving schedulability is not our main goal in this paper.

To address Q3, we propose the speedup factor for task drop. Although the conventional speedup factor for the MC scheduling problem is effective to evaluate the schedulability of scheduling algorithms, it does not evaluate the quality of task dropping solution. We propose to apply the speedup factor for a different MC scheduling problem, called the *task dropping problem*, which evaluates runtime performance of low-criticality tasks in addition to MC-schedulability. We derive that the speedup factor of MC-ADAPT for the task drop is 1.619, which indicates that MC-ADAPT can schedule any feasible task set under any runtime scenario by dropping the same number of low-criticality tasks as the optimal scheduling framework with optimal task dropping strategy if the processor is sped up by a factor of 1.619. To the best of our knowledge, this is the first work to quantify runtime performance of MC scheduling algorithms via the processor speedup factor [16]. In addition, we evaluate MC-ADAPT via simulation in terms of schedulability and resource utilization.

In summary, this paper makes the following contributions:

- We present the MC-ADAPT framework supporting online adaptive task dropping under task-level mode switch.
- We propose new scheduling algorithms, called EDF-AD and EDF-AD-E, that drop a minimal set of low-criticality tasks based on run-time schedulability analysis, without sacrifice in schedulability.
- We propose the speedup factor for the task dropping problem and derive that EDF-AD-E has a speedup factor of 1.619 for task drop.
- Our simulation shows the effectiveness of our framework in terms of schedulability and resource utilization.

2 RELATED WORK

Since Vestal's initial work [22] on MC systems, a rich number of studies have been introduced for MC real-time scheduling. A significant proportion of existing solutions make an assumption on MC system behavior that once a single high-criticality task violates its low-confidence WCET, all the other high-criticality tasks will simultaneously exhibit similar behavior, i.e., system-level mode switch. Upon system-level mode switch, typical scheduling approaches [2, 3, 9, 13] require the pessimistic strategy of dropping all active low-criticality tasks immediately. There is a method to delay dropping low-criticality tasks by adjusting the threshold of mode switch in offline computation [20] or runtime computation [11]. Bate et al. [5] presented a scheduling protocol for returning to the low-criticality mode so as to resume the execution of low-criticality tasks. Other works [6, 10, 11, 15, 17, 21] provided degraded service to low-criticality tasks after system-level mode switch, which includes stretching their periods [6, 15, 21], lowering their priorities [6], skipping their jobs [10], or reducing their execution times [11, 17]. However, all the above studies share the impractical assumption of system-level mode switch, which results that resources are still under-utilized in practice.

Relaxing the assumption of system-level mode switch, recent studies [12, 14, 19] considered task-level mode switch that enables low-criticality tasks to be penalized selectively in the event of individual mode switch. Huang et al. [14] proposed offline mapping from each high-criticality tasks to multiple low-criticality tasks: when the high-criticality task mode switches, the connected low-criticality tasks are dropped. Ren and Phan [19] proposed a similar technique with exclusive task grouping where each group has a single high-criticality tasks. Gu et al. [12] also presented task grouping technique that allows multiple high-criticality tasks. Within the predefined tolerance limit of a task group, they drop only low-criticality tasks within the task group. All the above studies make static task dropping decisions for a limited number of scheduling scenarios (which task is mode-switching or how many tasks in a task group are mode-switched) at design time. On the other hand, MC-ADAPT makes dynamic scheduling decision at runtime with online schedulability test considering runtime criticality of each high-criticality task. Our task dropping algorithm drops the minimum set of low-criticality tasks for each possible system state. It is worth to note that since the system state comprises not only

the information considered by the above approaches but also new additional information, such as dropped low-criticality tasks.

The processor speedup factor [16] is widely used to evaluate MC scheduling algorithms [2, 17]. Baruah et al. [2] proposed EDF-VD with a speedup factor of 4/3, which is optimal in uniprocessor MC scheduling. Since the conventional speedup factor in the existing work only evaluates schedulability of MC scheduling algorithms, we propose the speedup factor to evaluate the quality of task dropping solution.

3 SYSTEM MODEL AND FRAMEWORK OVERVIEW

3.1 System Model

We consider *dual-criticality* uniprocessor systems with two distinct criticality levels: HI (high) and LO (low).

Task Model. We consider an implicit-deadline sporadic task system (denoted τ) of n MC tasks. Each MC task τ_i is characterized by $(T_i, C_i^L, C_i^H, \chi_i)$, where

- $T_i \in \mathbb{R}$ is the minimum inter-job separation time (or *period*),
- $C_i^L \in \mathbb{R}$ is a low-confidence WCET (LO-WCET),
- $C_i^H \in \mathbb{R}$ is a high-confidence WCET (HI-WCET), and
- $\chi_i \in \{\mathsf{HI}, \mathsf{LO}\}$ is a task criticality level.

We can categorize individual tasks τ_i by their criticality levels χ_i. For notational convenience, let τ_H denote a set of tasks with HI-criticality levels (or a Hi-task set), i.e., $\tau_H \stackrel{\text{def}}{=} \{\tau_i \in \tau | \chi_i = \mathsf{HI}\}$. Likewise, τ_L denotes a set of tasks with LO-criticality levels (or a LO-task set).

Utilization. The LO- and HI-utilization of a task τ_i are defined as $u_i^L \stackrel{\text{def}}{=} C_i^L/T_i$ and $u_i^H \stackrel{\text{def}}{=} C_i^H/T_i$, respectively. For notational convenience, we have

$$U_L^L \stackrel{\text{def}}{=} \sum_{\tau_i \in \tau_L} u_i^L, \quad U_H^L \stackrel{\text{def}}{=} \sum_{\tau_i \in \tau_H} u_i^L, \quad U_H^H \stackrel{\text{def}}{=} \sum_{\tau_i \in \tau_H} u_i^H.$$

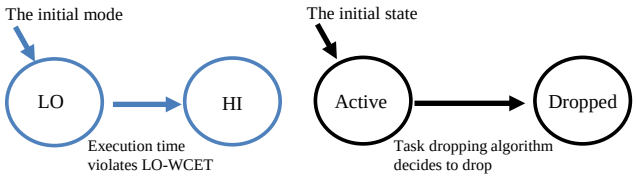
Behavior Model. We assume there exists some degree of uncertainty on the execution time of different jobs for a task. We consider task-level criticality mode (*task mode*). Each HI-task τ_i has its own task mode (denoted as M_i) that indicates its behavior. A task τ_i is said to be in LO-mode ($M_i = \mathsf{LO}$) if no job of the task has executed more than its LO-WCET (C_i^L) and be in HI-mode ($M_i = \mathsf{HI}$) otherwise.

Under task mode, we consider task-level mode switch, where an individual task changes its task mode independently. That is, each HI-task starts in LO-mode, and switches to HI-mode when its execution time violates C_i^L (called *mode switch*) (see Fig. 1a). Most of the existing MC schemes employ the system-level criticality-mode (called *system mode*); the system mode is LO if all HI tasks are in LO-mode, and it becomes HI when all HI-tasks are in HI-mode. System mode is a special case of task mode.

In addition to HI-tasks, we consider the execution state of a LO-task (see Fig. 1b): each LO-task is in either *active* state or *dropped* state. Initially, all LO-tasks are active (jobs of the tasks are released sporadically). On mode switch, some LO-tasks are allowed to be dropped in order to support HI-tasks with their additional resource requests. When a LO-task is dropped, no job of the task is released.

System Goal. It is generally important to maximize the performance of LO-tasks as well as the MC-schedulability [6]. Our system goal is to drop as few LO-task as possible under MC-schedulability, which consists of two conditions:

- **Condition A**: HI-tasks are always schedulable.
- **Condition B**: LO-tasks are schedulable if no HI-task is in HI-mode.



(a) The task mode of a HI-task (b) The execution state of a LO-task

Figure 1: The Behavioral Model of Tasks

3.2 The Overview of the MC-ADAPT Framework

We introduce an MC scheduling framework (called MC-ADAPT) that seeks to drop as few LO-tasks as possible under the MC-schedulability. The key features of MC-ADAPT include task-level mode switch and adaptive LO-task dropping. To enable such features and leverage them for achieving our goal raises several issues to address. We first need to design a new scheduling algorithm that supports task-level mode switch effectively. It is desirable to generalize or dominate the similar ones based on system-level mode switch. We then need to develop a method of task dropping that finds a minimal set of LO-tasks to drop while securing the additional resources requested by a mode-transiting task. This requires to analyze run-time variation on the resource demand of HI-tasks under task-level mode switch. When a task exhibits HI-behavior, the amount of additional resource demand for all HI-tasks to meet their deadlines can vary depending on a different runtime system state, i.e., a different combination of tasks in HI-mode, LO-mode, active state, and dropped state. This requires to calculate such resource demand precisely based on a runtime system state and determine which LO-tasks to be dropped so as to guarantee the required resources as well as minimize the number of dropped tasks at runtime in an efficient manner.

To develop the MC-ADAPT framework, we design a scheduling algorithm and its analysis building upon the principle of EDF-VD (Sec. 5), and enhance it to become a generalization of EDF-VD (Sec. 6).

4 RECAPITULATION OF EDF-VD SCHEDULING

In this section, we recapitulate EDF-VD [2] for the implicit-deadline task model, whose algorithm and analysis are simple while being speedup-optimal. Due to its simplicity, EDF-VD is extended into various directions (e.g., constrained-deadline task model [8, 9] and imprecise computation model [17]).

EDF-VD considers system-level mode switch that when a single HI-task switches to HI-mode, all the other HI-tasks switch to HI-mode simultaneously. Upon such an event, it changes the system mode from LO-mode to HI-mode and drops all the LO-tasks. Capturing the characteristics of MC tasks that HI-tasks are subject to

different WCET requirements in different modes, EDF-VD assigns different priorities to a HI-task in different modes (*virtual deadline* (VD) in the LO mode and real deadline in HI-mode).

We now explain how EDF-VD assigns VDs. For a HI-task τ_i, the VD of the task (V_i) is assigned by $V_i = xT_i$ where x is the VD coefficient[1] ($x \in \mathbb{R}$ s.t. $0 < x \leq 1$) with $x = U_H^L/(1 - U_L^L)$.

Schedulability analysis of EDF-VD consists of the following lemmas. We will reuse Lemma 4.1 for our new algorithm.

Lemma 4.1 (from [2]). *A task set τ is schedulable by EDF-VD when all HI-tasks are in LO-mode if*

$$U_L^L + \frac{U_H^L}{x} \leq 1. \tag{1}$$

Lemma 4.2 (from [2]). *A task set τ is schedulable by EDF-VD when any HI-tasks are in HI-mode if*

$$xU_L^L + U_H^H \leq 1. \tag{2}$$

By Lemmas 4.1 and 4.2, a given task τ is MC-schedulable by EDF-VD if Eqs. (1) and (2) hold.

5 THE MC-ADAPT FRAMEWORK

In this section, we present the MC-ADAPT framework that supports adaptive task dropping under task-level mode switch. To minimize the dropping of LO-tasks, we need a resource-efficient scheduling algorithm and task dropping algorithm to choose a minimal set of LO-tasks for additional resource request by a mode-transiting task (Sec. 5.1). In Sec. 5.2, we check whether a task is schedulable at a specific mode switch via online schedulability analysis. In Sec. 5.3, we check whether a task set is schedulable under any sequence of mode switches via offline schedulability analysis.

5.1 The EDF-AD Scheduling Algorithm

To minimize the dropping of LO-tasks under task-level mode switch, we propose the EDF-AD (EDF-Adaptive task Dropping) algorithm extending the principle of EDF-VD.

Runtime Scheduling Policy. To guarantee the schedulability of HI-tasks after mode switch, we apply VDs to HI-tasks in their LO-mode. EDF-AD adopts the same VD assignment[2] as EDF-VD, but changes the priorities of tasks according to task level mode switch. EDF-AD schedules the job with the earliest effective deadline and operates under the following rules:

- Schedule LO-tasks with their real deadlines.
- For each HI-task τ_i, schedule the task with its VD in its LO-mode (M_i = LO) and with its real deadline otherwise.
- At the mode switch of a HI-task τ_i, set M_i := HI and drop LO-tasks[3] by the EDF-AD task dropping algorithm.

If the task mode of a HI-task is changed to HI at the mode switch, the relative deadline of the task is postponed from its VD to its real deadline.

Task Dropping Algorithm. To drop as few LO-tasks as possible at mode switch, we need to know how many resources are required to satisfy MC-schedulability. To do it, we develop an *online schedulability test* and drop LO-tasks by the test.

To construct such a test, we introduce *system state* that captures the dynamic system behavior at mode switch, including the task mode (execution state) of each task.

Definition 5.1 (System state). For a given task set τ, a system state S is defined as four tuple of disjoint sets: $S = (\tau_{H1}, \tau_{H2}, \tau_{L1}, \tau_{L2})$ where

- τ_{H1} : the LO-mode HI-task set ($\tau_{H1} = \{\tau_i \in \tau_H | M_i = \text{LO}\}$),
- τ_{H2}: the HI-mode HI-task set (including the mode switching task) ($\tau_{H2} = \tau_H \setminus \tau_{H1}$),
- τ_{L1}: the active LO-task set, and
- τ_{L2}: the dropped LO-task set (including the dropping LO-tasks at mode switch) ($\tau_{L2} = \tau_L \setminus \tau_{L1}$).

The initial system state is $S_0 = (\tau_H, \emptyset, \tau_L, \emptyset)$.

We present the EDF-AD task dropping algorithm as follows:

- Before system start, sort LO-tasks in decreasing order of their task utilization.
- Drop the LO-task with the highest utilization among the active LO-task set (τ_{L1}) until the dropped LO-task set (τ_{L2}) satisfies the online schedulability test (Eq. (3)). This greedy algorithm minimizes the total number of the dropping LO-tasks during the entire running time [4].

We present the online schedulability test to determine which LO-tasks should be dropped at mode-switch:

$$U_{L2}^L \geq \frac{U_{L1}^L + U_{H1}^L/x + U_{H2}^H + U_L^L - 1}{1 - x}. \tag{3}$$

Its correctness is presented in Sec. 5.2.

The runtime complexity of MC-ADAPT (including runtime scheduling policy and task dropping algorithm) is $O(n)$, a linear complexity. Runtime scheduling policy takes $O(1)$ and the task dropping algorithm takes $O(n)$ (identifying resource deficiency takes $O(n)$ and selecting the drop candidates of LO-tasks (sorting LO-tasks is done offline) takes $O(n)$). The required memory space for the system state is at most n bits.

5.2 Online Schedulability Analysis

We analyze online schedulability at a specific mode switch, which means whether a given task set is schedulable by EDF-AD when the system state at mode switch is given. Let S_k be the system state after k-th mode switch ($k \geq 1$). To find the collective resource demand on a given interval, we compute the resource demand of each task depending on its task mode (execution state).

We consider online schedulability on two different kinds of system states: the initial system state (S_0) and the system state (S_k) that is switched from any feasible system state (S_{k-1}). Since S_0 is the same as system LO-mode in EDF-VD, we can reuse the result of EDF-VD for online schedulability on S_0.

Lemma 5.2. *A task set τ is schedulable by EDF-AD on S_0 if*

$$U_L^L + \frac{U_H^L}{x} \leq 1. \tag{4}$$

Proof. It is immediate from Lemma 4.1. □

[1] The computation of VD coefficient is derived from Eq. (1) of Lemma 4.1.

[2] The VD coefficient is derived from the schedulability analysis for the initial system state (all HI-tasks are in LO-mode and all LO-tasks are active), which is identical for both EDF-VD and EDF-AD.

[3] If a task is dropped by the scheduler, then the currently-released job of the task is immediately stopped (not guaranteed to meet its deadline) and no further job of the task is released. If a task is active (not dropped), all jobs released by the task meet their deadlines.

[4] Dropping the LO-task with the highest utilization is global optimal solution under a series of mode switches

Next, consider online schedulability on S_k.

Theorem 5.3. *Consider a task set τ. Assume that the task set is schedulable with S_{k-1}. Let S_k be the system state transited from S_{k-1}. Then, τ is schedulable by EDF-AD on S_k if*

$$U_{L1}^L + \frac{U_{H1}^L}{x} + xU_{L2}^L + U_{H2}^H \le 1. \tag{5}$$

We can derive the online schedulability test (Eq. (3)) in Sec. 5.1 from Theorem 5.3. Eq. (5) is rewritten to

$$(U_L^L - U_{L2}^L) + \frac{U_{H1}^L}{x} + xU_{L2}^L + U_{H2}^H \le 1 \quad (\because U_L^L = U_{L1}^L + U_{L2}^L)$$
$$\Leftrightarrow U_L^L + \frac{U_{H1}^L}{x} + U_{H2}^H - 1 \le U_{L2}^L - xU_{L2}^L,$$

which is Eq. (3).

Now, we present the proof strategy for Theorem 5.3 and present auxiliary lemmas for the proof. We prove it by contradiction. Suppose that a deadline is missed. Let I denote a *minimal*[5] instance of jobs released by τ on which a deadline is missed by EDF-AD. Without loss of generality, we assume that the first job in I is released at time 0 and the deadline miss occurs at time t_1[6]. For task τ_i and any time t, let $\mathsf{DEM}_i(t)$ be an upper bound of the demand[7] of task τ_i over time interval $[0, t)$ in I. Let $\mathsf{DEM}(t)$ be the sum of $\mathsf{DEM}_i(t)$ of all the tasks in τ. Since a deadline is missed at t_1, we have $\mathsf{DEM}(t_1) > t_1$. We will show that our calculation of $\mathsf{DEM}(t_1)$ is no greater than t_1, which leads to a contradiction.

To find $\mathsf{DEM}(t_1)$, we consider individual task demand over $[0, t_1)$. The following lemma bounds the demand for a HI-task in LO-mode and the demand of a LO-task in the active state.

Lemma 5.4. *Consider any time t. (a) For a HI-task in LO-mode ($\tau_i \in \tau_{H1}$), $\mathsf{DEM}_i(t) = (u_i^L/x)t$, and (b) for a LO-task in the active state ($\tau_i \in \tau_{L1}$), $\mathsf{DEM}_i(t) = u_i^L \cdot t$.*

Proof. **(a)** The task demand over $[0, t)$ is smaller than or equal to $C_i^L \cdot t/(xT_i)$. Thus, $\mathsf{DEM}_i(t) = (u_i^L/x)t$.
(b) The task demand over $[0, t)$ is smaller than or equal to $C_i^L \cdot t/T_i$. Thus, $\mathsf{DEM}_i(t_1) = u_i^L \cdot t_1$. □

For the demand for a HI-task in HI-mode and a LO-task in the dropped state, we utilize a characteristic of the jobs that are included in $\mathsf{DEM}(t_1)$.

Lemma 5.5 (from [2]). *Consider the minimal instance I. All jobs that execute in $[0, t_1)$ have deadline $\le t_1$.*

Based on Lemma 5.5, we bound the demand of a HI-task in HI-mode. For a HI-task τ_i, let J_i^* be the mode-switching job of τ_i in I and a_i^* be the release time of J_i^*.

Lemma 5.6. *If HI-task τ_i is mode switched ($\tau_i \in \tau_{H2}$), then*

$$\mathsf{DEM}_i(t_1) = \begin{cases} (u_i^L/x)t_1 & \text{if } t_1 < a_i^* + xT_i, \\ u_i^L \cdot a_i^* + u_i^H(t_1 - a_i^*) & \text{otherwise.} \end{cases}$$

[5]Since I is minimal, EDF-AD can schedule any proper subset of I.
[6]All jobs in I are necessary to construct the deadline miss. Otherwise, the unnecessary job can be removed from I, which contradicts the minimality of I.
[7]The demand of a task for a time interval indicates the worst-case resource demand to meet deadlines of jobs released by the tasks for the time interval [4].

Proof. At mode switch of τ_i, the deadline of J_i^* is changed from $a_i^* + xT_i$ to $a_i^* + T_i$ and the execution requirement is changed from C_i^L to C_i^H. By Lemma 5.5, the changed demand is not considered when $t_1 < a_i^* + T_i$. The task demand over $[0, t_1)$ is different depending on whether time t_1 is before the VD of J_i^* or not.

Case ($t_1 < a_i^* + xT_i$)**.** Since $t_1 < a_i^* + xT_i \le a_i^* + T_i$, the task demand is the same as Lemma 5.4a.

Case ($t_1 \ge a_i^* + xT_i$)**.** We calculate the task demand of jobs before time a_i^* and the task demand after time a_i^*. Since jobs before time a_i^* execute for LO-WCET (C_i^L), the task demand of jobs before a_i^* is $C_i^L \cdot a_i^*/T_i$. Since job J_i^* and its successive jobs execute HI-WCET (C_i^H), the task demand of jobs after time a_i^* is $C_i^H(t_1 - a_i^*)/T_i$. Then, $\mathsf{DEM}_i(t_1) = C_i^L \cdot a_i^*/T_i + C_i^H(t_1 - a_i^*)/T_i$. □

The following lemma bounds the demand of a LO-task in the dropped state.

Lemma 5.7. *Let τ_q be the last (k-th) mode-switched task in I of S_k. If LO-task τ_i is dropped ($\tau_i \in \tau_{L2}$), then*

$$\mathsf{DEM}_i(t_1) = \begin{cases} u_i^L \cdot t_1 & \text{if } t_1 < a_q^* + xT_q, \\ u_i^L(a_q^* + xT_q) & \text{otherwise.} \end{cases}$$

Proof. Since τ_q is the last mode-switched task, τ_i is dropped before or at the mode switch of J_q^*. The mode switch of J_q^* happens before or at its VD ($a_q^* + xT_q$). The task demand of τ_i over $[0, t_1)$ is different depending on whether time t_1 is before the VD of J_q^* or not.

Case ($t_1 < a_q^* + xT_q$)**.** In the worst case, the mode switch happens at the VD of J_q^*. Then, the upper bound of the demand is the same as Lemma 5.4b.

Case ($t_1 \ge a_q^* + xT_q$)**.** No job of the task executes after the mode switch of J_q^*. To execute before the mode switch, jobs must have a deadline no greater than the VD of J_q^*. Then, $\mathsf{DEM}_i(t_1) = C_i^L(a_q^* + xT_q)/T_i$. □

In the rest of the paper, we denote the last (k-th) mode-switched task in I of S_k by τ_q.

We consider the task demand on S_k based on the task demand on S_{k-1}. We know that S_{k-1} is a feasible system state and S_k is the transited state from S_{k-1} by the mode switch of HI-task τ_q. Let $\mathsf{DEM}_i^k(t)$ be the demand of task τ_i over $[0, t)$ on the system state S_k. We compute $\mathsf{DEM}_i^k(t_1)$ based on $\mathsf{DEM}_i^{k-1}(t_1)$ for task τ_i depending on whether time t_1 is before the VD of J_q^* or not (Lemmas 5.8 and 5.9).

Lemma 5.8. *If $t_1 < a_q^* + xT_q$, then $\mathsf{DEM}_i^k(t_1) = \mathsf{DEM}_i^{k-1}(t_1)$.*

Proof. Task τ_q belongs to τ_{H2} when the system state is S_k and belongs to τ_{H1} when the system state is S_{k-1}. Since $\tau_q \in \tau_{H2}$ on S_k, we have $\mathsf{DEM}_i^k(t_1) = (u_i^L/x)t_1$ by Lemma 5.6. Since $\tau_q \in \tau_{H1}$ on S_{k-1}, we have $\mathsf{DEM}_i^{k-1}(t_1) = (u_i^L/x)t_1$ by Lemma 5.4a.

Consider task τ_i that is dropped by τ_q. The task belongs to τ_{L2} when the system state is S_k and belongs to τ_{L1} when the system state is S_{k-1}. Since $\tau_i \in \tau_{L2}$ on S_k, we have $\mathsf{DEM}_i^k(t_1) = u_i^L \cdot t_1$ by Lemma 5.7. Since $\tau_q \in \tau_{L1}$ on S_{k-1}, we have $\mathsf{DEM}_i^{k-1}(t_1) = u_i^L \cdot t_1$ by Lemma 5.4b.

Consider task $\tau_i \in \tau$ that is not t_q and not dropped by τ_q. Since its task mode (execution state) is not changed from S_{k-1} to S_k, we have $\mathsf{DEM}_i^k(t) = \mathsf{DEM}_i^{k-1}(t)$ for any t. □

Lemma 5.9. *If $t_1 \geq a_q^* + xT_q$, then*

$$\mathsf{DEM}_i^k(t_1) \leq \mathsf{DEM}_i^{k-1}(a_q^*) + (t_1 - a_q^*) \begin{cases} u_i^L/x & \text{if } \tau_i \in \tau_{H1}, \\ u_i^L & \text{if } \tau_i \in \tau_{L1}, \\ u_i^H & \text{if } \tau_i \in \tau_{H2}, \\ x \cdot u_i^L & \text{if } \tau_i \in \tau_{L2}. \end{cases}$$

Proof. For $\tau_i \in \tau_{H1}$, we have $\mathsf{DEM}_i^k(t_1) = \mathsf{DEM}_i^{k-1}(a_q^*) + (t_1 - a_q^*)u_i^L/x$ by Lemma 5.4a. For $\tau_i \in \tau_{L1}$, we have $\mathsf{DEM}_i^k(t_1) = \mathsf{DEM}_i^{k-1}(a_q^*) + (t_1 - a_q^*)u_i^L$ by Lemma 5.4b.

Consider $\tau_i \in \tau_{H2}$. If $a_i^* \leq a_q^*$, we have $\mathsf{DEM}_i^k(t_1) = \mathsf{DEM}_i^{k-1}(a_q^*) + (t - a_q^*)u_i^H$ by Lemma 5.6. Otherwise, we have

$$\begin{aligned} \mathsf{DEM}_i^k(t_1) =& \mathsf{DEM}_i^{k-1}(a_q^*) + (a_i^* - a_q^*)u_i^L + (t_1 - a_i^*)u_i^H \\ & \text{(by Lemma 5.6)} \\ \leq& \mathsf{DEM}_i^{k-1}(a_q^*) + (t_1 - a_q^*)u_i^H. \end{aligned}$$

Consider $\tau_i \in \tau_{L2}$. By Lemma 5.5, the deadline of J_q^* is no greater than t_1: $a_q^* + T_q \leq t_1$. By Lemma 5.7, we have

$$\begin{aligned} \mathsf{DEM}_i^k(t_1) =& (a_q^* + xT_q)u_i^L \\ \leq& \mathsf{DEM}_i^{k-1}(a_q^*) + x(t_1 - a_q^*)u_i^L \qquad (\because T_q \leq t_1 - a_q^*). \end{aligned}$$

□

Based on the relation between the demand on S_k and the demand on S_{k-1}, we now prove Theorem 5.3.

Proof of Theorem 5.3. We summarize the proof strategy stated before. Let $\mathsf{DEM}^k(t)$ be the sum of $\mathsf{DEM}_i^k(t)$ of all the tasks in τ. Since S_{k-1} is a feasible system state, we have $\mathsf{DEM}^{k-1}(t) \leq t$ for any t. By proof by contradiction, we assume a deadline misses in I. Then, we have $\mathsf{DEM}^k(t_1) > t_1$. To lead to a contradiction, we only need to show that $\mathsf{DEM}^k(t_1) \leq t_1$. Let τ_q be the last mode-switched task in I. $\mathsf{DEM}^k(t_1)$ is different depending on whether t_1 is before the VD of J_q^* or not.

Case 1 ($t_1 < a_q^* + xT_q$)**.** We calculate $\mathsf{DEM}^k(t_1)$:

$$\begin{aligned} \mathsf{DEM}^k(t_1) &= \sum_{\tau_i \in \tau} \mathsf{DEM}_i^k(t_1) \\ &= \sum_{\tau_i \in \tau} \mathsf{DEM}_i^{k-1}(t_1) \qquad \textit{(by Lemma 5.8)} \\ &= \mathsf{DEM}^{k-1}(t_1), \end{aligned}$$

which is smaller than or equal to 1 by the assumption on DEM^{k-1}.

Task	χ_i	u_i^L	u_i^H
τ_1	HI	0.10	0.35
τ_2	HI	0.20	0.30
τ_3	LO	0.18	N/A
τ_4	LO	0.12	N/A
τ_5	LO	0.10	N/A

Table 1: The Parameters of an Example Task Set

Case 2 ($t_1 \geq a_q^* + xT_q$)**.** We calculate $\mathsf{DEM}^k(t_1)$:

$$\begin{aligned} \mathsf{DEM}^k(t_1) &= \sum_{\tau_i \in \tau} \mathsf{DEM}_i^k(t_1) \\ &\leq \sum_{\tau_i \in \tau} \mathsf{DEM}_i^{k-1}(a_q^*) + (t_1 - a_q^*)(U_{L1}^L + \frac{U_{H1}^L}{x} \\ &\quad + xU_{L2}^L + U_{H2}^H) \qquad \textit{(by Lemma 5.9)} \\ &\leq \mathsf{DEM}^{k-1}(a_q^*) + (t_1 - a_q^*) \quad \textit{(Eq. (5) with } S_k\textit{)} \\ &\leq a_q^* + (t_1 - a_q^*) \quad \textit{(by the assumption on } \mathsf{DEM}^{k-1}\textit{)} \\ &= t_1. \end{aligned}$$

From Cases 1 and 2, we showed that $\mathsf{DEM}^k(t_1) \leq t_1$. □

5.3 Offline Schedulability Analysis

We looked at online schedulability analysis at a mode switch. However, we do not yet know whether a task set is schedulable by EDF-AD under any sequences of mode switches, which is offline schedulability. To know it, we need to check whether EDF-AD can schedule the task set on any system state satisfying MC-schedulability (Condition A and B in Sec. 3), based on the online schedulability analysis (Lemma 5.2 and Theorem 5.3).

Theorem 5.10. *A task set τ is MC-schedulable by EDF-AD if*

$$U_L^L + \frac{U_H^L}{x} \leq 1, \tag{6}$$

$$xU_L^L + \sum_{\tau_i \in \tau_H} \max(\frac{u_i^L}{x}, u_i^H) \leq 1. \tag{7}$$

Proof. To show that τ is MC-schedulable, we need to satisfy both Conditions A and B. Since Eq. (6) holds, by Lemma 5.2, τ is schedulable on S_0, which satisfies Condition B. For Condition A, by Theorem 5.3, we show that Eq. (5) holds with any $\tau_{H2} \neq \emptyset$: since each HI-task is either LO-mode or HI-mode, and all LO-tasks may be dropped in the worst case,

$$xU_L^L + \sum_{\tau_i \in \tau_H} \max(\frac{u_i^L}{x}, u_i^H) \leq 1,$$

which holds from Eq. (7). □

We show an example task set schedulable by EDF-AD.

Example 5.11. Consider the example task set in Table 1. According to the VD assignment, we have $x = 0.3/(1 - 0.4) = 0.5$. We show that Eqs. (6) and (7) hold: $0.4 + 0.3/0.5 = 1$ and $0.5 * 0.4 + \max(0.1/0.5, 0.35) + \max(0.2/0.5, 0.30) = 0.2 + 0.35 + 0.4 = 0.95 \leq 1$. Then, the task set is MC-schedulable by Theorem 5.10.

6 AN ENHANCED MC-ADAPT FRAMEWORK

A straightforward extension of EDF-VD, which is EDF-AD, yields a counter-intuitive result that it does not dominate EDF-VD in schedulability. In Sec. 6.1, we find the characteristics of the subset of tasks that cause the schedulability loss. In Sec. 6.2, we present a new scheduling algorithm that isolates them from the other tasks.

6.1 The Schedulability Loss of EDF-AD

Let's look at an example schedulable by EDF-VD but not by EDF-AD.

Example 6.1. We modify the task set in Table 1 by changing u_1^H of τ_1 to 0.45. Since U_L^L and U_H^L are not changed, x will not be changed. The task set is schedulable by EDF-VD because Lemmas 4.1 and 4.2 hold: $0.4 + 0.3/0.5 = 1$ and $0.5 \cdot 0.4 + 0.75 = 0.95 \leq 1$. However, the task set is not schedulable by EDF-AD because Eq. (7) in Theorem 5.10 does not hold: $0.5 \cdot 0.4 + \max(0.1/0.5, 0.45) + \max(0.2/0.5, 0.30) = 1.05 > 1$.

We investigate which difference between EDF-VD and EDF-AD causes the schedulability loss. When checking Condition B in MC-schedulability, both EDF-VD and EDF-AD consider the initial system state S_0. Thus, the loss is related to check Condition A. We define *the critical task mode* as the combination of task modes for HI-tasks where the collective resource demand of HI-tasks is maximized. While the critical task mode for EDF-VD is system HI-mode (among system HI-mode and system LO-mode), the one for EDF-AD is not system HI-mode (all HI-tasks are in HI-mode). Since a HI-task executes with its VD in LO-mode, there may exist a HI-tasks whose resource utilization in LO-mode (C_i^L/V_i) is higher than the one in HI-mode (C_i^H/T_i). Thus, the critical task mode is the combination of the task mode of each HI-task where its resource utilization is maximized. However, EDF-VD adopting the system-level mode switch (from system LO-mode to system HI-mode) is irrelevant to the critical task mode of EDF-AD.

We investigate why some HI-task has higher resource utilization in LO-mode. All HI-tasks execute with their VD ($V_i = xT_i$) in their LO-mode and the VD coefficient is derived from the collective utilization of the task set such that $U_H^L/x \leq U_H^H$. However, the VD assignment may not be the best choice for an individual task, specially for the HI-task that has relatively small difference between HI-WCET and LO-WCET. Then, the task has $u_i^L/x > u_i^H$. From the understanding of the schedulability loss, we will present a resolution in the next subsection.

6.2 The EDF-AD-E Scheduling Algorithm

Since a fully-independent task-level mode switch may produce the schedulability loss, we apply a limited mode switch not to sacrifice schedulability. We propose another scheduling algorithm, called *EDF-AD-E*.

Scheduling Algorithm. We formally define the subset of HI-tasks that produces the schedulability loss.

Definition 6.2. HI-mode-preferred tasks (τ_F) are defined as a set of HI-tasks s.t. $C_i^L/V_i > C_i^H/T_i$: $\tau_F = \{\tau_i \in \tau_H \mid u_i^L/x > u_i^H\}$

The schedulability loss may happen when HI-mode-preferred tasks remain in LO-mode and the other tasks have mode-switched. In addition, since a HI-mode-preferred task has resource utilization in HI-mode lower than the one in LO-mode, it is better for the task to execute in HI-mode from system start. We now present the EDF-AD-E (Enhanced) algorithm as follows:

- The VD of each HI-task τ_i is assigned by $V_i = xT_i$ where $x = \min(1, (1 - U_H^H)/U_L^L)$.
- For HI-mode-preferred tasks, execute them in HI-mode from system start.
- All the other runtime scheduling policy (including task dropping algorithm) is the same as EDF-AD.

We cannot use the VD coefficient in EDF-AD because the offline schedulability of EDF-AD-E is different from EDF-AD. We compute the VD coefficient from EDF-AD-E offline schedulability (Theorem 6.5). The initial system state of EDF-AD-E is different from EDF-AD: $S_0 = (\tau_H \setminus \tau_F, \tau_F, \tau_L, \emptyset)$.

Online Schedulability Analysis. Since the EDF-AD-E scheduling algorithm is modified from EDF-AD, we need to check whether online schedulability analysis of EDF-AD is also applicable to EDF-AD-E. Since S_0 is different from EDF-AD, we re-derive online schedulability on S_0.

Lemma 6.3. *A task set τ is schedulable by EDF-AD-E on S_0 if*

$$U_L^L + \sum_{\tau_i \in \tau_H} \min(\frac{u_i^L}{x}, u_i^H) \leq 1. \tag{8}$$

Proof. On S_0, we have $\tau_{H1} = \tau_H \setminus \tau_F$, $\tau_{H2} = \tau_F$ and $\tau_{L1} = \tau_L$. Since the demand of task $\tau_i \in \tau_F$ over $[0, t)$ is no greater than $C_i^H \cdot t/T_i$, we have

$$\tau_i \in \tau_F, \mathsf{DEM_i}(t) = u_i^H \cdot t. \tag{9}$$

We show that the demand over $[0, t)$ is no greater than t:

$$\begin{aligned}
&\mathsf{DEM}(t_1) \\
&= \sum_{\tau_i \in \tau_H \setminus \tau_F} \mathsf{DEM_i}(t) + \sum_{\tau_i \in \tau_L} \mathsf{DEM_i}(t) + \sum_{\tau_i \in \tau_F} \mathsf{DEM_i}(t) \\
&= (\sum_{\tau_i \in \tau_H \setminus \tau_F} u_i^L/x + \sum_{\tau_i \in \tau_{L1}} u_i^L)t + \sum_{\tau_i \in \tau_F} \mathsf{DEM_i}(t) \\
&\qquad\qquad \textit{(by Lemma 5.4)} \\
&= (\sum_{\tau_i \in \tau_H \setminus \tau_F} u_i^L/x + U_L^L)t + \sum_{\tau_i \in \tau_F} u_i^H \cdot t \qquad \textit{(by Eq. (9))} \\
&= (U_L^L + \sum_{\tau_i \in \tau_H \setminus \tau_F} u_i^L/x + \sum_{\tau_i \in \tau_F} u_i^H)t \\
&\leq 1 \cdot t \qquad \textit{(by assumption)}
\end{aligned}$$

□

Online schedulability on S_k is the same as EDF-AD.

Theorem 6.4. *Consider a task set τ. Assume that the task set is schedulable with S_{k-1}. Let S_k be the system state that is transited from S_{k-1}. Then, the task set is schedulable by EDF-AD-E on S_k if Eq. (5) holds.*

Proof. The proof is the same as Theorem 5.3. □

Offline Schedulability Analysis. The following theorem derives the offline schedulability of EDF-AD-E.

Theorem 6.5. *A task set τ is MC-schedulable by EDF-AD-E if*

$$U_L^L + \sum_{\tau_i \in \tau_H} \min(\frac{u_i^L}{x}, u_i^H) \leq 1, \tag{10}$$

$$xU_L^L + U_H^H \leq 1. \tag{11}$$

Proof. To show that τ is MC-schedulable, we need to satisfy both Conditions A and B in MC-schedulability. Since Eq. (10) holds, by Lemma 6.3, τ is schedulable on S_0, which satisfies Condition B. For Condition A, by Theorem 6.4, we show that Eq. (5) holds with any $\tau_{H2} \neq \emptyset$: since each HI-task except HI-mode-preferred tasks is LO-mode or HI-mode, and all LO-tasks may be dropped in the worst case,

$$\begin{aligned} & xU_L^L + \sum_{\tau_i \in \tau_H \setminus \tau_F} \max(\frac{u_i^L}{x}, u_i^H) + \sum_{\tau_i \in \tau_F} u_i^H \leq 1 \\ \Leftrightarrow\ & xU_L^L + \sum_{\tau_i \in \tau_H \setminus \tau_F} u_i^H + U_F^H \leq 1 \qquad \textit{(by Def. 6.2)} \\ \Leftrightarrow\ & xU_L^L + U_H^H - U_F^H + U_F^H \leq 1, \end{aligned}$$

which holds from Eq. (11). □

Properties. EDF-AD-E strictly dominates EDF-VD in terms of MC-schedulability (Lemma 6.6 and Example 6.7).

Lemma 6.6. *If any task set is MC-schedulable by EDF-VD, the task set is also MC-schedulable by EDF-AD-E.*

Proof. Since the task set is MC-schedulable by EDF-VD, by Lemmas 4.1 and 4.2, Eqs. (1) and (2) hold. If Eqs. (10) and (11) holds, by Theorem 6.5, the task set is also MC-schedulable by EDF-AD-E. Eq. (10) holds: $U_L^L + \sum_{\tau_i \in \tau_H} \min(\frac{u_i^L}{x}, u_i^H) \leq U_L^L + \frac{U_H^L}{x} \leq 1$ from Eq. (1). Eq. (11) holds from Eq. (2). □

Example 6.7. We modify the task set in Table 1 by changing u_1^H of τ_1 to 0.55. We will schedule the task set by EDF-VD. Since U_L^L and U_H^L are not changed, x will not be changed. The task set is not schedulable by EDF-VD because Eq. (2) in Lemma 4.2 does not hold: $0.5{*}0.4{+}0.85 = 1.05 > 1$. We will schedule the task set by EDF-AD-E. By the VD assignment, we have $x = (1{-}0.85)/0.4 = 0.375$. Task τ_2 is a HI-mode-preferred task because $u_i^L/x = 0.2/0.375 = 0.53 > u_i^H = 0.30$. We show that Eqs. (10) and (11) hold: $0.4 + 0.1/0.375 + 0.3 = 0.96 \leq 1$ and $0.375 * 0.4 + 0.85 = 1$. Then, the task set is MC-schedulable by Theorem 6.5,

7 THE SPEEDUP FACTOR

In this section, we quantify the effectiveness of EDF-AD-E based on the metric of processor speedup factor [16]. The speedup factor ($\alpha \in \mathbb{R}$ s.t. $\alpha \geq 1$) is a reliable performance metric for comparing the worst-case behavior of different algorithms for solving the same problem. The smaller speedup factor of an algorithm indicates that the behavior of the algorithm is closer to that of the optimal algorithm. Previously, the speedup factor for the MC scheduling problem is effective to evaluate MC scheduling algorithms (e.g., [2]). However, it only evaluates MC-schedulability, and cannot evaluate the quality of task dropping. So, we propose the speedup factor for the task dropping problem which extends the existing MC scheduling problem with the runtime performance of LO-tasks. First, we define the task dropping problem.

Definition 7.1 (the task dropping problem). For a given feasible MC task set τ and a given subset of HI-tasks ($\tau_R \in \tau$), consider a task dropping problem for scheduling algorithm $\mathbb{A}$: if tasks in τ_R mode-switch on runtime, how many LO-tasks are required to be dropped for scheduling τ by scheduling algorithm $\mathbb{A}$?

Now, we define the speedup factor of scheduling algorithm $\mathbb{A}$ for the task dropping problem (Def. 7.1). Let OPT be the optimal clairvoyant scheduling algorithm with optimal task dropping[8]. The speedup factor of $\mathbb{A}$ for task drop is defined as the smallest real number α (≥ 1) such that the number of LO-tasks required to be dropped to schedule any given task set τ under any given mode switch sequence (specified by $\tau_R \in \tau$) by OPT on a speed-1 processor is the same as the one to schedule τ under the mode switch sequence by $\mathbb{A}$ on a speed-α processor.

The speedup factor for the MC scheduling problem evaluates scheduling algorithms in terms of MC-schedulability. Similarly, the speedup factor for the task dropping problem evaluates scheduling algorithms in terms of the number of the required task dropping under any possible scheduling scenarios. Although the existing work cannot provide any performance guarantee on LO-tasks via the speedup factor, we propose the first work to evaluate how many LO-tasks can be scheduled after mode switch via the speedup factor.

Next, we evaluate EDF-AD-E via the proposed metric.

Theorem 7.2. *EDF-AD-E has a speedup factor of $\frac{1+\sqrt{5}}{2}$ for task drop.*

To prove Theorem 7.2, we present an auxiliary lemma.

Lemma 7.3. *Consider a task set τ and a subset of HI-tasks $\tau_R \in \tau$. Consider a scheduling scenario that tasks in τ_R mode-switches on runtime. EDF-AD-E can schedule τ under the scenario by dropping a subset of LO-tasks $\tau_G \in \tau$ if*

$$U_L^L + \frac{U_H^L}{x} \leq 1, \tag{12}$$

$$U_{L1}^L + \frac{U_{H1}^L}{x} + xU_{L2}^L + U_{H2}^H \leq 1 \tag{13}$$

where $\tau_{H2} = \tau_R$ and $\tau_{L2} = \tau_G$.

Proof. We need to show that τ is schedulable on S_0 and any legitimate S_k considering τ_R and τ_G. For schedulability on S_0, by Lemma 6.3, we need to satisfy Eq. (8):

$$U_L^L + \sum_{\tau_i \in \tau_H} \min(\frac{u_i^L}{x}, u_i^H) \leq 1,$$

which holds by $\sum_{\tau_i \in \tau_H} \min(\frac{u_i^L}{x}, u_i^H) \leq U_H^L$ and Eq. (12).

Consider any legitimate S_k considering τ_R and τ_G. Since each task in τ_R is either HI-mode or LO-mode, τ_{H2} in S_k is any subset of τ_R. In S_k, we set $\tau_{L2} := \tau_G$ because any LO-task in τ_G may be dropped. To show that the task set is schedulable with S_k, by

[8] In MC systems, a clairvoyant scheme is the one that knows the time instant of mode switch before runtime scheduling.

Theorem 6.4, we need to satisfy Eq. (5):

$$U_{L1}^L + xU_{L2}^L + \sum_{\tau_i \in \tau_{H1} \setminus \tau_F} \frac{u_i^L}{x} + \sum_{\tau_i \in \tau_{H2} \setminus \tau_F} \max(\frac{u_i^L}{x}, u_i^H) + \sum_{\tau_i \in \tau_F} u_i^H \leq 1$$

$$\Leftrightarrow U_{L1}^L + xU_{L2}^L + \sum_{\tau_i \in \tau_{H1}} \min(\frac{u_i^L}{x}, u_i^H) + U_{H2}^H \leq 1,$$

which holds by $\sum_{\tau_i \in \tau_{H1}} \min(\frac{u_i^L}{x}, u_i^H) \leq U_{H1}^H$ and Eq. (13). □

Proof of Theorem 7.2. Consider a task set τ. Consider a scheduling scenario that tasks in a set of HI-tasks $\tau_R \in \tau$ mode-switch on runtime. We will prove that if τ is schedulable under the scenario by the optimal clairvoyant scheduling algorithm with dropping a set of LO-tasks $\tau_G \in \tau$ on a speed-1 processor, τ is also schedulable under the scenario by EDF-AD-E with dropping τ_G on a speed-$\frac{1+\sqrt{5}}{2}$ processor.

Let $\tau_{H1} := \tau_H \setminus \tau_R$, $\tau_{H2} := \tau_R$, $\tau_{L1} := \tau_L \setminus \tau_G$, and $\tau_{L2} := \tau_G$. Let b denote an upper bound on the utilization of τ on the initial state and the minimum utilization of τ on the worst-case task modes of HI-tasks:

$$\max(U_L^L + U_H^L, U_{L1}^L + U_{H1}^L + U_{H2}^H) \leq b. \quad (14)$$

To show that the task set is schedulable, by Lemma 7.3, it is required that Eqs. (12) and (13) hold. Suppose that there exists α s.t. $1/x \leq \alpha$ and $1 + x \leq \alpha$ for some $x \in \mathbb{R}$ s.t. $0 < x \leq 1$. We show that Eq. (12) holds:

$$U_L^L + \frac{U_H^L}{x} \leq U_L^L + \alpha U_H^L \leq \alpha(U_L^L + U_H^L),$$

which is smaller than or equal to 1 if $U_L^L + U_H^L \leq 1/\alpha$.

We show that Eq. (13) holds. We divide cases depending on whether $U_{H2}^H - U_{H2}^L \leq U_{L2}^L$ or not. When $U_{H2}^H - U_{H2}^L \leq U_{L2}^L$, we show that Eq. (13) holds:

$$\begin{aligned} &U_{L1}^L + \frac{U_{H1}^L}{x} + xU_{L2}^L + U_{H2}^H \\ &\leq U_{L1}^L + \alpha U_{H1}^L + (1+x)U_{L2}^L + U_{H2}^L \\ &\leq \alpha(U_L^L + U_H^L), \end{aligned}$$

which is smaller than or equal to 1 if $U_L^L + U_H^L \leq 1/\alpha$. When $U_{L2}^L < U_{H2}^H - U_{H2}^L$, we show that Eq. (13) holds:

$$\begin{aligned} &U_{L1}^L + \frac{U_{H1}^L}{x} + xU_{L2}^L + U_{H2}^H \\ &\leq U_{L1}^L + \alpha U_{H1}^L + U_{H2}^L + (1+x)(U_{H2}^H - U_{H2}^L) \\ &\leq \alpha(U_{L1}^L + U_{H1}^L + U_{H2}^H), \end{aligned}$$

which is smaller than or equal to 1 if $U_{L1}^L + U_{H1}^L + U_{H2}^H \leq 1/\alpha$. In sum, Eqs. (12) and (13) hold if $b \leq 1/\alpha$.

We need to find the range of α. To do it, we show the existence of x satisfying both of $1/\alpha \leq x$ and $x \leq \alpha - 1$:

$$1/\alpha \leq \alpha - 1 \Leftrightarrow \alpha^2 - \alpha - 1 \geq 0,$$

which is always true if $\alpha \geq \frac{1+\sqrt{5}}{2}$. □

8 EVALUATION

We looked at the speedup factor of the EDF-AD-E for task drop, which provides a theoretical upper bound on the number of LO-tasks to be dropped for a task set and its runtime scenario. In this section, we evaluate the runtime performance of EDF-AD-E in comparison with the existing approaches, via simulation with synthetic workloads. In addition, to show that our approach does not sacrifice schedulability, we compare EDF-AD-E with the existing approaches in terms of MC-schedulability.

Task Set Generation. We generate random task sets according to the workload-generation algorithm [2]. Let U^b be the upper bound of both LO-criticality and HI-criticality utilizations. A random task is generated as follows (all task parameters are randomly drawn in uniform distribution): for a task τ_i,

- U_i (task utilization) is a real number drawn from the range $[0.02, 0.2]$.
- T_i (task period) is an integer drawn from the range $[20, 300]$.
- R_i (the ratio of u_i^H/u_i^L) is a real number drawn from the range $[1, 4]$.
- P_i (the probability that the task is a HI-task) is a real number from the range [0,1]. If $P_i < P^{\mathsf{HI}}$ (default value of P^{HI} is 0.5), set $\chi_i := LO$ and $C_i^L := \lfloor U_i \cdot T_i \rfloor$. Otherwise, set $\chi_i := HI$, $C_i^H := \lfloor U_i \cdot T_i \rfloor$, and $C_i^L := \lfloor U_i \cdot T_i / R_i \rfloor$.

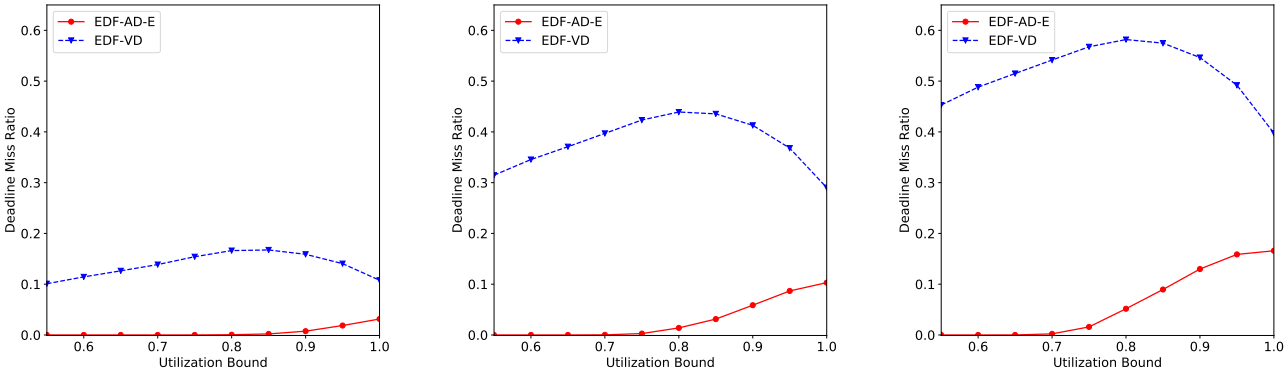
Repeat generating a task in the task set until $\max(U_H^L + U_L^L,\ U_H^H) > U^b$. Then, discard the task added last.

The Deadline Miss Ratio. We compare EDF-AD-E with EDF-VD [2] in terms of deadline miss ratio (DMR)[9] of LO-tasks. For a given randomly-generated task set schedulable by EDF-VD, we simulate the behavior of tasks with a given probability of mode switch for any HI-task, denoted as P^{MS} (the default value of P^{MS} is 0.4), for 10,000 time units. According to EDF-VD [2], on idle tick, the system is switched back to the initial state (all HI-tasks are in LO-mode and all LO-tasks are active).

Fig. 2 shows the average DMR with varying utilization bound U^b for different probability of mode switch: $P^{MS} = 0.1$, $P^{MS} = 0.4$ and $P^{MS} = 0.7$. For each utilization bound, we generate 5,000 systems. The result shows that EDF-AD-E significantly outperforms EDF-VD because the resource-efficient scheduling of EDF-AD-E minimizes the additional resource request at mode switch and the EDF-AD-E task dropping algorithm selects the minimal set of LO-tasks upon the resource request.

In higher utilization ($U^b > 0.85$), the DMR of EDF-VD is decreasing. We observed that the schedulable system tends to have a smaller number of HI-tasks in higher utilization. To pass the schedulability condition in higher utilization, the system should have unbalanced distribution of HI-tasks and LO-tasks (either relatively a larger number of HI-tasks or relatively a larger number of LO-tasks). Due to the selection of R_i, relatively large numbers of HI-tasks may not pass the schedulability test. Thus, the system tends to have relatively small numbers of HI-tasks, which affects the frequency of mode-switches. As the number of mode switches decreases, the DMR of EDF-VD (dropping all LO-tasks) decreases. Fig. 3 shows DMR varying P^{HI} from 0.05 to 0.95 in steps of 0.5 ($U^b = 0.8$). EDF-VD shows a higher DMR for a higher P^{HI} (a large

[9] DMR is the ratio of the number of the unfinished jobs over total number of job release for a given time interval. We assume that a LO-task in the dropped state releases its job but it does not execute.



(a) $P^{MS} = 0.1$ (b) $P^{MS} = 0.4$ (c) $P^{MS} = 0.7$

Figure 2: The DMR for Different P^{MS}

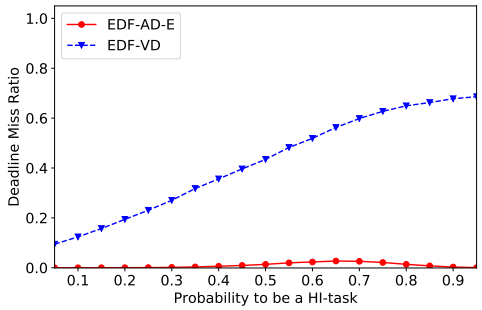


Figure 3: The DMR for Different P^{HI}

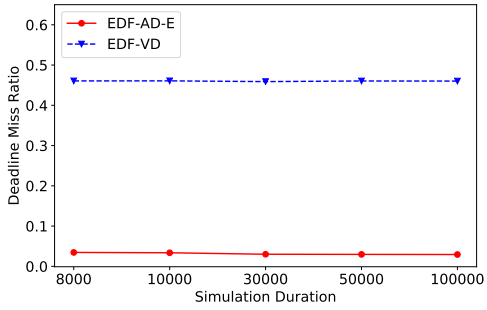


Figure 4: The DMR for Different Simulation Duration

number of mode switches at runtime) while EDF-AD-E shows a little variance for different values of P^{HI}.

Fig. 4 shows DMR varying simulation durations ($U^b = 0.85$ and $P^{MS} = 0.4$). It shows that the simulation duration does not affect the simulation results. Due to the return protocol to LO-mode, the DMR of EDF-VD and EDF-AD-E are converged in a large simulation duration.

MC-schedulability. We compare the MC-schedulability of EDF-AD and EDF-AD-E with the existing MC scheduling algorithms,

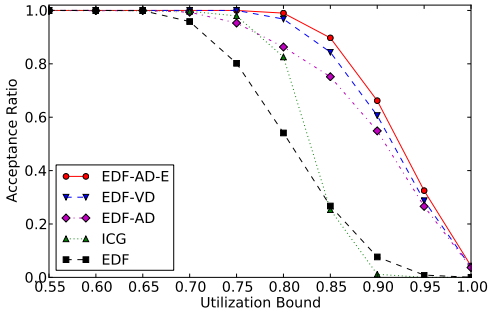


Figure 5: MC-schedulability Varying Utilization Bound

which are regular EDF, EDF-VD [2], and ICG [14][10]. We mathematically compute the schedulability of the randomly-generated systems via schedulability test of each scheduling algorithms.

Fig. 5 shows the acceptance ratio (the ratio of schedulable task sets) over varying utilization bound U^b from 0.55 to 1.0 in steps of 0.05. Each data point is based on 5,000 systems. Although EDF-AD has higher acceptance ratio than regular EDF, we confirmed that EDF-AD has lower MC-schedulability than EDF-VD for all utilization range and ICG for some utilization range. Our goal is to improve runtime performance of LO-tasks without sacrifice in schedulability. We already showed that EDF-AD-E strictly dominates EDF-VD in Sec. 6.2. The simulation result also confirmed that EDF-AD-E dominates EDF-VD.

9 CONCLUSION

We present the MC-ADAPT framework that makes online adaptive task dropping utilizing the dynamic system state under task-level mode switch. The framework focuses resource efficiency of mixed-criticality systems by improving the survivality of low-criticality tasks even under deviance of high-criticality tasks. To evaluate the quality of task dropping, we propose the speedup factor for

[10]The MC-schedulability of the ICG is maximum when the interference constraint graph is fully connected from HI-tasks to LO-tasks (which means that any mode switch drop all LO-tasks).

the task dropping problem while the speedup factor for the MC scheduling problem only evaluates MC scheduling algorithms in terms of the worst-case schedulability. We derive that the speedup factor of MC-ADAPT for task drop is 1.619.

As a future work, we would like to extend our framework to support HI-mode HI-tasks to resume into LO-mode under task-level criticality mode. It will be interesting to compare our new scheduling policy with bailout protocol [5], system-level resuming protocol. Orthogonally, we will include the runtime scheduling overheads (computation and memory space) in MC-ADAPT schedulability analysis. Finally, we plan to conduct more comprehensive experiments with other flexible MC scheduling approaches [12, 14, 19] and real-world case studies including runtime overheads of MC-ADAPT algorithms.

REFERENCES

[1] AUTOSAR. 2005. AUTomotive Open System ARchitecture. www.autosar.org. (2005).

[2] S. Baruah, V. Bonifaci, G. D"Angelo, H. Li, A. Marchetti-Spaccamela, S. Van der Ster, and L. Stougie. 2012. The Preemptive Uniprocessor Scheduling of Mixed-Criticality Implicit-Deadline Sporadic Task Systems. In *Euromicro Conference on Real-Time Systems (ECRTS)*. 145–154.

[3] S.K. Baruah, A. Burns, and R.I. Davis. 2011. Response-Time Analysis for Mixed Criticality Systems. In *Real Time System Symposium (RTSS)*. 34–43.

[4] Sanjoy K. Baruah, Louis E. Rosier, and Rodney R. Howell. 1990. Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic, Real-time Tasks on One Processor. *Real-Time Systems* 2, 4 (1990), 301–324.

[5] I. Bate, A. Burns, and R. I. Davis. 2015. A Bailout Protocol for Mixed Criticality Systems. In *Euromicro Conference on Real-Time Systems (ECRTS)*. 259–268.

[6] A. Burns and S. Baruah. 2013. Towards A More Practical Model for Mixed Criticality Systems. In *Workshop of Mixed Criticality Systems (WMC)*.

[7] Alan Burns and Robert Davis. 2016. Mixed Criticality Systems – A Review. http://www-users.cs.york.ac.uk/burns/review.pdf. (2016). the seventh edition.

[8] A. Easwaran. 2013. Demand-Based Scheduling of Mixed-Criticality Sporadic Tasks on One Processor. In *Real Time System Symposium (RTSS)*. 78–87.

[9] P. Ekberg and Wang Yi. 2012. Bounding and Shaping the Demand of Mixed-Criticality Sporadic Tasks. In *Euromicro Conference on Real-Time Systems (ECRTS)*. 135–144.

[10] Oliver Gettings, Sophie Quinton, and Robert I. Davis. 2015. Mixed Criticality Systems with Weakly-hard Constraints. In *Real-Time Networks and Systems (RTNS)*. 237–246.

[11] X. Gu and A. Easwaran. 2016. Dynamic Budget Management with Service Guarantees for Mixed-Criticality Systems. In *Real Time System Symposium (RTSS)*. 47–56.

[12] Xiaozhe Gu, A. Easwaran, Kieu-My Phan, and Insik Shin. 2015. Resource Efficient Isolation Mechanisms in Mixed-Criticality Scheduling. In *Euromicro Conference on Real-Time Systems (ECRTS)*. 13–24.

[13] Nan Guan, Pontus Ekberg, Martin Stigge, and Wang Yi. 2011. Effective and Efficient Scheduling of Certifiable Mixed-Criticality Sporadic Task Systems. In *Real Time System Symposium (RTSS)*. 13–23.

[14] P. Huang, P. Kumar, N. Stoimenov, and L. Thiele. 2013. Interference Constraint Graph - A new specification for mixed-criticality systems. In *Emerging Technologies and Factory Automation (ETFA)*. 1–8.

[15] Mathieu Jan, Lilia Zaourar, and Maurice Pitel. 2013. Maximizing the execution rate of low-criticality tasks in mixed criticality systems. In *Workshop of Mixed Criticality Systems (WMC)*.

[16] Bala Kalyanasundaram and Kirk Pruhs. 2000. Speed is As Powerful As Clairvoyance. *J. ACM* 47, 4 (2000), 617–643.

[17] 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 *Real Time System Symposium (RTSS)*. 35–46.

[18] P. J. Prisaznuk. 1992. Integrated modular avionics. In *National Aerospace and Electronics Conference (NAECON)*. 39–45.

[19] J. Ren and Linh Thi Xuan Phan. 2015. Mixed-Criticality Scheduling on Multiprocessors Using Task Grouping. In *Euromicro Conference on Real-Time Systems (ECRTS)*. 25–34.

[20] F. Santy, L. George, P. Thierry, and J. Goossens. 2012. Relaxing Mixed-Criticality Scheduling Strictness for Task Sets Scheduled with FP. In *Euromicro Conference on Real-Time Systems (ECRTS)*. 155–165.

[21] Hang Su and Dakai Zhu. 2013. An Elastic Mixed-Criticality task model and its scheduling algorithm. In *Design, Automation, and Test in Europe (DATE)*. 147–152.

[22] S. Vestal. 2007. Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution Time Assurance. In *Real Time System Symposium (RTSS)*. 239–243.