Future Generation Computer Systems 160 (2024) 406–419

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

IMC-PnG: Maximizing runtime performance and timing guarantee for imprecise mixed-criticality real-time scheduling

Jaewoo Lee [a], Jinkyu Lee [b,*]

[a] Department of Industrial Security, Chung-Ang University, Republic of Korea
[b] Department of Computer Science and Engineering, Sungkyunkwan University (SKKU), Republic of Korea

ARTICLE INFO

Keywords:
Mixed-Criticality Systems
Real-time Systems
Imprecise computing
Scheduling algorithm
Schedulability analysis

ABSTRACT

Mixed-Criticality (MC) systems have successfully overcome the limitation of traditional real-time systems based on pessimistic Worst-Case Execution Times (WCETs), by using different WCETs depending on different criticalities. One of the important yet unsolved problems of current MC systems is to achieve two goals (G1 and G2) for low-criticality tasks without compromising timing guarantees for high-criticality tasks: (G1) providing a certain (degraded) level of timing guarantees for *all low-criticality tasks*; (G2) while maximizing the *fully-serviced* (non-degraded) execution of low-criticality tasks. To address the problem, we propose IMC-PnG, an MC scheduling framework, which employs two salient features: a task-level virtual-deadline assignment under the imprecise computing model with efficient resource utilization (supporting G1), and an online scheduling algorithm that dynamically changes criticality levels of individual tasks at runtime (supporting G2). In simulation results with random workloads, we showed that IMC-PnG has up to 12.10% higher schedulability and up to 42.10% higher runtime performance (measured by the fully-serviced ratio of low-criticality tasks) than the existing approaches.

1. Introduction

Real-time systems aim at achieving (i) timing guarantees of tasks subject to timing constraints and (ii) efficient utilization of computing resources. Since classical real-time scheduling research that mainly focuses on (i) has relied on a pessimistically-calculated Worst-Case Execution Time (WCET) for each task, the concept of Mixed-Criticality (MC) has been introduced [1–3] in order to address (ii) without compromising (i). By employing multiple WCETs (optimistic/pessimistic ones) for each task, MC is capable of providing multiple levels of timing guarantees in different situations. That is, if no task executes for up to its optimistic WCET (i.e., in the normal mode), the system can support the timely execution of all tasks. Otherwise (i.e., there exists a task that executes for more than its optimistic WCET), the system turns to the critical mode and supports the timely execution of high-critical tasks only (that execute for up to their pessimistic WCETs) without taking care of timing guarantees of low-critical tasks.

Although the concept of MC operates as an interface to achieve (i) and (ii) at the same time, the early stage of MC studies cannot fully address both especially for *low-criticality tasks after the mode change* (i.e., in the critical mode), so-called graceful degradation of low-critical tasks, including (a) how to provide a certain (i.e., degraded) level of timing guarantees for *all low-critical tasks*, and (b) how to maximize (rather than guarantee) the *fully-serviced* (i.e., non-degraded) execution of low-critical tasks. As graceful degradation of low-critical tasks is desirable for system performance and safety [4], later MC studies for (a) support degraded service to low-critical tasks in the critical mode by stretching their periods [4–6] or skipping their jobs [7]. Recently, the imprecise computing model [8] is applied to the concept of MC, where a low-critical task consists of a mandatory execution part and an optional execution part, called IMC (Imprecise Mixed-Criticality) systems; a group of IMC studies guarantees the timely execution of the mandatory part of low-critical tasks after the mode change [4,9,10]. On the other hand, some studies address (b) to further achieve graceful degradation in the critical mode. For example, a group of studies [11–13] introduces a fine-grained mode change protocol and maximizes the execution of low-critical tasks by online scheduling decisions. However, no studies address both (a) and (b) through the interaction between multi-level timing guarantee and sophisticated runtime scheduling decision, which is a key to achieving both at the same time.

In this paper, our goals are (G1) to provide a weak level of timing guarantees for low-criticality tasks (by assuring timely execution of every mandatory part according to the IMC model, corresponding to (a)) without compromising timing guarantees for high-criticality tasks and (G2) to maximize runtime performance (by maximizing the number

* Corresponding author.
E-mail addresses: jaewoolee@cau.ac.kr (J. Lee), jinkyu.lee@skku.edu (J. Lee).

https://doi.org/10.1016/j.future.2024.06.015
Received 5 January 2024; Received in revised form 1 May 2024; Accepted 9 June 2024
Available online 10 June 2024

of optional parts performed in time, corresponding to (b)). Achieving both G1 and G2 are necessary for safety-critical cyber–physical systems. As an application that necessitates G1, automotive systems consist of high-critical hard-real-time components (such as engine, brake and steering systems), and low-critical soft-real-time components (such as navigation systems, GPS systems and signal lamps). In addition to timing guarantee in high-criticality components, some levels of timing guarantee in low-criticality components is necessary for the safe driving.

Although G1 is achieved, the system designer pursues G2, which means that they would like to maximize runtime performance of the system, increasing the utility (e.g., optional safety, fault tolerance, functionality) of the entire systems. For example, to achieve autonomous driving associated in automotive systems, an important task is perception, which is object detection (OD) for camera video stream. In particular, there are trade-off between accuracy and execution time in OD algorithms such as YOLO [14]. Utilizing the trade-off, some studies allocate different accuracy objectives on different tasks depending on their criticality [15,16]. When the system enters the critical mode (e.g., extreme weather), the system allocates more resources to high-critical OD tasks such as the front camera (increasing accuracy), and fewer resources to low-critical OD tasks such as the side/rear cameras under the IMC model. Even after the mode change, the system should provide a minimum-level of timing guarantees for low-criticality tasks such as OD associated with the rear camera (by executing every mandatory part in time), while maximizing the accuracy of low-criticality tasks (by running optional parts as much as possible).

To achieve the goal necessary for safety-critical cyber–physical systems, we propose the IMC scheduling framework maximizing the runtime Performance and the timing Guarantee, called IMC-PnG. Our framework has the following key features.

- **F1.** We propose IMC-PnG scheduling algorithm, applying classical EDF policy with Virtual Deadlines (VDs), EDF-VD [3], under IMC model. In IMC-PnG, we propose a runtime mode-change algorithm to minimize the number of degraded executions of low-criticality tasks by considering runtime criticality levels of individual tasks. This approach is the first work to adjust the number of degraded executions at runtime under IMC model (Section 5.1).
- **F2.** We develop the online schedulability analysis for IMC-PnG to determine whether the current system behavior by the decision of the IMC scheduling algorithm does not miss any deadlines; we also develop the offline schedulability analysis by analyzing the worst case of online scheduling scenarios of IMC-PnG (Section 5.2).
- **F3.** By identifying the duration of the peak resource demand of high-criticality tasks, we develop an advanced version of F1 and F2, resulting in an improved online schedulability analysis without compromising offline schedulability performance, which improves the runtime performance (Section 5.3).
- **F4.** In IMC-PnG, we assign the VD of each task individually. We propose a resource-efficient algorithm to decide the virtual deadline of individual tasks to maximize schedulability and the runtime performance of the task set (Section 5.4).

Using randomly generated workloads, we evaluated the performance of IMC-PnG compared to the existing approaches. In experimental results, IMC-PnG has up to 12.10% higher offline schedulability than the existing approaches. We also conducted to evaluate the runtime performance simulating various scheduling scenarios. As a result, IMC-PnG has up to 42.10% higher PFJ (percentage of fully-serviced jobs) of low-criticality tasks than the existing approaches.

The rest of this paper is structured as follows. Section 2 discusses related work. Sections 3 and 4 respectively explain the system model and background. Section 5 proposes the IMC-PnG scheduling framework and Section 6 evaluates the effectiveness of IMC-PnG. Section 7 discusses the remaining issues and Section 8 concludes the paper.

2. Related work

Since Vestal's seminal work [1] on MC systems, there have been many studies on MC domain (see a survey [17] for details). In classical MC systems, Baruah et al. [2] introduced the concept of mode-switch: when a HC task executes more than its low-criticality (or normal-confidence) WCET, the system enters in HI mode[1] where all LC tasks are dropped to ensure the execution of HC tasks.[2] EDF-VD [3] introduced virtual deadlines to adjust the priorities of HC tasks in LO mode. Based on the basic model of MC system, there has been a wide range of research direction on MC systems.

One direction is to improve schedulability. In multiprocessor MC domain, global or partitioned scheduling algorithm based on EDF-VD has low performance. Considering parallel execution in multiprocessors, MC-Fluid [18] improved schedulability by using per-task virtual-deadline assignment. Recently, Yang et al. [19] calculated per-task virtual-deadline from the analysis result of MC-Fluid and applied EDF-VD scheduling policy. In safety-critical cyber–physical systems, which is our target domain, some LC tasks need a minimum level of execution even after mode-switch, as well as maximizing runtime performance of LC tasks; however, this has not been considered in classical MC systems. For example, vision tasks in side/rear cameras of autonomous vehicles need to operate at a minimum level, even in extreme situations, and operate at a higher level as much as possible. Our work is the first work that extends classical MC systems with low-level timing guarantee of LC tasks and runtime performance optimization of LC tasks.

Another direction is graceful degradation after mode-switch, focusing the runtime performance of LC tasks. Instead of dropping all LC tasks at mode switch, various degraded service of LC tasks has been introduced, such as stretching task periods [4–6,20], reducing task execution times [21], and allowing not to execute some jobs [7]. Although system-level mode-switch approaches [2,3] cannot control the criticality mode of individual tasks, recent approaches [11–13] employed task-level mode-switch mechanism: when a HC task executes more than its normal-confidence WCET, the system only changes its runtime criticality mode. Chen et al. [12] introduced the kth HI mode to identify the number of HC tasks which are in HI mode. Our previous work [11,13] introduced task-level runtime criticality mode to further reduce the number of dropping LC tasks for different runtime criticality mode. Lee and Lee [13] introduced task-level mode recovery mechanism to re-execute the dropped LC tasks as soon as possible. Along with the task-level mode-switch paradigm, our work introduced the concept of stable-HI-mode tasks to maximize the runtime performance under IMC systems further.

The other direction is toward IMC systems which apply imprecise computing model [8] on MC systems. Although classical MC schemes drop LC tasks at mode-switch, Burns et al. [4] proposed to degrade LC tasks (similar to executing mandatory parts in imprecise computing) after the mode-switch. Liu et al. [22] formalized this mechanism with imprecise mixed-criticality systems, which was later extended to the constrained-deadline task model [23]. Guo et al. [9] improved schedulability under uniprocessor EDF scheduling algorithm by using theoretical MC-Fluid [18]. Pathan [10] considered both schedulability and runtime performance (in terms of the percentage of fully-executed jobs) from static offline schedulability analysis. To further maximize runtime performance, our work analyzes online schedulability considering dynamic runtime scenarios, assuming arbitrary switch of task-level criticality mode at any time.

MC systems have been studied considering power and energy. Guo and Baruah [24] proposed to schedule MC jobs on varying-speed multiprocessors. Huang et al. [25] proposed to schedule MC tasks by the

[1] Most of MC studies consider dual criticalities: HI (high) or LO (low).

[2] HC and LC stand for High-Criticality and Low-Criticality, respectively.

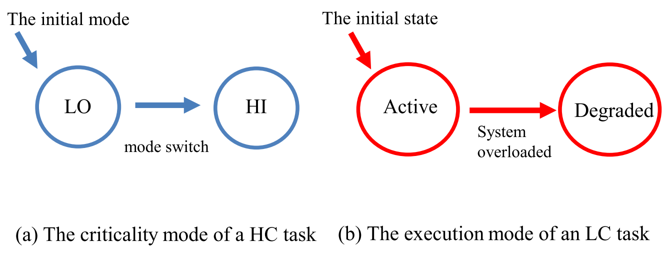


Fig. 1. Task behavioral model for IMC tasks.

minimum processor speedup. She et al. [26] does not sacrifice low-critical tasks of IMC workloads after mode-switch, by boosting the processor speed. Zhang and Chen [27] proposed partition heuristics on multicore IMC platform considering energy consumption.

Besides theoretical MC studies, practical implementation of MC systems is also important. Huang et al. [28] implemented a user-space MC scheduler on Linux and measured the preemption overheads and the cost of priority adjustment. Liu et al. [23] discussed how to implement the IMC scheduling algorithm on LITMUS-RT platform [29]. David et al. [30] observed that the cost of context switching between tasks of different criticality is higher than the switching cost between tasks of same criticality. Sundar and Easwaran [31] observed the effect of task degradation scheme on automotive test-beds.

3. System model

Task Model. For simplicity, we focus on a dual-criticality system with two distinct criticality levels, HI (high) and LO (low). We consider a set of implicit-deadline sporadic tasks (denoted by τ) of n MC tasks [3], each characterized by $\tau_i = (T_i, C_i^L, C_i^H, \chi_i)$, where

- $T_i \in \mathbb{R}$ is the minimum inter-job separation time (or period),
- $C_i^L \in \mathbb{R}$ is a LO-criticality WCET (Worst-Case Execution Time), denoted by L-WCET,
- $C_i^H \in \mathbb{R}$ is a HI-criticality WCET, denoted by H-WCET, and
- $\chi_i \in \{\mathsf{HI}, \mathsf{LO}\}$ is a task criticality level.

Depending on the task criticality level χ_i, a task is either a High-Criticality (HC) task or a Low-Criticality (LC) task. For a HC task with $\chi_i = \mathsf{HI}$, it has $C_i^L \le C_i^H$, which means that the task requires more execution budget in the extreme case. On the contrary, for a LC task with $\chi_i = \mathsf{LO}$, it has $C_i^L = C_i^H$, which means that the system will not assign more execution budget for the task even in the extreme situation.

Each MC task τ_i generates an infinite sequence of jobs $\{J_i^1, J_i^2, \dots\}$, each released with at least T_i inter-job separation time units. Once a job of τ_i is released at t, it executes for at most C_i^L or C_i^H (depending on the task/system mode) and should finish its execution no later than $t + T_i$. We consider a uniprocessor system where only one job can be executed at any time.

We consider an Imprecise Mixed-Criticality (IMC) task model, where reduced execution budgets for low-criticality tasks are applied in the overloaded system situation, which is a subset of MC task model derived from [4,22]. In the IMC task model, LC task consists of two sub-tasks: a mandatory sub-task (whose execution budget is denoted by C_i^M) and an optional sub-task (whose execution budget is denoted by C_i^O), satisfying $C_i^H = C_i^L = C_i^M + C_i^O$. In addition, the system allows to skip the optional sub-task of LC tasks when the system is overloaded, while the mandatory sub-task should be executed in any case. When HC tasks need more execution budgets, classical MC task model can drop LC tasks while this IMC model does not allow to drop LC tasks but skip optional sub-tasks of LC tasks.

Task Behavior. We consider task-level criticality mode (*task mode*) [11,13] as opposed to the system-level criticality mode (widely used in classic MC studies (e.g., [2,3])). For each HC task τ_i, we define M_i as the task mode of τ_i indicating its task behavior shown in Fig. 1(a). A task τ_i is said to be in LO mode ($M_i = \mathsf{LO}$) if no job of the task has executed for more than its L-WCET, and to be in HI mode ($M_i = \mathsf{HI}$) otherwise. In addition to HC tasks, we consider two execution modes of a LC task shown in Fig. 1(b); each LC task is in either an *active* mode ($M_i = A$) or a *degraded* mode ($M_i = D$). A LC task executes up to its active WCET ($C_i^A = C_i^M + C_i^O$) in its active mode, while the task executes up to its degraded WCET ($C_i^D = C_i^M$) in its degraded mode.

Scheduling semantics of the IMC model is different from the classical MC model in the overloaded system situation (i.e., some HC tasks execute more than their L-WCET). That is, in the overloaded system situation, while the classical MC model drops the entire execution of LC tasks, the IMC model skips optional execution of LC tasks without dropping the mandatory execution of any LC task. The system scenario for the IMC model are summarized as follows:

- Initially, all the HC tasks are in LO-mode and all the LC tasks are in active mode.
- When a HC task τ_i executes more than its L-WCET (C_i^L), the scheduler can change its task mode[3] (M_i) from LO mode to HI mode (called *mode-switch*), i.e., $M_i := \mathsf{HI}$.
- When the scheduler needs more resources (due to mode-switch of at least one HC task), one or more active LC tasks are selected to be degraded (by disallowing the execution on the optional part of the selected LC tasks) in order to support HC tasks with their additional resource requests.

When the scheduler decides a LC task to change its state into the degraded state, the current job of the task is immediately suspended if the amount of job execution already performed is more than its mandatory execution budget (C_i^M). The scheduling algorithm to be applied determines which LC task is degraded.

System Goal. We consider the IMC schedulability, similar to the existing IMC studies [4,22], which consists of the following three conditions:

- **IMC-A**: HC tasks are always schedulable.
- **IMC-B**: LC tasks are schedulable with their L-WCETs if all HC tasks are in LO mode.
- **IMC-C**: LC tasks are schedulable with their mandatory execution budgets if any HC task is not in LO mode.

Besides the IMC schedulability, we also consider the runtime performance of LC tasks, which entails the following goal.

- **IMC-R**: The ratio of fully serviced jobs (with both mandatory and optional execution) for LC tasks should be maximized.

In summary, the system goal is to achieve IMC-R while satisfying the constraints of IMC-A, IMC-B, and IMC-C.

Notations. For notational convenience, let τ_{H} and τ_{L} denote a set of HC tasks and LC tasks, respectively, i.e., $\tau_{\mathsf{H}} \overset{\text{def}}{=} \{\tau_i \in \tau \mid \chi_i = \mathsf{HI}\}$ and $\tau_{\mathsf{L}} \overset{\text{def}}{=} \{\tau_i \in \tau \mid \chi_i = \mathsf{LO}\}$. The utilization of a task τ_i is defined as $u_i^L \overset{\text{def}}{=} C_i^L/T_i$ and $u_i^H \overset{\text{def}}{=} C_i^H/T_i$ for HC tasks, and $u_i^A \overset{\text{def}}{=} C_i^A/T_i$ and $u_i^D \overset{\text{def}}{=} C_i^D/T_i$ for LC tasks.

We define the collective utilization of a task set as follows:

$$U_{\mathsf{H}}^H \overset{\text{def}}{=} \sum_{\tau_i \in \tau_{\mathsf{H}}} u_i^H, \qquad U_{\mathsf{H}}^L \overset{\text{def}}{=} \sum_{\tau_i \in \tau_{\mathsf{H}}} u_i^L,$$

$$U_{\mathsf{L}}^A \overset{\text{def}}{=} \sum_{\tau_i \in \tau_{\mathsf{L}}} u_i^A, \qquad U_{\mathsf{L}}^D \overset{\text{def}}{=} \sum_{\tau_i \in \tau_{\mathsf{L}}} u_i^D.$$

[3] The system-level mode-switch approach changes the task mode of all tasks while the task-level mode-switch approach changes the task mode of each task individually.

4. Background

Before presenting a new IMC scheduling framework, we recapitulate its base scheduling algorithm EDF-VD [3] under classic MC systems and review the IMC scheduling approaches.

EDF-VD scheduling algorithm. Different from the vanilla EDF, EDF-VD employs the concept of *virtual deadline (VD)*, which is the modified scheduling priority from the original deadline. That is, EDF-VD assigns different absolute deadlines[4] to jobs of each HC task in different modes (i.e., the VD in LO mode and the real deadline in HI mode).

EDF-VD assigns the virtual relative deadline of the task by $V_i := xT_i$, where $0 < x \leq 1$ is the VD coefficient. The role of VD is a priority boost for HC tasks. The job of a HC task with LO execution budget (C_i^L) may not complete its job execution until its VD,[5] which invokes a mode-switch. Then, the job may execute the additional execution budget ($C_i^H - C_i^L$) during the remaining time duration ($T_i - V_i$) between its VD and real relative deadline. Under the system-level mode-switch mechanism deployed by EDF-VD, when a single HC task switches to HI mode, all the other HC tasks switch to HI mode simultaneously.

We present the offline schedulability condition of EDF-VD [3]: a task set τ is schedulable by EDF-VD if

$$U_{\mathsf{L}}^A + \frac{U_{\mathsf{H}}^L}{x} \leq 1, \tag{1}$$

$$xU_{\mathsf{L}}^A + U_{\mathsf{H}}^H \leq 1. \tag{2}$$

Then, the VD coefficient can be derived from Eq. (1): $x = U_{\mathsf{H}}^L/(1 - U_{\mathsf{L}}^A)$.

Imprecise Mixed-Criticality Scheduling. Burns and Baruah [4] proposed an MC system where LC tasks reduce their execution budgets in HI mode, instead of skipping their execution. Since the system is similar to the notion of the imprecise computation model [32], Liu et al. [22] referred to the system as *Imprecise Mixed-Criticality (IMC)* system. IMC systems allow the reduced execution only for LC tasks in HI mode, whereas general imprecise systems always allow the reduced execution for all imprecise tasks.

AMC-IMC [4] proposed a fixed-priority IMC scheduling algorithm and its schedulability analysis. Later, Liu et al. [22] proposed EDF-VD-IMC extending EDF-VD [3] under IMC systems. Instead of dropping LC tasks at mode-switch, EDF-VD-IMC executes all LC tasks with a degraded execution budget in HI mode. The next lemma records schedulability analysis for EDF-VD-IMC:

Lemma 1 (*from [22]*). *A task set τ is schedulable by EDF-VD-IMC if*

$$U_{\mathsf{L}}^A + \frac{U_{\mathsf{H}}^L}{x} \leq 1, \tag{3}$$

$$xU_{\mathsf{L}}^A + (1-x)U_{\mathsf{L}}^D + U_{\mathsf{H}}^H \leq 1. \tag{4}$$

Eqs. (3) and (4) represent the schedulability condition in LO and HI mode, respectively. While the former is identical to the schedulability condition for EDF-VD (i.e., Eq. (1)), the latter is different from that for EDF-VD (i.e., Eq. (2)). That is, the term $(1-x)U_{\mathsf{L}}^D$ is added from Eq. (2) to Eq. (4), by considering that LC tasks still execute in degraded quality even after the mode-switch.

5. IMC-PnG scheduling framework

Different from MC studies, IMC studies have not matured yet as they have not fully addressed the system goal in terms of runtime performance (i.e., IMC-R) and offline schedulability (i.e., IMC-A, B, and C), as follows. First, early IMC studies [4,22] have considered schedulability only without addressing the runtime performance. One recent study [10] has calculated the allowable additional demand derived from the offline schedulability analysis and utilized it to support a given ratio of fully-serviced jobs of LC tasks in HI mode. However, since the study cannot utilize runtime information such as the difference between WCET and the actual execution time, its runtime performance is limited to the underlying offline schedulability analysis. Second, existing IMC studies have derived pessimistic schedulability. While the seminal work on IMC [4] results in limited schedulability as it is based on the fixed priority scheduling algorithm, Liu et al. [22] improved schedulability based on the EDF-VD [3] scheduling algorithm specialized for MC systems. However, it is still pessimistic due to applying the global priority adjustment using the system-wide VD coefficient. When it comes to theoretical fluid-based scheduling, Baruah et al. [9] (for uniprocessor) and Pathan [10] (for multiprocessor) proposed task-level priority adjustment (via execution rate assignment) based on MC-Fluid [18]. However, fluid-based IMC work is not practically implementable on most of embedded systems due to severe context-switching overheads.

To address the issues, we develop the IMC-PnG scheduling algorithm, based on EDF-VD-IMC [22] that considers schedulability only. We have three major approaches to consider both schedulability and the runtime performance of LC tasks.

1. We apply task-level mode-switch mechanism to minimize the number of degraded execution of LC jobs. In EDF-VD-IMC, all LC tasks are degraded immediately after system-level mode-switch. In classical MC approaches [11–13], task-level mode-switch helps runtime performance of LC tasks. Instead of degrading all LC tasks at system-level mode-switch, we can degrade the minimal numbers of LC tasks on a series of task-level mode-switch events, which improves the runtime performance of LC tasks.
2. To improve schedulability, we apply a per-task VD assignment scheme considering the task characteristic, instead of the VD assignment by the global VD coefficient (x) in EDF-VD-IMC. Inspired by MC-Fluid [18] in the classic MC domain, we propose an individual VD coefficient (x_i) for each HC task τ_i and optimize all VD coefficients to improve schedulability. Improved schedulability also helps the runtime performance of LC tasks by reducing the number of degraded execution of LC jobs at the mode-switch.
3. To improve runtime performance further, we carefully examine the demand of HC tasks entering in HI mode. For a HC task, we found that the demand of the mode-switching job (released in LO-mode and switched to HI-mode during its execution) may be different from that of subsequent jobs released in HI-mode (no mode switch during its execution). With a fine-grained segmentation of HC tasks on runtime, we can tightly bound the demand of HC tasks, which helps the runtime performance of LC tasks.

Fig. 2 illustrates the overall process of the IMC-PnG. In the design time, for a given task set, we first assign per-task VD for each HC task (Section 5.4). Then, we check the offline schedulability test for the task set with the assigned VDs (Section 5.2). If the task set cannot pass the test, we declare that the task set is unschedulable by IMC-PnG. At runtime, we schedule the task set according to the scheduling algorithm (Section 5.1). When a mode-switch happens, we invoke the mode-switch algorithm. In the algorithm, we check the online schedulability test. When the test is not satisfied, we pick an active LC task and degrade the task. Then, we check the test again. If the test is satisfied, we proceed to execute the scheduling algorithm.

This section presents the proposed approaches according to the following structure. First, targeting the basic model, Section 5.1 proposes the IMC-PnG scheduling framework, and Section 5.2 derives the online schedulability analysis as well as the offline one. Second, Section 5.3 improves the basic model and its online schedulability analysis. Finally, Section 5.4 proposes a resource-efficient algorithm to assign the VD of individual tasks.

[4] When no ambiguity arises, we use the term "deadline" for "absolute deadline".

[5] From now on, VD (when mentioned alone, not VD in EDF-VD) means the virtual relative deadline.

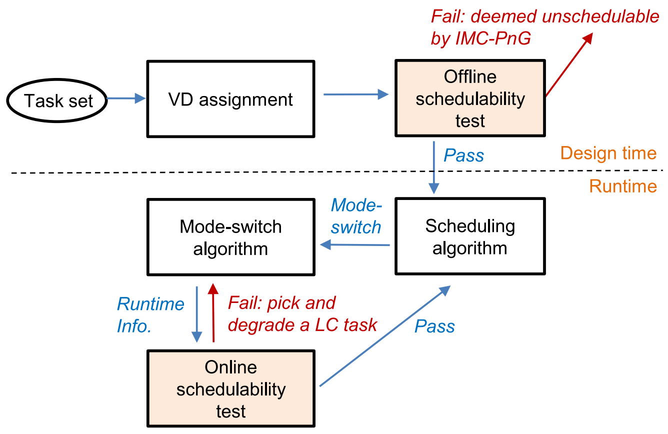

Task set
VD assignment
Offline schedulability test
Fail: deemed unschedulable by IMC-PnG
Pass
Design time
Runtime
Mode-switch algorithm
Mode-switch
Scheduling algorithm
Runtime Info.
Fail: pick and degrade a LC task
Pass
Online schedulability test

Fig. 2. The overall process of the IMC-PnG scheduling framework.

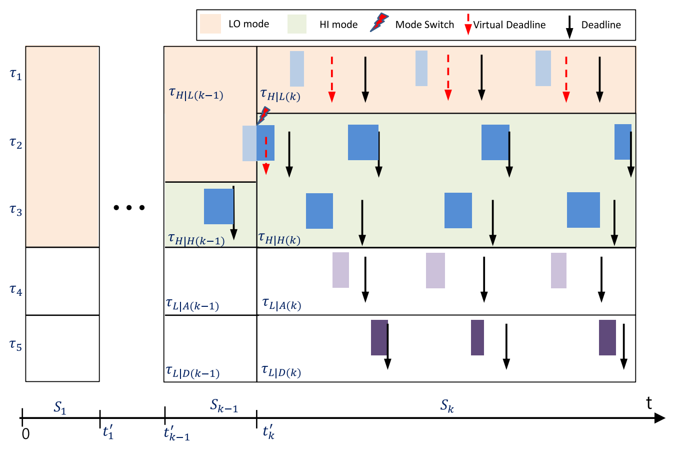


Fig. 3. Illustration of stage S_k, which describes the mode of tasks at runtime after t'_k, i.e., the kth mode-switch time instant (in this example, τ_2 belonged to $\tau_{\mathsf{H|L}(k-1)}$, but after the kth mode switch, τ_2 belongs to $\tau_{\mathsf{H|H}(k)}$).

5.1. Scheduling algorithm

Based on EDF-VD-IMC [22], we develop the IMC-PnG scheduling algorithm as follows. First, IMC-PnG assigns virtual deadlines to jobs of a HC task when the task is in HI-mode, as EDF-VD-IMC does. That is, whenever a job of a HC task τ_i is released at t_a in LO and HI mode, its virtual deadline $(t_a + V_i)$ and real deadline $(t_a + T_i)$ are used for prioritization, respectively. Second, the virtual deadline assignment for IMC-PnG is different from that for EDF-VD-IMC. While all the HC tasks under EDF-VD-IMC share the global virtual-deadline (VD) coefficient (i.e., x for all tasks), those under IMC-PnG employ task-level VD coefficients (i.e., x_i for each τ_i), meaning that the relative VD of each HC task (V_i) is assigned by $V_i = x_i \cdot T_i$. Note that Section 5.4 will develop how to assign task-level VD coefficients. Third, the mode-switch mechanism for IMC-PnG is also different from that for EDF-VD-IMC, as it operates in a task-level manner. Whenever a HC task in LO mode executes for more than its L-WCET, the HC task itself changes its task mode from LO to HI while other HC tasks keep their task mode unchanged (i.e., task-level mode-switch occurs), which is similar to existing task-level criticality-mode approaches [11,12].

We first present the runtime scheduling policy of the MC-RUN scheduling algorithm.

Runtime Scheduling Policy. As a first step to present runtime scheduling policy, we introduce a notion of *stage* that captures the dynamic system behavior at mode-switch, including the task mode for each HC task and the execution mode for each LC task shown in Fig. 1 of Section 3. Initially, all the HC tasks start in LO mode. At runtime, different HC task mode-switches at different time instants. As shown in Fig. 3, the kth stage is described for the system behavior after the kth mode-switch.

Definition 1 (*Stage*). Let t'_k and t'_{k+1} respectively denote the time instants at which the kth and $(k+1)$-th mode-switch occur. Stage S_k represents the system behavior in the time interval $[t'_k, t'_{k+1})$ and is defined as a tuple of four task sets as follows.

- $\tau_{\mathsf{H|L}(k)}$: a set of HC tasks whose task mode is LO in $[t'_k, t'_{k+1})$,
- $\tau_{\mathsf{H|H}(k)}$: a set of HC tasks whose task mode is HI in $[t'_k, t'_{k+1})$,
- $\tau_{\mathsf{L|A}(k)}$: a set of LC tasks whose execution state is "active" in $[t'_k, t'_{k+1})$, and
- $\tau_{\mathsf{L|D}(k)}$: a set of LC tasks whose execution state is "degraded" under in $[t'_k, t'_{k+1})$.

Note that t'_{k+1} is ∞ if the total number of mode-switches is k.

By definition, $\tau_{\mathsf{H|L}(k)}$ and $\tau_{\mathsf{H|H}(k)}$ are disjoint, and $\tau_{\mathsf{H}} = \tau_{\mathsf{H|L}(k)} \cup \tau_{\mathsf{H|H}(k)}$ holds. Likewise, $\tau_{\mathsf{L|A}(k)}$ and $\tau_{\mathsf{L|D}(k)}$ are disjoint, and $\tau_{\mathsf{L}} = \tau_{\mathsf{L|A}(k)} \cup \tau_{\mathsf{L|D}(k)}$ holds. Note that the initial stage S_0 is represented by $(\tau_{\mathsf{H}}, \emptyset, \tau_{\mathsf{L}}, \emptyset)$, which is identical to system-level LO mode in EDF-VD-IMC [22]. Similarly to U_{H}^H, U_{H}^L, U_{L}^A and U_{L}^D, we define collective utilization of each of the above disjoint task sets:

$$U_{z(k)}^{y} \overset{\text{def}}{=} \sum_{\tau_i \in \tau_{z(k)}} u_i^y, \tag{5}$$

where $z(k) \in \{\mathsf{H|L}(k), \mathsf{H|H}(k)\}$ and $y \in \{L, H\}$, or $z(k) \in \{\mathsf{L|A}(k), \mathsf{L|D}(k)\}$ and $y \in \{A, D\}$.

Based on the concept of stage, we present the runtime scheduling policy for IMC-PnG. We schedule the job of the earliest effective deadline, which is the same as EDF-VD-IMC with the following instructions:

P1. Initially, every LC task is set to active $(\forall \tau_i \in \tau_{\mathsf{L}}, M_i = A)$, and every HC task is set to LO mode $(\forall \tau_i \in \tau_{\mathsf{H}}, M_i = \mathsf{LO})$.

P2. When a job of HC task τ_i arrives at time t_a, it has C_i^L execution budget and is scheduled with its virtual deadline $(t_a + V_i)$ in LO mode. If the arriving job is in HI mode, it has C_i^H execution budget and is scheduled with its real deadline $(t_a + T_i)$.

P3. When a job of LC task τ_i arrives at time t_a, it is scheduled with its real deadline $(t_a + T_i)$. Its execution budget is set to C_i^A if the job is in active mode, and C_i^D otherwise.

P4. When a job of HC task τ_i in LO mode executes for more than its C_i^L (i.e., a mode-switch for τ_i occurs), run the mode-switch algorithm (i.e., Algorithm 1).

P5. When no job of any task waits in the ready queue (when the system is idle), the system changes its state to the initial state, i.e., all LC tasks are set to active $(M_i := A)$, and all HC tasks are set to LO mode $(M_i := \mathsf{LO})$.

Algorithm 1 describes the detailed system behavior at the runtime mode-switch. In Lines 4–8, we update the parameters of the mode-switching job and the task to which it belongs. In Lines 11–15, we check the online schedulability and determine which LC task needs to be degraded. Considering that individual tasks contribute to different resource demand (i.e., execution requirement for a unit of period) according to its affiliated set (i.e., $\tau_{\mathsf{H|L}(k)}$, $\tau_{\mathsf{H|H}(k)}$, $\tau_{\mathsf{L|A}(k)}$ and $\tau_{\mathsf{L|D}(k)}$), we may express a IMC-schedulability condition at a time instant using the following inequality:

$$\mathcal{F}(\tau_{\mathsf{H|L}(k)}, \tau_{\mathsf{H|H}(k)}, \tau_{\mathsf{L|A}(k)}, \tau_{\mathsf{L|D}(k)}) \le 1. \tag{6}$$

We will derive the function $\mathcal{F}(\cdot)$ in Section 5.2. The time-complexity for Algorithm 1 is $O(n)$ because Lines 11-15 (the 'while' part) repeat

Algorithm 1: Runtime mode-switch algorithm for IMC-PnG (degrading LC tasks according to the online schedulability test)

```
Data: the previous stage $S_{k-1}$, the job that incurs the $k$-th
      mode-switch $J^*$, its arrival time $t_a(J^*)$, its absolute deadline
      $t_d(J^*)$, and its execution budget $e(J^*)$
Result: $S_k$
1  $(\tau_{H|L(k)}, \tau_{H|H(k)}, \tau_{L|A(k)}, \tau_{L|D(k)}) \leftarrow S_{k-1}$;
2  foreach HC task $\tau_i \in \tau_{H|L}$ do
3      if $J^*$ is the instance of $\tau_i$ then
4          $t_d(J^*) \leftarrow t_a(J^*) + T_i$ ;
5          $e(J^*) \leftarrow C_i^H$ ;
6          $M_i \leftarrow$ HI ;
7          $\tau_{H|L(k)} \leftarrow \tau_{H|L(k)} \setminus \{\tau_i\}$ ;
8          $\tau_{H|H(k)} \leftarrow \tau_{H|H(k)} \cup \{\tau_i\}$;
9      end
10 end
11 while $\neg$ Eq. (6) do
12     Pick a LC task $\tau_j \in \tau_{L|A(k)}$ with the highest Like($\tau_j$) ;
13     $M_j \leftarrow D$ ;                                  /* Degrade $\tau_j$ */
14     $\tau_{L|A(k)} \leftarrow \tau_{L|A(k)} \setminus \{\tau_j\}$,   $\tau_{L|D(k)} \leftarrow \tau_{L|D(k)} \cup \{\tau_j\}$ ;
15 end
16 $S_k \leftarrow (\tau_{H|L(k)}, \tau_{H|H(k)}, \tau_{L|A(k)}, \tau_{L|D(k)})$ ;
```

as many times as the number of tasks in the worst case; for any schedulable task set, Eq. (6) holds if all LC tasks are degraded.

Even if IMC-schedulability is guaranteed on the system start, the IMC-schedulability condition (Eq. (6)) after subsequent mode-switches is not necessarily satisfied due to the increase of resource demand of HC tasks. Then, we inevitably have to decrease the resource demand of LC task by degrading some LC tasks, in order to satisfy the IMC-schedulability again. The problem is how to choose a good candidate to degrade among active LC tasks. To solve the problem, we present a heuristic solution, which is the Like() function indicating the benefit/penalty when the specific LC task is degraded. To minimize the total number of degraded jobs (related to IMC-R), the benefit/penalty is related to two factors: (a) how much resource (as a form of utilization) is reclaimed from degrading the task, and (b) how many jobs (for a given interval) are degraded additionally when the task is degraded. For (a), a LC task with the larger reclaimed utilization $(u_i^A - u_i^D)$ is preferred, which reduces the number of the degraded LC tasks. For (b), a LC task with the larger period (T_i) is preferred, which reduces the additional number of degraded jobs for a given time interval when the task is degraded. Therefore, the function is calculated by the reclaimed utilization $(u_i^A - u_i^D)$ multiplying task period:

$$\mathsf{Like}(\tau_i) = (u_i^A - u_i^D) \cdot T_i = C_i^A - C_i^D.$$

Simulation results from [13] demonstrated that the approach similar to the Like() function that considers both (a) and (b) has better performance than the corresponding approach that considers (a) only.

5.2. Schedulability analysis

In this subsection, we first analyze online schedulability at a specific mode-switch, which means whether a given task set is schedulable by IMC-PnG in a given mode-switch situation. Finding the worst-case online behavior, we analyze offline schedulability, which means whether a given task set is schedulable by IMC-PnG with any sequence of mode-switches.

In this subsection, we assume that VD coefficients (x_i) are given. Later in Section 5.4, we will find good VD coefficients for schedulability and runtime performance, based on this schedulability analysis.

Online Schedulability Analysis. To analyze online schedulability, we utilize mathematical induction: (i) analyzing the schedulability condition in the initial stage S_0, and (ii) for all $k \geq 1$, analyzing the schedulability condition under S_k (including the transitive stage from S_{k-1} to S_k) when the schedulability under S_{k-1} is satisfied.

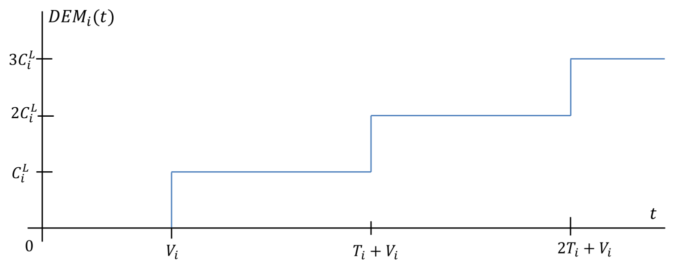


Fig. 4. Illustration of the resource demand of task $\tau_i \in \tau_H$ over time t when there is no mode-switch.

First, we consider schedulability before any mode-switch, which is schedulability under S_0: $(\tau_H, \emptyset, \tau_L, \emptyset)$. This schedulability can be derived offline without any runtime information.

Theorem 1. *Under the initial stage S_0, task set τ is schedulable by IMC-PnG if*

$$U_L^A + \sum_{\tau_i \in \tau_H} \frac{u_i^L}{x_i} \leq 1. \tag{7}$$

To prove Theorem 1, we need to bound the resource demand (i.e., execution requirement) on the system. Since the collective resource demand can be derived from the resource demand of each task, we define the demand of a task as follows:

Definition 2 (*Task Demand*)**.** $\mathsf{DEM}_i(t)$ is defined as the resource demand for task τ_i in a consecutive time interval of length t, which is the amount of necessary execution of jobs of τ_i within a consecutive time interval of length t in order to avoid any job deadline miss of τ_i under the IMC-PnG scheduling algorithm.

Until the first mode-switch, $\mathsf{DEM}_i(t)$ is similar to the demand of τ_i for an interval of length t under the classical real-time (single-criticality) systems [33]. That is, if $\tau_i \in \tau_H$ invokes its jobs in a strictly periodic manner from 0 with the system scenario which has no mode-switch, $\mathsf{DEM}_i(t)$ is calculated by $\lfloor \frac{t+T_i-V_i}{T_i} \rfloor \cdot C_i^L$ [13], as shown in Fig. 4. Note that it is challenging to develop how to calculate $\mathsf{DEM}_i(t)$ *after* some mode-switches, which will be addressed in this subsection.

By the definition of $\mathsf{DEM}_i(t)$, we only need to check the following condition for schedulability.

$$\forall t, \sum_{\tau_i \in \tau} \mathsf{DEM}_i(t) \leq t. \tag{8}$$

To calculate Eq. (8), we need to know the demand of each task (i.e., $\mathsf{DEM}_i(t)$) depending on task criticality and task mode. In the next lemma, we can compute the demand of each HC task in LO mode and that of each LC task in active mode.

Lemma 2. *For LC task τ_i that is in active mode, the following conditions hold:*

$$\mathsf{DEM}_i(t) \;\leq\; u_i^A \cdot t$$

Proof. Since each job of τ_i executes for its L-WCET (i.e., C_i^L) until its real deadline (i.e., T_i), we have $\mathsf{DEM}_i(t) \leq C_i^L/T_i \cdot t = u_i^L \cdot t$. □

Lemma 3. *For HC task τ_i that is in LO mode, the following conditions hold:*

$$\mathsf{DEM}_i(t) \;\leq\; u_i^L/x_i \cdot t$$

Proof. Since each job of τ_i executes for its L-WCET (i.e., C_i^L) until its virtual deadline (i.e., $x_i \cdot T_i$), we have $\mathsf{DEM}_i(t) \le C_i^L/(x_i \cdot T_i) \cdot t = u_i^L/x_i \cdot t$. □

Based on Lemmas 2 and 3, we prove Theorem 1 by considering the collective demand.

Proof of Theorem 1. In the initial stage S_0, every HC task is in LO mode, and every LC task is in active mode. Then, we need to show that Eq. (8) holds:

$$\begin{aligned}\mathsf{DEM}(t) &= \sum_{\tau_i \in \tau_{\mathsf{L}}} \mathsf{DEM}_i(t) + \sum_{\tau_i \in \tau_{\mathsf{H}}} \mathsf{DEM}_i(t) \\ &= \sum_{\tau_i \in \tau_{\mathsf{L}}} u_i^A \cdot t + \sum_{\tau_i \in \tau_{\mathsf{H}}} u_i^L/x_i \cdot t \qquad \text{(by Lemmas 2 and 3)} \\ &\le \Big(U_{\mathsf{L}}^A + \sum_{\tau_i \in \tau_{\mathsf{H}}} u_i^L/x_i\Big) \cdot t,\end{aligned}$$

which is less than or equal to t since Eq. (7) holds. □

Next, we derive the schedulability condition under S_k (including the transitive stage from S_{k-1} to S_k), assuming the system is schedulable under stage S_{k-1}. From now on, we prove the following online schedulability condition, which needs runtime information in each stage.

Theorem 2. *Suppose a task set τ is schedulable by IMC-PnG under S_{k-1}. When the stage is changed to S_k, τ is schedulable by IMC-PnG if the following inequality holds.*

$$U_{\mathsf{L|A}(k)}^A + U_{\mathsf{L|D}(k)}^D + \sum_{\tau_i \in \tau_{\mathsf{H|L}(k)}} \frac{u_i^L}{x_i} + \sum_{\tau_i \in \tau_{\mathsf{H|H}(k)}} \frac{u_i^H - u_i^L}{1 - x_i} \le 1. \tag{9}$$

In Section 5.1, we express IMC-schedulability as $\mathcal{F}(\cdot)$. Now, we can set $\mathcal{F}(\cdot)$ in Eq. (6) to the left-hand-side of Eq. (9):

$$\mathcal{F}(\cdot) = U_{\mathsf{L|A}(k)}^A + U_{\mathsf{L|D}(k)}^D + \sum_{\tau_i \in \tau_{\mathsf{H|L}(k)}} \frac{u_i^L}{x_i} + \sum_{\tau_i \in \tau_{\mathsf{H|H}(k)}} \frac{u_i^H - u_i^L}{1 - x_i}.$$

As Theorem 1 does, proving Theorem 2 necessitates the calculation of resource demand. The amount of the resource demand of a task (shown in Definition 2) is changed by each mode-switch. That is, when a mode-switch happens for a HC task, the demand of the task is changed from L-WCET (C_i^L) to H-WCET (C_i^H). When the scheduler decides to degrade a LC task, the demand of the task is changed from the active WCET (C_i^A) to the degraded WCET (C_i^D).

The following lemma calculates the demand of a HC task after a mode-switch. In the lemma, we derive the demand of the HC task from the two facts: (1) the mode-switching job (denoted by J^*) of the HC task, whose demand until its VD at t^* is C_i^L, additionally requires an amount of $C_i^H - C_i^L$ (due to the mode-switch) until its real deadline; and (2) all the subsequent jobs after J^* require their C_i^H until their real deadlines, which are depicted in Fig. 5.

Lemma 4. *Suppose that a job of HC task τ_i, called J^*, mode-switches from* LO *to* HI. *Then, the following condition holds for time $t \ge t^*$:*

$$\mathsf{DEM}_i(t) \le \mathsf{DEM}_i(t^*) + \max\Big(u_i^H, \frac{u_i^H - u_i^L}{1 - x_i}\Big) \cdot (t - t^*), \tag{10}$$

where t^* is the VD of J^*.

Proof. Let r^* be the release time of J^*. Then, we can compute the VD and the deadline of J^* by $t^* = r^* + x_i \cdot T_i$ (equivalently $t^* = r^* + V_i$) and $d^* = r^* + T_i$, respectively. Even if J^* mode-switches at $[r^*, t^*)$, the demand until t^* (including that invoked by the current job J^*), $\mathsf{DEM}_i(t^*)$, does not change from LO mode because additional demand due to the mode-switch of J^* occurs at d^* (meaning that the demand changes at $2 \cdot T_i$, not $T_i + V_i$ as shown in Fig. 5). Then, $\mathsf{DEM}_i(t)$ until t^* can be upper-bounded by Lemma 3.

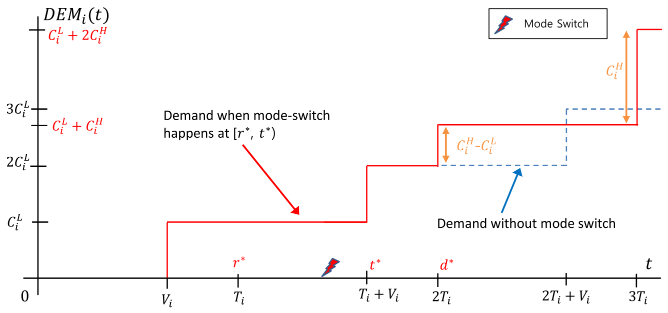


Fig. 5. The change of the resource demand for task τ_i over time t when a job of τ_i mode-switches within $[T_i, T_i + V_i)$, which is $[r^*, t^*)$ for mode-switching job J^*; the blue line indicates the demand before mode-switch, identical to the demand in Fig. 4, and the red line indicates the demand after mode-switch.

Consider $t^* \le t < d^*$. Since there is no additional demand before d^* (while additional demand occurs at d^*), the demand of τ_i until time t is no more than $\mathsf{DEM}_i(t^*)$, implying Eq. (10) holds.

Consider $t \ge d^*$. Due to the mode-switch of J^*, the additional demand from t^* to d^* is $C_i^H - C_i^L$, as shown in Fig. 5. For the subsequent jobs of τ_i (released in HI mode) after the mode-switch, each job executes for its H-WCET(C_i^H) until its real relative deadline T_i, which is also depicted in Fig. 5. Then,

$$\begin{aligned}\mathsf{DEM}_i(t) \;&\le\; \mathsf{DEM}_i(t^*) + (C_i^H - C_i^L) + C_i^H \cdot \frac{t - d^*}{T_i} \\ &= \mathsf{DEM}_i(t^*) + \frac{C_i^H - C_i^L}{T_i - x_i \cdot T_i} \cdot (d^* - t^*) + u_i^H \cdot (t - d^*) \\ &\quad (\because T_i - x_i \cdot T_i = d^* - t^*) \\ &\le \mathsf{DEM}_i(t^*) + \max\Big(u_i^H, \frac{u_i^H - u_i^L}{1 - x_i}\Big) \cdot (t - t^*). \\ &\quad (\because a \cdot (t - d^*) + b \cdot (d^* - t^*) \;\le\; \max(a, b) \cdot (t - t^*) \\ &\qquad\qquad \text{where } a > 0,\ b > 0)\end{aligned}$$ □

To derive a tighter schedulability condition, we present a constraint on the VD coefficient in Constraint 1 (to be presented), which makes it possible to narrow down its range[6] to $u_i^L/u_i^H \le x_i \le 1$. That is, the range that deviates from $u_i^L/u_i^H \le x_i \le 1$, which is the range of $x_i < u_i^L/u_i^H$, leads to unnecessary resource demand in LO mode without reducing resource demand in HI mode, to be proved in Lemma 5.

Lemma 5. *Consider a HC task τ_i and its VD coefficient $x_i^* = u_i^L/u_i^H$. There is no coefficient $x_i' < u_i^L/u_i^H$ that incurs less resource demand than x_i^*. In other words, the following inequality holds.*

$\forall t, \mathsf{DEM}_i^*(t) \le \mathsf{DEM}_i'(t)$ *holds,*

where $\mathsf{DEM}_i^*(t)$ and $\mathsf{DEM}_i'(t)$ are the demand of τ_i with $x_i^* = u_i^L/u_i^H$ and $x_i' < u_i^L/u_i^H$, respectively.

Proof. Divide cases depending on whether τ_i mode-switches.

(a) τ_i does not mode-switch: the task executes in LO-mode, applying Lemma 3. Since $x_i^* > x_i'$, we have $\mathsf{DEM}_i^*(t) \le \mathsf{DEM}_i'(t)$ for any t by Lemma 3.

(b) A job of τ_i, called J^*, mode-switches: the task switches its mode from LO-mode to HI-mode, applying Eq. (10) in Lemma 4.

We compare $\mathsf{DEM}_i^*(t)$ and $\mathsf{DEM}'(t)$. Let t^* be the VD of J^*. We have $\mathsf{DEM}_i^*(t) = \mathsf{DEM}_i^*(t^*) + u_i^H \cdot (t - t^*)$ because $\frac{u_i^H - u_i^L}{1 - x_i^*} = \frac{u_i^H - u_i^L}{1 - u_i^L/u_i^H} = u_i^H$ holds by Lemma 4. Next, we consider $\mathsf{DEM}'(t)$. If $u_i^L/x_i' > u_i^H$, we have $\frac{u_i^H - u_i^L}{1 - x_i'} <$

[6] The original range of VD coefficient is $0 \le x_i \le 1$ for task τ_i.

u_i^H, which is $u_i^H - u_i^L < u_i^H - u_i^H \cdot x_i$. Then, $\mathsf{DEM}'(t) = \mathsf{DEM}_i'(t^*) + u_i^H \cdot (t - t^*)$, by Lemma 4. Then, we have $\mathsf{DEM}_i^*(t) = \mathsf{DEM}'(t)$ in Case (b).

Therefore, we have $\mathsf{DEM}_i^*(t) \leq \mathsf{DEM}_i'(t)$ for any t for Cases (a) and (b). □

Lemma 5 enables to narrow down the range of the VD coefficient, recorded as follows.

Constraint 1. *For any HC task τ_i, the range of the VD coefficient is constrained to $u_i^L / u_i^H \leq x_i \leq 1$.*

Considering Constraint 1, we can simplify Lemma 4 into the next lemma.

Lemma 6. *With Constraint 1, suppose that a job of HC task τ_i, called J^*, mode-switches from* LO *to* HI. *Then, the following condition holds for $t \geq t^*$:*

$$\mathsf{DEM}_i(t) \leq \mathsf{DEM}_i(t^*) + \frac{u_i^H - u_i^L}{1 - x_i} \cdot (t - t^*), \tag{11}$$

where t^* is the VD of J^*.

Proof. From Lemma 4, we have Eq. (10). From Constraint 1, we have $\frac{u_i^H - u_i^L}{1 - x_i} \geq u_i^H$, which is equivalent to $u_i^H - u_i^L \geq u_i^H - u_i^H \cdot x_i$. Then, Eq. (10) can be rewritten as Eq. (11). □

The next lemma computes the demand of a LC task which is degraded at the mode-switching job (J^*) of some HC task. In the lemma, we drive the demand using the fact that the jobs with deadlines no later than t^* (i.e., the VD of J^*) execute for C_i^A in the worst case.

Lemma 7. *Suppose that LC task τ_i is degraded by the mode-switching job, called J^*. Then, the following conditions hold for $t \geq t^*$:*

$$\mathsf{DEM}_i(t) \leq \mathsf{DEM}_i(t^*) + u_i^D \cdot (t - t^*), \tag{12}$$

where t^* is the VD of J^*.

Proof. Although τ_i is degraded by J^*, the jobs of τ_i with deadlines no later than the VD of J^* (i.e., t^*) are already executed in active mode. Then, $\mathsf{DEM}_i(t^*)$ can be computed by Lemma 2. Jobs of τ_i with deadlines later than t^* will be executed in the degraded state, which implies $\mathsf{DEM}_i(t) \leq \mathsf{DEM}_i(t^*) + u_i^D(t - t^*)$, which is Eq. (12). □

Based on the demand calculation of different tasks with different modes (Lemmas 2, 3, 6, and 7), we prove Theorem 2.

Proof of Theorem 2. Let t^* be the VD of the last mode-switching job. Since τ is schedulable under S_{k-1} before the last mode-switch and the resource demand does not change before t^* (by Lemmas 6 and 7), we have $\sum_{\tau_i \in \tau} \mathsf{DEM}_i(t) \leq t$ for any $t \leq t^*$.

We need to prove that the task set is schedulable when $t > t^*$. Now, we show that $\sum_{\tau_i \in \tau} \mathsf{DEM}_i(t) \leq t$ for any $t > t^*$:

$$\begin{aligned}
&\sum_{\tau_i \in \tau} \mathsf{DEM}_i(t) \\
\leq &\sum_{\tau_i \in \tau} \mathsf{DEM}_i(t^*) + \\
&\left(U^A_{\mathsf{L|A}(k)} + \sum_{\tau_{\mathsf{H|L}(k)}} \frac{u_i^L}{x_i} + \sum_{\tau_{\mathsf{H|H}(k)}} \frac{u_i^H - u_i^L}{1 - x_i} + U^D_{\mathsf{L|D}(k)} \right) \cdot (t - t^*) \\
&\qquad\qquad\qquad (\textit{by Lemmas } 2, 3, 6, \textit{ and } 7) \\
\leq &\sum_{\tau_i \in \tau} \mathsf{DEM}_i(t^*) + (t - t^*) \qquad (\because \text{Eq. } (9) \textit{ holds}) \\
\leq &\; t^* + (t - t^*), \qquad (\textit{by the supposition of Theorem } 2) \\
= &\; t. \quad \square
\end{aligned}$$

By Theorem 2, τ is schedulable by IMC-PnG if IMC-PnG employs the task degrading algorithm associated with Eq. (9).

Offline Schedulability Analysis. Although we derived online schedulability analysis at each specific mode-switch situation, we need to develop offline schedulability analysis, which indicates whether a task set is IMC-schedulable (defined in Section 3) by IMC-PnG under any sequences of mode-switches. To this end, we need to find the worst-case conditions of all possible mode-switch sequences when the scheduler degrades LC tasks by the online schedulability test (i.e., Eq. (9)), recorded in the following theorem.

Theorem 3. *A task set τ is IMC-schedulable by IMC-PnG that employs the task degrading algorithm associated with Eq. (9), if the following two conditions hold:*

$$U^A_{\mathsf{L}} + \sum_{\tau_{\mathsf{H}}} \frac{u_i^L}{x_i} \leq 1, \tag{13}$$

$$U^D_{\mathsf{L}} + \sum_{\tau_{\mathsf{H}}} \frac{u_i^H - u_i^L}{1 - x_i} \leq 1. \tag{14}$$

Proof. For IMC schedulability, we need to show that conditions IMC-A, IMC-B, and IMC-C in Section 3 are satisfied. Divide cases depending on whether all HC tasks are in LO mode or not.

(a) All HC tasks are in LO mode: we need to satisfy conditions IMC-A and IMC-B. From Eq. (13), τ is schedulable on S_0 by Theorem 1, which corresponds conditions IMC-A and IMC-B.

(b) At least one HC task is in HI mode: we need to satisfy conditions IMC-A and IMC-C. We now show that Eq. (9) holds for $\tau_{\mathsf{H|H}(k)} \neq \emptyset$. The situation where all HC tasks are in HI mode yields the largest value for HC tasks to contribute to the left-hand-side of Eq. (9). From condition IMC-C, the smallest value for LC tasks to contribute to the left-hand-side of Eq. (9) occurs when all LC tasks' execution mode is "degraded". Considering that the degrading algorithm can adjust the execution mode of LC tasks based on the task mode of HC tasks, conditions IMC-A and IMC-C are satisfied if Eq. (14) holds. □

For scalability, we need to know the time-complexity of the offline schedulability test (i.e., Theorem 3) for IMC-PnG. For a given task set, Theorem 3 takes $O(n)$ because Eqs. (13) and (14) iterate all tasks, i.e., $O(n) + O(n) = O(n)$.

5.3. Advanced scheduling algorithm and its schedulability analysis with stable-HI-mode

Although we successfully derived the online/offline schedulability analysis in the previous subsection, there exists pessimism in the online schedulability analysis; that is, we may derive a tighter online schedulability analysis, if we address a decrease in resource demand of a HI-mode HC task after completing the mode-switch process. To this end, we divide the set of HI-mode HC tasks (i.e., $\tau_{\mathsf{H|H}(k)}$) and handle them differently, in order to maximize the number of fully serviced jobs for LC tasks.

We now revise the scheduling algorithm and its schedulability analysis. Depending the existence of mode-switching jobs, we divide HI-mode HC tasks into two groups, which are the group with mode-switching jobs and the group whose jobs are all released in HI-mode. We revise the stage (Definition 1 in Section 5.1) considering stable-HI-mode tasks, shown in Fig. 6; note that LC tasks τ_4 and τ_5 in Fig. 3 are omitted because they are identical to Fig. 3.

Definition 3 (*The Revised Stage with Stable-HI-mode Tasks*). We categorize HI-mode HC tasks ($\tau_{\mathsf{H|H}(k)}$) into two disjoint sets: transient-HI-mode tasks ($\tau_{\mathsf{H|HT}(k)}$) and stable-HI-mode tasks ($\tau_{\mathsf{H|HS}(k)}$). To distinguish $\tau_{\mathsf{H|HT}(k)}$ and $\tau_{\mathsf{H|HS}(k)}$, we define the stable-HI-mode flag, denoted by SH_i for each HC task τ_i: initially, set SH_i := `false`. If the current job is released in HI, which means that the task is now in stable-HI-mode, SH_i

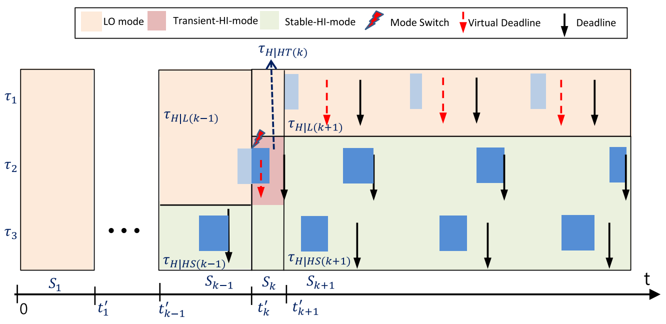


Fig. 6. Illustration of stable-HI-mode tasks and the revised stage; in this example, the mode-switching job belongs to $\tau_{\mathsf{H|HT}(k)}$ and all the subsequent jobs belong to $\tau_{\mathsf{H|HS}(k+1)}$ after t'_{k+1}, i.e., the $(k+1)$-th mode-switch time instant.

is set to `true`. When there are no active jobs in the ready queue (the idle state), the stable-HI-mode flag of the task is initialized (SH_i := `false`).

Let t'_k denote the time instant at which the kth event (the mode-switch or the change of SH_i for any HC task τ_i) occurs. The revised stage S'_k is defined as a tuple of five task sets in time interval $[t'_k, t'_{k+1})$, denoted by $(\tau_{\mathsf{H|L}(k)}, \tau_{\mathsf{H|HT}(k)}, \tau_{\mathsf{H|HS}(k)}, \tau_{\mathsf{L|A}(k)}, \tau_{\mathsf{L|D}(k)})$ as follows.

- $\tau_{\mathsf{H|HT}(k)}$ (transient-HI-mode tasks): a set of HC tasks whose task mode is HI and SH_i is `false` in $[t'_k, t'_{k+1})$, which means that the task is not yet in stable-HI-mode.
- $\tau_{\mathsf{H|HS}(k)}$ (stable-HI-mode tasks): a set of HC tasks whose task mode is HI and SH_i is `true` in $[t'_k, t'_{k+1})$.

Considering stable-HI-mode tasks, we revise three items (P1, P4 and P5) in the runtime scheduling policy of IMC-PnG as follows:

P1'. Initially, every LC task is set to active and every HC task is set to LO mode. For every HC task τ_i, set SH_i to `false`.

P4'. When a job of HC task τ_i in LO mode executes for more than its C_i^L (i.e., a mode-switch for τ_i occurs), run the mode-switch algorithm (i.e., Algorithm 1). When a mode-switching job of HC task τ_i completes its execution, set SH_i to `true`.

P5'. When no job of any task waits in the ready queue (when the system is idle), the system changes its state to the initial state, i.e., all LC tasks are set to active (M_i := A), all HC tasks are initialized (M_i := LO and SH_i :=`false`).

For stable-HI-mode tasks, we can tightly bound its resource demand. Since both $\tau_{\mathsf{H|HT}(k)}$ and $\tau_{\mathsf{H|HS}(k)}$ are subsets of $\tau_{\mathsf{H|H}(k)}$, we can compute the demand of $\tau_{\mathsf{H|HT}(k)}$ and $\tau_{\mathsf{H|HS}(k)}$ by Lemma 6 in Section 5.2. By using the characteristics of stable-HI-mode tasks, the next lemma presents the tight bound for the demand of $\tau_{\mathsf{H|HS}(k)}$. Although the job in transient-HI-mode tasks requires the remaining execution time $(C_i^H - C_i^L)$ until its remaining deadline after the mode-switch $(T_i - V_i)$, the job in stable-HI-mode tasks only needs to execute C_i^H until its real deadline (T_i), which allows tight resource demand calculation.

Lemma 8. *Consider HC task τ_i. Suppose that a job of HC task τ_i, called J^*, mode-switches from LO to HI and its subsequent jobs are released in HI mode (after $t \geq d^*$, the task belongs to stable-HI-mode tasks). Then, the following condition holds for time $t \geq d^*$:*

$$\mathsf{DEM}_i(t) \leq \mathsf{DEM}_i(d^*) + u_i^H \cdot (t - d^*), \tag{15}$$

where d^* is the deadline of J^*.

Proof. We can compute the demand until d^* (including the mode-switching job, J^*), i.e., $\mathsf{DEM}_i(d^*)$ by Lemma 6. We can tightly compute the task demand of jobs subsequent to J^*. The first job after the mode-switching job is released after d^* (the deadline of J^*) and each job executes for its H-WCET (C_i^H) until its real deadline T_i. Then,

$$\begin{aligned}\mathsf{DEM}_i(t) \leq\ & \mathsf{DEM}_i(d^*) + C_i^H \cdot \frac{t - d^*}{T_i} \\ =\ & \mathsf{DEM}_i(d^*) + u_i^H (t - d^*),\end{aligned}$$

implying Eq. (15) holds. □

By using the tight demand of stable-HI-mode tasks, we can improve the online schedulability, which leads to better runtime performance of LC tasks. The left term of Eq. (16) is always smaller than the left term of Eq. (9) in Theorem 2 because HC tasks entering stable-HI-mode in Eq. (16) demand less resources than those in Eq. (9).

Theorem 4. *Suppose a task set τ is schedulable by IMC-PnG under S'_{k-1}. When the stage is changed to S'_k, τ is schedulable by IMC-PnG if the following inequality holds.*

$$U^A_{\mathsf{L|A}(k)} + U^D_{\mathsf{L|D}(k)} + \sum_{\tau_i \in \tau_{\mathsf{H|L}(k)}} \frac{u_i^L}{x_i} + \sum_{\tau_i \in \tau_{\mathsf{H|HT}(k)}} \frac{u_i^H - u_i^L}{1 - x_i} + U^H_{\mathsf{H|HS}(k)} \leq 1. \tag{16}$$

Proof. Let t^* be either the VD of the last mode-switching job or its deadline. Since τ is schedulable before the last mode-switch and the resource demand does not change before t^*, we have $\sum_{\tau_i \in \tau} \mathsf{DEM}_i(t^*) \leq t^*$. To prove IMC-schedulability when $t > t^*$, we need to show that $\sum_{\tau_i \in \tau} \mathsf{DEM}_i(t) \leq t$ for $t > t^*$:

$$\begin{aligned}& \sum_{\tau_i \in \tau} \mathsf{DEM}_i(t) \\ \leq\ & \sum_{\tau_i \in \tau} \mathsf{DEM}_i(t^*) + (t - t^*) \cdot \\ & \left(U^A_{\mathsf{L|A}(k)} + \sum_{\tau_{\mathsf{H|L}(k)}} \frac{u_i^L}{x_i} + \sum_{\tau_{\mathsf{H|HT}(k)}} \frac{u_i^H - u_i^L}{1 - x_i} + U^D_{\mathsf{L|D}(k)} + U^H_{\mathsf{H|HS}(k)} \right) \\ & \qquad\qquad (\textit{by Lemmas } 2, 3, 6, 7, \textit{and} 8) \\ \leq\ & \sum_{\tau_i \in \tau} \mathsf{DEM}_i(t^*) + (t - t^*), \qquad (\because \text{Eq. } (16)\ \textit{holds})\end{aligned}$$

which is less than or equal to t with a similar step in the proof of Theorem 4. □

This advanced version of IMC-PnG does not change the worst-case scheduling pattern at runtime. Therefore, Theorem 3 is still applicable to determine the offline schedulability of IMC-PnG.

5.4. The assignment of virtual deadlines

In the previous subsection, we analyzed schedulability assuming that VD coefficients are given. Now, based on the analysis, we discuss how to assign VD coefficients for higher schedulability and better runtime performance. In the existing EDF-VD-IMC [22], a global VD coefficient x is applied to compute VD for all HC tasks: $V_i = x_i \cdot T_i$ where $x_i = x$. For higher schedulability than EDF-VD-IMC, we use different VD coefficients for different HC tasks, considering the task characteristics. Then, we transform the assignment of VD coefficients into an optimization problem. To construct the optimization problem, we will utilize Theorem 3.

Our goal is to find an optimal set of x_i to satisfy both Eqs. (13) and (14) in Theorem 3. To make a simple optimization problem, we introduce a variable z_i:

$$z_i \overset{\text{def}}{=} u_i^L / x_i.$$

Considering Constraint 1, the range of z_i is $u_i^L \leq z_i \leq u_i^H$. Replacing x_i with z_i in Eqs. (13), and (14), we construct the below optimization problem:

Definition 4 (*Assignment Problem of VD Coefficients*). Given a task set τ, we define a non-negative real number z_i for each task $\tau_i \in \tau_{\mathsf{H}}$. z_i^* is an optimal point of z_i on the following optimization problem:

$$\text{minimize} \quad W := \sum_{\tau_{\mathsf{H}}} \frac{z_i \cdot (u_i^H - u_i^L)}{z_i - u_i^L}, \tag{17}$$

$$\text{subject to} \quad U_{\mathsf{L}}^A + \sum_{\tau_{\mathsf{H}}} z_i \le 1, \quad \text{and} \tag{18}$$

$$\forall \tau_i \ , u_i^L \le z_i \le u_i^H. \tag{19}$$

We explain why solving the problem in Definition 4 is equivalent to finding the optimal VD coefficients. Eq. (17) is transformed from the second term of left-hand-side of Eq. (14) with z_i. Eq. (18) is transformed from Eq. (13) with z_i. For a feasible task set, the solution of the optimization problem (Definition 4) presents the value of W no larger than $(1 - U_{\mathsf{L}}^D)$, which satisfies Eq. (14). In the solution, we can find the set of z_i^*, which can be transformed into the set of x_i^*, i.e., the optimal set of x_i satisfying Eqs. (13) and (14) in Theorem 3 (IMC-schedulability).

By linear programming, a heuristic solution for the optimization problem could be found. Pathan [10] has used linear programming. For a similar optimization problem that assigns execution rates in a fluid scheduling model, Lee et al. [18,34] have found the optimal solution by using Lagrange Multiplier Method, which will be utilized for solving our optimization problem in Definition 4. The overview of our strategy for the optimal solution of the problem is to utilize the partial derivative of W for each z_i:

$$\frac{\partial W}{\partial z_i} = \frac{(1 - z_i) \cdot (u_i^H - u_i^L)}{(z_i - u_i^L)^2}.$$

Then, we increase z_i of the smallest $\partial W / \partial z_i$ from the initial condition ($\forall \tau_i \in \tau_{\mathsf{H}}, \ z_i = u_i^L$) until Eq. (18) holds.

We explain the detailed solution steps for our strategy. First, we set $z_i = u_i^L$ for each HC task. Then, the utilization slack for Eq. (18) is $(1 - U_{\mathsf{L}}^A + U_{\mathsf{H}}^L)$ since the left-hand-side of Eq. (18) is $(U_{\mathsf{L}}^A + U_{\mathsf{H}}^L)$. Second, we select τ_i with the smallest $\partial W / \partial z_i$ and increment z_i because $\partial W / \partial z_i$ is inversely proportional to z_i. Third, we repeat the previous step until the utilization slack is exhausted (i.e., Eq. (18) barely holds), which is the optimal point of z_i for the problem. More precise steps and the formal proof for our assignment algorithm can be founded in Lee et al. [34].

6. Evaluation

In this section, we evaluate performance of IMC-PnG compared to the existing approaches. We explain how to setup our simulations in Section 6.1, and present simulation results on IMC systems in Section 6.2. Finally, we present simulation results on classical MC systems where LC tasks can be dropped at mode-switch in Section 6.3.

6.1. Simulation setup

To evaluate IMC-PnG, we prepare randomly-generated synthetic workloads and utilize them for two ways. First, via offline schedulability test, we will evaluate IMC-PnG in terms of the offline schedulability, compared to existing approaches. Second, via runtime simulation, we will evaluate IMC-PnG in terms of the percentage of fully-serviced jobs for LC tasks.

Random Workload Generation. We generate random workload according to the workload-generation algorithm [3,11,12,35]. We define the upper bound of both collective utilization in LO mode and HI mode: $U^b = \max(U_{\mathsf{L}}^L + U_{\mathsf{H}}^L, U_{\mathsf{H}}^H)$. An individual task τ_i is generated as follows:

- Task period, T_i, is an integer chosen in the range $[20, 150]$.
- The ratio of HI-WCET over LO-WCET, R_i, is a real number drawn from the range $[1, 4]$.

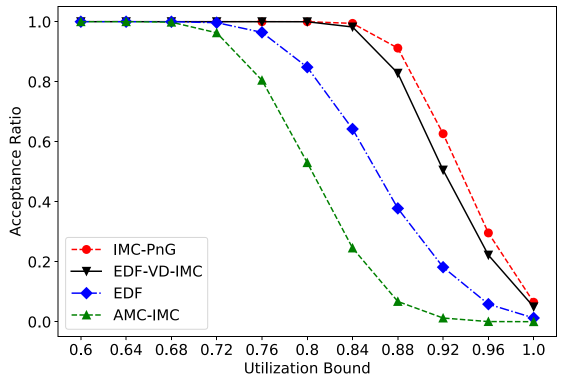


Fig. 7. Acceptance ratio of different scheduling algorithms varying utilization bound.

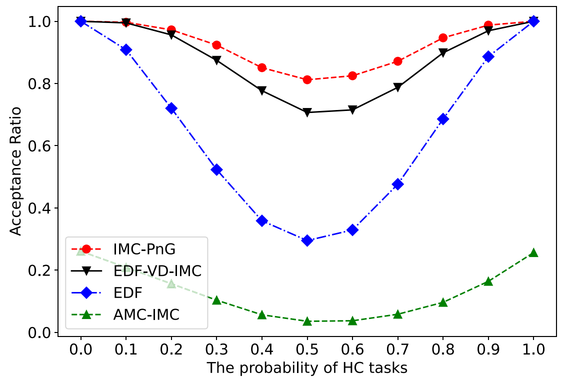


Fig. 8. Acceptance ratio of different scheduling algorithms varying P^{HC}.

- Task utilization, u_i, is a real number drawn from the range $[0.02, 0.2]$.
- The probability of being a HC task, P^{HC}, is set to 0.5 unless specified. We uniformly generate a real number in $[0.0, 1.0]$. If the number is not larger than P^{HC}, the task is a HC task; we set $\chi_i :=$ HI, $C_i^H := \lfloor u_i \cdot T_i \rfloor$, and $C_i^L := \lfloor u_i \cdot T_i / R_i \rfloor$. Otherwise (i.e., the number is larger than P^{HC}), the task is a LC task; we set $\chi_i :=$ LO, $C_i^A := \lfloor u_i \cdot T_i \rfloor$, and $C_i^D := \lfloor u_i \cdot T_i / R_i \rfloor$.

Note that all task parameters are randomly drawn in uniform distribution. Until $\max(U_{LC}^L + U_{HC}^L, \ U_{HC}^H) > U^b$ holds, repeat to generate a task and discard the lastly-added task. The number of workloads for each utilization bound is 5000.

Runtime Simulation. To evaluate runtime performance of different IMC approaches, we simulate the behavior of tasks for 32,000 time units.[7] In the simulation, the probability of mode-switch (P^{MS}) for each HC tasks's job is set to 0.1, and the duration of HI-mode (L) is set to 200 unless specified.

6.2. Simulation results on IMC systems

In this subsection, we evaluate IMC-PnG on IMC systems. In Fig. 7, we evaluate the performance of different algorithms in terms of the acceptance ratio for the randomly generated workloads. In Figs. 10 and 11, we also evaluate the runtime performance of different algorithms in terms of the Percentage of Fully-serviced Jobs (PFJ).

Acceptance Ratio. We compare IMC-PnG with EDF-VD-IMC [22] (the base algorithm of IMC-PnG), AMC-IMC [4] (the fixed priority

[7] The reason for this simulation duration is presented in the last experiment (associated with Fig. 12) in Section 6.2.

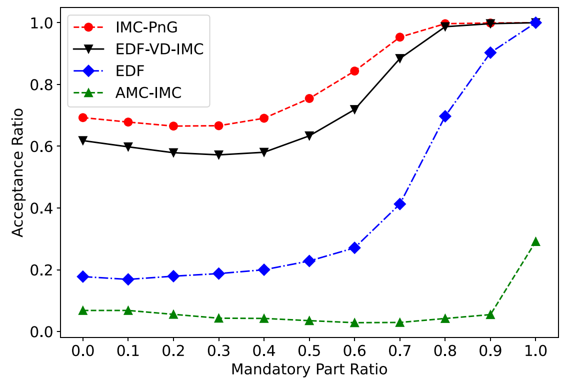


Fig. 9. Acceptance ratio of different scheduling algorithms varying the ratio of the mandatory part in LC task.

algorithm for IMC systems), and the vanilla EDF approach (the dynamic priority algorithm for non-MC real-time systems).

Fig. 7 shows the acceptance ratio (schedulable ratio) varying utilization bound U^b from 0.60 to 1.0 in step of 0.04. The acceptance ratio indicates the ratio of schedulable workloads over total randomly-generated workloads. The number of total workloads for Fig. 7 is 55,000 (=5000 × 11). The figure shows that AMC-IMC has the lowest acceptance ratio because it employs the fixed-priority approach. The (vanilla) EDF approach without considering IMC environment has a better acceptance ratio than AMC-IMC due to the superiority of EDF in terms of schedulability. The existing EDF-VD-IMC approach has a higher acceptance ratio than AMC-IMC and EDF because it schedules IMC tasks with their VDs and degrades all LC tasks at the mode-switch. Our IMC-PnG has up to 12.10% higher schedulability than EDF-VD-IMC approaches because it schedules IMC tasks with their VDs which are individually assigned to improve schedulability.

We also consider how the acceptance ratio is affected by various task parameters, such as the probability of being a HC task (i.e., P^{HC}) and the ratio of the mandatory part in LC task. Fig. 8 shows the acceptance ratio varying P^{HC} from 0.0 to 1.0 in step of 0.1. If P^{HC} is 0 or 1, the task set is non-MC workloads (only LC tasks or only HC tasks), where IMC-PnG, EDF-VD-IMC behave identical to EDF. As P^{HC} is more close to 0.5, we observed that workloads are more difficult to schedule, and IMC-PnG have better schedulability than other algorithms. Fig. 9 shows the acceptance ratio varying the ratio of the mandatory part (in LC tasks) from 0.0 to 1.0 in step of 0.1. When the ratio is 0, the workloads are classic MC workloads without the mandatory part in LC tasks. The results are consistent with the experimental result in classic MC systems (See Fig. 13). When the ratio is 1, the workloads are non-MC workloads (LC tasks are not degraded after mode-switch), where IMC-PnG and EDF-VD-IMC behave identical to EDF. When the ratio is between 0 and 1, IMC-PnG outperforms other algorithms under IMC workloads, e.g., IMC-PnG has 12.46% higher schedulability than EDF-VD-IMC when the ratio is 0.6.

Percentage of Fully-serviced Jobs. To evaluate the runtime performance of IMC-PnG, we consider two different versions of IMC-PnG:

- **IMC-PnG-B**: the basic version of IMC-PnG (Section 5.1–5.2) without considering stable-HI-mode tasks.
- **IMC-PnG-E**: the advanced version of IMC-PnG considering stable-HI-mode tasks and the revised stage (applying Section 5.3 on IMC-PnG).

We compare two versions of IMC-PnG with EDF-VD-IMC [22]. All the workloads are schedulable with EDF-VD-IMC and IMC-PnG.

Fig. 10 shows the PFJ varying utilization bound U^b (from 0.7 to 0.98 in the step of 0.4) for different L, the duration of HI-mode: $L = 0$, 200 and 400. The number of total workloads for Fig. 10 is 120,000, which comes from 5000 × 8 (the number of different U^b)×3 (the number of different L). When $L = 0$, we observed that IMC-PnG family (IMC-PnG-B and IMC-PnG-E) has up to 11.40% higher PFJ compared to EDF-VD-IMC. When $L = 200$ and $L = 400$, IMC-PnG-E has up to 31.75% and 42.10% higher PFJ compared to EDF-VD-IMC, respectively. This is because IMC-PnG family adopts task-level mode-switch and runtime mode-switch algorithm to degrade only necessary LC tasks at each mode-switch. We also observed that IMC-PnG-E shows up to 2.70% and 7.59% higher PFJ than IMC-PnG-B in the case of $L = 200$ and $L = 400$, respectively. This is because the revised runtime scheduling policy of IMC-PnG-E with the stable-HI-mode tasks effectively reduces degrading the LC tasks at mode-switch, especially for a longer duration of HI mode.

Fig. 11 shows the PFJ varying utilization bound U^b (from 0.7 to 0.98 in the step of 0.4) for different P^{MS}: $P^{MS} = 0.05$, 0.10 and 0.15. The number of total workloads for Fig. 11 is 120,000 (=5000 × 8 × 3). In the result, we observed that IMC-PnG-E has up to 21.45%, 31.57% and 37.83% higher PFJ compared to EDF-VD-IMC for $P^{MS} = 0.05$, 0.10 and 0.15, respectively. There is a larger performance gap between IMC-PnG-E and EDF-VD-IMC in higher P^{MS}. Also, IMC-PnG-E shows up to 0.95%, 2.73% and 4.54% higher PFJ compared to IMC-PnG-B for $P^{MS} = 0.05$, 0.10 and 0.15, respectively. This is because the IMC-PnG-E effectively reduces degrading the LC tasks, especially for a higher P^{MS}.

Fig. 12 shows the PFJ under different simulation duration: 4000, 8000, 16000, 32000, 64000, and 128000. This simulation is executed with $U^b = 0.9$. The result shows that the PFJs of algorithms are converged for larger simulation durations (≥ 8000), which is also observed in similar existing work [13]. Therefore, we choose 32,000 time units as the default simulation duration.

6.3. Simulation results on classical MC systems

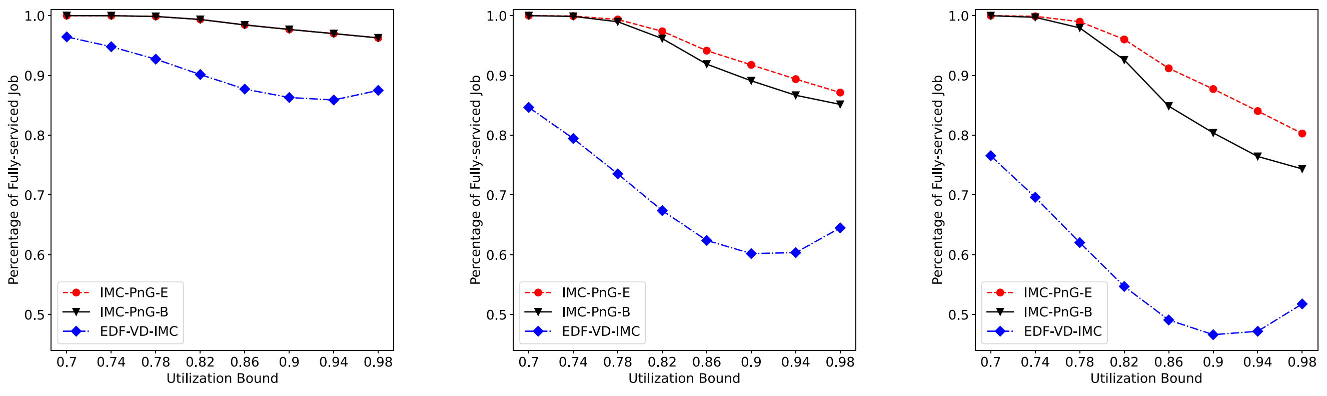
In the previous subsection, we evaluate IMC-PnG on IMC systems. Now, we evaluate IMC-PnG on classical MC systems. We compare IMC-PnG with existing classical MC approaches: MC-Fluid [18] (fluid-based scheduling approach), MC-ADAPT [11] (a dynamic-priority approach with task-level mode-switch),[8] FMC [12] (another dynamic-priority approach with task-level mode-switch), EDF-VD [3] (a dynamic-priority approach with system-level mode-switch), and AMC [2] (a fixed-priority approach).

Acceptance Ratio. We evaluate the performance of IMC-PnG in terms of the acceptance ratio for the randomly generated workloads. Fig. 13 shows the acceptance ratio varying utilization bound U^b from 0.60 to 1.0 in the step of 0.04. The number of total workloads for Fig. 13 is 55,000 (=5000 × 11). The figure shows that AMC has a low acceptance ratio in higher utilization ($U^b \geq 0.84$) because it employs the fixed-priority approach, not EDF. The FMC approach has the lowest acceptance ratio among the dynamic-priority approaches because it focuses on reducing the deadline miss ratio, not schedulability. MC-ADAPT and IMC-PnG outperform EDF-VD because they adopt different VDs for different tasks. Our IMC-PnG shows up to 3.50% and 8.26% higher acceptance ratio compared to MC-ADAPT and EDF-VD, respectively. This is because the optimal VD assignment strategy (Section 5.4) improves schedulability. Although MC-Fluid has the same schedulability as IMC-PnG, fluid-based scheduling strategy in MC-Fluid is not practical because it incurs a large context switching overheads.

7. Discussion

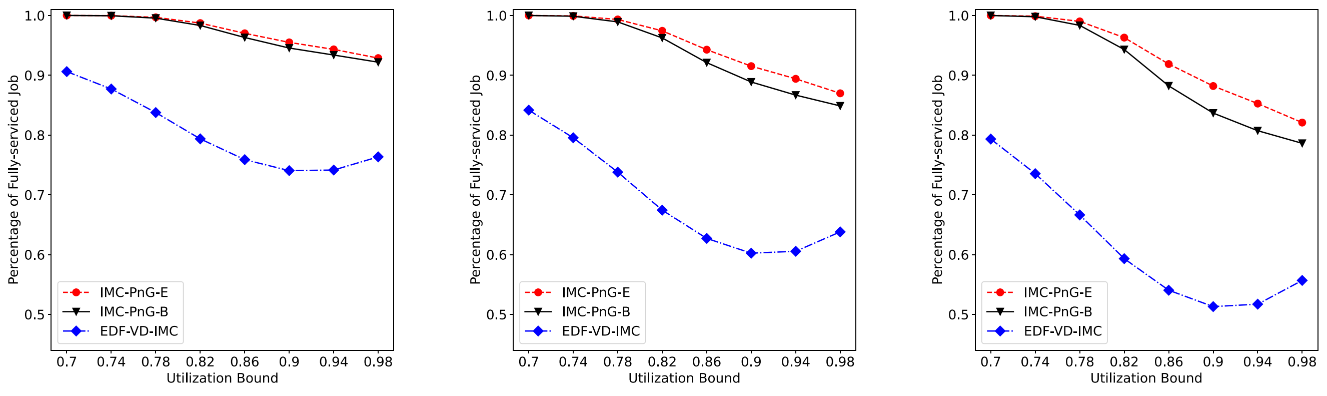
Extension to Multi-criticality Systems and Multicore Systems. One direction to extend IMC-PnG is toward multi-criticality systems. As preliminary work, we only consider dual-criticality systems. We will extend IMC-PnG in consideration of multi-criticality IMC tasks (e.g., HC,

[8] MC-FLEX [13] has the same offline schedulability as MC-ADAPT.



(a) $L = 0$ (b) $L = 200$ (default) (c) $L = 400$

Fig. 10. PFJ comparison varying the duration of HI-mode (L).



(a) $P^{MS} = 0.05$ (b) $P^{MS} = 0.10$ (default) (c) $P^{MS} = 0.15$

Fig. 11. PFJ comparison varying P^{MS}.

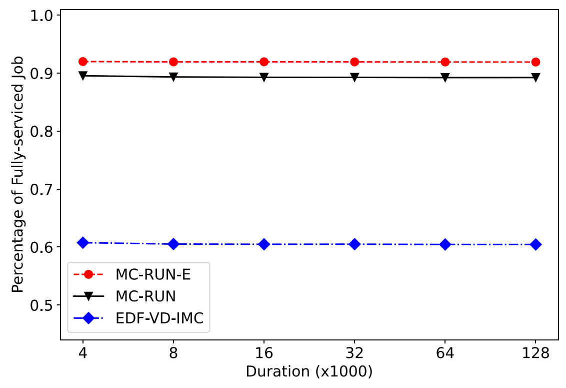


Fig. 12. PFJ comparison varying simulation duration.

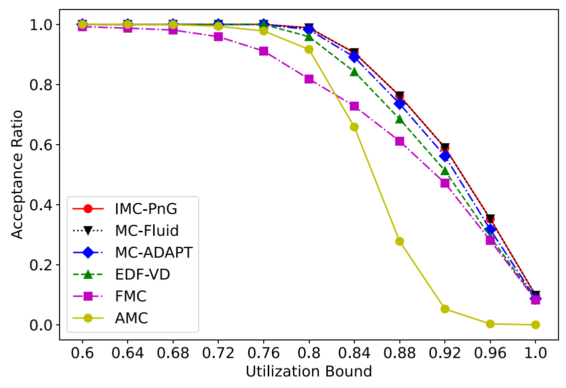


Fig. 13. Acceptance ratio of different scheduling algorithms under varying utilization bound (classical MC systems).

MC (Middle-Criticality), and LC tasks) with multiple WCET estimate (e.g., L-WCET, M-WCET, and H-WCET). Then, we need to revise the system goals and Theorems 3 and 4 according to the revised system model.

Another extension direction is toward multicore systems. One simple extension is the partitioned scheduling approach by partitioning tasks into cores. We may migrate LC tasks into under-utilized cores to maximize the PFJ of LC tasks.

Practical Issues. One practical issue of IMC-PnG is runtime overheads, which mainly consists of the EDF-VD scheduling algorithm and the runtime mode-switch algorithm (i.e., Algorithm 1). According to Liu et al. [23], the EDF-VD scheduler can be implemented through the small modification of the vanilla EDF scheduler, which exhibits $O(1)$ time-complexity. At every mode switch, the mode-switch algorithm takes $O(n)$, which is described in Section 5.1. In future, we will investigate the implementation overheads of IMC-PnG.

Another issue is the switch-back protocol to maximize runtime PFJ performance of LC tasks. Currently, the system resets the execution mode of all LC tasks at the system idle event, which is pessimistic in terms of PFJ. We plan to apply the task-level switch-back protocol into IMC-PnG, which is introduced in MC-Flex [13]. This task-level switch-back protocol can re-activate the optional execution parts of the degraded LC task as early as possible.

8. Conclusion

We present the IMC-PnG scheduling framework to provide a certain level of timing guarantee for low-critical tasks while maximizing the execution of low-critical tasks under MC systems. To this end, we propose IMC-PnG that employs individual VD assignment and online mode-change algorithm on IMC systems. We also present online and offline schedulability analysis under task-level runtime criticality mode. Finally, we present an individual VD assignment algorithm to maximize schedulability and runtime performance. Via simulation, we demonstrated that our approach has up to 12.10% higher schedulability and up to 42.10% higher PFJ than the existing approaches. To overcome the limitation of this work (as a theoretical scheduling framework), we need to investigate diverse aspects of IMC-PnG in terms of implementation on real platforms. In future work, we plan to conduct a case-study with autonomous driving vehicles, by applying IMC-PnG to autonomous driving platforms such as AUTOWARE [36] and find practical issues on IMC-PnG. Also, we plan to extend our work considering energy consumption and more expressive system models such as the constrained-deadline task model.

CRediT authorship contribution statement

Jaewoo Lee: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Conceptualization. **Jinkyu Lee:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Conceptualization.

Declaration of competing interest

None.

Data availability

No new data were created or analysed in this study. Data sharing is not applicable to this article.

Acknowledgments

This research was supported in part by Basic Science Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2022R1F1A1062660). This study was also supported in part by the BK21 FOUR program(Education and Research Center for Securing Cyber-Physical Space) through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(5199990314137). This work was also supported by the National Research Foundation of Korea(NRF), South Korea grant (2022R1A4A3 018824) funded by the Korea government (MSIT).

References

[1] S. Vestal, Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance, in: Real Time System Symposium, RTSS, 2007, pp. 239–243.
[2] S. Baruah, A. Burns, R. Davis, Response-time analysis for mixed criticality systems, in: Real Time System Symposium, RTSS, 2011, pp. 34–43.
[3] S. Baruah, V. Bonifaci, G. D''Angelo, H. Li, A. Marchetti-Spaccamela, S. Van der Ster, L. Stougie, The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems, in: Euromicro Conference on Real-Time Systems, ECRTS, 2012, pp. 145–154.
[4] A. Burns, S. Baruah, Towards a more practical model for mixed criticality systems, in: Workshop of Mixed Criticality Systems, WMC, 2013, pp. 1–6.
[5] M. Jan, L. Zaourar, M. Pitel, Maximizing the execution rate of low-criticality tasks in mixed criticality systems, in: Workshop of Mixed Criticality Systems, WMC, 2013, pp. 1–6.
[6] H. Su, D. Zhu, An elastic mixed-criticality task model and its scheduling algorithm, in: Design, Automation, and Test in Europe, DATE, 2013, pp. 147–152.
[7] O. Gettings, S. Quinton, R.I. Davis, Mixed criticality systems with weakly-hard constraints, in: Real-Time Networks and Systems, RTNS, 2015, pp. 237–246.
[8] J. Liu, W.-K. Shih, K.-J. Lin, R. Bettati, J.-Y. Chung, Imprecise computations, Proc. IEEE 82 (1) (1994) 83–94, http://dx.doi.org/10.1109/5.259428.
[9] S. Baruah, A. Burns, Z. Guo, Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors, in: 2016 28th Euromicro Conference on Real-Time Systems, ECRTS, 2016, pp. 131–138, http://dx.doi.org/10.1109/ECRTS.2016.12.
[10] R.M. Pathan, Improving the quality-of-service for scheduling mixed-criticality systems on multiprocessors, in: M. Bertogna (Ed.), 29th Euromicro Conference on Real-Time Systems, ECRTS 2017, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 76, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017, pp. 19:1–19:22, http://dx.doi.org/10.4230/LIPIcs.ECRTS.2017.19.
[11] J. Lee, H.S. Chwa, L.T.X. Phan, I. Shin, I. Lee, MC-ADAPT: Adaptive task dropping in mixed-criticality scheduling, ACM Trans. Embed. Comput. Syst. 16 (5s) (2017) 163:1–163:21, http://dx.doi.org/10.1145/3126498.
[12] G. Chen, N. Guan, D. Liu, Q. He, K. Huang, T. Stefanov, W. Yi, Utilization-based scheduling of flexible mixed-criticality real-time tasks, IEEE Trans. Comput. 67 (4) (2018) 543–558, http://dx.doi.org/10.1109/TC.2017.2763133.
[13] J. Lee, J. Lee, MC-FLEX: Flexible mixed-criticality real-time scheduling by task-level mode switch, IEEE Trans. Comput. 71 (8) (2022) 1889–1902, http://dx.doi.org/10.1109/TC.2021.3111743.
[14] J. Redmon, A. Farhadi, YOLOv3: An incremental improvement, 2018, pp. 1–6, arXiv.org, arXiv:1804.02767v1, URL http://arxiv.org/abs/1804.02767v1.
[15] Y. Hu, S. Liu, T. Abdelzaher, M. Wigness, P. David, On exploring image resizing for optimizing criticality-based machine perception, in: 2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA, 2021, pp. 169–178, http://dx.doi.org/10.1109/RTCSA52859.2021.00027.
[16] W. Kang, S. Chung, J.Y. Kim, Y. Lee, K. Lee, J. Lee, K.G. Shin, H.S. Chwa, DNN-SAM: Split-and-merge DNN execution for real-time object detection, in: 2022 IEEE 28th Real-Time and Embedded Technology and Applications Symposium, RTAS, 2022, pp. 160–172, http://dx.doi.org/10.1109/RTAS54340.2022.00021.
[17] A. Burns, R.I. Davis, A survey of research into mixed criticality systems, ACM Comput. Surv. 50 (6) (2017) http://dx.doi.org/10.1145/3131347.
[18] J. Lee, K.-M. Phan, X. Gu, J. Lee, A. Easwaran, I. Shin, I. Lee, MC-Fluid: Fluid model-based mixed-criticality scheduling on multiprocessors, in: 2014 IEEE Real-Time Systems Symposium, 2014, pp. 41–52, http://dx.doi.org/10.1109/RTSS.2014.32.
[19] K. Yang, A. Bhuiyan, Z. Guo, F2VD: Fluid rates to virtual deadlines for precise mixed-criticality scheduling on a varying-speed processor, in: 2020 IEEE/ACM International Conference on Computer Aided Design, ICCAD, 2020, pp. 1–9.
[20] H. Su, N. Guan, D. Zhu, Service guarantee exploration for mixed-criticality systems, in: 2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications, 2014, pp. 1–10, http://dx.doi.org/10.1109/RTCSA.2014.6910499.
[21] X. Gu, A. Easwaran, Dynamic budget management with service guarantees for mixed-criticality systems, in: Real Time System Symposium, RTSS, 2016, pp. 47–56.
[22] D. Liu, J. Spasic, N. Guan, G. Chen, S. Liu, T. Stefanov, W. Yi, EDF-VD scheduling of mixed-criticality systems with degraded quality guarantees, in: Real Time System Symposium, RTSS, 2016, pp. 35–46.
[23] D. Liu, N. Guan, J. Spasic, G. Chen, S. Liu, T. Stefanov, W. Yi, Scheduling analysis of imprecise mixed-criticality real-time tasks, IEEE Trans. Comput. 67 (7) (2018) 975–991, http://dx.doi.org/10.1109/TC.2018.2789879.
[24] Z. Guo, S. Baruah, Mixed-criticality scheduling upon varying-speed multiprocessors, in: 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, 2014, pp. 237–244, http://dx.doi.org/10.1109/DASC.2014.50.
[25] P. Huang, P. Kumar, G. Giannopoulou, L. Thiele, Run and be safe: Mixed-criticality scheduling with temporary processor speedup, in: 2015 Design, Automation and Test in Europe Conference and Exhibition, DATE, 2015, pp. 1329–1334.
[26] T. She, S. Vaidhun, Q. Gu, S. Das, Z. Guo, K. Yang, Precise scheduling of mixed-criticality tasks on varying-speed multiprocessors, in: Proceedings of the 29th International Conference on Real-Time Networks and Systems, RTNS '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 134–143, http://dx.doi.org/10.1145/3453417.3453428.
[27] Y.-W. Zhang, R.-K. Chen, Energy-efficient scheduling of imprecise mixed-criticality real-time tasks based on genetic algorithm, J. Syst. Archit. 143 (2023) 102980, http://dx.doi.org/10.1016/j.sysarc.2023.102980.
[28] H.-M. Huang, C. Gill, C. Lu, Implementation and evaluation of mixed-criticality scheduling approaches for periodic tasks, in: 2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium, 2012, pp. 23–32, http://dx.doi.org/10.1109/RTAS.2012.16.

[29] J.M. Calandrino, H. Leontyev, A. Block, U.C. Devi, J.H. Anderson, Litmus-RT : A testbed for empirically comparing real-time multiprocessor schedulers, in: 2006 27th IEEE International Real-Time Systems Symposium, RTSS'06, 2006, pp. 111–126, http://dx.doi.org/10.1109/RTSS.2006.27.

[30] R.I. Davis, S. Altmeyer, A. Burns, Mixed criticality systems with varying context switch costs, in: 2018 IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS, 2018, pp. 140–151, http://dx.doi.org/10.1109/RTAS.2018.00024.

[31] V.K. Sundar, A. Easwaran, A practical degradation model for mixed-criticality systems, in: 2019 IEEE 22nd International Symposium on Real-Time Distributed Computing, ISORC, 2019, pp. 171–180, http://dx.doi.org/10.1109/ISORC.2019.00040.

[32] J. Liu, K.-J. Lin, W.-K. Shih, A.-s. Yu, J.-Y. Chung, W. Zhao, Algorithms for scheduling imprecise computations, Computer 24 (5) (1991) 58–68, http://dx.doi.org/10.1109/2.76287.

[33] S. Baruah, A. Mok, L. Rosier, Preemptively scheduling hard-real-time sporadic tasks on one processor, in: Real-Time Systems Symposium, 1990. Proceedings., 11th, 1990, pp. 182–190, http://dx.doi.org/10.1109/REAL.1990.128746.

[34] J. Lee, S. Ramanathan, K.-M. Phan, A. Easwaran, I. Shin, I. Lee, MC-Fluid: Multi-core fluid-based mixed-criticality scheduling, IEEE Trans. Comput. 67 (4) (2018) 469–483, http://dx.doi.org/10.1109/TC.2017.2759765.

[35] X. Gu, A. Easwaran, K.-M. Phan, I. Shin, Resource efficient isolation mechanisms in mixed-criticality scheduling, in: Euromicro Conference on Real-Time Systems, ECRTS, 2015, pp. 13–24.

[36] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, T. Azumi, Autoware on board: Enabling autonomous vehicles with embedded systems, in: 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems, ICCPS, 2018, pp. 287–296.

Jaewoo Lee (jaewoolee@cau.ac.kr) received his B.S. and M.S. from Seoul National University in 2006 and 2008, respectively. He received his Ph.D. from University of Pennsylvania in 2017. He is currently an associate professor in the department of industrial security, Chung-Ang University. His research interests include cyber–physical systems, real-time embedded systems, and security systems.

Jinkyu Lee (jinkyu.lee@skku.edu) is an associate professor in Department of Computer Science and Engineering, Sungkyunkwan University (SKKU) Republic of Korea, where he joined in 2014. He received the B.S., M.S., and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea, in 2004, 2006, and 2011, respectively. He has been a visiting scholar/research fellow in the Department of Electrical Engineering and Computer Science, University of Michigan, U.S.A. in 2011–2014. His research interests include system design and analysis with timing guarantees, QoS support, and resource management in real-time embedded systems, mobile systems, and cyber–physical systems. He won the best student paper award from the 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) in 2011, and the Best Paper Award from the 33rd IEEE Real-Time Systems Symposium (RTSS) in 2012.