MCED-H: An efficient scheduling policy for energy harvesting mixed-criticality real-time systems

Mostafa Tamimipour, Hakem Beitollahi, Maryline Chetto

MCED-H: An efficient scheduling policy for energy harvesting mixed-criticality real-time systems

Mostafa Tamimipour[a], Hakem Beitollahi[a,b], Maryline Chetto[c]

[a] *School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran*
[b] *Department of Computer Science, Soran University, Soran 44008, Kurdistan Region, Iraq*
[c] *Laboratory of Computer Science (LS2N), University of Nantes, Nantes, France*

Real-time systems traditionally rely on batteries, which have limited lifespans. Energy harvesting offers a sustainable alternative but introduces challenges due to the unpredictable nature of environmental energy sources. Integrating energy harvesting with mixed-criticality (MC) features further complicates system design. This paper presents the MCED-H algorithm, a novel approach for scheduling MC tasks in energy-constrained environments. MCED-H (Mixed-Criticality ED-H) is an enhanced scheduling algorithm based on ED-H, which itself is an extension of EDF for energy-harvesting systems, tailored specifically for mixed-criticality real-time applications. By employing virtual deadlines and adaptive mode switching, MCED-H effectively manages energy resources and ensures the timely completion of critical tasks. Experimental results demonstrate significant improvements in system reliability and energy efficiency compared to state-of-the-art techniques.

1. Introduction

Many embedded systems rely on batteries, which have limited lifespans, leading to frequent replacements and associated costs [1]. For example, replacing batteries in implanted devices in the medical sector often requires invasive procedures [2]. Energy harvesting has emerged as a promising alternative to address these challenges [3]. By capturing ambient energy from solar, wind, and thermal energy, energy harvesting can extend battery life or even eliminate the need for batteries [4]. However, energy harvesting systems face challenges such as energy variability and conversion losses [5].

Vestal introduced the concept of Mixed Criticality (MC) systems in 2007 [6], where tasks are classified based on their criticality levels and the consequences of missing deadlines [7]. For example, missing dead lines for a task like air conditioning might degrade system performance. In contrast, missing deadlines for safety critical tasks in automotive or aerospace systems can have catastrophic consequences [8]. To address these issues, standards such as ARINC 653 and ISO 26262 have been developed [9,10]. In this work, we focus on two criticality levels: high criticality level (HI) and low criticality level (LO). HI tasks require higher execution accuracy and have longer execution times in critical modes. Scheduling MC systems is complex due to the uncertainty of mode transitions and the stringent timing requirements of HI tasks [7]. According to the DO 178B standard, there are five assurance levels ranging from Level A (the most critical) to Level E (the least critical). In this paper, LO tasks are assumed to correspond to Levels D or E, where their suspension in high criticality mode is considered acceptable under certification standards [11].

Low power techniques such as Dynamic Voltage and Frequency Scaling (DVFS) and Dynamic Power Management (DPM) have been employed to reduce energy consumption in real time systems. DVFS involves adjusting the processor's voltage and frequency to match the computational demands of tasks, while DPM optimizes power consump tion by selectively powering down system components. However, these techniques primarily focus on managing energy consumption within a fixed power budget [12,13].

This research investigates the scheduling issue in MC systems that integrate energy harvesting technology to improve system lifespan and potentially eliminate batteries. Mixed Critical Energy Harvesting (MCEH) systems face unique challenges, including the unpredictable nature of energy sources, the stringent timing requirements of high criticality tasks, and the dynamic power consumption patterns resulting from mode transitions. This paper proposes a novel scheduling al gorithm to address these challenges by guaranteeing a reliable and energy neutral system operational mode.

The key contributions can be summarized as follows:

* Corresponding author at: Department of Computer Science, Soran University, Soran 44008, Kurdistan Region, Iraq.
E-mail addresses: mostafa tamimipour@comp.iust.ac.ir (M. Tamimipour), beitollahi@iust.ac.ir, hakem.beitollahi@soran.edu.iq (H. Beitollahi), maryline.chetto@univ-nantes.fr (M. Chetto).

- **Integration of Energy Harvesting and Mixed-Criticality:** The MCED H algorithm simultaneously considers both time and en ergy constraints, a novel approach compared to traditional MC and energy harvesting systems.
- **New Virtual Deadlines for HI Tasks:** Introducing virtual dead lines for HI tasks is a novel approach to improve task schedula bility and energy efficiency.
- **Energy-Aware Task Selection:** The algorithm prioritizes the ex ecution of LO tasks in HI mode based on energy usage, optimizing resource utilization.
- **Mode Switching Criteria:** Using slack time and slack energy as indicators for mode transitions enhances system flexibility.

This paper is organized as follows: Section 2 explores the related works. Section 3 presents the system model. Section 4 describes the notation and definitions. Section 5 discusses previous algorithms used in the proposed method and the behavior of the system. In Section 6, we explain our algorithm in each mode after commenting a motiva tional example. Section 7 shows the experimental results. Section 8 provides a detailed discussion of the limitations and potential areas for improvement. Finally, Section 9 concludes the paper.

2. Related work

Research on energy harvesting and real time systems has gained significant attention in recent years. While advancements have been made in both fields, integrating energy harvesting into MC real time systems still needs to be explored. This paper addresses this gap by proposing a novel scheduling algorithm. We first review research on energy harvesting systems and then discuss previous works on MC systems, restricted to uniprocessor platforms.

Early research in energy harvesting focused on optimizing energy consumption and storage. For example, the Lazy algorithm, named LSA and introduced in 2007, aimed to maximize energy efficiency by keeping the system idle when possible [14]. LSA uses the EDF (Earliest Deadline First) scheduler [15]. PFP_{ASAP}, a fixed priority scheduling algorithm proposed in 2013 [16], assumes linear energy consumption and incorporates a schedulability test. It has been demonstrated to be an optimal algorithm under the given system assumptions. A new algorithm called ED H was introduced in 2014, which builds upon EDF by incorporating task scheduling based on time and energy slack. ED H is a clairvoyant algorithm, meaning it assumes perfect knowledge of future task arrivals to permit procrastination. While ED H demonstrates optimal performance under ideal conditions, its clairvoyant nature limits its applicability to real world scenarios [17].

EGDVFS [18], introduced in 2019, extends the ED H algorithm by incorporating DVFS to reduce energy consumption further. By dy namically adjusting the processor's voltage and frequency, EGDVFS can save energy, especially during periods of low energy availability. However, using DVFS may introduce additional overhead and impact task execution times. The Celebi algorithm [19], introduced in 2020, offers a different perspective on energy harvesting task scheduling by categorizing tasks into computational tasks, energy consuming tasks, and tasks that cannot be executed due to energy constraints. Unlike previous algorithms, Celebi operates in an offline mode, allowing for pre processing and optimization of task schedules. However, the offline nature of Celebi may limit its applicability to dynamic environments with unpredictable energy availability.

In 2021, [20] proposed a method to determine the required charg ing time before task execution when the available energy is insufficient. The author further establishes schedulability tests for non preemptive task sets scheduled using EDF. In 2024, EH DM was introduced and presented a scheduling algorithm for real time energy harvesting sys tems, focusing on energy efficiency and battery lifespan extension [21]. The proposed approach combines fixed priority scheduling based on the deadline monotonic policy with DVFS to manage energy consump tion effectively. It optimizes energy usage, reduces consumption, and extends the operational life of energy harvesting systems. In 2024, [22] explores the use of Integer Linear Programming to optimize preemp tion points in real time tasks for energy harvesting systems. The goal is to balance real time constraints with energy availability, enabling autonomous IoT devices with minimal maintenance. In 2025 [23] presents a novel scheduling approach for energy harvesting real time systems powered by ambient energy. It introduces a limited preemp tion model that balances preemptive and non preemptive execution while ensuring energy and timing constraints are met. An EDF based scheduler is developed, integrating energy availability checks to en hance efficiency. Additionally, a checkpointing strategy is proposed to minimize overhead and prevent energy starvation.

Vestal introduced the concept of MC systems in 2007 [6]. MC sys tems categorize tasks based on criticality levels, reflecting the severity of consequences if task deadlines are missed. This approach addresses the limitations of traditional real time systems by providing a frame work for handling tasks with different importance levels. The EDF VD algorithm [24], introduced in 2011, extends the EDF scheduling policy to handle MC systems by assigning virtual deadlines to high criticality tasks. By reserving time for potential high criticality workload in the low criticality mode, EDF VD aims to improve schedulability and pre vent deadline misses during mode transitions.

The Semi Clairvoyant concept introduced in 2019 offers a different approach to handling MC systems by assuming prior knowledge of task execution times in low criticality mode (LO mode) and high criticality mode (HI mode) upon task arrival [25]. While this approach can improve scheduling performance, it relies on accurate predictions of task behavior, which may not be feasible in real world scenarios. The MCQPA algorithm [26], introduced in 2022, proposes a test based approach to scheduling MC systems. By evaluating task sets to determine potential deadline misses during mode transitions, MCQPA aims to improve system reliability. The EAU algorithm [27], proposed in 2022, combines EDF VD with DVFS to address both schedulability and energy efficiency in MC systems. By adapting the system's speed to the actual workload, EAU aims to reduce energy consumption while meeting task deadlines. However, the effectiveness of EAU may be influenced by the accuracy of workload predictions and the overhead associated with DVFS.

A recent study [28] introduced in 2024 a reward based resource reservation model to address this issue. Instead of completely dis carding LO tasks in HI mode, this method allocates resources dy namically based on a reward driven mechanism, ensuring a better balance between HI and LO task execution. This approach enhances overall system efficiency by employing a resource reservation mech anism that adjusts CPU time and energy distribution according to task priority and system conditions. NPFP, introduced in 2025, in cludes a semi clairvoyant mechanism, enabling improved scheduling predictions [29]. The method prioritizes tasks based on estimated exe cution costs and their impact on system performance, thereby reducing deadline misses, especially for high priority tasks. The key advan tage of this approach is its ability to mitigate delays in critical task execution without requiring complete execution time predictability. However, it introduces additional computational overhead and does not always guarantee optimal scheduling due to the inherent uncertainty in execution estimates.

A common approach in MC systems (all above mentioned tech niques) is to suspend LO tasks when the system enters HI mode. While this approach prioritizes the execution of critical tasks, it can lead to reduced system performance and resource underutilization. To mitigate these issues, various techniques have been proposed, including Im precise Mixed Criticality (IMC), Elastic Mixed Criticality (EMC), Flex ible Mixed Criticality (FMC), and Precise Mixed Criticality (PMC) [30]. However, these methods primarily focus on task scheduling and do not explicitly consider energy constraints, which are crucial in energy

Table 1
Comparison of this work with previous research on uniprocessor real-time scheduling considering energy harvesting and mixed-criticality systems.

Years	Scheduling algorithm & references	Energy harvesting	Mixed criticality	Application model
2007	PFP [6]	×	✓	Periodic
2007	Lazy [14]	✓	×	Periodic, Sporadic
2011	EDF-VD [24]	×	✓	Sporadic
2013	PFP_{ASAP} [16]	✓	×	Periodic
2014	ED-H [17]	✓	×	Periodic, Sporadic
2019	EG-DVFS [18]	✓	×	Periodic, Sporadic
2019	Semi-Clairvoyance [25]	×	✓	Periodic, Sporadic
2020	Celebi [19]	✓	×	Periodic, Sporadic
2021	EDF based [20]	✓	×	Periodic
2022	MCQPA [26]	×	✓	Periodic, Sporadic
2022	EAU [27]	×	✓	Periodic
2024	ED-DM [21]	✓	×	Periodic
2024	EDF based [22]	✓	×	Periodic, Sporadic
2024	RBRR [28]	×	✓	Periodic
2025	EDF based [23]	✓	×	Sporadic
2025	NPFP-SC [29]	×	✓	Sporadic
2025	MCED-H	✓	✓	Periodic, Sporadic

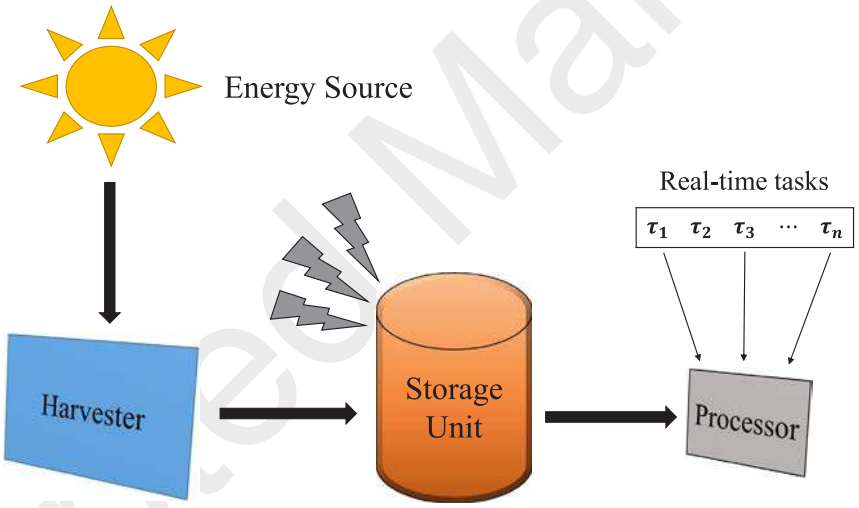


Fig. 1. Framework of a real-time energy harvesting system.

harvesting systems. Table 1 gives a synthetic description of research works related to real time scheduling in energy harvesting and MC systems, restricted to uniprocessing platforms.

While significant research has been conducted on energy harvesting and MC systems individually, the integration of these two domains remains an underexplored area. Existing approaches for handling LO tasks in HI mode, such as IMC, EMC, FMC, and PMC, primarily focus on task scheduling without considering energy constraints. This paper addresses these limitations by proposing a novel MCEH scheduling algorithm that prioritizes the execution of LO tasks without compromising the timely completion of HI tasks.

3. System model

Fig. 1 illustrates the core components of a MCEH system. A harvester first captures renewable energy and then converts it into electricity by a converter. For example, a solar panel converts sunlight into electrical energy. The device utilizes an energy harvester to operate solely on renewable energy harvested from environmental sources. The system does not operate on a battery; however, to ensure continuous operation during periods without energy harvested, an energy storage unit is necessary. The harvested energy is temporarily stored in a unit such as a capacitor. The processor utilizes the stored energy to schedule and execute tasks while adhering to the real time constraints of the system.

3.1. Energy product

The amount of energy obtained from natural sources fluctuates over time due to factors such as weather conditions and environmental changes. Additionally, energy conversion efficiency is typically less than 100%, resulting in a portion of the harvested energy being lost as heat or other forms of energy. To model this behavior, we define the energy product, P_p, as the amount of energy successfully converted and stored per unit time. To model energy variations over time, as shown in Fig. 2, we define $E_p(t_1, t_2)$ as the total energy harvested

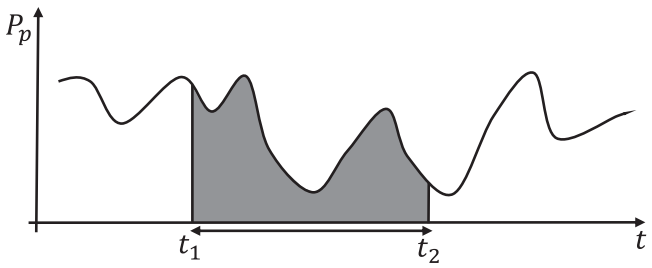


Fig. 2. Amount of energy harvested between t_1 and t_2.

between time points t_1 and t_2, calculated as the integral of $P_p(t)$ over that interval ($E_p(t_1, t_2) = \int_{t_1}^{t_2} P_p(t)dt$). This representation allows us to analyze energy availability and consumption patterns within specific time frames. We assume that energy harvesting and task execution can occur simultaneously. To simplify the scheduling problem, we assume a constant energy input rate, P_p, during each time unit [17].

3.2. Energy storage

Energy is stored in a storage unit, such as a capacitor, with a max imum capacity denoted by C. For simplicity, we assume ideal storage behavior without energy loss or self discharge. The amount of stored energy can vary between 0 and C units, affecting the system's ability to meet energy demands and execute tasks. The amount of energy stored in the storage unit at time t, denoted as $E(t)$, varies dynamically due to the combined effects of energy harvesting and task execution. The rate of change in $E(t)$ is influenced by the energy input from the harvester, P_p, and the energy consumed by tasks. Understanding the dynamics of energy storage is crucial for effective task scheduling and system performance optimization.

To characterize the of energy storage unit, we define two thresholds: discharged and fully charged. An energy storage unit is considered discharged when its stored energy, $E(t)$, is between 0 and e_{Max}, where e_{Max} represents the maximum energy consumption of any individual task in the system. Conversely, a energy storage unit is considered fully charged when $E(t)$ is between C and $C + e_{Max}$, where C is the energy storage unit's capacity. By defining these thresholds, we can implement strategies to manage energy storage and task execution based on the charge level of the energy storage unit [17].

3.3. Task model

Tasks are executed on a uniprocessor system with preemptive scheduling. While task preemption incurs a small energy overhead, we assume this overhead to be negligible for simplicity. Due to the inter mittent nature of energy harvesting, the system may enter idle states when insufficient energy is available for task execution. Similarly, idle power consumption is considered negligible compared to the energy consumed by active tasks.

The system comprises a set of n periodic tasks, denoted as $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$. To model task criticality, we assign a criticality level X_i to each task, where $X_i \in \{Lo, Hi\}$. Tasks with $X_i = Lo$ are considered low criticality, while tasks with $X_i = Hi$ are high criticality. Each task τ_i is characterized by a 6 tuple $(T_i, D_i, C_i^{Lo}, E_i^{Lo}, C_i^{Hi}, E_i^{Hi}, X_i)$, where T_i is the task period, D_i is the relative deadline (assumed to be equal to T_i in this work), C_i^{Lo} and C_i^{Hi} are the worst case execution times in low and high criticality modes, respectively. E_i^{Lo} and E_i^{Hi} are the corresponding energy consumptions. The criticality level X_i indicates whether the task is LO or HI. These parameters form the basis for task scheduling and resource allocation in the MCEH system.

To account for the varying computational demands of tasks in different operating conditions, we differentiate between LO mode and HI mode execution times and energy consumptions. For LO tasks, the execution time (C_i^{Lo}) and energy consumption (E_i^{Lo}) remain constant across both modes. However, for HI tasks, the execution time (C_i^{Hi}) and energy consumption (E_i^{Hi}) are higher in HI mode compared to LO mode. These differences in resource requirements introduce additional complexities in task scheduling and energy management for MCEH systems.

We assume that actual task execution time (C_i) and energy con sumption (E_i) are not directly proportional, as suggested by [31]. For LO tasks, both C_i and E_i remain constant across modes. However, for HI tasks, executing in HI mode increases both C_i and E_i compared to LO mode ($\frac{C_i^{Hi}}{C_i^{Lo}} = \frac{E_i^{Hi}}{E_i^{Lo}}$ [32 34]). We further assume a linear relationship between the increase in C_i and E_i for HI tasks, maintaining a constant energy consumption per unit time (e_i) regardless of the operating mode. This assumption simplifies the energy modeling process while capturing the essential characteristics of task energy behavior.

We assume that energy consumption during task execution exclu sively depletes the energy stored in the storage unit, preventing any increase in its energy level. Consequently, the energy harvesting rate, P_p, must be less than or equal to the minimum energy consumption per unit time of any task, e_i (i.e., $P_p \leq min(e_i)$), to ensure sustainable system operation. This assumption simplifies the energy management model by avoiding complex interactions between energy harvesting and task execution (see Table 2).

4. Notations and definitions

The purpose of this section is to introduce important terms and symbols needed to understand the remainder of the paper. A real time system is considered feasible if there exists a schedule that guarantees the completion of all tasks within their respective deadlines. If no such schedule can be found, the system is deemed infeasible. Feasibility anal ysis is crucial for determining the schedulability of task sets, especially in energy constrained and MC environments.

In MC systems, a task set is considered feasible if there exists a schedule that guarantees the completion of all LO and HI tasks within their deadlines while adhering to the specified mode requirements. LO tasks are executed in LO mode, and HI tasks are executed in either LO or HI mode, depending on system conditions. Ensuring the feasibility of MC systems is challenging due to the potential for mode transitions and the varying resource demands of tasks in different modes.

In real time energy harvesting systems, system feasibility is deter mined by the interplay of time and energy constraints. Unlike tradi tional real time systems, where feasibility is solely dependent on timing parameters such as execution times of the tasks, energy harvesting systems must also consider the availability of sufficient energy to exe cute tasks. As a consequence, infeasibility i.e. possible deadline missing can arise from either insufficient processor availability or insufficient environmental energy to supply the processing device.

In MCEH systems, a task set is considered feasible if there exists at least one schedule that guarantees the completion of all tasks within their deadlines. Feasibility depends on factors such as energy harvesting rate, capacity of energy storage unit, task energy consumption, and mode transitions in addition to execution times. The scheduler must ensure that sufficient energy is available to execute tasks in both LO and HI modes.

Before scheduling a task set, it is essential to determine its feasi bility. To assess feasibility, we calculate utilization metrics for both LO and HI modes. Utilization analysis helps to evaluate whether character istics of system resources, including energy harvester, energy storage unit, and processing power, are compatible with the timing and energy constraints of the task set.

$U_{i\,Lo}^{Lo} = \frac{C_i^{Lo}}{T_i}$ is the utilization of τ_i which is a LO task in LO mode. $U_{Lo}^{Lo} = \sum_{\tau^{Lo} \in \Gamma} \frac{C_i^{Lo}}{T_i}$ is the utilization of a task set for LO tasks in LO mode. $U_{i\,Hi}^{Lo} = \frac{C_i^{Lo}}{T_i}$ is the utilization of τ_i which is a HI task in LO mode.

Table 2
List of symbols and their descriptions.

Notation	Description
MC	Mixed-Criticality
MCEH	Mixed-Critical Energy Harvesting
LO	Low criticality level
HI	High criticality level
LO mode	Low criticality mode
HI mode	High criticality mode
P_p	Amount of energy stored in the energy storage unit over one unit of time
$E_p(t_1, t_2)$	Amount of energy stored in the energy storage unit from time t_1 to t_2
$E(t)$	Amount of energy stored in the energy storage unit at time t
e_{Max}	Highest amount of energy consumption in one unit of time
T_i	Period of task τ_i
D_i	Relative Deadline of task τ_i
C_i^{Lo}	Execution time of task τ_i in LO mode
E_i^{Lo}	Energy consumption of task τ_i in LO mode
C_i^{Hi}	Execution time of task τ_i HI mode
E_i^{Hi}	Energy consumption of task τ_i HI mode
X_i	Criticality level of task τ_i
e_i	Amount of energy consumption of task τ_i in a unit of time
$U_{i\,Lo}^{Lo}$	Utilization of LO task τ_i in LO mode
$U_{i\,Hi}^{Lo}$	Utilization of HI task τ_i in LO mode
$U_{i\,Hi}^{Hi}$	Utilization of HI task τ_i in HI mode
$eU_{i\,Lo}^{Lo}$	Energy utilization of LO task τ_i in LO mode
$eU_{i\,Hi}^{Lo}$	Energy utilization of HI task τ_i in LO mode
$eU_{i\,Hi}^{Hi}$	Energy utilization of HI task τ_i in HI mode
U_{Lo}^{Lo}	Sum of utilizations of LO tasks in LO mode
U_{Hi}^{Lo}	Sum of utilizations of HI tasks in LO mode
U_{Hi}^{Hi}	Sum of utilizations of HI tasks in HI mode
U^{Lo}	Utilization in LO mode
U^{Hi}	Utilization in HI mode
eU_{Lo}^{Lo}	Sum of energy utilizations of LO tasks in LO mode
eU_{Hi}^{Lo}	Sum of energy utilizations of HI tasks in LO mode
eU_{Hi}^{Hi}	Sum of energy utilizations of HI tasks in HI mode
eU^{Lo}	Energy Utilization in LO mode
eU^{Hi}	Energy Utilization in HI mode
t_c	Current time
$ST_{\tau_i}(t_c)$	Slack time of task τ_i at time t_c
$h(t_1, t_2)$	Processing demand in $[t_1, t_2]$
$ST_{\tau}(t_c)$	Slack time of task set τ at time t_c
$SE_{\tau_i}(t_c)$	Slack energy of task τ_i at time t_c
$g(t_1, t_2)$	Energy demand in $[t_1, t_2]$
$SE_{\tau}(t_c)$	Slack energy of task set τ at time t_c
$PSE_{\tau}(t_c)$	Preemption Slack Energy

$U_{Hi}^{Lo} = \sum_{\tau^{Hi} \in \Gamma} \frac{C_i^{Lo}}{T_i}$ is the utilization of the task set for HI tasks in LO mode. $U^{Lo} = U_{Lo}^{Lo} + U_{Hi}^{Lo}$ is the utilization in LO mode. $U_{i\,Hi}^{Hi} = \frac{C_i^{Lo}}{T_i}$ is the utilization of τ_i which is a HI task in HI mode. $U_{Hi}^{Hi} = \sum_{\tau^{Hi} \in \Gamma} \frac{C_i^{Hi}}{T_i}$ is the utilization of the task set for HI tasks in HI mode. The utilization of HI tasks in HI mode is $U^{Hi} = U_{Hi}^{Hi}$.

In addition to time based utilization, energy consumption also plays a crucial role in determining task set feasibility. By calculating energy utilization metrics, we can verify that the energy source is compatible with the energy requirements of tasks. $eU_{i\,Lo}^{Lo} = \frac{E_i^{Lo}}{T_i}$ is the energy utilization of τ_i which is a LO task in LO mode. $eU_{Lo}^{Lo} = \sum_{\tau^{Lo} \in \Gamma} \frac{E_i^{Lo}}{T_i}$ is the energy utilization of the task set for LO tasks in LO mode. $eU_{i\,Hi}^{Lo} = \frac{E_i^{Lo}}{T_i}$ is the energy utilization of τ_i which is a HI task in LO mode. $eU_{Hi}^{Lo} = \sum_{\tau^{Hi} \in \Gamma} \frac{E_i^{Lo}}{T_i}$ is the energy utilization of the task set for HI tasks in LO mode. $eU^{Lo} = eU_{Lo}^{Lo} + eU_{Hi}^{Lo}$ is the energy utilization in LO mode. $eU_{i\,Hi}^{Hi} = \frac{E_i^{Lo}}{T_i}$ is the energy utilization of τ_i which is a HI task in HI mode. $eU_{Hi}^{Hi} = \sum_{\tau^{Hi} \in \Gamma} \frac{E_i^{Hi}}{T_i}$ is the energy utilization of the task set for HI tasks in HI mode. The energy utilization of HI tasks in HI mode is $eU^{Hi} = eU_{Hi}^{Hi}$

We will now discuss feasibility conditions after calculating the utilization of tasks in different modes. For a task set to be feasible in LO mode, the following necessary conditions must be met:

- $U^{Lo} \leq 1$: The total utilization of all tasks in LO mode must be less than or equal to 1 to ensure schedulability based on time constraints.
- $eU^{Lo} \leq P_p$: The total energy consumption of all tasks in LO mode must be less than or equal to the available energy harvesting rate (P_p), assumed to be constant to guarantee sufficient energy for task execution.

Similarly, for a task set to be feasible in HI mode, the following conditions must be satisfied:

- $U^{Hi} \leq 1$: The total utilization of all tasks in HI mode must be less than or equal to 1.
- $eU^{Hi} \leq P_p$: The total energy consumption of all tasks in HI mode must be less than or equal to the available energy harvesting rate.

Failure to meet any of these conditions indicates potential system overload and deadline misses. If these condition is not met, the sys tem will not have sufficient energy to execute all tasks within their deadlines, leading to energy starvation.

5. Background material and preliminaries

5.1. Base algorithm

To establish a foundation for our proposed algorithm, we introduce the ED H algorithm [17], an online scheduling approach based on EDF. ED H incorporates a look ahead mechanism, considering the next D time units (where D is the maximum relative task deadline) to make scheduling decisions. By anticipating future task arrivals and future energy availability, ED H aims to efficiently organize energy consumption so as to meet task deadlines despite energy limitations. In ED H, there are five rules for scheduling tasks. Before we introduce these rules, we define the essential concepts that ED H requires, namely slack time and slack energy.

Slack time of a task represents the maximum amount of time this task can be delayed without jeopardizing its deadline. In the ED H algorithm, slack time is used to determine when the processor can be kept idle without compromising task deadlines. For the task τ_i at t_c, the slack time is:

$$ST\tau_i(t_c) = d_i - t_c - h(t_c, d_i) \tag{1}$$

$h(t_c, d_i)$, known as processor demand, represents the total processing time it takes to execute all the tasks with release times and deadlines between t_c and d_i. The slack time of task set τ between t_c to d_i is:

$$ST_\tau(t_c) = \min_{d_i > t_c} ST_{\tau_i}(t_c) \tag{2}$$

Identically, we may define the slack energy of every task τ_i in τ as follows:

$$SE\tau_i(t_c) = E(t_c) + E_p(t_c, d_i) - g(t_c, d_i) \tag{3}$$

$g(t_c, d_i)$, known as the energy demand, represents the amount of energy consumed by all tasks with release times and deadlines between t_c and d_i. The slack energy of task set τ at time t_c is:

$$SE_\tau(t_c) = \min_{t_c < r_i < d_i} SE_{\tau_i}(t_c) \tag{4}$$

Roughly speaking, the slack energy of the task set τ at current time t_c gives the energy surplus that could be consumed at time t_c without compromising the feasibility of τ.

Before authorizing τ_i to be executed at time t_c, we have to verify that its execution will not lead to energy starvation for any higher priority task released in the future, i.e., with a deadline less than d_i. We define this as Preemption Slack Energy (PSE). In other terms, we have to test if the slack energy of such task is positive. Consequently, $PSE(t)$ is the minimal slack energy from all the jobs with deadline less than d_i. If $PSE = 0$, the system cannot execute the highest priority task without risking deadline misses for future tasks. By carefully considering PSE, MCED H can make informed scheduling decisions and optimize energy consumption.

The ED H algorithm utilizes slack energy to determine when the system can operate in a power saving mode or execute additional tasks without risking deadline misses. The necessary and basic definitions of the ED H have now been covered. The next step is to discuss the rules for the ED H.

Let $Q_r(t_c)$ be the queue of tasks ready to execute at time t_c:

- **Rule 1:** A future running task in $Q_r(t_c)$ is selected based on the EDF priority order.
- **Rule 2:** if $Q_r(t_c) = \emptyset$, the processor is idle from time t_c to $t_c + 1$.

Table 3
Parameters of the task set for illustration.

Tasks	T_i	D_i	C_i^{Lo}	E_i^{Lo}	C_i^{Hi}	E_i^{Hi}	X_i
τ_1	5	5	1	3	1	3	LO
τ_2	10	10	3	6	3	6	LO
τ_3	20	20	3	6	4	8	HI
τ_4	20	20	2	6	3	9	HI

- **Rule 3:** if $Q_r(t_c) \neq \emptyset$ and one of the following conditions is met, the processor will invariably be idle from time t_c to $t_c + 1$: (
  1) $E(t_c) = 0$. (
  2) $PSE(t_c) = 0$.
- **Rule 4:** if $Q_r(t_c) \neq \emptyset$ and one of the following conditions is met, the processor will invariably be busy from time t_c to $t_c + 1$: (
  1) $E(t_c) = C$. (
  2) $ST_\tau(t_c) = 0$.
- **Rule 5:** It is equally possible that the processor be idle or busy from time t_c to $t_c + 1$ if $Q_r(t_c) \neq \emptyset$, $0 < E(t_c) < C$, $ST(t_c) > 0$, and $PSE(t_c) > 0$.

As proved in [17], ED H is an optimal algorithm for RTEH systems. An algorithm is optimal if it can schedule tasks in such a way that all tasks meet their deadlines and the scheduler can find this schedule.

5.2. System behavioral

The MCEH system operates in either LO or HI mode, depending on task characteristics and energy availability. Initially, the system operates in LO mode, executing tasks with their LO mode parameters (C_i^{Lo}, E_i^{Lo}). A transition to HI mode occurs when the execution time of a HI task exceeds its allocated time in LO mode (C_i^{Lo}). Due to the assumed linear relationship between execution time and energy consumption, the system switches to HI mode simultaneously for both time and energy. Once in HI mode, a selective subset of LO tasks is executed until the system returns to LO mode at the end of the hyperperiod.

6. MCED-H

First, we explain the problem with a motivational example. Then we explain the algorithm in two parts: LO mode and HI mode.

6.1. Motivational example

While the ED H algorithm has shown promise in energy harvesting systems, its performance in MC environments remains unexplored. To illustrate the limitations of ED H in handling MC tasks, we present a motivating example.

As you can see in Table 3, we have a task set that includes 4 tasks. The capacity of the energy storage unit is 10. P_p is 2 and $E(0) = 0$. Here, it is assumed that there is only one mode for scheduling, as in ED H. The first step is to determine whether the task set is feasible. $U = 0.75 \leq 1$ and $eU = 1.8 \leq P_p$ so it is feasible. $ST\tau(0) = 4$ and $SE\tau(0) = 4$. Since $E(0) = 0$, to satisfy rule 3, we must keep the system idle in order to recharge the energy storage unit. We keep it idle for 4 s. At time 4, $E(4) = 8$, the energy storage unit is not yet filled. However, we must execute τ_1 to satisfy rule 4. At time 5, $E(5) = 7$, $ST\tau(5) = 1$ and $SE\tau(5) = 8$. The system stays idle up to time 6. At time 6, $E(6) = 9$ and $ST\tau(6) = 0$. All tasks are executed up to time 20, as shown in Fig. 3.

As demonstrated in the example, applying the ED H algorithm to a MC task set can lead to deadline misses for HI tasks. If the system transitions to HI mode at time 18 or 20 due to increased computational demands, tasks τ_3 and τ_4 may not complete execution before their deadlines. This highlights the need for a scheduling algorithm that can

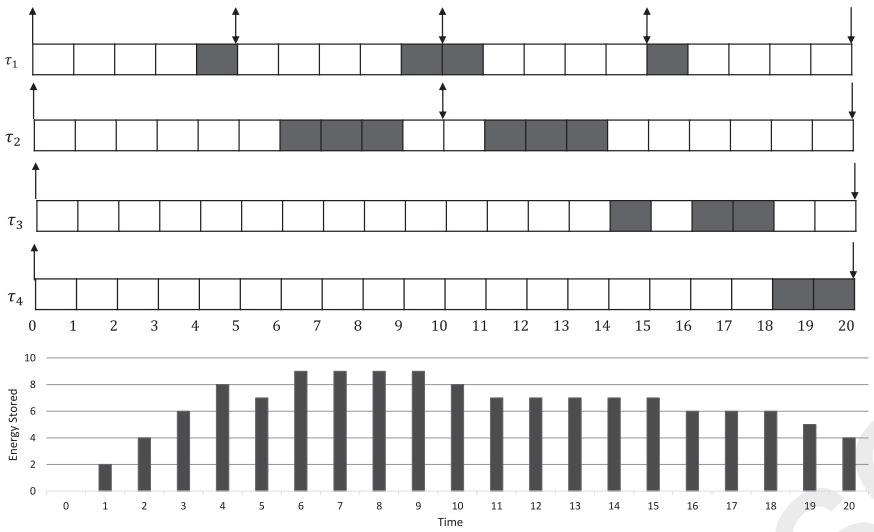


Fig. 3. Illustration of the ED-H scheduler.

proactively manage mode transitions and ensure the timely completion of critical tasks.

The EDF VD, which assigns virtual deadlines to HI tasks, is a com mon approach for scheduling MC systems. However, EDF VD is greedy and disregards for energy constraints. This makes it unsuitable for en ergy harvesting environments. To address these limitations, we propose the MCED H algorithm, which incorporates energy management and task criticality into the scheduling process.

The virtual deadline assignment for HI tasks in MCEH systems must consider both time and energy constraints. To balance the need for timely task completion with efficient energy utilization, the virtual deadline is calculated by incorporating task execution time, task energy consumption, and available energy. By minimizing the virtual deadline while ensuring task schedulability, we aim to maximize energy storage and system efficiency.

6.2. Scheduling tasks in LO mode

In LO mode, tasks are scheduled using the ED H algorithm, which prioritizes tasks based on their deadlines and available energy. To accommodate HI tasks, we introduce virtual deadlines for HI tasks. These virtual deadlines are calculated to ensure that HI tasks can be executed within their original deadlines even if the system transitions to HI mode. By incorporating virtual deadlines into the LO mode schedul ing process, we aim to optimize energy usage and improve system reliability. To illustrate the concept of virtual deadlines more clearly, we will use a simplified example that excludes energy consumption, the capacity of the energy storage unit, and the energy harvesting rate. This abstract example will focus solely on task scheduling and deadline management.

Each HI task has two potential execution times: C_i^{Lo} and C_i^{Hi}, where $C_i^{Lo} < C_i^{Hi}$. Fig. 4.A illustrates a HI task with a deadline of t, where C_i^{Lo} is represented by x and the additional execution time in HI mode (α) is added to C_i^{Lo} to determine C_i^{Hi}. To illustrate the need for virtual deadlines, consider a HI task with deadline t, where the execution time in LO mode (C_i^{Lo}) is x. We assume the task begins execution at time t_1, such that $t - t_1 = x$, resulting in zero slack time. While the task completes execution in LO mode at time t, a mode switch to HI mode at any point after t_1 would result in a deadline miss due to the extended execution time in HI mode ($C_i^{Hi} = x + \alpha$).

To address the potential for deadline misses due to mode transitions, we introduce a virtual deadline (D^*) for the HI task. The virtual deadline is calculated by subtracting the difference between C_i^{Lo} and C_i^{Hi} (α) from the original task deadline (t). By starting task execution at

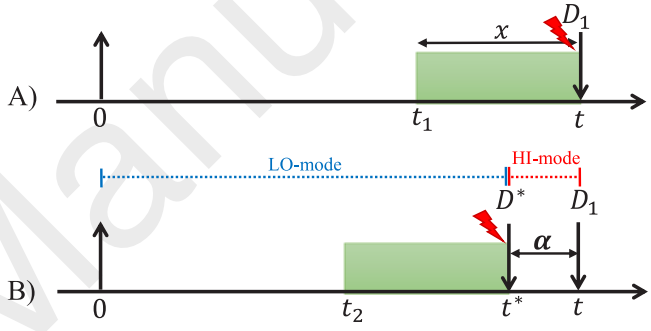


Fig. 4. Illustration of the virtual deadline process.

t_2 (where $t^* - t_2 = x$), the system ensures that sufficient time is available to complete the task even if a mode switch to HI mode occurs at time t^*. This approach helps to guarantee the timely completion of HI tasks while optimizing resource utilization, as shown in Fig. 4.B.

Despite our previous discussion about incorporating both time and energy factors into this virtual deadline computation, we only focus on energy considerations. Then we will prove why there is no need to consider the time factor for computing the virtual deadline.

$$eU_{Lo}^{Lo} + \frac{eU_{Hi}^{Lo}}{P_p - eU_{Hi}^{Hi} + eU_{Hi}^{Lo}} \leq P_p \tag{5}$$

If a task set satisfies Eq. (5), we can apply a virtual deadline to HI tasks. It has two advantages for scheduling. First, when the system switches into HI mode, there is enough time and energy to schedule HI tasks. Second, after applying the virtual deadline, there is still enough energy and time to schedule LO and HI tasks in the LO mode. Consider λD_i as the virtual deadline for HI tasks in the LO mode. λ can be obtained from $\frac{P_p - (eU_{Hi}^{Hi} - eU_{Hi}^{Lo})}{P_p}$. In our approach, λ is a factor added to the deadline of HI tasks to calculate the virtual deadline in LO mode. This factor ensures when the system switches to HI mode, HI tasks are able to meet their deadlines by accounting for the energy consumption differences between HI and LO modes. The value of λ is determined based on the energy utilization difference between HI and LO modes for HI tasks. Specifically, we calculate the difference in energy consumption for HI tasks in the two modes as $eU_i^{Hi} - eU_i^{Lo}$. This represents the additional energy required when switching from LO to HI mode, which is crucial for scheduling HI tasks without missing deadlines. The remaining energy for tasks in LO mode is determined as

$P_p - (eU_i^{Hi} - eU_i^{Lo})$, where P_p represents the average power harvested by the system per unit of time. The obtained value is energy based, and to derive the virtual deadline in terms of time, we convert it by dividing it by P_p. Thus, λ acts as a coefficient that adjusts the original deadline to create the virtual deadline for HI tasks, ensuring that enough energy is available to meet their deadlines when the system switches its mode. Therefore, λ plays a crucial role in determining the virtual deadline for HI tasks, enabling the system to manage energy and time efficiently during mode switching.

In addition, the deadlines for all LO tasks remain unchanged. Here, we describe the process by which Eq. (5) was obtained. $eU^{Lo} \leq P_p$

$$eU_{Lo}^{Lo} + eU_{Hi}^{Lo} \leq P_p$$
$$\sum_{\tau^{Lo} \in \Gamma} \frac{E_i^{Lo}}{T_i} + \sum_{\tau^{Hi} \in \Gamma} \frac{E_i^{Lo}}{\lambda_* T_i} \leq P_p$$
$$eU_{Lo}^{Lo} + \sum_{\tau^{Hi} \in \Gamma} \frac{E_i^{Lo}}{(P_p - (eU_{Hi}^{Hi} - eU_{Hi}^{Lo}))T_i} \leq P_p$$
$$eU_{Lo}^{Lo} + \frac{1}{(P_p - (eU_{Hi}^{Hi} - eU_{Hi}^{Lo}))} \sum_{\tau^{Hi} \in \Gamma} \frac{E_i^{Lo}}{T_i} \leq P_p$$
$$eU_{Lo}^{Lo} + \frac{eU_{Lo}^{Hi}}{(P_p - (eU_{Hi}^{Hi} - eU_{Hi}^{Lo}))} \leq P_p$$

To determine the scheduling horizon, we introduce d_{max}, representing the maximum absolute deadline among all ready tasks in the system. While virtual deadlines are calculated for HI tasks to ensure their timely completion, the system's look ahead window is based on the initial deadlines (D_i) of all tasks. This approach allows for proactive resource allocation and energy management.

In HI mode, although LO tasks are not fully executed, they are still considered in the scheduling process. This inclusion enables the system to make informed decisions about resource allocation and potential mode transitions. Upon transitioning to HI mode, a selective subset of LO tasks is executed, and the deadlines of HI tasks revert to their original values (D_i). The system remains in HI mode until it reaches the hyperperiod. The decision to switch from LO mode to HI mode is triggered when the slack time (ST_τ) and slack energy (SE_τ) both reach zero.

Let us explain why we do not consider time for the virtual deadline calculation. Let us assume a task set where P_p is x. In addition, we can determine how much energy is consumed per unit of time by each task. $\frac{E_i}{C_i} = e_i$. This means that e_i is the amount of energy required per unit of time for a task to run properly. There is a specific assumption in this algorithm that is $P_p \leq min(e_i)$. So $1 \leq \frac{e_i}{P_p}$. To assess task set feasibility, we consider both time based and energy based utilization metrics. Time based utilization (U) is calculated as the sum of task execution times (C_i) divided by their periods (T_i), i.e., $\sum_n^{i=1} \frac{C_i}{T_i} \leq 1$. Energy based utilization (eU) is computed as the sum of the task energy consumption per unit time (e_i) and execution time (C_i) divided by the energy harvesting rate (P_p) and period (T_i), i.e., $\sum_n^{i=1} \frac{e_i \times C_i}{P_p \times T_i} \leq 1$.

By comparing U and eU/P_p, we observe that energy constraints are more stringent than time constraints in this scenario. When the system switches into HI mode, as mentioned, their time and energy consumption increase to a certain extent. The energy consumption added to tasks requires more time to meet deadlines than the added execution time of the same task. Consequently, the MCED H algorithm focuses on energy based considerations for virtual deadline calculations. Hence, there is no need to calculate λ for time.

For the illustrative task set, the calculated utilization values are $U^{Lo} = 0.75$ and $eU^{Lo} = 1.8$. These values satisfy Equation (5), indicating that the task set is feasible in LO mode. The scaling factor λ is calculated as $\lambda = P_p - (eU_{Hi}^{Hi} - eU_{Hi}^{Lo})/P_p$, resulting in a value of 0.875. By multiplying the original deadlines of HI tasks by λ, we obtain their virtual deadlines, as shown in Table 4. The use of virtual deadlines in LO mode allows the system to proactively manage energy resources and improve the likelihood of meeting HI task deadlines even in the event of a mode switch. The schedule in LO mode produced by MCED H is shown in Fig. 5.

As shown in Fig. 5.A, the MCED H algorithm successfully schedules all tasks within their deadlines in LO mode. However, the system may transition to HI mode at time 14 or 16 due to the potential execution of HI tasks with longer execution times (C_i^{Hi}) in HI mode. To ensure the timely completion of HI tasks, the MCED H algorithm must effectively manage mode transitions and task scheduling in HI mode.

Table 4
Task parameters after virtual deadline assignment.

Tasks	T_i	D_i	C_i^{Lo}	E_i^{Lo}	C_i^{Hi}	E_i^{Hi}	X_i
τ_1	5	5	1	3	1	3	LO
τ_2	10	10	3	6	3	6	LO
τ_3	20	17.5	3	6	4	8	HI
τ_4	20	17.5	2	6	3	9	HI

Algorithm 1 Applying virtual deadline on HI tasks.

```
Calculate U_Lo and eU_Lo
if U_Lo ≤ 1 and eU_Lo ≤ P_p then
    Calculate eU_Lo^Lo, eU_Hi^Lo and eU_Hi^Hi
    if Condition satisfies Eq. (5) then
        Calculate λ
        for each τ_i ∈ τ^Hi do
            D_i^* = λ × D_i
            Assign D_i^* as the new virtual deadline for τ_i
            Adjust priority levels according to D_i^*
        end for
        Remain in LO mode
    else
        Switch to HI mode
    end if
else
    Switch to HI mode
end if
while system is running in the LO mode do
    if ST ≤ 0 and PSE ≤ 0 then
        Switch to HI mode
    end if
end while
```

As illustrated in Figs. 5.B and 5.C, the system transitions to HI mode at times 14 and 16, respectively. Upon entering HI mode, LO tasks are suspended, and HI tasks are executed using their HI mode parameters (C_i^{Hi}, E_i^{Hi}). The order of HI task execution is determined by their original deadlines (D_i) to minimize the risk of deadline misses. To address potential energy constraints in HI mode, the system may need to adopt energy saving techniques or adjust task priorities dynamically.

Algorithm 1 outlines the core steps of the MCED H algorithm. Initially, the algorithm checks the feasibility of the task set in LO mode by evaluating Equation (5). If the conditions are met, the system proceeds to calculate virtual deadlines for HI tasks to improve their schedulability. The ED H algorithm is then employed to schedule tasks in LO mode, considering both original and virtual deadlines.

The system continuously monitors system conditions (slack time, slack energy) to determine the need for a mode switch. If energy or time resources become critically low, the system transitions to HI mode, suspending LO tasks and focusing on executing HI tasks based on their original deadlines. The algorithm aims to minimize the time spent in HI mode to optimize overall system performance.

While executing tasks in LO mode, the algorithm continuously monitors system conditions to ensure adherence to both time and energy constraints. If either Equation (2) or Equation (4) (representing time based and energy based utilization, respectively) is violated, it indicates a potential risk of deadline misses. In such cases, the system transitions to HI mode to prioritize the execution of critical tasks.

By assigning virtual deadlines to HI tasks, the MCED H algorithm proactively allocates system resources to ensure their timely completion. This approach helps to prevent deadline misses even in the event

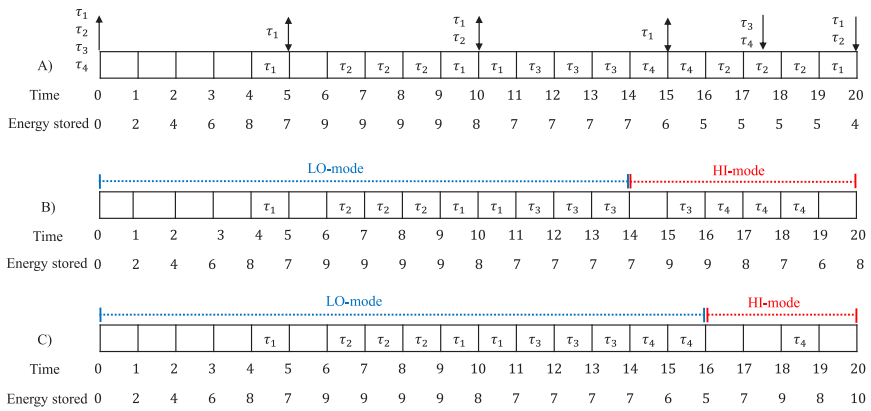


Fig. 5. Scheduling with MCED-H in LO mode.

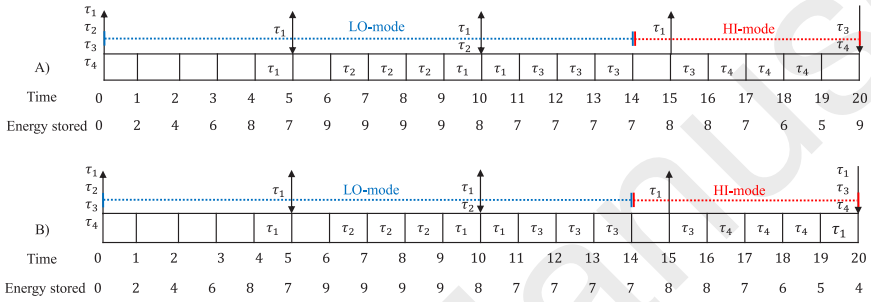


Fig. 6. Example to illustrate MCED-H scheduling.

of unexpected mode transitions. The system continuously monitors energy consumption and task execution progress to detect potential violations of time or energy constraints. If these constraints are not met, the system switches to HI mode to prioritize the execution of critical tasks.

The time complexity of the proposed algorithm is $O(n \log n)$. The highest time complexity arises from sorting the queue after applying the virtual deadlines to the tasks. Otherwise, the computation of utilization, energy utilization, and λ involves a time complexity of $O(n)$, as it requires iterating through all tasks and performing basic arithmetic operations.

6.3. Scheduling tasks in HI mode

Traditional MC systems often suspend all LO tasks during HI mode, leading to reduced system performance. To address this limitation, var ious approaches such as IMC, PMC, FMC, and EMC have been proposed. However, these methods primarily focus on task scheduling without explicitly considering energy constraints, making them unsuitable for MCEH systems.

To balance the need for executing LO tasks in HI mode with energy efficiency, the MCED H algorithm adopts a selective approach. By carefully considering task priorities, energy availability, and system resources, the algorithm determines which LO tasks can be executed without compromising the timely completion of HI tasks.

To determine which LO tasks can be executed in HI mode, we calculate the slack time (ST) and slack energy (SE) for each LO task at the current time (Q_r). Tasks are prioritized based on their energy consumption, with lower energy consumption tasks given preference. If the calculated ST and SE for an LO task are both non negative, the task is included in the HI mode schedule. Otherwise, the task is suspended. This approach aims to maximize resource utilization while minimizing the risk of energy depletion or deadline misses.

As shown by Fig. 6A, the system is scheduled until time 14. The system then switches to HI mode. In this mode, we execute tasks with

Algorithm 2 Choosing LO tasks to execute in HI mode at t_c.

1: whenever system switches to HI mode Or d_{max} has changed.
2: $Chosen(\tau_i)$= find τ_i^{Lo} which has the lowest energy consumption in $Q_r(t_c)$
3: **if** $ST_\tau(t_c) \geq 0$ and $PSE_\tau(t_c) \geq 0$ **then**
4: keep $Chosen(\tau_i)$ in $Q_r(t_c)$
5: **else**
6: suspend $Chosen(\tau_i)$ from $Q_r(t_c)$
7: **end if**

C_i^{Hi} and E_i^{HI}. All HI tasks will return to their initial value, which is 20, and we will suspend all LO tasks. At time 14, without taking into account the LO tasks, ST and SE are equal to 2 and 9, respectively. There are 2 units of time and 9 units of energy that LO tasks can consume. There are two tasks in Q_r, which are τ_1 and τ_2. Of these two tasks, τ_1 has lower energy consumption. The values of ST and SE are calculated from these three tasks, which are 1 and 4, respectively. Both values are greater than zero, so we keep the τ_1 in Q_r and move to τ_2. Now, we calculate the value of ST and SE for all tasks. Here it can be seen that the value of ST and SE is equal to -2 and -2. Both parameters are less than zero. This signifies that one of the tasks will miss its deadline. So, we suspend task τ_2 from the system. Now, we schedule 3 tasks without deadline miss, as shown in Fig. 6.B.

Algorithm 2 outlines the process for selecting LO tasks to be ex ecuted in HI mode. The algorithm is triggered whenever the system transitions to HI mode or when the maximum task deadline (d_{max}) changes. For each LO task in the ready queue (Q_r), the algorithm calculates ST and SE based on current system conditions. LO tasks with positive ST and SE values are prioritized for execution, with lower energy consumption tasks given preference. The selected LO task is included in the HI mode schedule, and its ST and SE values are recalculated for the remaining tasks. This iterative process continues until no eligible LO tasks are found.

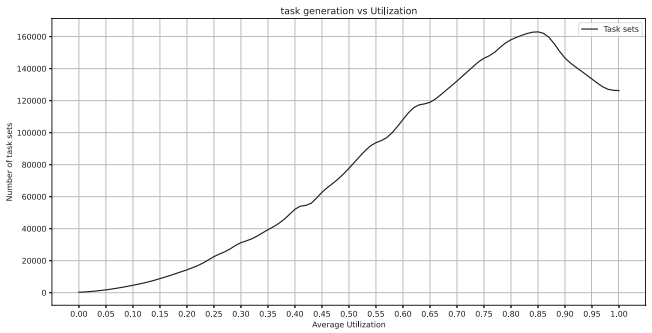


Fig. 7. Number of task set that generated at each point.

By executing eligible LO tasks in HI mode, the MCED H algorithm improves system quality, prevents energy waste, and enhances resource utilization. Assuming the queue is implemented as a priority queue, this operation can be done in $O(log\ n)$ time, where n is the number of LO tasks in the queue. The complexity of calculating slack time and slack energy is $O(n)$. Since the calculation of slack time and slack energy comes after finding the LO task, the overall complexity becomes $O(n\ log\ n)$.

7. Experimental results

To evaluate the performance of the MCED H algorithm, we con ducted a comparative study with EDF, EDF VD, ED H, and MCQPA. These algorithms are all based on the EDF priority assignment rule. The average utilization, defined as $\frac{U^{Lo}+U^{Hi}}{2}$, is used as a measurement metric.

To evaluate the performance of different algorithms under various workload conditions, we generated two types of task sets. The first set of task sets was created using the UUniFast algorithm [35] with a target utilization of 0.8. For each generated task set, energy consumption val ues (E_i) were randomly assigned within a specified range. The second set of task sets explored all possible combinations of utilization values within a defined interval to provide a more exhaustive evaluation. Task criticality levels (LO/HI) were assigned randomly to each task in both sets.

7.1. System model and task set generation

For both task generation methods, the system was configured with a constant energy harvesting rate (P_p) of 2 units. The capacity of the energy storage unit is 500 units. Initially, the energy storage unit was assumed to be empty ($E(0) = 0$). Each task set consisted of 10 periodic tasks with equal periods and deadlines, randomly assigned as either LO or HI with a 50% probability. The execution time of a HI task in LO mode (C_i^{Lo}) was determined by multiplying its HI mode execution time (C_i^{Hi}) by a critical factor randomly selected between 0.6 and 0.8, as suggested in [36]. Energy consumption values (E_i^{Lo} and E_i^{Hi}) were proportionally adjusted based on the execution times. Only feasible task sets, where both time based and energy based utilization constraints were satisfied, were included in the experiments.

7.1.1. The point generator

To comprehensively evaluate algorithm performance, we generated a range of task sets by systematically varying utilization values (U_{Lo}^{Lo}, U_{Hi}^{Lo} and U_{Hi}^{Hi}) ranging from 0.03 to 1.00 in increments of 0.01. The energy utilization value for each task set is randomly selected within the range of $P_p\times$ average utilization to P_p, to ensure that the gener ated task set is feasible. Energy consumption values (E_i^{Lo}, E_i^{Hi}) were assigned randomly within a specified range. Only feasible task sets, satisfying both time based and energy based utilization constraints, were considered for analysis. In Fig. 7, you can see how many different task sets we generate at each point.

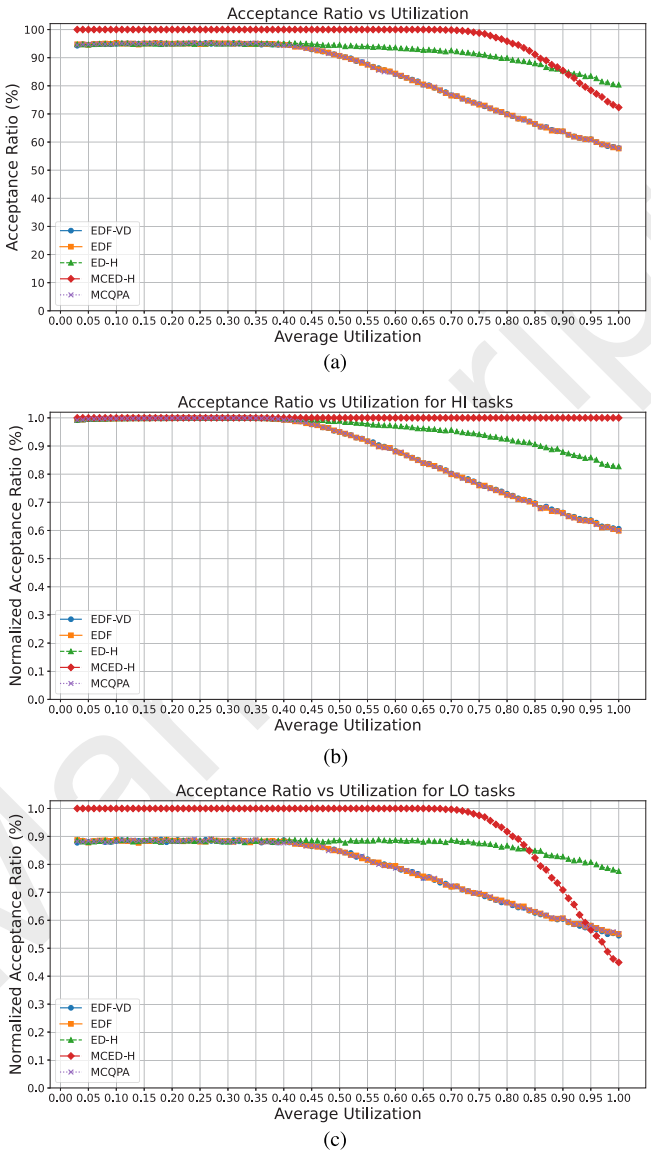


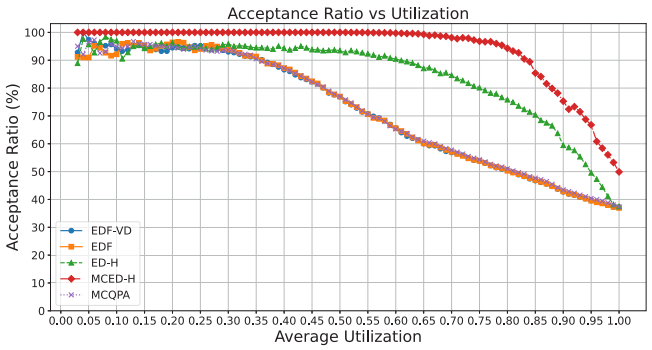
Fig. 8. UUniFast experimental results.

7.1.2. UUniFast

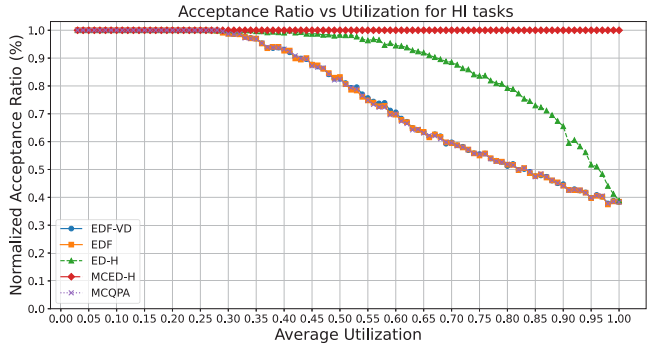
To evaluate the performance of MCED H under varying workloads, we employed the UUniFast algorithm to generate task sets with av erage utilizations ranging from 0.03 to 1.00 in increments of 0.01. For each average utilization level, 10000 task sets were generated. Energy consumption values for each task were randomly assigned using UUniFast to maintain consistency with the task utilization distribution. By systematically varying task set parameters, we aimed to comprehen sively assess the algorithm's behavior across a wide range of workload conditions.

7.2. Experiments

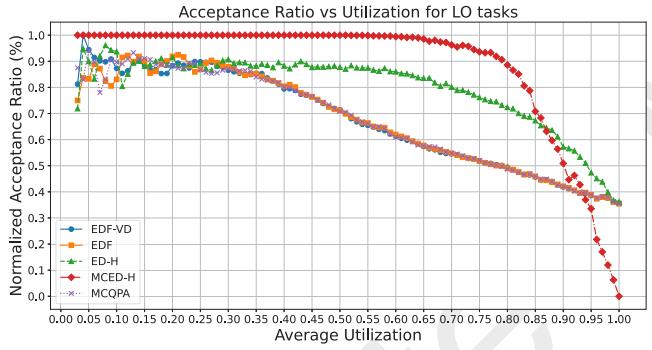
To assess the performance of the algorithms, we calculated the acceptance ratio, defined as the percentage of tasks that meet their deadlines within a given task set. Figs. 8 and 9 illustrate the acceptance ratio for LO and HI modes, respectively, as a function of average task set utilization. In these figures, at each point, all tasks in LO mode and HI mode, if they meet their deadline, will show their highest value. On the other hand, if a number of tasks in each LO mode or HI mode miss their deadline, the lower value will be shown in the figures.



(a)



(b)



(c)

Fig. 9. The Generator experimental results.

To evaluate the ability of different algorithms to meet deadlines for both LO and HI tasks, we calculated the acceptance ratio for each task type separately. The acceptance ratio represents the percentage of tasks that meet their deadlines within a given task set. Figs. 8 and 9 illustrate the acceptance ratio for LO and HI tasks, based on UUniFast and exhaustively generated task sets, respectively.

To further analyze the performance of the algorithms in handling LO and HI tasks separately, we introduce the concept of "success rate". The success rate represents the percentage of tasks that meet their deadlines within a specific task type (LO or HI). Figs. 8.B and 9.B illustrate the success rate of LO tasks, while Figs. 8.C and 9.C depict the success rate of HI tasks for both UUniFast and exhaustively generated task sets.

Fig. 8.A shows the first point where the average utilization for the MCED H algorithm falls below 100%, at 0.66, with a success rate of 99.99%. At the same average utilization point, EDF achieves a success rate of 79.96%, and ED H achieves 92.90%. This represents a 25.06% improvement for MCED H over EDF and a 7.64% improvement over ED H. At the same utilization, the success rates for LO tasks are 75.19% for EDF, 88.46% for ED H, and 99.99% for MCED H, as depicted in Fig. 8.C. This corresponds to a 32.97% improvement for MCED H over EDF and a 13.04% improvement over ED H. For HI tasks, as shown in Fig. 8.B, the success rates are 83.49% for EDF, 96.06% for ED H, and 100% for MCED H, marking a 19.78% improvement over EDF and a 4.11% improvement over ED H. When the average utilization reaches its maximum, Fig. 8.A indicates success rates of 57.69% for EDF, 80.38% for ED H, and 72.31% for MCED H. Despite slightly lower overall performance here, MCED H still shows notable results for HI tasks. Fig. 8.C reveals that at maximum utilization, the success rates for HI tasks are 59.88% for EDF, 82.74% for ED H, and 100% for MCED H. This corresponds to a 67.04% improvement over EDF and a 20.85% improvement over ED H, underscoring the robustness of the MCED H algorithm in critical scenarios.

In high utilization scenarios (e.g., with utilization values exceeding 0.90, as shown in Fig. 8.A), ED H outperforms MCED H in terms of overall success rate. This higher performance is due to the combined success rates of both HI and LO tasks. However, upon separately examining the success rates for HI and LO tasks (Figs. 8.B and 8.C), we find that MCED H guarantees a 100% success rate for HI tasks, which is critical in MC systems. As a result, although MCED H has a lower overall success rate compared to ED H, it prioritizes the timely execution of HI tasks. On the other hand, the success rate for LO tasks in MCED H is lower compared to ED H, which contributes to the overall degradation in performance observed in high utilization scenarios. It is important to note that in MC systems, the fulfillment of HI task deadlines is crucial. The MCED H algorithm's primary objective is to ensure that critical HI tasks meet their deadlines, even if this results in lower success rates for LO tasks. Therefore, the lower overall success rate of MCED H is a trade off for guaranteeing the timely execution of critical tasks, making it more suitable for MC systems.

In the Point Generator scenario, the first utilization point we examine is 0.52. The success rates for EDF, ED H, and MCED H are 74.24%, 93.48%, and 99.96%, respectively, as shown in Fig. 9.A. Here, MCED H outperforms EDF by 34.67% and ED H by 6.93%. At lower utilization levels, MCED H consistently achieves a 100% success rate. At the highest utilization in Fig. 9.A, the success rates are 36.96% for EDF, 37.49% for ED H, and 49.88% for MCED H. This indicates a 34.92% improvement over EDF and a 33.03% improvement over ED H. Throughout this scenario, as shown in Fig. 9.B, the success rate for HI tasks remains at 100% for MCED H. These experimental results confirm that the MCED H algorithm provides significant improvements across all scenarios, particularly for HI tasks. The algorithm achieves substantial gains in success rates, ranging from 4.11% to 67.04% when compared to ED H and from 19.78% to 67.04% when compared to EDF, depending on the scenario.

The superior performance of MCED H in meeting HI task deadlines can be attributed to the implementation of virtual deadlines and the proactive mode switching mechanism. By assigning virtual deadlines to HI tasks in LO mode, the algorithm ensures sufficient time and energy resources are allocated for their execution, even in the event of a mode transition. The use of slack time and slack energy as indicators for mode switching enables the system to dynamically adapt to changing workload conditions, preventing deadline misses for critical tasks.

8. Discussion and future work

While the proposed approach offers significant advancements and addresses several challenges, it simplifies certain aspects to make the problem more manageable. Static power consumption and energy leakage from the energy storage unit are inherent characteristics of modern processors, especially with the shrinking size of transistors and the non ideal nature of energy storage units. Ignoring these factors can lead to an overestimation of energy efficiency. To address this limitation, we propose incorporating energy loss due to leakage and static power into the energy model by introducing a parameter called 'Leakage,' which represents the constant reduction of energy over time. This parameter would account for energy loss due to leakage currents and static power.

However, further research is needed, as the leakage rate may vary during system operation depending on factors like temperature and workload. In light of this, future work could explore dynamically mod eling leakage and static power to improve the accuracy of the energy model. Future work could explore dynamically modeling leakage and static power to improve the accuracy of the energy model. Techniques such as DPM and DVFS can be employed to mitigate the impact of these energy losses. These techniques help optimize power consumption by adjusting the system's operating state, thereby reducing static power consumption and minimizing the effects of energy leakage.

The proposed algorithm assumes a constant energy input rate. However, in real world systems, energy harvesting sources such as solar or wind can experience significant fluctuations over short periods, potentially impacting system reliability. Adaptive techniques or predic tion models could be incorporated to adjust the scheduling based on real time or predicted energy availability, enhancing system resilience and preventing failures due to energy shortages. Another potential solution to mitigate energy fluctuations is using multi source energy harvesting systems. Systems can achieve greater stability and reliability by combining multiple energy sources, such as solar, wind, or kinetic energy. Extending the proposed algorithm to support multi source en ergy harvesting would enhance its applicability, ensuring more reliable and energy efficient operation in dynamic environments.

To prevent system quality degradation and minimize energy wastage, we execute Lo tasks selectively in HI mode. Previous works execute tasks based on the IMC, FMC, PMC, and EMC concepts. Each approach may reduce task quality, and they execute tasks to prevent system quality degradation. In contrast, our proposed approach exe cutes tasks selectively while maintaining their quality. In future work, we plan to align our approach with these four concepts to leverage the benefits of both approaches.

While the current work focuses on single core systems, modern processors are moving towards multi core architectures. The MCED H algorithm can be extended to multi core and distributed systems to ac commodate this shift. Tasks can be allocated to cores in multi core sys tems based on metrics such as energy consumption and execution time. Challenges such as coordinated scheduling across nodes, communica tion overhead, and fault tolerance need to be addressed for distributed systems. Future research will explore these directions to enhance the applicability of MCED H to more complex system architectures.

9. Conclusion

This paper introduced the MCED H algorithm, a novel approach for scheduling MC tasks in energy harvesting systems. By incorporating virtual deadlines and a flexible mode switching mechanism, MCED H effectively addresses the challenges posed by energy constraints and task criticality. Experimental results demonstrate the algorithm's superior performance in meeting HI task deadlines while maintaining acceptable LO task completion rates compared to existing methods.

CRediT authorship contribution statement

Mostafa Tamimipour: Writing original draft, Software, Method ology, Investigation, Conceptualization. **Hakem Beitollahi:** Writing review & editing, Validation, Supervision, Project administration, Formal analysis. **Maryline Chetto:** Validation, Supervision, Project administration, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing finan cial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] M.M. Sandhu, S. Khalifa, R. Jurdak, M. Portmann, Task scheduling for energy-harvesting-based IoT: A survey and critical analysis, IEEE Internet Things J. 8 (18) (2021) 13825–13848.

[2] C. Maraveas, T. Bartzanas, Sensors for structural health monitoring of agricultural structures, Sensors 21 (1) (2021) 314.

[3] S. Priya, D.J. Inman, Energy Harvesting Technologies, vol. 21, Springer, 2009.

[4] A. Khaligh, O.C. Onar, Energy Harvesting: Solar, Wind, and Ocean Energy Conversion Systems, CRC Press, 2017.

[5] V. Pecunia, S.R.P. Silva, J.D. Phillips, E. Artegiani, A. Romeo, H. Shim, J. Park, J.H. Kim, J.S. Yun, G.C. Welch, et al., Roadmap on energy harvesting materials, J. Phys.: Mater. 6 (4) (2023) 042501.

[6] S. Vestal, Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance, in: 28th IEEE International Real-Time Systems Symposium, RTSS 2007, IEEE, 2007, pp. 239–243.

[7] A. Burns, R.I. Davis, Mixed Criticality Systems-A Review:(February 2022), York, 2022.

[8] S. Baruah, H. Li, L. Stougie, Towards the design of certifiable mixed-criticality systems, in: 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium, IEEE, 2010, pp. 13–22.

[9] J. Littlefield-Lawwill, R. Viswanathan, Advancing open standards in integrated modular avionics: An industry analysis, in: 2007 IEEE/AIAA 26th Digital Avionics Systems Conference, IEEE, 2007, pp. 2–B.

[10] D. Heffernan, C. MacNamee, P. Fogarty, Runtime verification monitoring for automotive embedded systems using the ISO 26262 functional safety standard as a guide for the definition of the monitored properties, IET Softw. 8 (5) (2014) 193–203.

[11] L.A. Johnson, et al., DO-178B: Software considerations in airborne systems and equipment certification, Crosstalk Oct. 199 (1998) 11–20.

[12] M.K. Bhatti, C. Belleudy, M. Auguin, Hybrid power management in real time embedded systems: an interplay of DVFS and DPM techniques, Real-Time Syst. 47 (2) (2011) 143–162.

[13] J. Luo, N. Jha, Static and dynamic variable voltage scheduling algorithms for real-time heterogeneous distributed embedded systems, in: Proceedings of ASP-DAC/VLSI Design 2002. 7th Asia and South Pacific Design Automation Conference and 15h International Conference on VLSI Design, IEEE, 2002, pp. 719–726.

[14] C. Moser, D. Brunelli, L. Thiele, L. Benini, Real-time scheduling for energy harvesting sensor nodes, Real-Time Syst. 37 (2007) 233–260.

[15] C.L. Liu, J.W. Layland, Scheduling algorithms for multiprogramming in a hard-real-time environment, J. ACM 20 (1) (1973) 46–61.

[16] Y. Abdeddaïm, Y. Chandarli, D. Masson, The optimality of PFPasap algorithm for fixed-priority energy-harvesting real-time systems, in: 2013 25th Euromicro Conference on Real-Time Systems, IEEE, 2013, pp. 47–56.

[17] M. Chetto, Optimal scheduling for real-time jobs in energy harvesting computing systems, IEEE Trans. Emerg. Top. Comput. 2 (2) (2014) 122–133.

[18] H. El Ghor, M. Chetto, Energy guarantee scheme for real-time systems with energy harvesting constraints, Int. J. Autom. Comput. 16 (2019) 354–368.

[19] B. Islam, S. Nirjon, Scheduling computational and energy harvesting tasks in deadline-aware intermittent systems, in: 2020 IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS, IEEE, 2020, pp. 95–109.

[20] M. Karimi, H. Choi, Y. Wang, Y. Xiang, H. Kim, Real-time task scheduling on intermittently powered batteryless devices, IEEE Internet Things J. 8 (17) (2021) 13328–13342.

[21] S.A. Chafi, M.K. Benhaoua, Energy harvesting deadline monotonic approach for real-time energy autonomous systems, Scalable Comput.: Pr. Exp. 25 (6) (2024) 5734–5744.

[22] P.-E. Hladik, H.e. Zahaf, S. Faucou, A. Queudet, ILP representation for limited preemption in energy-neutral single-core systems, in: Proceedings of the 32nd International Conference on Real-Time Networks and Systems, 2024, pp. 1–11.

[23] H.-E. Zahaf, P.-E. Hladik, S. Faucou, A. Queudet, Checkpointing for single core energy-neutral real-time systems, J. Syst. Archit. 161 (2025) 103371.

[24] S.K. Baruah, V. Bonifaci, G. d'Angelo, A. Marchetti-Spaccamela, S. Van Der Ster, L. Stougie, Mixed-criticality scheduling of sporadic task systems, in: Algorithms–ESA 2011: 19th Annual European Symposium, Saarbrücken, Germany, September 5–9, 2011. Proceedings 19, Springer, 2011, pp. 555–566.

[25] K. Agrawal, S. Baruah, A. Burns, Semi-clairvoyance in mixed-criticality schedul-ing, in: 2019 IEEE Real-Time Systems Symposium, RTSS, IEEE, 2019, pp. 458–468.

[26] A. Chaudhari, S. Baruah, Efficient schedulability analysis of semi-clairvoyant sporadic task systems with graceful degradation, in: Proceedings of the 30th International Conference on Real-Time Networks and Systems, 2022, pp. 116–126.

[27] Y.-W. Zhang, Energy aware algorithm based on actual utilization for periodic tasks in mixed-criticality real-time systems, Comput. Stand. Interfaces 79 (2022) 103563.
[28] A. Ali, S. Zeb, M. Alruwaili, A.M. Khattak, B. Hayat, K.-I. Kim, Mixed criticality reward-based systems using resource reservation, IEEE Access (2024).
[29] Y.-W. Zhang, C. Ouyang, Semi-clairvoyant scheduling in non-preemptive fixed-priority mixed-criticality systems, J. Syst. Archit. 159 (2025) 103332.
[30] Y.-W. Zhang, R.-K. Chen, A survey of energy-aware scheduling in mixed-criticality systems, J. Syst. Archit. 127 (2022) 102524.
[31] R. Jayaseelan, T. Mitra, X. Li, Estimating the worst-case energy consumption of embedded software, in: 12th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS'06, IEEE, 2006, pp. 81–90.
[32] R. Sun, J. Zhan, W. Jiang, Q. Dong, Y. Ye, Energy optimization of mixed-criticality distributed real-time embedded systems, J. Circuits Syst. Comput. 30 (05) (2021) 2150085.
[33] Y.-W. Zhang, R.-K. Chen, Energy aware fixed priority scheduling in mixed-criticality systems, Comput. Stand. Interfaces 83 (2023) 103671.
[34] Y.-W. Zhang, R.-K. Chen, Z. Gu, Energy-aware partitioned scheduling of imprecise mixed-criticality systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2023).
[35] E. Bini, G.C. Buttazzo, Measuring the performance of schedulability tests, Real-Time Syst. 30 (1) (2005) 129–154.
[36] M. Völp, M. Hähnel, A. Lackorzynski, Has energy surpassed timeliness? Scheduling energy-constrained mixed-criticality systems, in: 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium, RTAS, IEEE, 2014, pp. 275–284.