
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2024 SCPE. Volume 25, Issues 6, pp. 5734–5744, DOI 10.12694/scpe.v25i6.2670

ENERGY HARVESTING DEADLINE MONOTONIC APPROACH FOR REAL-TIME ENERGY AUTONOMOUS SYSTEMS

CHAFI SAFIA AMINA* AND BENHAOUA MOHAMMED KAMAL†

Abstract. This paper presents an innovative scheduling algorithm designed specifically for real-time energy harvesting systems, with a primary focus on minimizing energy consumption and extending the battery's lifespan. The algorithm employs a fixed priority assignment which is the deadline monotonic policy, we have chosen it for its optimality and superior performance compared to other fixed priority scheduling methods.

To achieve a balance between energy efficiency and system performance, we incorporated a DVFS (Dynamic Voltage and Frequency Scaling) technique into the algorithm. This adaptive approach enables precise control over the processor's operating frequency, effectively managing energy consumption while ensuring satisfactory system functionality.

The core objective of our scheduling algorithm centers on optimizing energy utilization in real-time energy harvesting systems, specifically tailored to extend the battery's operational life. Rigorous evaluations, including comprehensive comparisons against established fixed priority scheduling algorithms, validate the algorithm's efficacy in significantly reducing energy consumption while preserving the system's overall functionality.

By combining the deadline monotonic policy and DVFS technique, our proposed algorithm emerges as a promising solution for energy-autonomous systems, contributing to the advancement of sustainable energy practices in real-time applications. As energy harvesting technologies continue to progress, our algorithm holds valuable potential to provide critical insights for enhancing the efficiency and reliability of future energy harvesting systems.

Key words: Real-time systems, energy harvesting, embedded systems, power management, dynamic voltage and frequency selection (DVFS), Deadline monotonic policy, fixed priority assignment.

1. Introduction. The pervasive influence of real-time systems in today's technological landscape has propelled them to the forefront of various aspects of human life. From facilitating automated control systems to enabling the emergence of autonomous vehicles and revolutionizing medical fields, real-time systems have assumed a pivotal role in shaping modern society. As their significance continues to grow, researchers are earnestly dedicating their efforts to augmenting these systems and devising sophisticated scheduling algorithms to meet their stringent real-time requirements.

One of the most critical determinants of success in real-time systems is the task deadline. The seamless execution and accomplishment of tasks within their specified timeframes are instrumental in ensuring the overall effectiveness and reliability of these systems. Hence, meeting stringent temporal constraints becomes an imperative pursuit in the pursuit of optimized performance.

Concurrently, energy consumption poses as a significant challenge for real-time systems, with wireless devices being particularly affected. These devices, which often operate on limited battery life, are frequently deployed in settings where access to conventional power sources is scarce or absent altogether. As a consequence, the pursuit of innovative approaches to curtail energy usage while simultaneously upholding real-time imperatives represents a pivotal realm of ongoing research.

Numerous research endeavors have been dedicated to managing and minimizing energy consumption in various systems. One of the initial approaches to address this challenge was Dynamic Power Management (DPM), wherein the processor is transitioned into an idle mode to minimize energy usage when no tasks are ready for execution. Another notable approach is Dynamic Voltage and Frequency Scaling (DVFS) [4], which dynamically adjusts the processor's operating frequency to reduce energy consumption while ensuring

*Computer Science Department, LAPECI Laboratory, Oran 1 University, Oran, Algeria (chafi.amina@edu.univ-oran1.dz).
†Computer Science Department, Laboratoire Technologique en Intelligence Artificielle (LABTEC-IA), University of Mascara, Mascara, Algeria (k.benhaoua@univ-mascara.dz).

all deadlines are met. Extensive studies have highlighted the efficacy of these methods in effectively utilizing available energy resources [1, 2, 3].

Despite these advancements in energy efficiency, a major obstacle persists - the system's dependence on battery life. When batteries become discharged, the system's functionality is compromised, necessitating recharging or replacement. Regrettably, in some regions, access to power sources for recharging may be limited or non-existent [5, 6, 7, 8].

To overcome the challenges associated with battery charging and to enhance its lifespan, researchers have developed energy harvesting systems. Energy harvesting has garnered significant interest due to its potential to offer effective and sustainable solutions. For instance, wireless sensor devices can harvest energy from their ambient environment in real-time, enabling them to operate indefinitely. Such an approach holds the promise of theoretically achieving an infinite device lifetime. Energy can be harvested from various sources, including sunlight through photovoltaic (PV) cells (solar systems), wind, and vibrations, among others.

Within this context, we propose an algorithm for energy harvesting real-time systems, based on a variant of the deadline monotonic scheduling policy, with the primary objective of minimizing energy consumption. By strategically managing energy utilization, the proposed algorithm aims to extend the system's operational life. Also, we proposed a feasibilty test based on timing and energy constraints.

The remainder of this paper is organized as follows: Section 2 provides a comprehensive background and outlines related research. Section 3 presents the task model and architecture employed in our system. In Section 4, the background of the deadline monotonic scheduling approach is discussed. The proposed EH-DM algorithm is described in Section 5. In section 6, we discuss the worst case execution of our proposed scheduling algorithm. We proved the optimality of our algorithm in section 7. The feasibility test is presented in Section 8. Section 9 elaborates on how the processor's operating speed is computed. Section 10, presents the evaluation of the performance of EH-DM. Finally, Section 11 presents our concluding remarks.

2. Related work. In the context of real-time energy autonomous systems, researchers are dedicated to developing scheduling algorithms specifically tailored for systems utilizing energy harvesting. Over time, numerous approaches have been proposed to tackle this challenging problem. One of the pioneering solutions was introduced by Kansal et al. [9], who devised a model to accurately capture the power generated by a solar energy source. To address the scheduling problem, the authors employed a periodic approach and formulated it as a linear program.

In each period, the scheduling algorithm adapts the work cycle to account for any discrepancies between the actual energy levels and the anticipated values. This adaptive approach ensures that the system operates optimally even in the face of varying energy availability.

The work by Kansal et al. [9] represents a significant contribution to the field of real-time energy harvesting systems, laying the groundwork for subsequent research in designing efficient and adaptable scheduling algorithms to maximize system performance and extend battery life.

In the realm of task scheduling for real-time systems powered by renewable energy storage, Moser et al. [10] made significant contributions by introducing the Lazy Scheduling Algorithm (LSA). This novel real-time scheduling approach represents a variant of the well-established Earliest Deadline First (EDF) policy.

The fundamental operation of LSA is as follows: all ready tasks are prioritized based on their respective deadlines. The task with the earliest deadline is designated as the top priority and is executed first. Notably, LSA enables the system to run this task at full processor speed while ensuring the timely completion of all deadlines.

The ED-H algorithm, proposed by Chetto [11], is another approach in the domain of real-time energy harvesting systems. This algorithm adopts a distinctive strategy for task execution, permitting a task to be processed only under two conditions: when there is sufficient energy available or when the slack time is zero. In cases where these conditions are not met, the processor remains idle until enough energy is replenished, all while ensuring that the time constraints of the task are strictly adhered to.

Although the ED-H algorithm effectively manages energy resources during task execution, it introduces a potential drawback. The requirement for sufficient energy may lead to intervals of processor idleness, potentially resulting in missed task deadlines due to energy scarcity.

To address this concern and optimize the performance of real-time energy autonomous systems, further

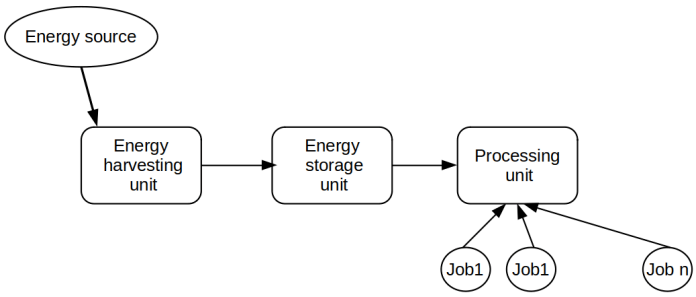

Fig. 3.1: Architecture model.

research is warranted. The focus can be directed towards refining the ED-H algorithm to strike a better balance between energy conservation and meeting strict task deadlines. By mitigating the issue of processor idleness and potential deadline misses caused by energy starvation, significant advancements can be made in enhancing the overall efficiency and reliability of real-time energy harvesting systems.

Fixed priority tasks scheduling is a well-explored area in research, with many studies addressing this important problem. Among these studies, the first algorithm in this context, PFP_{ASAP}, was proposed in [17] and [18]. This algorithm has been demonstrated to be an optimal scheduling solution for non-concrete task sets, proving its effectiveness in managing task priorities efficiently.

Another noteworthy algorithm is PFP_{ALAP}, which was introduced in [19]. This approach aims to delay job execution as late as possible, ensuring that jobs are executed only when sufficient replenished energy is available to support their execution. By adopting this strategy, PFP_{ALAP} optimizes energy utilization while still meeting task deadlines effectively.

In addition to the above, the PFP_{ST} scheduling heuristic was proposed in [20]. This algorithm prioritizes executing jobs as soon as enough energy is available in the storage unit to support their execution. When the required energy is not immediately available, PFP_{ST} switches to energy harvesting mode to gather the necessary energy for task execution.

3. System model.

3.1. Architecture model. The system considered in our work comprises an energy harvesting unit powered by a renewable energy source, an energy storage unit (reservoir), and a processing unit that operates at a given frequency.

The real-time system has access to both the timing characteristics of a task and the characteristics related to the available energy in the system. This enables the system to schedule tasks in an optimal manner, considering both timing and energy constraints. Figure 3.1 illustrates the architecture of the system.

3.1.1. Energy harvesting system unit. The system is powered by an energy source that collects energy from an external source and converts it into electrical power. We assume that the energy harvested from the ambient environment is a function of time. The energy is continuously harvested, and we denote $P_s(t)$ as the recharging function of the reservoir. The energy harvested during any time interval $[t_1, t_2]$, denoted as $E(t_1, t_2)$, is given as follows:

$$E_s(t_1, t_2) = \int_{t_1}^{t_2} P_s(t), dt \tag{3.1}$$

where:

- $E_s(t_1, t_2)$ represents the energy harvested within the time interval $[t_1, t_2]$.
- $P_s(t)$ denotes the worst-case charging rate on the harvested source power output.

For the sake of simplicity, we assume $P_s(t)$ to be a constant function, meaning that the energy harvested remains constant during each time slot. This allows for offline prediction, and it is computed using the following

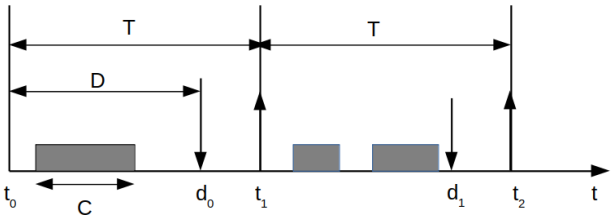

Fig. 3.2: Task model

formula:

$$E_s(t_1, t_2) = (t_2 - t_1)P_s \tag{3.2}$$

We supposed in our work that the energy is recharged continuously even during the execution of a job.

3.1.2. Energy storage unit. The system includes a battery to store the harvested energy. We assume that the energy level in the reservoir must remain within two limits: E_{min}, which is the minimum level of energy required to keep the system running, and E_{max}, which represents the maximum capacity of the reservoir. Thus, it follows that $E_{min} \leq E(t) \leq E_{max}$. The capacity of the storage unit, denoted as C, is the difference between these two limits: $C = E_{max} - E_{min}$.

It is essential for the capacity of the reservoir to be sufficient to execute at least one time unit of the jobs; otherwise, the taskset is considered infeasible.

3.1.3. Processing unit. The system incorporates a DVFS (Dynamic Voltage and Frequency Scaling) module, where the energy consumed by a task depends on the processor operating frequency.

Additionally, we make the assumption that the energy consumed during idle states or when charging and discharging the reservoir is negligible.

3.2. Task model. We consider a set of n periodic, synchronous, independent, and preemptable tasks denoted as $\Gamma = \tau_1, \tau_2, ..., \tau_n$.

Each task is represented by $\tau_i = (r_i, C_i, D_i, E_i, T_i, P_i)$, where $r_i, C_i, D_i, E_i, T_i, P_i$ correspond to the release time, worst-case execution time (WCET), relative deadline, worst-case energy consumption (WCEC), minimum inter-arrival time between two consecutive jobs, and priority of task τ_i, respectively. We assume that the energy consumption and the execution time are fully independent, and that $D_i \leq T_i$.

Each task consists of an infinite sequence of jobs. A job of task τ_i arrives at time $(r_i + kT_i)$ for each integer k such that $k \geq 0$, and it must be completed within D_i time units from its arrival time. During the interval $[r_i, d_i]$ where d_i is the absolute deadline of the job's i^{th} task , the job must receive C_i units of execution.

As mentioned in [11], the energy utilization factor U_e is computed as the sum of the ratios between the task's worst-case energy consumption (WCEC) and its period: $U_e = \sum_i^n \frac{E_i}{T_i}$. Similarly, the processor utilization factor U_p is calculated as the sum of the ratios between the task's worst-case execution time (WCET) and its period: $U_p = \sum_i^n \frac{C_i}{T_i}$.

4. Background algorithm. Deadline monotonic [12] is one of the major scheduling algorithms for fixed priority assignment. It assigns priorities to tasks based on their deadlines, where the priority of each task is inversely proportional to its deadline.

Deadline monotonic has been proven to be optimal for fixed priority scheduling and performs well when tasks have larger periods but relatively short deadlines. However, it loses its optimality when we integrate energy constraints into the scheduling process.

A sufficient schedulability test based on utilization was proposed in [13], but it is considered very pessimistic:

$$U(n) = \sum_{i=1}^{n} \frac{C_i}{D_i} \leq n(2^{\frac{1}{n}} - 1) \tag{4.1}$$

where n is the number of tasks.

Another exact test based on response time analysis was proposed for arbitrary fixed priorities, including deadline monotonic, in [14]. The worst-case response time (WCRT) is the maximum time interval between the arrival and finish instants of a task. The response time for each task is calculated using the following iterative formula:

$$\forall i \quad R_i = I_i + C_i \tag{4.2}$$

Where I_i represents the interference caused by the execution of higher-priority tasks:

$$I_i = \sum_{k \in hp(i)} \left\lceil \frac{R_i}{T_k} \right\rceil \times C_k \tag{4.3}$$

Furthermore, it has been proven that a task set is feasible with fixed priority assignment if and only if the response time of each task satisfies the condition:

$$\forall i \quad R_i \leq D_i \tag{4.4}$$

5. EH-DM Algorithm. Before presenting our algorithm let us give some definitions of slack time and slack energy, the reader can refer to [11] and [16] for more details

5.1. Slack time. The slack time at a current time t_c assigned to a job is computed as follow:

$$ST_{\tau_i}(t_c) = d_i - t_c - tdbf(\tau_i, t_c, d_i) - \sum_{d_k \leq d_i} C_k(t_c) \tag{5.1}$$

Where :

- $\sum_{d_k \leq d_i} C_k(t_c)$ is the remaining execution time of uncompleted tasks within $[t_c, d_i]$.
- $tdbf(\tau_i, t_c, d_i)$ is the time demand bound function in time interval $[t_c, d_i]$

The slack time of a set of jobs at instant t_c represents the maximum amount of time during which the processor could remain idle at time t_c, while respecting the timing constrains of all jobs. The slack time of a set of jobs is given by:

$$ST(t_c) = \min_{d_i > t_c} ST_{\tau_i}(t_c) \tag{5.2}$$

5.2. Slack energy. The notion of slack energy was firstly introduced by Chetto in [11], Slack energy of an instance of the current time t_c is computed as follow

Slack energy associated to a job of task τ_i is given by

$$SE_{\tau_i}(t_c) = E(t_c) + P_s(t_c, d_i) - edbf(t_c, d_i) \tag{5.3}$$

The slack energy of a task set is the minimum slack energy among all tasks :

$$SE(t_c) = \min_{t_c < r_i < d_i < d} SE_{\tau_i}(t_c) \tag{5.4}$$

Where : $edbf(t_c, d_i)$ is the energy demand bound function in time interval $[t_c, d_i]$.

Hereafter, we describe our algorithm, namely EH-DM (Energy Harvesting Deadline Monotonic), which aims to schedule tasks while minimizing energy consumption and guaranteeing all deadlines using an energy harvesting system.

EH-DM algorithm is designed to schedule tasks according to the deadline monotonic scheduling policy. The enhancement that EH-DM offers is that it considers not only timing constraints but also the level of energy in the reservoir. Based on this information, it decides whether to execute tasks at full speed, lower speed, or to let the processor idle.

Let us consider that we have a set of n periodic tasks in the ready queue initially. EH-DM follows the following rules when scheduling tasks:

1. Priorities are assigned offline according to deadline monotonic priority.
2. The processor stays idle in the time interval $[t_c, t_c + 1]$ if ready queue is empty $L(t_c) = \emptyset$.
3. The processor is idle within $[t_c, t_c + 1]$ if the ready queue is not empty but the energy reservoir is empty, $L(t_c) \neq \emptyset$ and $E(t_c) = 0$; it must charge energy to continue executing the rest of tasks.
4. The processor is busy in $[t_c, t_c + 1]$ if there is at least a ready task in the ready queue and a sufficient energy to execute at least a time slot for the ready task; $L(t_c) \neq \emptyset$ and $0 < E(t_c) <= C_{max}$; here we have two cases:
   - **Case 1 :** If the slack time is equal to 0 $ST = 0$, then tasks are executed at full processor speed s = 1.
   - **Case 2 :** If the slack time is positive $ST > 0$, then speed is reduced $s < 1$.

In rule 1, the priorities are assigned according to Deadline Monotonic policy and in the case when two tasks have the same deadline the priorities are assigned according to the task's index, the lower the index the higher the priority.

In rule 3, the processor should remain idle until it recharges sufficient energy for the execution of at least one time slot of the current task.

In rule 4, the processor can be active if there is enough energy for the execution of the current task. There are two cases:

In the first case, the processor should execute tasks at the maximum speed if $ST = 0$. In this situation, we cannot reduce the operating speed because doing so would increase the execution time of the task, potentially causing tasks to miss their deadlines. In the second case, the operating speed is reduced if $ST > 0$ in order to reduce the energy consumption of the tasks. We did not consider the cases when $SE = 0$ or $SE > 0$ because in both situations, it is better to reduce the speed to gain more energy for the execution of the remaining tasks.

Algorithm 1 EH-DM Algorithm

Input set of jobs
Output schedule

```
s = speedCompute(jobs)
for i = 0 to Hyperperiod do
   L = Readyjobs
   if (L = ∅) then
      Processor = idle
   else
      if E(t_c) = 0 then
         Processor = idle
      else
         if ST(t_c) = 0 then
            Execute(Job, 1)
         else
            Execute(Job, s)
         end if
      end if
   end if
end for
```

5.3. Example. Consider a set of two tasks given by τ_1 and τ_2 such that $\tau_1 = (0, 1, 4, 1, 5)$ and $\tau_2 = (0, 3, 8, 3, 10)$; the battery has a capacity of $C = 5$ where it is initially empty $(E(0) = 0)$ for the worst case; the energy harvested from the environment is constant $E_p = 1$.

6. Worst case scenario. The objective of this section is to describe the worst-case scenario that a taskset can encounter during its execution using an EH-DM scheduler.

The worst case scenario can be described as follows: All tasks are synchronously activated when the battery

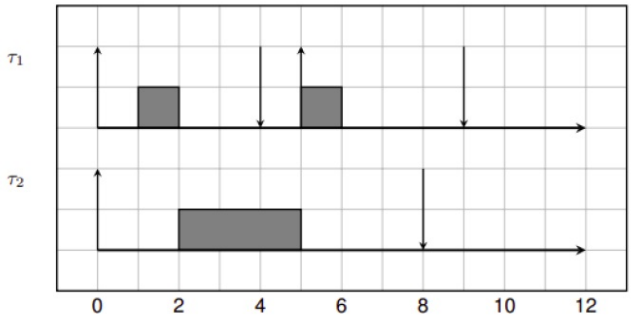

Fig. 5.1: EH-DM Algorithm

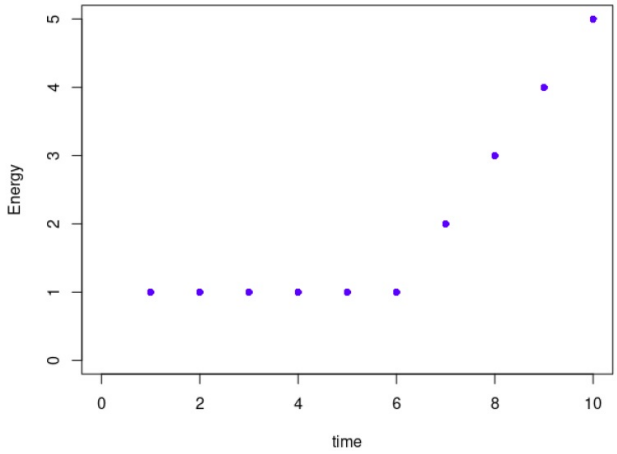

Fig. 5.2: Remaining energy in the reservoir

is empty. In this situation, the execution of tasks is postponed to replenish the necessary energy for executing at least one time unit of the current ready job. Additionally, the worst case arises whenever the energy harvested at each instant is lower than the energy consumed by the execution of one time unit of the current job. This means that the system consumes more energy than it replenishes.

As a consequence, tasks with lower priorities are at risk of missing their deadlines in such worst-case scenarios.

THEOREM 6.1. *Let* Γ *denote a set of n periodic tasks with constraint or implicit deadlines, ordered by deadline monotonic policy. The EH-DM worst-case scenario for any task of* Γ *occurs when the tasks are requested simultaneously when the battery is empty* ($E = 0$)*, and the harvested energy for each instant is lower than the energy consumed by a time unit of the current job.*

7. Optimality of EH-DM. The aim of our work is to develop an optimal fixed priority scheduling algorithm for autonomous systems, which takes into consideration timing constraints and energy constraints. We prove the optimality of our algorithm below.

THEOREM 7.1. *EH-DM is an optimal scheduling algorithm for periodic task sets with synchronous activation and constrained or implicit deadlines.*

Proof. Let us consider a taskset Γ of n periodic tasks. We suppose that Γ is not schedulable by EH-DM but it is schedulable by another fixed priority energy harvesting scheduling algorithm using the same priority assignment. This means that there exists at least one job denoted J_m that misses its deadline. According to

EH-DM rules, a job misses its deadline only if the energy available in the storage unit during the time interval of the execution of job J_m is lower than the required energy for its execution. Hence:

$$E_{J_m} < E(a_m) + (d_m - a_m) \times P_s(t) \tag{7.1}$$

If EH-DM is not optimal, then there should exist another fixed priority schedule that makes it schedulable; supposing that such a schedule exists, this means that at least one job is executed even if the available energy is not sufficient for its execution, which is not possible. Therefore, we prove that EH-DM is an optimal fixed priority scheduling algorithm for autonomous real-time systems. □

8. Feasibility analysis. In this section, we analyze the feasibility of using the EH-DM scheduling algorithm.

We consider the worst case to study the feasibility of this algorithm, assuming that all tasks are activated at the same time. This scenario encounters a critical instant when tasks with the lowest priority are activated simultaneously with all others having higher priority. Additionally, we assume that the energy storage unit is empty at $t = 0$.

In energy harvesting systems, scheduling algorithms must guarantee the feasibility of tasks concerning both timing constraints and energy constraints [15]. Therefore, we need to take into consideration two types of starvation as described in [11]:

- **Time starvation:** Occurs when a job reaches its deadline without completing its execution while there is energy in the reservoir ($E(t) > 0$).
- **Energy starvation:** Occurs when a job reaches its deadline without completing its execution, and the energy in the reservoir is exhausted ($E(t) = 0$).

8.1. Timing feasibility. To assess the timing feasibility of the EH-DM scheduling algorithm, we can use an exact test based on response time analysis, as proposed in [14]:

A task set is feasible with a fixed priority assignment if and only if the response time of each task R_i is less than or equal to its deadline D_i:

$$\forall i \quad R_i \leq D_i \tag{8.1}$$

8.2. Energy feasibility. To assess the energy feasibility of a task τ_i that is activated at time a_i, we can use the following condition:

τ_i is energy feasible if the available energy E_{τ_i} at the beginning of its execution is sufficient to cover its energy consumption during the execution interval, taking into account the harvested energy and the energy consumed by higher priority tasks. The condition can be expressed as follows:

$$\forall i \quad E_{\tau_i} \leq E(a_i) + R_i \times E_p - E_I(a_i, d_i) \tag{8.2}$$

where:

- E_{τ_i} is the available energy for task τ_i at time a_i.
- $E(a_i)$ is the initial energy level at time a_i.
- $R_i \times E_p$ is the energy harvested within the execution interval of τ_i.
- $E_I(a_i, d_i)$ is the energy consumed by tasks that have higher priority than τ_i within the interval $[a_i, d_i]$.

To examine feasibility for a taskset, we should iteratively verify the following two conditions for all jobs:

$$\forall i \quad R_i \leq D_i \text{ and } E_{\tau_i} \leq E(a_i) + R_i \times E_p - E_I(a_i, d_i) \tag{8.3}$$

9. Computing speed. The operating speed of the processor is computed offline for each time interval, starting from a job's arrival time and ending with an absolute deadline of the same job or another job. The speed is determined as the ratio between the demand function and the length of the interval. Finally, we choose the maximum value to guarantee the feasibility of all jobs. For each interval $[a_i, d_j]$, the speed is given by:

$$s = \frac{DF(a_i, d_j)}{d_j - a_i} \tag{9.1}$$

Algorithm 2 FeasibiltyTest Algorithm

Input: taskset
Output: feasibility
  for $i = 1$ to n **do**
    if ($R_i > D_i$ OR $E_i > E(a_i) + R_i \times E_p - E_I(a_i, d_i)$ **then**
      return False
    end if
  end for
  return True

Table 9.1: Algorithms overview.

Algorithms	Year	developer	Scheduling policy	Speed	Idle periods	Optimality
PFP_{ALAP}	2012	Chandarli et al.	DM	max	-	-
PFP_{ASAP}	2013	Abdeddaïm et al.	DM	max	-	+
PFP_{ST}	2011	Chetto et al.	FP Algorithm	max	-	-
EH-DM	2023	Chafi et al.	DM	DVFS	+	+

where $DF(a_i, d_j)$ is the demand function in the interval $[a_i, d_j]$; which is calculated as the sum of execution time of all jobs having their arrival time and absolute deadline within the interval $[a_i, d_j]$ and it is given by:

$$DF(a_i, d_j) = \sum_{a_i \leq a_k < d_k \leq d_j} C_k \tag{9.2}$$

Algorithm 3 speedCompute Algorithm

Input: job[m]
Output: speed
  Initialisation :
  speed = 0
  for $i = 1$ to m **do**
    for $j = 1$ to m **do**
      $= s = \frac{df}{job[j].d - job[i].a}$
      $speed = max(speed, s)$
    end for
  end for
  return *speed*

10. Performance evaluation. In this paper, we have proven the optimality of EH-DM algorithm for periodic, independent and preemptive tasks. In this section, we examine the performance of our algorithm and compare it with its competitors by analyzing the behaviour of each algorithm (PFP_{ALAP}, PFP_{ASAP}, PFP_{ST}) on the example from Tab. 10.1.

In this example, we consider the taskset presented in Table 10.1, and the capacity of the energy storage unit is $C = 5$, with $E_{max} = 5$ and $E_{min} = 0$. The energy is continuously replenished with a constant charging rate of $P_s(t) = 0.5$. Here, we specifically focus on the worst-case scenario.

We analyzed the performance of each method in terms of energy consumption, and then we measured the remaining energy in the reservoir at every instant during the hyperperiod. The results are presented in Figure 10.1.

We observed that when considering the worst-case scenario and starting the execution of tasks with an empty battery ($E(0) = 0$), both Algorithm PFP_{ALAP} and PFP_{ST} miss the deadline of τ_3 at $t = 18$, and the

Table 10.1: Example 2

	C	D	E	T
τ_1	1	4	1	5
τ_2	2	9	2	10
τ_3	4	18	4	20

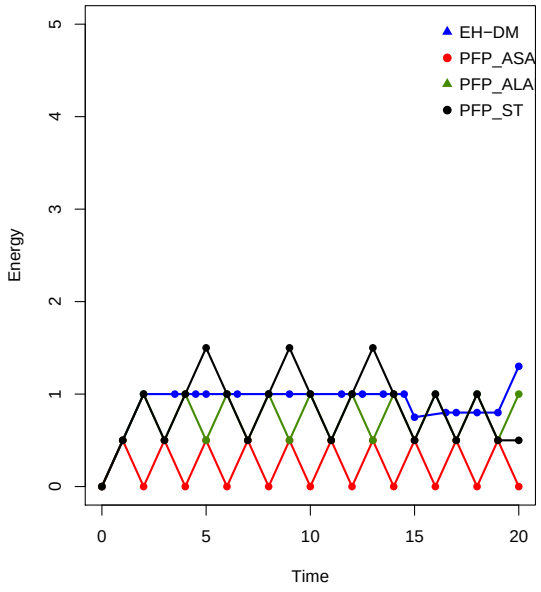

Fig. 10.1: Comparison between EH-DM, PFP_{ASAP}, PFP_{ALAP} and PFP_{ST}

last jobs of both τ_1 and τ_2 at $t = 19$ due to energy starvation. A deadline miss occurs at $t = 19$ for the last instances of τ_1 and τ_2 when using PFP_{ASAP}. This problem occurs due to energy starvation and the addition of idle periods to recharge the battery, which in turn delays the execution of some jobs and causes deadline misses. However, when using EH-DM, we observed that the execution time was extended without missing any deadline, and the energy was sufficient for the execution of all tasks, even if we started with an empty battery. This is due to the DVFS technique and its effectiveness in reducing energy consumption.

11. Conclusion. Due to the limitations of traditional battery power, there is an increasing demand for energy harvesting capabilities in various applications, including health, military, environmental, etc. An increasingly significant area of research is how to get an embedded device to operate perpetually and effectively. The major challenge is to perform effective processor time utilization while also optimizing energy consumption.

The aim of our work is to increase battery lifetime for real-time systems with energy harvesting, using fixed priority assignment. We proposed our first algorithm EH-DM to deal with this problem. EH-DM algorithm is a variant of the deadline monotonic algorithm for energy harvesting systems. The advantage of this algorithm is that it makes a better use of the energy and of the processor because it lets the processor idle only in cases where there are no tasks to execute. Otherwise, the processor remains busy, either operating at full speed or

with reduced speed to avoid wasting a lot of energy.

We will continue this study for future works to improve this algorithm in terms of complexity and development on real systems and to propose a feasibility test for this algorithm.

Acknowledgement. This work was supported in part by PHC-Tassili Project: 24MDU118 and PNR Project.

REFERENCES

[1] Li. Guohui, Fangxiao Hu, and Ling Yuan, *An energy-efficient fault-tolerant scheduling scheme for aperiodic tasks in embedded real-time systems.*, Third International Conference on Multimedia and Ubiquitous Engineering. IEEE, 2009.

[2] Allavena, Andre, and Daniel Mosse, *Scheduling of frame-based embedded systems with rechargeable batteries.*, Workshop on Power Management for Real-time and Embedded systems (in conjunction with RTAS 2001). 2001.

[3] Moser, C., Brunelli, D., Thiele, L., & Benini, L., *Real-time scheduling for energy harvesting sensor nodes.*, Real-time scheduling for energy harvesting sensor nodes. Real-Time Systems, 37, 233-260.

[4] Lin, X., Wang, Y., Chang, N., & Pedram, M., *Concurrent task scheduling and dynamic voltage and frequency scaling in a real-time embedded system with energy harvesting* , IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 35(11), 1890-1902.

[5] Chen, Jing, Tongquan Wei, and Jianlin Liang, *State-aware dynamic frequency selection scheme for energy-harvesting real-time systems*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 22.8 (2013): 1679-1692.

[6] Rusu, Cosmin, Rami Melhem, and Daniel Mossé, *Multi-version scheduling in rechargeable energy-aware real-time systems*; Journal of Embedded Computing 1.2 (2005): 271-283.

[7] Quaglia, Francesco, *A cost model for selecting checkpoint positions in Time Warp parallel simulation*, IEEE Transactions on Parallel and Distributed Systems 12.4 (2001): 346-362.

[8] Qiu, Qinru, Shaobo Liu, and Qing Wu, *Task merging for dynamic power management of cyclic applications in real-time multiprocessor systems*, International Conference on Computer Design. IEEE, 2006.

[9] KANSAL, Aman, HSU, Jason, ZAHEDI, Sadaf, et al., *Power management in energy harvesting sensor networks*, ACM Transactions on Embedded Computing Systems (TECS), 2007, vol. 6, no 4, p. 32-es.

[10] Moser, C., Brunelli, D., Thiele, L., & Benini, L. , *Real-time scheduling for energy harvesting sensor nodes*, Real-Time Systems 37.3 (2007): 233-260.

[11] Chetto, Maryline, *Optimal scheduling for real-time jobs in energy harvesting computing systems*, IEEE Transactions on Emerging Topics in Computing 2.2 (2014): 122-133.

[12] Audsley, N. C., Burns, A., Richardson, M. F., and Wellings, A. J., *Hard real-time scheduling: The deadline-monotonic approach*,(1991), IFAC Proceedings Volumes, 24(2), 127-132.

[13] Liu, Chung Laung, and James W. Layland, *Scheduling algorithms for multiprogramming in a hard-real-time environment*, Journal of the ACM (JACM) 20.1 (1973): 46-61.

[14] Bini, Enrico, and Giorgio C. Buttazzo, *Schedulability analysis of periodic fixed priority systems*, IEEE Transactions on Computers 53.11 (2004): 1462-1473.

[15] Abdeddaïm, Y., Chandarli, Y., Davis, R. I., & Masson, D., *Response time analysis for fixed priority real-time systems with energy-harvesting*, Real-Time Systems 52.2 (2016): 125-160.

[16] El Ghor, Hussein, and Maryline Chetto, *Energy guarantee scheme for real-time systems with energy harvesting constraints*, International Journal of Automation and Computing 16.3 (2019): 354-368.

[17] Abdeddaïm, Yasmina, and Damien Masson, *Real-time scheduling of energy harvesting embedded systems with timed automata*, IEEE International Conference on Embedded and Real-Time Computing Systems and Applications. IEEE, 2012.

[18] Abdeddaïm, Yasmina, Younès Chandarli, and Damien Masson, *The optimality of PFPasap algorithm for fixed-priority energy-harvesting real-time systems*, 25th Euromicro Conference on Real-Time Systems. IEEE, 2013.

[19] Chetto, Maryline, Damien Masson, and Serge Midonnet, *Fixed priority scheduling strategies for ambient energy-harvesting embedded systems*, IEEE/ACM International Conference on Green Computing and Communications. IEEE, 2011.

[20] Chandarli, Younès, Yasmina Abdeddaïm, and Damien Masson., *The fixed priority scheduling problem for energy harvesting real-time systems*, IEEE International Conference on Embedded and Real-Time Computing Systems and Applications. IEEE, 2012.