Optimal Real-Time Scheduling Algorithm for Wireless Sensors with Regenerative Energy

Hussein El Ghor, Maryline Chetto

Noname manuscript No.
(will be inserted by the editor)

Optimal Real-Time Scheduling Algorithm for Wireless Sensors with Regenerative Energy

Hussein El Ghor · Maryline Chetto

Received: date / Accepted: date

Abstract Dynamic Voltage and Frequency Scaling (DVFS) is a promising and broadly used energy efficient technique to overcome the main problems arising from using a finite energy reservoir capacity and uncertain energy source in real-time embedded systems. This work investigates an energy management scheme for real-time task scheduling in variable voltage processors located in sensor nodes and powered by ambient energy sources. We use DVFS technique to decrease the energy consumption of sensors at the time when the energy sources are limited. In particular, we develop and prove an optimal real-time scheduling framework with speed stretching, namely Energy Guarantee Dynamic Voltage and Frequency Scaling (EG-DVFS), that jointly accounts not only for the timing constraints, but also for the energy state incurred by the properties of the system components. EG-DVFS relies on the well-known ED-H scheduling algorithm combined with DVFS technique where the sensor processing frequency is fine tuned to further minimize energy consumption and to achieve an energy autonomy of the system. Further, an exact feasibility test for a set of periodic, aperiodic or evern sporadic tasks is presented.

Keywords Real-time systems · dynamic voltage and frequency scaling · energy harvesting · slack energy · optimal scheduling

1 Introduction

Wireless technology enables us to locate sensors in remote areas for monitoring and sensing purposes. Such systems are generally deployed in remote

H. El Ghor
University Institut of Technology, Lebanese University, B.P. 813 Saida, Lebanon.
E-mail: husseinelghor@ul.edu.lb

M. Chetto
IRCCyN Laboratory, University of Nantes, 1 Rue de la Noe, F-44321 Nantes, France.
E-mail: maryline.chetto@univ-nantes.fr

places where human beings have no access to reach these remote places all the time to manually recharge or replace the battery after it is discharged. In order for wireless sensor networks to become a ubiquitous part of our environment, energy harvesting sources to power sensor nodes must be employed to significantly increase the autonomy of the nodes and to overcome the energy limitations of battery-powered systems. Energy scavenging is essentially a technology that converts renewable generation into electrical energy which can be used to power electronic devices [1]. Several implemented prototypes like Heliomote [2] or Prometheus [3] demonstrate the dominance of renewable energy to achieve perpetual operation.

Hence, energy scavenging represents a new candidate for the natural progression of energy optimization techniques, especially when the system should be designed to operate perpetually. However, the energy captured from the environmental energy sources is often inconstant and unpredictable [4], and therefore, a novel power management framework should be developed dedicated to energy scanvenging systems that is able to operate properly to achieve energy autonomy [5].

Various power management techniques for reducing the energy consumption have been investigated in the earlier works. One of them is Dynamic Voltage Frequency Scaling (DVFS) [6] and the other is Dynamic Power Management (DPM) [7]. DVFS is a popular and broadly used power management method in real-time systems. This method can be used to decrease the power consumption of a system at the time when the sources are limited [8], [9]. Dynamic power management, on the other side, is a technique used to decrease the power consumption of a system. In dynamic power management the power manager puts the devices which are not being used in sleep mode and some power is saved [10], [11]. Although DVFS and DPM techniques are both dominant in reducing the energy consumption of the system, any portable device such as wireless sensor network will eventually deplete the energy reservoir. In addition, and while considering the energy reservoir as the only energy source, such design techniques will be wortheless.

Based on the green computing concept, developing energy-efficient mechanisms for real-time applications in wireless sensor networks becomes increasingly attractive. Nonetheless, the amount of energy harvested each day at a particular time varies in a nondeterministic manner. Therefore, energy management schemes are important in energy harvesting embedded systems as it helps in reducing the deadlines miss rate [12], [13]. Under this context, DVFS technique has been widely employed to reduce energy consumption in energy harvesting systems.

The main objective of this paper is to design a scheduler that is able to assign periodic tasks to time slots while still guaranteeing all timing and power requirements during the whole lifetime of the application. This can be achieved by integrating the ED-H scheduling algorithm, that is proved in [16] to be an optimal scheduler for preemptive uniprocessor scheduling of independent tasks, with a dynamic voltage and frequency scaling technique to guarantee predictable task executions even in the face of energy shortage and

to achieve lowest energy dissipation. This means adopting a strategy based on the complementary employment of an energy guarantee scheme for variable voltage processors dedicated to energy harvesting systems.

Given a real-time energy savenging system that is known to be feasible, we propose an Energy Guarantee - Dynamic Voltage and Frequency Scaling (EG-DVFS) scheduling algorithm with DVFS technique that allows to guarantee that all deadlines are met while still relying on a smaller solar cells as well as limited size of the energy storage element. The targeted system is predominantly energy efficient and the dissipated energy by tasks is not fixed since we can scale the processor speed to reduce energy consumption. Provided that the average scavenging power is sufficient for continuous operation, EG-DVFS makes best use of the available energy so as to minimize the necessary energy reservoir capacity.

Thus, our system operates in energy neutral mode. This means that instead of saving some energy to elongate the operating time of energy reservoir, the system only consumes the energy recharged by the harvested power from the environmental energy source while still minimizing the energy consumption of the system, by slowing down the processor speed to the lowest possible state, leading to the absence of no deadline violation whenever possible.

The rest of the paper is organized as follows: Section 2 presents the prior research related to this work, whereas the energy scavenging system model and assumptions are presented in Section 3. Section 4 gives the necessary background materials that are essential for understanding the paper. The Energy Guarantee - Dynamic Voltage and Frequency Scaling (EG-DVFS) scheduling algorithm is presented in section 5. The optimality of EG-DVFS is proved in section 6. Section 7 concludes the paper.

2 Prior Work

Energy scavenging system design have been extensively studied using dynamic voltage and frequency selection technique. Researchers mainly focus on reducing the energy consumption by the tasks, hence reducing the processor power while still meeting the tasks deadline. Fot this sake, several scheduling algorithms have been developed that focus on energy efficient techniques for energy scavenging real-time systems.

The first valuable work that really studies the problem of power management while considering ambient energy sources has been studied in [12]. Authors build a system that switches between busy mode and idle mode depending on harvested energy from the source. Later, Moser at al. [15] targets the case of scheduling tasks with deadlines on a uniprocessor system that is powered by a rechargeable energy reservoir. Authors propose an optimal idling energy-clairvoyant real-time scheduler named lazy scheduling algorithm (LSA). In LSA, the ready task with the earliest deadline is authorized to execute only if the system is able to keep on running at full processor speed and without violating its deadline. Disadvantages of LSA are the following:

first, tasks run at maximum processor speed and hence, deadines of some future tasks may not be respected because of energy shortage. Second, the total energy dissipated by a task is necessarily proportional to its execution time, which is not the real case and finally the slack time is not used for energy savings.

Recently, many researchers have extensively studied low power systems in accordance with Dynamic Voltage and Frequency Scaling. In [14], authors proposed an energy-aware DVFS (EA-DVFS) algorithm that aims to exploit the slack time as much as possible to reduce the deadline miss rate. This can be achived by using a good tradeoff between the saved energy and the processor speed. The available energy mainly depends on the energy stored in the reservoir and the energy harvested from the renewable energy source. In case of insufficient available energy, the processor slows down the task execution; otherwise, the tasks are executed at maximum processor speed. The advantage of EA-DVFS is that it increases the percentage of feasibly executed tasks and reduces the storage capacity in case of low overload. However, EA-DVFS sill suffer from some inconvenients: First, authors perform the energy availability test based only on the single current task. Second, the scheduler can continue its operation as long as the energy is sufficient to complete executing a task whose relative deadline is no more than the remaining operation time of system at maximum processor speed [14]. For example, let us consider that the energy reservoir has only 1% energy and the system can execute the current task at full speed without exhausting the energy reservoir. Then, the EA-DVFS scheduler will run the task at maximum processor speed, which is not the correct behavior. Second, when using the task slacks, the proposed algorithm only considers the currect task instead of take into account all tasks in the ready queue. Hence, slack time is not fully exploited for reducing energy consumption.

To further improve such inconvenients, authors in [5], propose harvesting-aware DVFS (HA-DVFS) scheduler. HA-DVFS algorithm works as follows: HA-DVFS scales down the processor speed whenever possible for reducing energy dissipation. Indeed, the processor will speed up the execution of tasks in case of energy overflow. In this case, the current running task will finish its execution earlier before its deadline and thus, more slack time will be available for the succeeding tasks where the proseccor will have the opportunity to be slowed down for more energy saving.

To fully exploit the slack time, Chetto [16] proposed a scheduling algorithm, ED-H (Earliest Deadline - Harvesting), that accounts on the limits of both time and energy. ED-H relies on two basic concepts: slack time and slack energy. ED-H was proved in [16] to be optimal. However, in order to build an optimal schedule, ED-H needs to know the future task characteristics and the profile of the used energy source.

Later in [17], authors proposed an algorithm based on the DVFS technique that dynamically concentrates all disperse free time together to harvest energy by dynamically scheduling harvesting tasks and service tasks.

3 System Model and Assumptions

Hereafter, we consider a real-time energy scavenging system that consists of three major units: energy source module, energy dissipation module and energy reservoir of limited capacity (see figure).

3.1 System Model

3.1.1 Energy Source Module

In this paper we talk about a real-time embedded system which is powered by an energy source unit that harvests the energy from external environmental sources like wind, sun, etc. Energy harvested from time t_1 to t_2 can be calculated using the following formula:

$$E_s(t_1, t_2) = \int_{t_1}^{t_2} P_s(t)dt \tag{1}$$

Where $P_s(t)$ is the worst case charging rate (WCCR) on the harvested source power output.
We assume that it is not possible to determine the exact amount of energy harvested before hand but we can certainly predict the energy harvested by shadowing the previous energy source profile.

3.1.2 Energy Dissipation Module

We consider a real-time embedded system equipped with a DVFS technique as the primary power management technique. We assume that the DVFS-enabled varaible speed processor works with N discrete frequencies ranging from $f_{min} = f_1 \leq f_2 \leq \cdots \leq f_N = f_{max}$ where f_1 is the minimum and f_N is the maximum frequency at which the processor can work.

We characterize here a set of independent and non-preemptive periodic tasks that can be denoted as follows: $\Gamma = \{\tau_i | 1 \leq i \leq n\}$. A five-tuple $(r_i, C_i, D_i, T_i, E_i)$ is used to characterize a periodic real-time task τ_i, where r_i, C_i, D_i, T_i and E_i indicate the arrival time, the worst case execution time, the relative deadline, the period and the worst case energy consumption of task τ_i, respectively. The instances of the tasks that are ready to get processed enter the ready queue where the task with the earliest deadline has the highest priority and should be executed first and preemption is allowed.

The power consumption of the tasks running in the processor depends on the processors frequency. This means that the power consumption of a task, say τ_i, is dependent on its voltage and frequency level which is given by the relationship

$$P_{dyn} = C_L V^2 f \tag{2}$$

Where C_L is the gate load capacitance.
We define a slowdown factor as the ratio of the current frequency to the maximum frequency of the processor and is denoted by S_n. This factor can range from S_{min} to 1.

$$S_n = \frac{f_n}{f_{max}} \tag{3}$$

When we stretch a task τ_i by a slowdown factor S_k, then its actual execution time, denoted by $C_i(a)$, at frequency f_k will be C_i/S_k. We assume that each task τ_i from a task set Γ have different power dissipation which also changes with its frequencies. Thus a task will have maximum power dissipation at its maximum frequency and it decreases as the frequency decreases. For convenience, we define power dissipation of a task as a function of task index and its slowdown factor $P_D(\tau_i, S_k)$. The energy dissipation of the task can be found as shown below:

$$E_D(St_i, Ft_i) = P_D(\tau_i, S_k) \times C_i/S_k \tag{4}$$

Where St_i and Ft_i are respectively the start time and finish time of task τ_i.

3.1.3 Energy Storage Module

Energy reservoir is also required in an energy scavenging real-time system since the system needs it to continue operation even in abscence of energy harvested from the environment. In our work, we use an energy reservoir, for example a rechargeable battery, that has a nominal capacity C and it is used to store the extra amount of energy harvested for the future use at times of crisis. In our case, we ignore the amount of energy wasted in the process of charging and discharging and hence we use an ideal energy storage unit.
There is an upper limit to the storage device denoted by C_{max} which is the maximum capacity of the energy reservoir. The lower limit of the capacitor, denoted as C_{min}, cannot be zero since there is an energy reserved in the capacitor for worst case scenarios. During the normal operation mode and at a given time t we have

$$C_{min} \leq C(t) \leq C_{max} \tag{5}$$

and

$$C(t_2) \leq C(t_1) + E_s(t_1, t_2) - E_D(t_1, t_2) \quad \forall\, t_2 > t_1 \tag{6}$$

3.2 Terminology

In this subsection, we state some definitions that will be helpful in the remainder of this paper.

Definition 1 A schedule ω for a task set Γ is said to be *time-valid* if the deadlines of all tasks of Γ are met in ω, considering that $\forall\ 1 \leq i \leq n$, $E_i = 0$.

Definition 2 A schedule ω for a task set Γ is said to be *energy-valid* if the deadlines of all tasks of Γ are met in ω, considering that $\forall\ 1 \leq i \leq n$, $C_i = 0$.

Definition 3 A schedule ω for a task set Γ is said to be *valid* if the deadlines of all tasks of Γ are met in ω, starting with a storage fully charged, with the energy generated by a given energy source.

Definition 4 A task set Γ is said to be *temporally-feasible* if there exists a valid schedule for Γ without considering its energy constraints.

Definition 5 A task set Γ is said to be *energy-feasible* if there exists an energy valid schedule for Γ.

Definition 6 A system is said to be *feasible* if there exists at least one valid schedule for ω with the given energy source and energy storage. Otherwise, it is *infeasible*.

When systems are infeasible, the limiting factors are either both time and energy or only time or only energy. We focus in this article on feasible systems only.

Definition 7 A scheduling algorithm ω for a task set Γ is said to be *optimal* if it finds a valid schedule whenever one exists.

4 Background Materials

4.1 Classical Concepts for Real-Time Scheduling

we recall some concepts related to real-time scheduling. To formally present the background material, we define the following factors: the processor demand and the slack time.

Definition 8 The processor demand of a task set Γ on the time interval $[t_1, t_2)$ is

$$h(t_1, t_2) = \sum_{t_1 \leq r_k, d_k \leq t_2} C_k \tag{7}$$

Denote now t_c as the current time where we have to schedule a task set Γ by a certain scheduling algorithm ω. In the real-time scheduling theory (without energy consideration), and for a given task set Γ, the term *slack time* is defined as the longest interval of time starting at t_c during which the processor may be idle continuously while still respecting all the timing constraints.

Definition 9 The slack time of a task τ_i at current time t_c is

$$ST_{\tau_i}(t_c) = d_i - t_c - h(t_c, d_i) - AT_i \tag{8}$$

Where AT_i is the total remaining execution time of uncompleted tasks currently ready at t_c and in the time interval $[t_c, d_i)$.

Hence, $ST_{\tau_i}(t_c)$ gives the time available by the processor after executing uncompleted tasks with deadlines at or before d_i.

The slack time of the task set Γ is then the minimum of the slack times of all tasks in Γ.

Definition 10 The slack time of a task set Γ at current time t_c is

$$ST_{\Gamma}(t_c) = \min_{d_i > t_c} ST_{\tau_i}(t_c) \tag{9}$$

Equation 9 represents the maximum continuous time that the processor can be inactive starting from time t_c while still respecting the deadlines of all the tasks in Γ.

4.2 Earliest Deadline-Havesting (ED-H) Scheduler

Before presenting the ED-H scheduler, we have to introduce some novel concepts related to the feasibe scheduling when considering both time and energy constraints. These factors are the energy demand and the slack energy.
Let us denote r_k , d_k and E_k as the release time, deadline and worst case energy consumption of task τ_k respectively.

Definition 11 The energy demand of a task set Γ on the time interval $[t_1, t2)$ is

$$g(t_1, t_2) = \sum_{t_1 \leq r_k, d_k \leq t_2} E_k \tag{10}$$

In [18], we defined the concept of slack energy of a task τ_i at a current time t_c as the amount of energy surplus in the reservoir that can be used from t_c till the start time of τ_j and without violating its timing and energy requirements.

Definition 12 The slack energy of a task τ_i at current time t_c is

$$SE_{\tau_i}(t_c) = E(t_c) + E_s(t_c, d_i) - g(t_c, d_i) \tag{11}$$

The slack energy of the task set Γ is determined by the minimum slack energy of all the tasks.

Definition 13 The slack energy of a task set Γ at current time t_c is

$$SE_{\Gamma}(t_c) = \min_{t_c < r_i < d_i < d} SE_{\tau_i}(t_c) \tag{12}$$

Where d is the deadline of the active task at time t_c.

ED-H scheduler [16] is an extension of EDF with energy budgeting that manages timing and energy requirements as well as the replenishment rate of the storage unit for monoprocessor embedded systems with solar energy harvesting capability. The novelty of ED-H is that it prevents energy starvation. The main intuition behind ED-H is that it dynamically adapts the processor state to the runtime harvested power variations without violating the deadlines of the tasks. This means that ED-H has to verify the following two conditions to let the processor busy:

1. The available energy in the reservoir is enough to execute the ready task with the highest priority even for the next unit timeslot.
2. The energy consumption in that time-slot must guarantee the energy feasibility of all future occurring tasks considering their timing and energy requirements and the replenishment rate of the storage.

When these conditions are verified, ED-H will schedule tasks according to the EDF policy. However, ED-H may result in unnecessary deadline violations if one of these conditions is not fulfilled. The processor will then stay idle so that the energy reservoir is fully replenished or the slack time becomes equal to zero. Note that the ED-H degenerates to the EDF scheduling policy if the processor utilization is equal to one.

ED-H was proved in [16] to be optimal for a considered real-time energy harvesting system.

Theorem 1 *The ED-H scheduling algorithm is optimal for the real-time energy harvesting model.*

Proof See [16].

Moreover, Chetto produced the necessary feasibility condition for a given task set Γ that is scheduled by the ED-H algortithm. The test for the schedulability of ED-H is reduced to test time-feasibility and energy-feasibility separately [16].

As for the time-feasibility condition, consider that for every task τ_i in a given task set Γ, $E_i = 0$.

Lemma 1 *A task set Γ is time-feasible by ED-H if and only if*

$$\sup_{0 \leq t_1 < t_2 < d_{max}} \frac{h(t_1, t_2)}{t_2 - t_1} \leq 1 \tag{13}$$

Where $d_{max} = \max_{1 \leq i \leq n} d_i$

Proof See [16].

Now, consider the energy-feasibility condition where it is assumed that for every task τ_i in a given task set Γ, $C_i = 0$.

Lemma 2 *A task set Γ is energy-feasible by ED-H if and only if*

$$\sup_{0 \leq t_1 < t_2 < d_{max}} \frac{g(t_1, t_2)}{C + E_s(t_1, t_2)} \leq 1 \tag{14}$$

Where $d_{max} = \max_{1 \leq i \leq n} d_i$

Proof See [16].

From Lemma 1, and Lemma 2, we can deduce the feasibility test for a given task set Γ.

Lemma 3 *A task set Γ is feasible by ED-H if and only if*

$$\sup_{0 \leq t_1 < t_2 < d_{max}} \frac{h(t_1, t_2)}{t_2 - t_1} \leq 1 \quad \textit{and} \quad \sup_{0 \leq t_1 < t_2 < d_{max}} \frac{g(t_1, t_2)}{C + E_s(t_1, t_2)} \leq 1 \tag{15}$$

Proof See [16].

5 Energy Guarantee - Dynamic Voltage and Frequency Scaling (EG-DVFS) Algorithm

5.1 Overview of the Scheduling Algorithm

The naive approach of simply scheduling tasks with full processor speed may produce some deadline violations. This may occur when after the execution of task, say τ_1, another task, say τ_2, is ready with the highest priority. If now the required energy to complete the execution of τ_2 at full processor speed is not available in the energy storage unit even when using the available slack time, the ED-H scheduler produces a deadline violation. If the processor speed would be scaled down to decrease the energy consumption of τ_2 instead of executing at full speed, this deadline violation might have been avoidable. These considerations directly lead us to the principle of Energy Guarantee - Dynamic Voltage and Frequency Scaling (EG-DVFS) algorithm: scale the processor speed only if it is necessary to decrease the energy consumption of tasks while still guaranteeing the deadlines of tasks. In other words, we need to perform a check at run-time for the energy availability and after that adjust the scheduling. If the available energy is not enough for a task's scheduling, we must make adjustments to deal with the energy shortage.

Initially, all tasks are executed according to EDF scheduler where the system operates at full processor speed. We assume that the start time of any given task is derived under the assumption that the task executes at the constant processor speed until its completion.
Let us consider that, there are M task instances in the ready queue. St_i and Ft_i are denoted as the start time and finish time of task τ_i respectively. Obviously, the start time of the first task instance, say τ_1, in the ready queue is equal to its release time.

$$St_1 = r_1 \tag{16}$$

Thus, we can compute the start time of the remaining task instances as:

$$St_i = max(r_i, Ft_{i-1}) \qquad 2 \leq i \leq M-1 \tag{17}$$

When applying the EG-DVFS policy for a ready task, say τ_i, with the highest priority, we have to ensure that the available energy is sufficient to to completely execute τ_i. This means that we have to compute the remaining energy in the energy storage unit at the end of the task execution using the following equation:

$$C(Ft_i) = C(St_i) + E_s(St_i, Ft_i) - E_i(St_i, Ft_i) \tag{18}$$

If $C(Ft_i) > 0$, then EG-DVFS instantaneouly executes τ_i at the scheduled start time and with full processor speed ($S_i = 1$).
On the other side, energy conflicts occur if the required energy is simply not available at the finishing time Ft_i. This means that when the energy is required to execute τ_i, the energy storage level is not sufficient to meet the deadline (energy reservoir is exhausted at Ft_i). In this step, we have to *"stretch"* the task executions to minimize their energy dissipation by executing at lower energy and frequency levels of the processor. Thus, EG-DVFS has to decide the slowdown factor for τ_i based on the processor utilization and energy state. Following this rule, the tasks can be slowed down by a factor S_n in cases where we have insufficient energy to run the task at its full speed which in turn increases the worst case execution time. Thus, we have to compute the slack time of the system at St_i and the task's execution time will be stretched to the actual execution time $C_i(a)$ where:

$$C_i(a) = C_i + ST(St_i) \tag{19}$$

The finishing time of task instance τ_i can be calculated as

$$Ft_i = St_i + C_i(a) \tag{20}$$

Consequently the start time of the next task instance will be:

$$St_{i+1} = max(r_{i+1}, Ft_i) \tag{21}$$

The slowdown factor is thus computed thanks to the following equation:

$$S_i = \frac{C_i}{C_i(a)} \tag{22}$$

It is worthwhile to mention that since, every task τ_i must complete the execution before its absolute deadline , then the following equation must hold:

$$r_i + d_i - c_i(a) \geq max\{r_i, t\} \tag{23}$$

Where t is the current time instance.

Furthermore, the execution of any real-time task τ_i is considered to be preemptable which means that when a task of higher priority becomes ready, it will preempt the execution of the current task of lower priority.

This implies that the start time of a task τ_i is equal to $St_i = max(r_i, Ft_{i-1})$ when τ_i finishes its execution without being preempted by other task. When preemption occurs, the start time of the newly arrived task is equal to r_i. That is

$$St_i = \begin{cases} max(r_i, Ft_{i-1}) & \text{if } r_i + d_i - C_i(a) \geq max(r_i, t) \\ r_i & \text{otherwise} \end{cases} \quad (24)$$

However, when we resume execution of the preempted task , it has the opportunity to run at the same or different processor speed, depending upon the system state.

5.2 Description of the EG-DVFS Scheduler

In what follows, we consider a given task set Γ that is known to be feasible for the real-time energy harvesting model. Let $Q_r(t)$ be the queue of uncompleted tasks ready for execution at t. The EG-DVFS scheduling algorithm obeys the following rules.

- **Rule 1:** The future running task in $Q_r(t)$ is ordered according to the EDF policy.
- **Rule 2:** The processor is imperatively idle in $[t, t+1)$ if $Q_r(t) = \phi$.
- **Rule 3:** The incoming power is wasted in $[t, t+1)$ if $Q_r(t) = \phi$ and $C(t) = C$.
- **Rule 4:** The processor operates at maximum speed in $[t, t+1)$ if $Q_r(t) \neq \phi$ and at least one of the following conditions is satisfied:
  1. $C(t) \approx C$.
  2. $ST_\Gamma(t) = 0$.
- **Rule 5:** The processor is slowed down by applying the DVFS technique in $[t, t+1)$ if $Q_r(t) \neq \phi$ and at least one of the following conditions is satisfied:
  1. $C(t) \approx 0$.
  2. $SE_\Gamma(t) \approx 0$.

  The ready task with the highest priority, say τ_k, is stretched by a slowdown factor S_n, its actual execution time $(C_k(a))$ at frequency f_n will be C_k/S_n.
- **Rule 6:** The processor operates maximum speed if $Q_r(t) \neq \phi$, $0 < C(t) < C$, $ST_\Gamma(t) > 0$ and $SE_\Gamma(t) > 0$.

All tasks in the ready queue are ordered according to the EDF policy. The only case that the processor is imperatively idle is when the ready queue is empty. The incoming power is wasted only when the ready queue is empty and the energy reservoir is fully replenished.
Rules 4.1 and *4.2* say that the processor is active and can operate at full speed if either the energy storage unit is fully charged or using slack time at current time t would cause a deadline violation of at least one future task. *Rules 5.1*

and *5.2* state that the processor must scale down its speed to decrease the energy consumption of the tasks if the energy rservoir is depleted or executing any task at full speed would prevent at least one future task from being executed before its deadline because of energy starvation (slack time is zero). In this case EG-DVFS must decide the speed at which the task should be executed depending on the energy availability. This means that for the execution of the ready task to be accomplished, we must consider the amount of energy harvested during its execution time and then decide the amount by which the task can be stretched in order to avoid energy deficiency at any point of time. *Rule 6* says that the processor operates at full speed when the energy reservoir capacity at time t is not empty even if there is slack time.

6 Properties of EG-DVFS Scheduling

6.1 Optimality of EG-DVFS

First, we will demonstrate that EG-DVFS is optimal with respect to minimizing the processor energy consumption and the maximum lateness. Let L_{max} be the maximum lateness that upper-bounds the time by which any task misses its deadline.

$$L_{max} = \max_{\tau_i}(Ft_i - d_i) \tag{25}$$

Theorem 2 *Given a task τ_i in a task set Γ with arbitrary arrival times r_i, computation times C_i and deadlines d_i, the EG-DVFS algorithm that at every scheduling instant t_i executes the task with the earliest deadline among all the ready tasks with a processing slowdown factor S_i is optimal with respect to minimizing the processor energy consumption and the maximum lateness. The optimum processing factor is*

$$S_i = \begin{cases} \frac{C_i}{C_i + ST_\Gamma(t_i)} & \text{if } SE_\Gamma(t_i) \approx 0 \text{ or } C(t_i) \approx 0 \\ 1 & \text{otherwise} \end{cases} \tag{26}$$

Proof Suppose that there can be another slowdown factor S_i' where we slow down the ready task with the earliest deadline, say τ_k, to the maximum possible extent just before its deadline in order to achieve the maximum energy savings. Clearly S_i' must be no more than S_i. Here, we have the following cases:
Case 1: The energy reservoir is fully replenished or there is no slack time. This case directly follows Rule 4. In this case the processor will execute at full processor speed, then $S_i' = S_i = 1$. This contradicts that there exists a slowdown factor to achieve more energy saving.
Case 2: The energy is fully depleted or the slack energy is zero. This case directly follows Rule 5. Here the processor must slow down its speed to achieve energy saving while still guaranteeing the deadlines of future tasks. To stretch the task execution at time t_i, we use the slack time $ST_\Gamma(t_i)$ of the system t_i. To obtain more energy saving, we must stretch the task execution by using another

slack time, say $ST'_\Gamma(t_i)$ greater than $ST_\Gamma(t_i)$. However, following the definition of slack time, we get that $ST_\Gamma(t_i)$ is the longest interval of time starting at t_i during which the processor may be idle continuously while still respecting all the timing constraints. Thus, using $ST'_\Gamma(t_i) > ST_\Gamma(t_i)$ can give rise to conditions where we have sufficient energy to execute the present task with S'_i speed, but still there is a timing constraint for the future tasks. Consequently even if the future task runs with the slowdown factor equals one which is highest frequency, it cannot meet its deadline as the task before it was slowed down to more than its maximum extent. This contradicts that S'_i exists.
This concludes that S_i is the optimal slowdown factor that achieves the best energy saving. In additon, since EG-DVFS stretches the task execution to its maximum extent without violating future deadlines, then the deadline of tasks will be approximately tight since tasks will finish very close to their deadlines and consequently, the maximum lateness L_{max} will be the smallest possible.

In what follows, we will prove that EG-DVFS Scheduling algorithms optimal in the sense that if EG-DVFS is not able to schedule a given task set Γ, then no other scheduling algorithm is able to schedule it. For this sake, we have to show that EG-DVFS makes the best use of the available time and energy.
The scheduling policy presented in this article is inherently energy-driven. Thus, a deadline violation occurs if EG-DVFS fails to completely execute a task, say τ_i, assigned by an energy dissipation E_i before its deadline d_i. A deadline violation can occur due one of the two following situations:

- **Time starvation:** A deadline d cannot be guaranteed when a task is not able to finish its execution before its deadline and the energy reservoir is not depleted at time d (i.e. $C(d) > 0$).
- **Energy starvation:** A deadline d cannot be guaranteed when a task is not able to finish its execution before its deadline because the energy reservoir is exhausted when the deadline violation occurs (i.e. $C(d) \approx 0$)

We suppose that when system is initialized, we have a full energy storage capacity, i.e., $C(0) = C$. Furthermore, the processor is idle at a given time t only if the ready queue is empty.
The proof of theorems 1 and 2 directly follows the proof of the optimality of ED-H in [16].

Theorem 3 *Let Γ be a set of tasks that are scheduled by the EG-DVFS algorithm. Suppose that a task with deadline d and release time r is missed because of time starvation with $C(d) > 0$. Then there exists a time instant t such that the processor load $\frac{h(t,d)}{d-t}$ in the interval $[t, d]$ exceeds one and no schedule exists where d and all previous deadlines are met.*

Proof Let us suppose that t_0 is the maximal time before d where the processor was running with a low processor speed. Clearly, such a time exists. At first, the processor constantly operates with full speed on the tasks within the time interval $[t_0, d)$. We consider here the two following cases:

Case 1: There is at least one ready task at time t_0.
The processor stops to slowdown the processor at time t_0 if:

Case 1a: The energy reservoir is fully recharged at time t_0, i.e. $C(t_0) = C$ from rule 4.1.

Case 1b: The slack time of the task set Γ becomes zero at time t_0, i.e. $ST_\Gamma(t_0) = 0$ from rule 4.2.

However, since the processor is constantly operating at full speed on tasks in time interval $[t_0, d)$ and $E(d) > 0$, then there is no slack time in $[t_0, d)$. Therefore, EG-DVFS will directly execute tasks with deadline at or before d which are ready at time t_0 and released within $[t_0, d)$ by applying the EDF strategy. But since no slack time is found, this implies that $d - t_0 > h(t_0, d)$ which contradicts that d is missed.

Case 2: There is no ready task at time t_0.
In this case t_0 must coincide with the arrival of a task, say τ_k, whose release time $r_k = t_0$ that verifies $r_k \leq r$. Here we have 2 cases:

Case 2a: $d_k \leq d$.
This means that τ_k is completely executed before d_2 since d is the first violated deadline and tasks are scheduled according to the EDF policy in EG-DVFS.

Case 2a1: According to rule 4, the processor operates at maximum speed at time r_k either because there is no slack time at r_k ($ST(r_2) = 0$) or because the energy reservoir at r_2 is fully replenished. But since $E(d) > 0$, then $ST(r_2)$ is no greater than zero. Consequently the processor demand in the time interval $[r_k, d)$, given by $h(r_k, d) = \sum_{r_k \leq r_j, d_j \leq d} C_j \leq d - r_k$ which contracdicts that the deadline d is missed. And no other scheduler can produce a valid schedule on $[r_k, d)$.

Case 2a2: According to rule 6, the processor operates at full speed at time r_k when $0 < C(t) < C$, $ST_\Gamma(t) > 0$ and $SE_\Gamma(t) > 0$. As the slack time of the task set Γ at time r_2 is greater than zero, this means that the processor demand, which is the sum of the WCET of all tasks with arrival and deadline within time interval $[r_k, d)$, is smaller than the allowed time $(d - r_k)$, i.e. $h(r_k, d) = \sum_{r_k \leq r_j, d_j \leq d} C_j < d - r_k$. This contradicts that d is missed.

Now, we can show that a related result also holds for the energy starvation case, too.

Theorem 4 *Let Γ be a set of tasks that are scheduled by the EG-DVFS algorithm. Suppose that a task with deadline d and release time r is missed because of energy starvation with $C(d) = 0$. Then there exists a time instant t such that the energy load $\frac{g(t,d)}{C+E_s(t,d)}$ in the interval $[t, d]$ exceeds one and no schedule exists where d and all previous deadlines are met.*

Proof Let us consider that deadline d is the first deadline of the task set that is missed by EG-DVFS due to energy starvation, i.e. $C(d) = 0$. Let t_0 be the largest time before d ($t_0 < d$) where a task with deadline greater than d is released such that the energy reservoir is fully replenished at t_0 ($C(t_0) = C$) and no task τ_i waiting just before t_0. As t_0 is the last time instance with the

above properties, then the processor is idle in the time interval $[t_0-1, t_0)$ since no task is ready (Rule 2). The processor is now operating with full speed at least in the time interval $[t_0, t_0+1)$. Here we have 2 cases:
Case 1: No task with deadline greater than d is executed within the time interval $[t_0, d)$.
The energy reservoir is fully charged time t_0, then according to Rule 4.1, the processor will at full speed and the stored energy is used to advance all waiting tasks with release time greater than or equal to t_0 and deadline at or before d with an energy demand equal to $g(t_0, d)$. But, since the task set Γ is feasible, then the demanded energy to complete the execution at full processor speed is no more than the energy reservoir capacity plus the energy drained from the source, or shortly $g(t_0, d) < C + E_s(t_0, d)$. From above, we can conclude that all tasks ready within the time interval $[t_0, d)$ are completely executed without depleting the energy reservoir, consequently $C(d) > 0$ which contradicts that d is missed.
Case 2: At least one task with deadline greater than d is executed within the time interval $[t_0, d)$.
Since there is no task waiting with deadline smaller than d, the processor would operate on task, say τ_2 with $d_2 > d$. Consider t_2 be the latest time where τ_2 executed. But since $d < d_2$ and tasks are executed according to the earliest deadline rule in EG-DVFS, then r_2 must be no more than r_1. At time t_2, one of the following situations occurs.

Case 2a: Tasks are executed at full processor speed all the times in $[t_2, d)$. Let's consider now that a task, say τ_3 is released after t_2 an with deadline $d_3 < d_2$, then τ_2 is preempted by τ_3. Hence, $SE_\Gamma(r_3)$ is greater than zero. This implies that $g(r_3, d) < C(r_3) + E_s(r_3, d)$. Consequently, the maximum amount of energy required by tasks released at or after r_3 and with deadline at or before d is $g(r_3, d)$. This contradicts that the energy reservoir is empty at d and hence the deadline d is not violated.

Case 2b: The processor works at lower speed from $t_3 - 1$ to t_3 with $t_3 > t_2$ and at full speed all in the time interval $[t_3, d)$. The processor stops working with lower speed at time t_3 by Rule 4 if $C(t_3) = C$ or $ST_\Gamma(t_3) = 0$. Here, there is no task ready at t_3 with deadline smaller than d since t_0 is the latest one. In addition, no task with deadline after d is executed in the time interval $[t_2, d)$ which includes time t_3. Thus all the energy drained from the source is used to advance waiting tasks with deadline greater than d. To maintain a maximum energy level at r, the processor will contineously scale the processor speed till time r where $C(r) = C$. But since the task set Γ is feasible, then $g(r, d) < C(r) + E_s(r, d)$. That contradicts deadline violation of d and $E(d) = 0$.

From the above two theorems, we conclude the following: First, in the time starvation case, there exists some interval of time $[t, d)$ before the deadline d is violated such that the processor load $\frac{h(t,d)}{d-t}$ exceeds one. And second, in the energy starvation case, there exists some interval of time $[t, d)$ before the deadline d is violated such that the energy load $\frac{g(t,d)}{C+E_s(t,d)}$ exceeds one. These

considerations lead us to Theorem 3 one of the major results for uniprocessor scheduling in real-time energy harvesting (RTEH) systems with DVFS scaling:

Theorem 5 *The EG-DVFS scheduling algorithm is optimal for RTEH systems with DVFS scaling.*

Proof The proof of this theorem follows immediately from Theorems 3 and 4. We consider in the above theorem the case of scheduling a task set Γ by EG-DVFS algorithm where a deadline d of task τ is missed and it is the first deadline in Γ that is missed by EG-DVFS. In what follows, we distinguish between the case where d is violated because of time starvation (Theorem 3) and energy starvation (Theorem 4) respectively.
According to theorem 3, time starvation occurs when there exists some interval of time $[t, d)$ before the deadline d is violated such that the processor load $\frac{h(t,d)}{d-t}$ exceeds one where $C(d) > 0$. Then, if EG-DVFS is unable to respect the deadline d and all earlier deadlines simultaneously in Γ, then no other scheduling algorithm can do that. Same to Theorem 4, where energy starvation occurs when there exists some interval of time $[t, d)$ before the deadline d is violated such that the energy load $\frac{g(t,d)}{C+E_s(t,d)}$ exceeds one. This case also concludes that it is impossible for another scheduling algorithm to guarantee the execution of task τ with deadline d and all earlier deadlines simultaneously. But, since every time deadline d is violated by EG-DVFS, we have either the time starvation reason or the energy starvation reason, and we prove that in both reasons, it is impossible for another scheduler to meet deadline d and all earlier deadlines simultaneously, then conclude that EG-DVFS is optimal.

6.2 Feasibility Analysis

In this section, we are concerned with the EG-DVFS algorithm by which, given a task set Γ, we must be capable of answering the following question: if Γ is feasible by ED-H, does it remain feasible by EG-DVFS? According to Definition 6, a task set is said to be feasible if there exists at least one valid schedule with the given energy source $E_s(t)$ and energy storage C.
As we demonstrated in Theorem 5, we proved that EG-DVFS is optimal for a RTEH system with DVFS scaling, we must derive the conditions under which the EG-DVFS scheduler avoids deadline violations for a given task set Γ. In what follows, we will prove that a task set that is feasible by ED-H remains feasible by EG-DVFS. To do this, we will separate our proof to time-feasibility condition and and energy-feasibility condition.

Theorem 6 *A task set Γ that is time-feasible with ED-H remains feasible with EG-DVFS.*

Proof In the time constraint case, we consider that tasks have processing requirements only. This means that for every task $\tau_k \in \Gamma$, $E_k = 0$. In the classical scheduling theory, we say that Γ is feasible if the processor demand

in every time interval must be less than or equal to the length of this interval, or simply the processor utilization must be no more than one. Let us consider U_p and U'_p as the processor utilization of ED-H and EG-DVFS respectively. If ED-H is feasible then $U_p \leq 1$. We must prove that if $U_p \leq 1$, then $U'_p \leq 1$.
Suppose that the task set Γ is not time-feasible by EG-DVFS, then $U'_p > 1$. But $U'_p = \sum_{i=1}^{n} \frac{C_i(a)}{T_i} = \sum_{i=1}^{n} \frac{C_i+ST(St_i)}{T_i} = U_p + \sum_{i=1}^{n} \frac{ST(St_i)}{T_i}$. Then $U_p + \sum_{i=1}^{n} \frac{ST(St_i)}{T_i} > 1$.
By multiplying the whole equality by the hyperperiod T_{LCM}, we get $T_{LCM}U_p + T_{LCM} \sum_{i=1}^{n} \frac{ST(St_i)}{T_i} > T_{LCM}$. But since Γ is periodic, then $T_{LCM} \sum_{i=1}^{n} \frac{ST(St_i)}{T_i}$ is equal to the total slack time ST_{tot}. By substitution, we get $T_{LCM}U_p + ST_{tot} > T_{LCM}$, then $ST_{tot} > T_{LCM}(1 - U_p)$.
According to ED-H, if Γ is time-feasible, then $U_p \leq 1$ and the total slack time during a whole hyperperiod T_{LCM} is equal to $T_{LCM}(1 - U_p)$. This contradicts that $U'_p > 1$. Consequently, EG-DVFS is time-feasible since it is not processor-overloaded in any time interval.

According to the energy constraint case, tasks in Γ have only energy constraints. This means that for every task $\tau_k \in \Gamma$, $C_k = 0$.

Theorem 7 *A task set Γ that is energy-feasible with ED-H remains feasible with EG-DVFS.*

Proof Since Γ is energy-feasible by ED-H, then in every interval of time $[t_1, t_2)$, the energy demand is necessarily no more than the available energy in $[t_1, t_2)$, i.e. $g(t_1, t_2) \leq C(t_1) + E_s(t_1, t_2)$. Let $g'(t_1, t_2)$ be the energy demanded by tasks according to EG-DVFS. We consider two cases:
Case 1: The processor executes all tasks with full speed.
In this case, the demanded energy is the same in EG-DVFS and ED-H, i.e. $g'(t_1, t_2) = g(t_1, t_2)$. But since Γ is energy-feasible by ED-H, then $\forall (t_1, t_2) \subset [0, d_{max})$, $g(t_1, t_2) \leq C + E_s(t_1, t_2)$, consequently $g'(t_1, t_2) \leq C + E_s(t_1, t_2)$. This concludes that when ED-H is energy-feasible, then EG-DVFS is also energy-feasible.
Case 2: The processor scales down its speed to execute some tasks τ_k in Γ.
In this case, the consumed energy by tasks is decreased by scaling down the processor speed to save energy. Hence, the energy demanded in EG-DVFS must be less than that of ED-H i.e. $g'(t_1, t_2) < g(t_1, t_2)$. But since Γ is energy-feasible by ED-H, then $\forall (t_1, t_2) \subset [0, d_{max})$, $g(t_1, t_2) \leq C + E_s(t_1, t_2)$, Hence, $g'(t_1, t_2) \leq C + E_s(t_1, t_2)$, and consequently Γ is energy-feasible by EG-DVFS.

From theorems 6 and 7, we can derive the feasibility of EG-DVFS.

Theorem 8 *A task set Γ that is feasible with ED-H remains feasible with EG-DVFS.*

7 Conclusion

In this paper, we studied the problem of energy guarantee scheduling and dynamic voltage/frequency selection technique targeting at real-time systems

with ambient energy sources. To this end, we proposed an Energy Guarantee DVFS (EG-DVFS) algorithm that considers both energy and timing constraints of the energy harvesting systems. The purpose from EG-DVFS was successfully achieved by compensating the extra/less energy harvested from the environment in such a way that the perpetual and self-sustaining operation of the system can be achieved. EG-DVFS is proved to be optimal and appropriate in minimizing the maximum lateness and the processor energy consumption. We also proved that if a given a task set Γ is feasible by ED-H, then it is also feasible by EG-DVFS. In this sense, EG-DVFS is able to feasibly schedule any task set as long as both the processor load and the energy load are no more than one.

There are number of places where we can work in order to enhance the above mentioned algorithm further. First, we will explore dynamic task allocation and frequency selection scheme for multiprocessor systems based on the scheduling scheme proposed in this paper. Second, we will adapt the proposed scheduler to fixed priority environments.

References

1. S. Priya and D.-J. Inman, Energy Harvesting Technologies. New York, NY, USA: Springer-Verlag, (2009).
2. V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. B. Srivastava, Design considerations for solar energy harvesting wireless embedded systems, in Proc. Int. Symp. Inf. Process. Sensor Netw., pp. 457-462, (2005).
3. X. Jiang, J. Polastre, and D. E. Culler, Perpetual environmentally powered sensor networks, in Proc. Int. Symp. Inf. Process. Sensor Netw., pp. 463-468, (2005).
4. Marco Severini, Stefano Squartini, Francesco Piazza, Energy-aware lazy scheduling algorithm for energy-harvesting sensor nodes, Neural Computing and Applications, 23:1899-1908, December (2013).
5. Shaobo Liu, Jun Lu, Qing Wu, and Qinru Qiu, Harvesting-Aware Power Management for Real-Time Systems With Renewable Energy, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 20, no. 8, August (2012).
6. Lei Miao, Yong Qi, Di Hou, Chang-li Wu and Yue-hua Dai, Dynamic Power Management and Dynamic Voltage Scaling in Real-time CMP Systems, In Proceedings of IEEE NAS, (2007).
7. S. Liu, Q. Qiu, and Q. Wu, Task merging for dynamic power management of cyclic applications in real-time multi-processor systems, in Proc. Int. Conf. Comput. Design, pp. 397-404, Oct. (2006).
8. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava, Power optimization of variable-voltage core-based systems, IEEE Trans. Comput. Aided Design Integr. Circuits Syst., vol. 18, no. 1, pp. 1702-1713, Dec. (1999).
9. J. Luo and N. K. Jha, Static and dynamic variable voltage scheduling algorithms for real-time heterogeneous distributed embedded systems, in Proc. VLSI Design, pp. 719-726, (2002).
10. Y. Lu, L. Benini and G. D. Micheli, Low-power task scheduling for multiple device, in Proc. Int. Workshop Hardw. Softw. Codesign, pp. 39-43, (2000).
11. Amit Sinha and Anantha Chandrakasan, Dynamic Power Management in Wireless Sensor Networks, IEEE Design and Test of Computers, March-April (2001).
12. A. Kansal, J. Hsu, S. Zahedi and M. B. Srivastava, Power management in energy harvesting sensor networks, in ACM Transactions on Embedded Computing Systems (TECS07), vol. 6, no. 4, Sep. (2007).
13. S. Liu, Q. Wu and Q. Qiu, An adaptive scheduling and voltage/frequency selection algorithm for real-time energy harvesting systems, in DAC, (2009).

14. S. Liu, Q. Qiu and Q. Wu, Energy Aware Dynamic Voltage and Frequency Selection for Real-Time Systems with Energy Harvesting, In Proc. of DATE, pp. 236-241, (2008).
15. C. Moser, D. Brunelli, L. Thiele, L. Benini, Real-time scheduling for energy harvesting sensor nodes, Real-Time Systems, Volume 37, Issue 3, pp. 233260, (2007).
16. M. Chetto, Optimal Scheduling for Real-Time Jobs in Energy Harvesting Computing Systems. IEEE Transactions on Emerging Topics in Computing (TETC), IEEE Computer Society, (2014).
17. Y. Tan, X. Yin, X., A dynamic scheduling algorithm for energy harvesting embedded systems. J Wireless Com Network, (2016).
18. Hussein EL Ghor, Maryline Chetto, Rafic Hage Chehade, A real-time scheduling framework for embedded systems with environmental energy harvesting, Computers and Electrical Engineering 37, pp. 498-510, (2011).