Energy Guarantee Scheme for Real-time Systems with Energy Harvesting Constraints

Article *in* International Journal of Automation and Computing · January 2019

DOI: 10.1007/s11633-018-1166-3

Hussein El Ghor
Lebanese University
23 PUBLICATIONS **83** CITATIONS

Maryline Chetto
University of Nantes
112 PUBLICATIONS **763** CITATIONS

International Journal of Automation and Computing

DOI: 10.1007/s11633-018-1166-3

Energy Guarantee Scheme for Real-time Systems with Energy Harvesting Constraints

Hussein El Ghor[1] Maryline Chetto[2]

[1]Laboratory of Embedded and Networked Systems, Faculty of Technology, Lebanese University, B. P. 813 Saida, Lebanon

[2]Laboratory of Numerical Sciences of Nantes, University of Nantes, 2 Avenue of Professor Jean Rouxel, 4475 Carquefou, France

Abstract: The growth of environmental energy harvesting has been explosive in wireless computing systems especially when replacing or recharging batteries manually is impracticable. This work investigates the scheduling of periodic weekly hard real-time tasks under energy constraints. Based on this motivation, we proposed a real-time scheduling algorithm, namely energy guarantee dynamic voltage and frequency scaling (EG-DVFS), that utilizes the earliest deadline-harvesting (ED-H) scheduling algorithm combined with dynamic voltage and frequency scaling. This one is qualified as real-time since tasks must satisfy their timing constraints. We assume that the preemptable tasks receive dynamic priorities according to the earliest deadline first (EDF) rule. EG-DVFS adjusts the processor's behavior by characterizing the properties of the energy source module, capacity of the stored energy as well as the harvested energy in a future duration. Specifically, tasks are executed at full processor speed if the amount of energy in the battery is enough to finish its execution. Otherwise, the processor slows down task execution to the lowest possible processor speed while still guaranteeing to meet all the timing constraints. EG-DVFS mainly depends on the on-line computation of the slack time and the slack energy with dynamic voltage and frequency selection in order to achieve an improved system performance. Experimental results show that EG-DVFS can achieve capacity savings up of up to 33% when compared to ED-H.

Keywords: Real-time systems, energy harvesting, embedded systems, power management, dynamic voltage and frequency selection (DVFS), ED-H scheduler.

1 Introduction

Energy management is a central problem in the design of real-time systems including embedded wireless devices. Various power management techniques for reducing the energy consumption have been investigated in the literature. One of them is dynamic power management (DPM)[1] that achieves energy efficiency by putting a device which is not being used in a low power state or sleep mode. The device becomes active again when some requests arrive, and work in high power state[2]. Another power management technique is dynamic voltage and frequency selection (DVFS) which is applied to decrease energy dissipation by lowering the operating frequency of the processor[3].

Advancements in wireless technologies enable us to position embedded wireless computing systems in remote areas for monitoring or sensing purposes. For example, sensor nodes can be employed in critical scenarios such as natural catastrophes and artificial disruptions[4]. Sensor nodes help in monitoring physical and environmental conditions such as temperature and pressure[5]. It is not permitted to manually recharge or replace the battery in many remote places. Consequently, renewable energy sources that are found in all parts of our environment should be employed. Harvesting energy from ambient sources seems to be an appropriate approach to increase the life-time of wireless systems.

By definition, energy harvesting is the process by which energy is captured from external sources and converted into electricity to power small, autonomous devices, making them self-sufficient, often for a long period of time[6].

Several technologies for deriving ambient energy from the environment have been demonstrated including solar power, thermal energy, wind energy, salinity gradients, kinetic energy, and many others[7]. Many modules for energy harvesting have been developed in the literature. Heliomote[8] and Prometheus[9] are the first prototypes which proved the feasibility of energy harvesting for small autonomous devices.

Nonetheless, harvested energy typically varies with time in a non-deterministic manner. Therefore, energy management schemes are required in energy harvesting embedded systems so as to guarantee an acceptable quality of service characterized by the deadlines miss rate[10].

In the context of battery-operated real-time systems, two constraints need to be studied: energy and timing constraints. Hence, a high performance scheduling policy should take into consideration the properties of the energy source, the limitation in the energy storage capacity as well as power consumption of the executed tasks.

Research Article

Manuscript received November 24, 2017; accepted November 14, 2018

Recommended by Associate Editor Zheng-Tao Ding

In a real-time energy harvesting system, the role of any scheduler is to assign the tasks to time slots while still respecting all timing and power requirements during the whole lifetime of the application. Conventional task scheduling and power management techniques are no longer convenient under energy harvesting considerations. They cannot adapt their behavior to the uncertainty in the available energy profile. A new power management technique with energy harvesting awareness should be proposed in order to suitably exploit both the processing capability and the available ambient energy.

1.1 Problem formulation

In this paper, we target a system of three components: a general purpose dynamic voltage and frequency scaling processor, an energy harvester and a rechargeable energy storage unit with a limited capacity such as battery or supercapacitor.

For this reason, we address the real-time scheduling problem in a uniprocessor platform. We study the case where a task can be preempted and later resumed at any time without any time loss associated with such preemption. All the tasks must be successfully scheduled within their timing requirements without any deadline failure so as to avoid intolerable damage.

Many challenges lie ahead so as to make a real-time energy scavenging system work effectively. Among them is the development of energy management techniques and scheduling algorithms that produce a valid schedule whenever possible while still respecting all deadlines of the tasks. Specifically, we consider the following features:

1) An energy harvesting unit is used to harvest the energy from one or several external sources in order to charge the energy storage unit.

2) Tasks may be the recurring invocations of periodic tasks. The parameters of each task can be determined prior to system run-time.

3) The energy source acts as a function of time. Hence, it is not possible to determine the exact amount of energy harvested beforehand, but we can certainly predict the energy harvested on near future at run time by shadowing the previous energy source profile.

4) The scheduler is able to alter the microprocessor's operating voltage at run-time.

5) The instantaneous power consumption of any task is assumed to be greater than the incoming power from the harvesting unit.

6) No harvesting energy is wasted because of energy overflow except when there are no ready tasks and the storage unit is fully replenished.

1.2 Contributions

In this work, we propose the so-called energy guarantee dynamic voltage and frequency scaling (EG-DVFS) power management algorithm. EG-DVFS utilizes the earliest deadline-harvesting (ED-H) scheduling rule combined with dynamic voltage and frequency scaling facilities to guarantee predictable execution for every task even in the face of energy shortage.

The work presented in this paper provides the following contributions to research:

1) We present an online algorithm that permits us to answer the three following questions: How and when to put the processor in idle versus active state? How to select the active task? How to compute the frequency of the processor for executing the selected task?

2) Our power management approach is implemented without any prior information about the energy source module that is considered to be uncontrollable and time-varying.

3) The EG-DVFS policy is based on trading two notions: slack time and slack energy. It intelligently dynamically selects the processing speed for every task depending on energy and time considerations.

4) The DVFS technology enables us to reduce energy consumption while still guaranteeing the absence of no deadline violation whenever possible.

1.3 Outline

The rest of this paper is organized as follows: In Section 2, we present related works. The system model and terminology are introduced in Section 3. In Section 4, we give necessary background materials. The EG-DVFS policy is described in details in Section 5. Simulation results and discussions are presented in Section 6. Finally, Section 7 concludes the paper.

2 Related work

The emerging technology of the energy scavenging systems design has earned lot of interest in the past years. The first valuable work that really tackles the problem of power management for energy harvesting systems has been studied in [11]. Kansal et al.[11] build a model that captures the energy supply of a solar energy source by tracing its instantaneous power profile. Power scaling algorithms for tuning system duty cycles are related to the power consumption that in turn affects the system performance. The system switches between busy mode and idle mode depending on harvested energy from the source. Authors formulate the problem as a linear program and is solved periodically. Within each period, it is necessary to adapt the duty cycle when the observed energy values are different from the predicted ones. The main disadvantage lies in that tasks have no real-time pattern. Scheduling real-time tasks under the strong variation of energy sources remains a key challenge even today[12]. In what follows, we review the main scheduling techniques of real-time tasks that are executed in a timely manner in an energy harvesting system.

Moser et al.[13] considered the case of scheduling tasks with deadlines on a monoprocessor system that is powered by a rechargeable energy storage unit. The authors work with the following assumptions: 1) Tasks may be periodic or aperiodic and 2) energy loss is insignificant. They propose an optimal real-time scheduling algorithm, named "lazy scheduling algorithm (LSA)". LSA is a variation of the well-known earliest deadline first (EDF) scheduler[14], but it is an idling energy-clairvoyant scheduler. LSA works as follows: The task is executed only if it has the earliest deadline among all ready tasks and the system is able to keep on running at full processor speed and without violating its deadline. Liu and Layland[14] assume that the energy consumption for every task in the computing system is directly connected to its execution time through the constant recharging power of the processor. Disadvantages of this algorithm are the following: First DVFS is not considered, this means that tasks are executed at full processor speed and consequently, some future tasks may violate their deadlines because of energy shortage. Secondly, authors assume the total energy consumed by a task is necessarily proportional to its execution time, which is not the real case and finally slack time is not used for energy savings.

In practice, the total energy which can be consumed by a task must not depend on the worst case execution time[15]. This is due to the fact that the worst case instantaneous power consumed by every task depends on several factors like the circuitry and the instrumentation used by the task during and/or after its execution. In reality, the highest power consumption of a running task mainly comes from its actuating operation or from data transmission ordered by the task. Under this assumption, we presented a scheduling algorithm, ED-H[16], that accounts for the limits of both time and energy. ED-H relies on two basic concepts: slack time and slack energy. In ED-H, tasks run according to the earliest deadline first rule. However, before we authorize a task to execute, we have to ensure that the energy storage capacity has sufficient energy to complete the execution of all future occurring tasks. When this condition is not verified, the processor has to stay idle so that the storage unit recharges as much as possible and as long as all the deadlines can still be met despite execution postponement. The power management strategy, ED-H, was proved in [16] to be optimal. However, in order to build an optimal schedule, ED-H needs to know the characteristics of the future tasks and the energy source profile.

This work was later extended in [17] where tasks are obliged to be scheduled according to an on-line algorithm that ignores the arrival time of tasks and their future energy production. We can know the energy consumption of tasks only when they are released on the processor. For this manner, we presented the framework of an on-line monoprocessor scheduling algorithm, namely energy harvesting-earliest deadline first (EH-EDF). The main advantage behind this scheduler is that it is completely on-line and not clairvoyant such as EDF and hence it can be easily implemented in any real-time operating system.

To achieve better system performance and energy efficiency, several researchers focus on extending the classical priority driven schedulers to variable-voltage processors to save power by slowing down the processor just enough to meet the deadlines. Allavena and Mosse[18] describe an off-line scheduler that uses voltage and frequency selection (DVFS) for a frame based system. While they permit the reduction of power consumption by slowing down task execution under deadline constraints, their approach relies on the unrealistic assumption that both the instantaneous consumption power and production power are constant. Later in [19], we set the full hypothesis to reduce central processing unit (CPU) energy consumption by proposing an optimal energy efficient scheduling algorithm for aperiodic real-time jobs. Specifically, we apply the concept of real-time process scheduling to a dynamic voltage and frequency scaling (DVFS) technique.

Later on, Liu et al.[20] propose an energy aware dynamic voltage and frequency scaling algorithm, called EA-DVFS, for periodic tasks. EA-DVFS can efficiently use the slack to reduce the deadline miss rate. Before executing a task, the processor must decide whether to run with full power or reduced power based on the energy available in the energy reservoir. When we have sufficient energy to complete the task execution, the processor will operate at its full speed. Otherwise, the task is stretched and executed at a lower speed. In case of low workload, EA-DVFS algorithm reduces deadline miss rate by 50% compared to LSA and decreases the minimum storage size by 25% when the deadline miss rate is zero. 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, this work has some shortcomings:

1) Authors define the term "sufficient available energy" on a single current task. The system considers that there is sufficient energy if the remaining operation time of system at the full speed is more than the relative deadline of the task. Let′s suppose that there is only 1% left in the energy reservoir, then the system will operate at full speed and consequently the deadline of the task will be violated. That is not the desired behavior.

2) For the aim of energy savings, task slacks are not fully exploited. This is due to the fact that when authorizing tasks to be scheduled, EA-DVFS considers only one task instead of considering all tasks in the ready task queue and hence not all task slacks are exploited.

To overcome the above inconvenients, Liu et al.[21] propose a harvesting-aware DVFS (HA-DVFS) algorithm to improve the system performance by fully exploiting the task slack under timing and energy constraints. HA-DVFS combines the adaptive scheduling techniques with dynamic voltage and frequency selection to reduce the deadline miss rate when compared to LSA and EA-DVFS. With the aim of achieving full system energy autonomy, Lin et al.[22] propose a global control real-time embedded system with an energy harvesting capability (RTES-EH) scheduler. The global controller aims to adopt a photovoltaic panel as the energy harvesting source, a supercapacitor as the energy reservoir, and a real-time sensor node as the embedded device that performs an energy-harvesting aware real-time task scheduling with dynamic voltage and frequency scaling.

Srbinovski et al.[23] present an algorithm to adapt the sampling frequency according to the available energy. Tan and Yin[24] propose an algorithm based on dynamic voltage and frequency scaling technique that dynamically concentrates all dispersed free time together to harvest energy by dynamically scheduling harvesting tasks and service tasks. Xu et al.[25] target the problem of energy-efficiency in real-time systems with DVFS under the constraint of reliability. For this reason, they present a global dynamic scheduling algorithm to maximize the energy efficiency while ensuring the reliability.

3 System model and terminology

3.1 System model

The real-time energy harvesting system (Fig. 1) considered in this work consists of three major units: energy harvesting unit (EHU), energy storage unit (ESU) and energy dissipation unit (EDU). The energy harvesting unit harvests the energy from external sources like sun, wind, etc.

Apart from the applications running in the energy dissipation unit, there is additional software running in the uniprocessor system, namely the scheduler. Earliest deadline first (EDF) is the first dynamic priority scheduler used in our algorithm[26]. The other scheduler we used is the DVFS which slows down task execution under deadline constraints depending on the energy harvested and energy in the storage unit.

3.1.1 Energy harvesting unit (EHU)

We assume that the ambient energy is captured and converted into electrical power. The energy source is considered to be unpredictable but we still can predict the expected availability in a short-term perspective with a worst case charging rate (WCCR) on the harvested source power output, namely $P_s(t)$. Clearly, we do not make any consideration about the nature and dynamics of the energy source so as to make our model more easily implemented in any real application where the energy source properties may not be available beforehand.

The energy harvested in an interval of time $[t_1, t_2]$ is denoted by $E_s(t_1, t_2)$ and can be calculated using the following formula:

$$E_s(t_1, t_2) = \int_{t_1}^{t_2} P_s(t)\mathrm{d}t. \qquad (1)$$

3.1.2 Energy storage unit (ESU)

We use in our work an ideal energy storage unit (supercapacitor or battery) that can be recharged up to a nominal capacity C. Since we use an ideal energy storage unit, we assume that the amount of energy wasted in the charging and discharging process is neglected. The energy level has to remain between two boundaries $C_{\min}$ and $C_{\max}$ with $C = C_{\max} - C_{\min}$. The lower limit of the energy storage unit ($C_{\min}$) is not zero since there must always be a reserved energy in the energy storage unit for worst case scenarios.

3.1.3 Energy dissipation unit (EDU)

We consider a real-time system equipped with a DVFS-enabled processor. The variable speed processor is assumed to be working with N discrete frequencies ranging from $f_{\min} = f_1 \leq f_2 \leq \cdots \leq f_n = f_{\max}$. The power consumption of the tasks running in the processor depends on the processor's frequency. Thus, the power consumption and voltage level correspondent to clock frequency f_n are denoted as P_n and V_n, respectively. We suppose that P_n is the overall power consumption of the EDU that contains both dynamic power consumption and

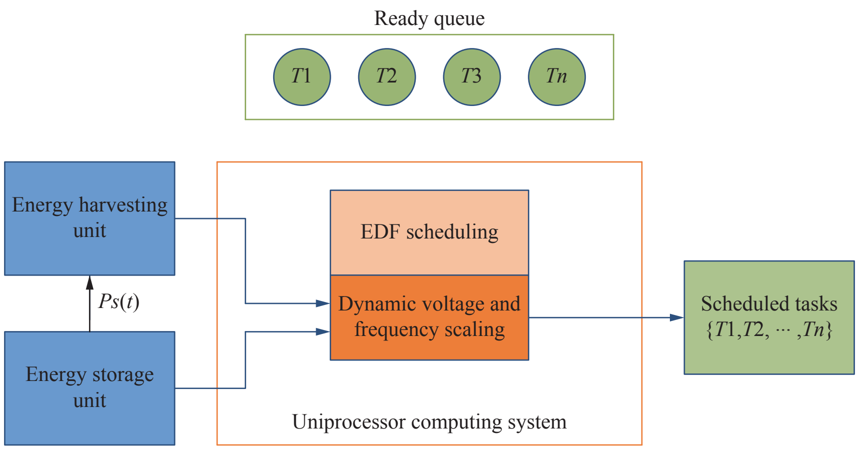

Ready queue
T1
T2
T3
Tn
Energy harvesting unit
Ps(t)
Energy storage unit
EDF scheduling
Dynamic voltage and frequency scaling
Uniprocessor computing system
Scheduled tasks {T1,T2, ··· ,Tn}

Fig. 1 A real-time energy harvesting system model

leakage power consumption.

We consider a slow down factor S_n as the normalized frequency of f_n with respect to the maximum frequency $f_{\max}$. S_n ranges from $S_{\min}$ to 1:

$$S_n = \frac{f_n}{f_{\max}}. \tag{2}$$

We consider in our work that each task has different power dissipation that varies relative to its corresponding frequencies. Consequently, a task executes with maximum power dissipation when the frequency is maximum and this power consumption decreases as the frequency decreases.

Consequently, the power dissipation of a task must be defined as function of the task index and its corresponding slow down factor $P_i(\tau_i, S_i)$.

We consider here a set of independent and 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 task τ_i, where C_i, D_i, T_i and E_i indicate the worst case execution time (WCET), the relative deadline, the period and the worst case energy consumption (WCEC), respectively. Release time r_i of task τ_i is equal to kT_i, $k = 0, 1, 2, \cdots$. When we stretch a task τ_i by a slow down factor S_i, then its actual execution time $(C_i(a))$ at frequency f_i will be $\dfrac{C_i}{S_i}$. When the processor is running at its maximum frequency, then $C_i(a) = C_i$. We assume that $0 \leq C_i \leq D_i \leq T_i$ for each $1 \leq i \leq n$.

Tasks are scheduled on a monoprocessor system. Task set Γ is said to be feasible if all tasks meet the deadlines.

The energy dissipation (E_i) of a task τ_i is computed as

$$E_i = P_i(\tau_i, S_i) \times \left(\frac{C_i}{S_i}\right). \tag{3}$$

4 Background material

4.1 EDF scheduling

The problem of scheduling periodic tasks on one processor without energy constraints has attracted considerable research efforts in the past thirty years[27]. The most popular approach is the dynamic priority algorithms, including the earliest deadline first (EDF) algorithm. EDF places tasks in a priority queue. Whenever a task finishes or a new task is released, the task closest to its deadline will be executed. This means that, at each time instant t, EDF schedule the ready task whose deadline is closest to t. EDF is proved to be an optimal scheduling algorithm. EDF fully exploits the processor, reaching a utilization bound up to 100%.

Generally, EDF implementation offers tasks according to their priority either earliest deadline as soon as possible (EDS)[28] or earliest deadline as late as possible (EDL)[28, 29]. EDF is a work-conserving (also called non-idling) scheduling algorithm where at each instant, we choose for execution the ready job with the closest absolute deadline.

EDF is proved in [30] to be a class one scheduler for energy harvesting applications because of its simplicity in implementation and its optimality for non-idling settings.

4.2 Classical concepts for real-time scheduling

In this subsection, we recall some definitions related to real-time scheduling. Let us consider t_c as the current time where we have to schedule a task set Γ by a certain scheduling algorithm ω.

At a current time t_c, we define the slack time of a task set Γ 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. Calculating the slack time at run-time is performed by the so-called dynamic EDL schedule[29].

Definition 1. The slack time of a task τ_i at current time t_c is

$$ST_{\tau_i}(t_c) = d_i - t_c - h(t_c, d_i) - AT_i \tag{4}$$

where AT_i is the total remaining execution time of uncompleted tasks currently ready at t_c and in the time interval $[t_c, d_i)$. $h(t_c, d_i)$ is the processor demand of a task set Γ on 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.

Definition 2. The slack time of a task set Γ at current time t_c is

$$ST_{\Gamma}(t_c) = \min_{d_i > t_c} ST_{\tau_i}(t_c). \tag{5}$$

Equation (5) represents the maximum continuous processor time that could be available from time t_c while still guaranteeing the deadlines of all the tasks.

4.3 ED-H scheduling algorithm

Despite its optimality for non-idling settings and robustness properties, EDF behaves poorly because it consumes the energy greedily. When considering energy as a limiting factor, simply executing tasks according to the EDF rule may lead to some possible deadline misses. Hence, to avoid deadline violation in energy scavenging systems, we presented an on-line scheduler called earliest deadline-harvesting scheduling algorithm (ED-H). ED-H is an EDF-based real-time scheduler for monoprocessor energy-harvesting systems with considerations of both energy and time constraints. We take the hypothesis to be that the energy consumption of any task can be performed with any power.

This means that before authorizing a task to execute, the scheduler must ensure that the energy storage is sufficient to execute this task during at least one time unit. When this condition is not satisfied, the processor has to postpone the execution of the task as late as possible so that the energy reservoir recharges as much as possible and as long as all the deadlines can still be met.

The idea behind the ED-H is to order tasks according to the earliest deadline first (EDF) rule. This rule is natural since tasks have hard deadlines. Executing them in accordance with their relative urgency appears to be the best approach even if they are not systematically executed as soon as possible.

The major difference between ED-H and EDF is on the operation of the processor. This means that ED-H has to decide when to let the processor busy in executing the ready tasks and when the processor has to be idle. Before executing a task, the energy availability of the system is checked to ensure that energy in the resevoir is sufficient to verify the scheduling of all future occurring tasks, by considering both their timing and energy requirements and the replenishment rate of the energy reservoir[17]. Clearly, this means that there is sufficient slack time.

To formally present ED-H, we need to illustrate some novel concepts particularly helpful when studying the feasibility of a task set when jointly consider both energy and deadline requirements: the energy demand and the slack energy.

Let r_k, d_k and E_k be release time, deadline and worst case energy consumption of a task τ_k, respectively.

Hereafter, for short, we will actually refer to the dynamic slack energy (respectively the dynamic slack time) as the slack energy (respectively the slack time) at current time t_c when producing a schedule for a task set Γ by a certain scheduling algorithm.

We define the slack energy of the system at current time t_c as the maximum amount of energy that can be consumed from t_c continuously while still satisfying all the timing constraints of the tasks[17].

Definition 3. The slack energy of a task τ_i at current time t_c is

$$SE_{\tau_i}(t_c) = E(t_c) + E_s(t_c, d_i) - g(t_c, d_i) \tag{6}$$

where $E_s(t_c, d_i)$ is the amount of energy that is produced by the renewable energy source between t_1 and t_2.

Since the main principle of ED-H is to execute a task as long as no future starvation could occur, this leads us to define a new terminology, named preemption slack energy (PSE). The PSE of a task set Γ at current time t_c ($PSE_\Gamma(t_c)$) is the maximum amount of energy that could be consumed by the currently active task while still guaranteeing energy feasibility for tasks that may preempt it[30].

Definition 4. The preemption slack energy of a task set Γ at current time t_c is

$$PSE_\Gamma(t_c) = \min_{t_c<r_i<d_i<d} SE_{\tau_i}(t_c) \tag{7}$$

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

Let $Q_r(t)$ be the queue of uncompleted tasks which are ready for execution at time t. $SE_\Gamma(t)$ and $ST_\Gamma(t)$ are respectively the slack energy and the slack time of the task set Γ at time t. The ED-H scheduling algorithm follows the below rules:

Rule 1. The EDF priority order is used to select the future running task in $Q_r(t)$.

Rule 2. The processor is imperatively idle in $[t, t+1)$ if $Q_r(t) = \phi$.

Rule 3. The processor is imperatively idle in $[t, t+1)$ if $Q_r(t) \neq \phi$ and either $E(t) = 0$ or $SE_\Gamma(t) = 0$.

Rule 4. The processor is imperatively busy in $[t, t+1)$ if $Q_r(t) \neq \phi$ and either $E(t) = C$ or $ST_\Gamma(t) = 0$.

Rule 5. The processor can equally be idle or busy if $Q_r(t) \neq \phi$, $0 < E(t) < C$, $ST_\Gamma(t) > 0$ and $SE_\Gamma(t) > 0$.

The ED-H scheduler achieves full energy autonomy for monoprocessor scheduling while considering both time and energy harvesting constraints.

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

Proof. See [16]. □

4.4 Motivational example

Consider a task set Γ that is composed of three periodic tasks, $\Gamma = \{\tau_i | 1 \leq i \leq 3\}$ and $\tau_i = (C_i, D_i, T_i, E_i)$. Let $\tau_1 = (1, 3, 5, 30)$, $\tau_2 = (2, 7, 10, 80)$ and $\tau_3 = (3, 12, 20, 180)$. We assume that the energy reservoir has capacity C equal to 200 energy units at $t = 0$. For ease of simplicity, we consider that the rechargeable power P_s is constant along the hyperperiod and equal to 10.

Before beginning the schedule, we have to verify the feasibility conditions. The processor utilization $U_p = \sum_{i=0}^{n} \dfrac{C_i}{T_i} = 0.55 < 1$. Consequently, the necessary feasibility condition related to timing constraints, $U_p \leq 1$, is satisfied. On the other side, the necessary feasibility condition related to energy constraints, $U_e \leq P_s$, is not respected because $U_e = \sum_{i=0}^{n} \dfrac{E_i}{T_i} = 23 > 10$.

Let us schedule Γ according to ED-H within the first hyperperiod ([0, 20]). We show that Γ is not schedulable because of energy starvation at time $t = 12$ (Fig. 2). The system stops immediately and the deadline miss rate amounts to 42%. In detail:

1) At time $t = 0$, all task instances are ready and $C(0) = 200$. τ_1, as the highest priority task, runs and finishes at $t = 1$. $C(1) = C(0) - E1 + (P_S \times C_1) = 180$ energy units. Since there is no task instance in the ready queue released after $t = 0$ with deadline less than or equal to the current time ($t = 1$), the slack energy does not require to be computed at $t = 0$. We just have to verify that the energy level in the reservoir permits us to satisfy the energy requirement of task τ_1.

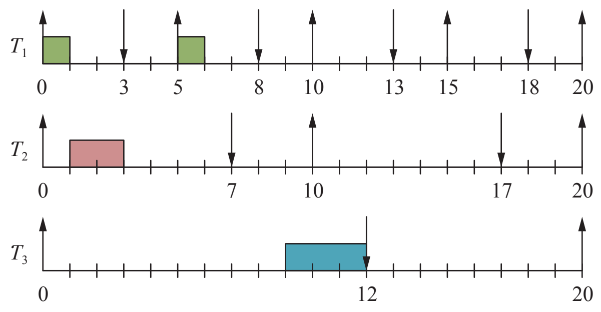


Fig. 2 Task scheduling according to ED-H

2) At time $t = 1$, τ_1 has the highest priority task, ready to be processed, runs and finishes at $t = 3$. $C(3) = 120$ energy units.

3) At $t = 3$, τ_3 is now ready to be processed and $C(3) = 120$. But there is no sufficient energy in the storage unit for execution. So, we have to insert an idle time to let the processor inactive as long as the energy storage unit is not fully replenished ($C = C_{\max}$) and the latest start time of the next periodic task has not been attained. The slack time at time $t = 3$ is equal to 2. Hence, the energy storage capacity is recharged until $t = 5$ where $C(5) = 140$.

4) At $t = 5$, τ_1 is released, runs and finishes at $t = 6$. $C(6) = 120$ energy units.

5) Now, τ_3 is the highest priority task ready to be processed, but there is no sufficient energy for execution. Another time, we have to insert an idle time. The slack time at time $t = 6$ is equal to 3. Hence, the energy storage capacity is recharged until $t = 9$ where $C(5) = 150$.

6) At $t = 9$, τ_3 has the highest priority task, ready to be processed, runs and finishes at $t = 12$ where the energy storage unit is now empty.

7) At $t = 12$, τ_1 has the highest priority task, but the energy reservoir is now empty and the processor cannot be idle. Consequently, we stop scheduling at time $t = 12$.

5 Energy guarantee-dynamic voltage and frequency scaling (EG-DVFS) algorithm

In this work, we propose an energy guarantee dynamic voltage and frequency scaling algorithm that scales the processor speed for executing a task based on the energy found in the storage unit as well as the available harvesting energy and deadline of a task which in fact scales down to conserve energy for future running tasks. The EG-DVFS scheduling scheme jointly accounts for the requirements arising from both the energy and time domain.

5.1 Presentation of the algorithm

EG-DVFS algorithm is designed to order tasks according to the EDF scheduling policy. The difference between EG-DVFS and classical EDF (or EDS) is to decide when to execute tasks at full processor speed and when to decrease the speed processor while meeting all deadlines.

Initially, we try to execute all task instances according to the EDF scheduler where the system operates at full processor speed. Let us consider that there are M task instances in the ready queue. The start time of the task is derived under the assumption that the task executes at the constant processor speed until its completion. St_i and Ft_i are respectively considered as the start time and finish time of task τ_i.

We consider that the start time of the first task instance τ_1 in the ready queue is equal to its release time.

$$St_1 = r_1. \tag{8}$$

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

$$St_i = \max(r_i, Ft_{i-1}) \tag{9}$$

where $2 \leq i \leq M - 1$.

Before authorizing a task to execute, we must ensure that there is sufficient energy to completely execute this task during the next time unit, which represents the worst case situation. Thus, we have to compute the remaining energy in the energy storage unit at the end of the task execution using the following equation:

$$C(Ft_i) = C(St_i) + E_s(St_i, Ft_i) - E_i(St_i, Ft_i) \tag{10}$$

when the energy in the storage capacity is sufficient to execute a task, then this task will be executed at the scheduled start time and with full processor speed ($S_i = 1$).

When we cannot verify this condition, the processor has to execute the task at the head of the queue at the derived speed that is computed by scaling down the operating processor speed as much as possible and as long as the system will be able to meet all the deadlines. This means that we have to decide the slow down factor for all task instances based on the processor utilization and energy state. 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

$$C_i(a) = C_i + ST(St_i). \tag{11}$$

The finishing time of task instance τ_i can be calculated as

$$Ft_i = St(i) + C_i(a). \tag{12}$$

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

$$St_{i+1} = \max(r_{i+1}, Ft_i). \tag{13}$$

The slow down factor is thus computed by the following equation:

$$S_i = \frac{C_i}{C_i(a)}. \tag{14}$$

Since, every task τ_i must complete the execution before its absolute deadline, then the following equation must hold:

$$r_i + d_i - c_i(a) \geq \max\{r_i, t\} \tag{15}$$

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 another task. When preemption occurs, the start time of the newly arrived task is equal to r_i. That is

$$St_i = \begin{cases} \max(r_i, Ft_{i-1}), & \text{if } r_i + d_i - C_i(a) \geq \max(r_i, t) \\ r_i, & \text{otherwise.} \end{cases} \tag{16}$$

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.

The EG-DVFS algorithm works as follows: First, EG-DVFS checks if there are ready instances in the queue to be executed. If not, the processor is made idle until the next task release. Otherwise, EG-DVFS selects the highest priority instance ready for execution.

Before executing the ready task instance with the highest priority, EG-DVFS tests if there is sufficient energy in the energy storage unit. Moreover, EG-DVFS calculates the slack energy of all instances with a higher priority but not ready. The slack energy of the system is the minimum of all these slack energies.

If the system slack energy is positive and there is sufficient energy for execution, then the task instance will be executed with the highest speed. Otherwise, EG-DVFS scales down the frequency of the processor. The question is: by how much? EG-DVFS answers this question by computing the slack time of the system. Now, the scheduler is able to scale down the processor speed to the lowest possible level while EG-DVFS jointly considers slack time and energy state is more conservative. Consequently, we stretch the execution time of the ready task instance to its actual execution time without violating deadlines. Upon stretching the execution time of a task instance, the recharging energy increases (execution time increases) and the energy dissipation decreases.

EG-DVFS can now compute the slow down factor of the corresponding task instance and the energy dissipation can then be chosen.

5.2 Frequency tuning in case of energy overflow

When the energy reservoir reaches its full charge or overflow occurs, the incoming ambient energy overflows the storage and the extra energy is wasted. This overflow completely counteracts our previous effort to save energy by slowing down task execution. Hence, the execution speed of a task must be fine-tuned to eliminate energy overflow while still meeting energy and time constraints. In this work, and in case of insufficient energy, tasks are executed at the minimum possible operating frequency which respects the deadline of the task and it will be increased to the next higher level to avoid energy overflow.

In case of insufficient energy, EG-DVFS stretches the task execution time to its maximum level which is just enough to meet the deadline of the task. It then checks if an energy overflow occurs. In other words, if $C(Ft_i) = C(St_i) + E_s(St_i, Ft_i) - E_i(St_i, Ft_i)$ holds, the overflow occurs. In this case, EG-DVFS increases the speed level to next higher level, updates $C(t)$ and rechecks whether the energy still overflows. This process is then repeated until $C(t)$ is less than or equal to C.

When fine-tuning the operated frequency, overflow will be eliminated and more slack time will be saved for remaining tasks.

5.3 EG-DVFS algorithm

The major components of EG-DVFS algorithm are the following: $C(t)$, $SE(t)$ and $ST(t)$. We assume that t is the current time, $C(t)$ is the amount of energy that is stored in the energy reservoir at time t. In addition, we consider that $SE(t)$ and $ST(t)$ are respectively the slack energy of the system and the slack time of the system at time t.

The function execute() enables the processor to run the ready job with the earliest deadline at its corresponding frequency.

We describe in Algorithm 1 the pseudo code of the EG-DVFS scheduler:

Algorithm 1. Energy guarantee dynamic voltage and frequency selection (EG-DVFS) algorithm.

Require: A Set of M EDF-based periodic tasks $\Gamma = \{\tau_i | \tau_i = (r_i, C_i, D_i, T_i, E_i) \ \ i = 1, \cdots, M\}$, current time t, energy reservoir with capacity bounded between $C_{\max}$ and $C_{\min}$, energy level of the battery $C(t)$, source power $P_s(t)$.

Ensure: A processor working with N discrete frequencies ranging from f_1 ($f_{\min}$) to f_N ($f_{\max}$).

```
1) Sort task instances according to the EDF rule
2) Determine the start time of task instances
3) for $i$ = 1:$M$ do
4)   if $i == 1$ then
5)       $St_1 = r_1$
```

```
6)  else
7)      Calculate $St_i$ from equation (16)
8)  end if
9) end for
10) Assume we have to execute task instance $\tau_i$ with full processor speed ($S_i = 1$).
11) Calculate the energy that is remained in the energy storage unit at the end of the execution.
12) $C(Ft_i) = C(St_i) + \int_{St_i}^{Ft_i} P_s(t)\mathrm{d}t - E_i(St_i, Ft_i)$
13) if ($C(Ft_i) \geq C_{\min}$ and $SE(St_i) > 0$) then
14)   execute()
15) else
16)   Compute $ST(St_i)$
17)   Actual execution time $C_i(a) = C_i + ST(St_i)$
18)   $Ft_i = C_i + ST(St_i)$
19)   Slow down factor $S_i = \frac{C_i}{C_i(a)}$
20)   Update execution time
21)   Select the relative energy consumption ($E_i$)
22)   Compute $C(Ft_i)$ using equation (10)
23)   if $C(Ft_i) > C$ then
24)     for $j = i + 1$ to $N$ do
25)       Update $C(Ft_i)$ using equation (10)
26)       if $C(Ft_i) \leq C$ then
27)         break
28)       end if
29)     end for
30)   end if
31) $St_{i+1} = max(r_{i+1}, Ft_i)$
32) Slow down factor $S_i = \frac{C_i}{C_i(a)}$
33) Update execution time
34) Remove task $\tau_i$ from ready task list
35) end if
```

The main contributions of EG-DVFS are:

1) Optimization process is based on both energy and timing constraints.

2) It fully explores the tradeoff between slack time as well as slack energy to save energy when the ready queue contains multiple tasks at the same time.

3) It allows DVFS techniques to stretch the processor speed of periodic tasks such that overall energy consumption of the system is reduced and a greater number of task are accepted.

4) Avoid wasting the overflow energy. We only waste energy when there are no ready tasks in the queue and the storage unit is fully replenished.

5.4 Efficiency

The computations of slack time and slack energy are the major keys to the operation of the EG-DVFS algorithm. As proved in [31], the slack time of a periodic task set at a given time instant can be obtained on-line by computing the dynamic EDL scheduling algorithm, with complexity $O(K.n)$, where n is the number of periodic tasks, and K is equal to $\left\lfloor \frac{R}{p} \right\rfloor$, where R and p are, respectively the longest deadline and the shortest period of current ready tasks. Moreover, the complexity for computing the slack energy is $O(K.n)$ too[16]. As EDeg has low and constant space requirements, this makes it easily implementable on many low-power, unsophisticated hardware platforms including micro-controllers. We can conclude that the time complexity for EG-DVFS is $O(K.n)$.

5.5 Illustrative example 1

Consider the above example where a task set $\Gamma = \{\tau_i | 1 \leq i \leq 3\}$ and $\tau_i = (C_i, D_i, T_i)$. Let $\tau_1 = (1, 3, 5)$, $\tau_2 = (2, 7, 10)$ and $\tau_3 = (3, 12, 20)$. We assume that the energy reservoir has capacity C equal to 200 energy units at $t = 0$. For simplicity, we consider that the rechargeable power P_s is constant along the hyperperiod and equal to 10. The processor is assumed to be working with six discrete slow down factors $S_i = \{1, 0.75, 0.5, 0.3, 0.2, 0.1\}$. The power dissipation of tasks τ_i is shown in Table 1.

Table 1 Energy dissipation of tasks τ_i

Energy dissipation	$S = 1$	$S = 0.75$	$S = 0.5$	$S = 0.3$	$S = 0.2$	$S = 0.1$
Task τ_1	30	20	9	5	1	0.5
Task τ_2	80	50	35	10	5	2
Task τ_3	180	120	60	27	15	12

In this example, we have to schedule Γ according to EG-DVFS within the first hyperperiod. We show that Γ is schedulable since all tasks are executed before their deadlines and without depleting the energy reservoir. In details:

1) At time $t = 0$ (Fig. 3), all tasks are ready and stored in the ready queue. τ_1 has the earliest deadline and is executed till $t = 1$ where $C(1) = 180$ energy units.

2) At time $t = 1$, τ_2 is executed until $t = 3$ where $C(1) = 120$ energy units.

3) At $t = 3$, τ_3 is the highest priority task ready to be processed but it cannot run at maximum speed because of insufficient energy in the battery. So, we have to slow down the processor in such a way that the deadline is not violated. The slack time is equal to one. Thus, the actual execution time for τ_3 is equal to four and the slow down factor S_3 is $S_3 = \frac{3}{4} = 0.75$. Consequently, the energy dis-

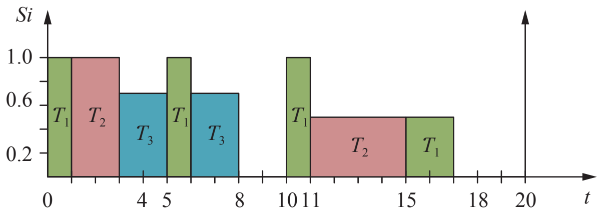


Fig. 3 Task scheduling according to EG-DVFS

sipation for τ_3 is $E_3 = 120$ (see Table 1).

In addition, we have to calculate the slack energy of the system at time $t = 3$.

$$SE(t=3) = C(3) + \int_3^{12} P_s(t)\mathrm{d}t - (E_1 + E_3) = 60 > 0. \quad (17)$$

Now, τ_3 is executed from $t = 3$ to $t = 5$ with a slow down factor $S_3 = 0.75$ where $C(5) = 20$ energy units.

4) At $t = 5$, τ_3 is preempted by τ_1 that runs and finishes at $t = 6$ where the energy storage unit is now empty.

5) At $t = 6$, τ_3 continues its execution where its energy consumption is equal to 120 – 100 = 20 energy units. τ_3 finishes its execution at $t = 8$ where the energy storage unit is again empty.

6) The processor is idle from $t = 8$ to $t = 10$ where $C(10) = 20$ energy units.

7) At $t = 10$, τ_1 is released and has the highest priority task, ready to be processed, runs and finishes at $t = 11$ where $C(11) = 0$ energy units.

8) At $t = 11$, τ_2 is the highest priority task ready to be processed but it cannot run at maximum speed because of insufficient energy in the battery. So, we have to slow down the processor in such a way that the deadline is not violated. The slack time is equal to four. Thus, the actual execution time for τ_2 is equal to six and the slow down factor S_2 is $S_2 = \dfrac{2}{6} = 0.33$. Since, $S_2 = 0.33$ is not found in Table 1, then we have to choose the nearest value greater than 0.33. Consequently, S_2 becomes equal to 0.5 and the actual execution time becomes equal to four.

According to Table 1, when the slow down factor of τ_2 is 0.5, then the energy dissipation is $E_2 = 35$ energy units.

Now, τ_2 can be executed from $t = 11$ to $t = 15$. This is because when we stretch the execution time, the harvested energy increases and the energy dissipation decreases. The energy left is $C(15) = 5$ energy units.

9) At $t = 15$, τ_1 is released, but again there is no sufficient energy. τ_1 is then stretched until $t = 17$ where the slow down factor is $S_1 = 0.5$ and $C(17) = 16$ energy units.

10) Now, the system is idle until the end of the hyperperiod where $C(t = 20) = 46$ energy units.

5.6 Illustrative example 2

Consider a periodic task set Γ that is composed of three tasks, $\Gamma = \{\tau_i | 1 \leq i \leq 3\}$ and $\tau_i = (C_i, D_i, T_i, E_i)$. Let $\tau_1 = (1, 5, 6, 12)$, $\tau_2 = (2, 8, 10, 17)$ and $\tau_3 = (3, 7, 15, 19)$. We assume that the energy storage capacity is $C = 10$ energy units at $t = 0$. For simplicity, we assume that $C_{\min} = 0$, $C_{\max} = C$ and the rechargeable power is constant along the hyperperiod and equal to 5 ($P_s = 5$).

Let us verify the timing and energy feasibility conditions. First, the processor utilization $U_p = \sum_{i=0}^{n} \dfrac{C_i}{T_i} = \dfrac{17}{30} \leq 1$. Consequently, the necessary feasibility condition related to timing constraints, $U_p \leq 1$ is satisfied. Second, the energy utilization $U_e = \sum_{i=0}^{n} \dfrac{E_i}{T_i} = \dfrac{149}{30} \leq 5$ and the necessary feasibility condition related to energy constraints, $U_e \leq P_s$, is satisfied.

5.6.1 Scheduling under ED-H

Let us schedule Γ according to ED-H within the first hyperperiod, from 0 to 30. We verify that Γ is not schedulable because of energy starvation at time $t = 7$ (Fig. 4). The system stops immediately and the deadline miss rate amounts to 80%.

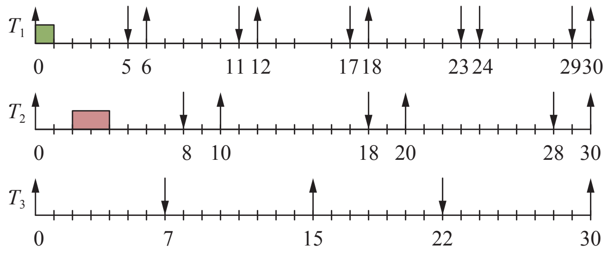


Fig. 4 Task scheduling according to ED-H

In details:

1) At time $t = 0$, all task instances are ready and $C(0) = 10$. τ_1, as the highest priority task, runs and finishes at $t = 1$. $C(1) = C(0) - E_1 + (P_s \times C_1) = 10 - 12 + 5 = 3$ energy units. As there is no task instance released after 0 with deadline less than that of τ_1, the slack energy does not require to be computed at $t = 0$. We just have to verify that the energy level in the storage unit permits to satisfy the energy requirement of task τ_1.

2) At time $t = 1$, τ_2 has the highest priority task and is ready to be processed. But there is not sufficient energy to complete the execution. Consequently, the processor is let idle until $t = 2$ where $E(2) = 8$ energy units.

3) At time $t = 2$, τ_2 as the highest priority task, ready to be processed, runs and finishes at $t = 4$. $E(7) = 11$ energy units.

4) At time $t = 4$, τ_3 has the highest priority task and ready to be processed. However, the energy storage capacity is not sufficient to complete the execution and there is no slack time. Hence, the system stops immediately without completing the execution.

5.6.2 Scheduling under EH-DVFS

Now, Γ is scheduled according to EH-DVFS within the first hyperperiod. The processor is assumed to be working with six discrete slow down factors $S_i = \{1, 0.75, 0.5, 0.3, 0.2, 0\}$. The power dissipation of tasks τ_i is shown in Table 2.

We verify that Γ is schedulable and the energy storage capacity is again full at the end of the hyperperiod. (Fig. 5).

In detail:

Table 2 Energy dissipation of tasks τ_i

Energy dissipation	$S=1$	$S=0.75$	$S=0.5$	$S=0.3$	$S=0.2$	$S=0.1$
Task τ_1	12	9	5	3	2	1
Task τ_2	17	13	8	5	3	2
Task τ_3	19	14	10	7	5	3

1) At time $t=0$, all task instances are ready and $C(0)=10$. τ_1, as the highest priority task, runs and finishes at $t=1$. $C(1)=C(0)-E_1+(P_s\times C_1)=10-12+5=3$ energy units. As there are no task instance released after 0 with deadline less than that of τ_1, the slack energy does not require to be computed at $t=0$. We have just to verify that the energy level in the storage unit is able to satisfy the energy requirement of task τ_1.

2) At time $t=1$, τ_2 has the highest priority task and is ready to be processed. But there is not sufficient energy to complete the execution. So, we have to slow down the processor in such a way that the deadline is not violated. The slack time is equal to one. Thus, the actual execution time for τ_2 is equal to three and the slow down factor S_2 is $S_2=\frac{2}{3}=0.66$. Consequently, the energy dissipation for τ_2 is $E_3=13$ (see Table 2).

3) Now, τ_2 is executed from $t=1$ to $t=4$ with a slow down factor $S_2=0.66$ where $C(4)=5$ energy units.

4) At time $t=4$, τ_3 is the highest priority task and is ready to be processed, runs and finishes at $t=7$. $E(7)=1$ energy units.

5) The second task instance of τ_1 is ready to be executed but again we have to slow down the processor. The slack time is equal to three but we need only two to fully replenish the energy storage unit. Thus, the actual execution time for τ_1 is equal to three and the slow down factor S_1 is $S_1=\frac{1}{3}=0.33$. Consequently, the energy dissipation for τ_1 is $E_1=5$. Consequently, τ_1 is executed from $t=7$ to $t=10$ with a slow down factor $S_1=0.33$ where the energy storage unit is full again.

6) At time $t=10$, τ_2 has the highest priority task and is ready to be processed. The energy storage capacity is sufficient to execute τ_2 with full processor speed.

In addition, we have to compute the slack energy of the system at time $t=10$.

$$SE(t=10)=C(10)+\int_{10}^{18}P_s(t)\mathrm{d}t-(E_1+E_2)=21>0. \tag{18}$$

Consequently, τ_2 is executed from $t=10$ to $t=12$ where $C(12)=3$ energy units.

7) This procedure continues until $t=30$ where the energy storage unit is full.

6 Experimental results and discussions

6.1 Simulation details

Extensive simulation experiments have been performed to evaluate the performance of the EG-DVFS algorithm in terms of energy saving and performance improvement. We developed a simulator in C/C^{++}. In the simulator, we implement EG-DVFS with respect to ED-H. We use the task generator of periodic tasks as the one described by Martineau[32]. Experiments are built by selecting as input several parameters: the number of synthesized tasks n, the hyperperiod of task periods, processor utilization U_p, energy utilization U_e and the recharging power $P_s(t)$. For each experiment set, the algorithm was simulated 100 times, generating a task configuration of the scheduled task set. The worst case execution times of tasks are randomly generated such that the processor utilization $\sum_{i=1}^{n}\frac{C_i}{T_i}=U_p\leq 1$. The energy consumptions of tasks are also randomly generated based on the energy utilization factor such that $\sum_{i=1}^{n}\frac{E_i}{T_i}=U_e\leq\overline{P_s}$ where $\overline{P_s}$ is the average recharging power.

The simulator generates 30 tasks with least common multiple of the periods equal to 3 360. The worst-case computation times are set according to the processor utilization U_p. We also consider that $C_i\leq D_i\leq T_i$ for every task τ_i.

The rechargeable power $P_s(t)$ may be constant or may vary according to time. For this reason, a random number generator is found at the input of the simulator to produce for every quantum of time within the hyperperiod, a power energy profile with minimum value 5 and a maximum value 35.

To estimate the energy consumption of tasks, we use an Intel XScale processor that supports five frequency levels[33]. Values of the discrete frequencies, supply voltage and consumed power of the processor are listed in Table 3.

The effect of processor utilization on the task set feasibility ratio and the average energy consumption can be seen from the Figs. 6–8.

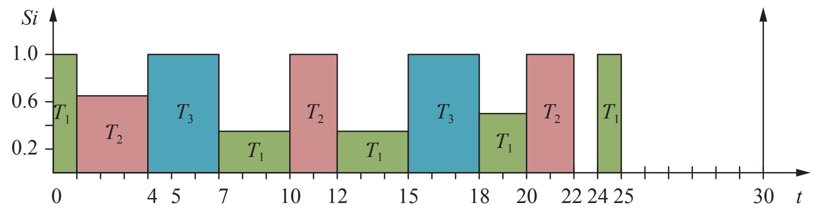
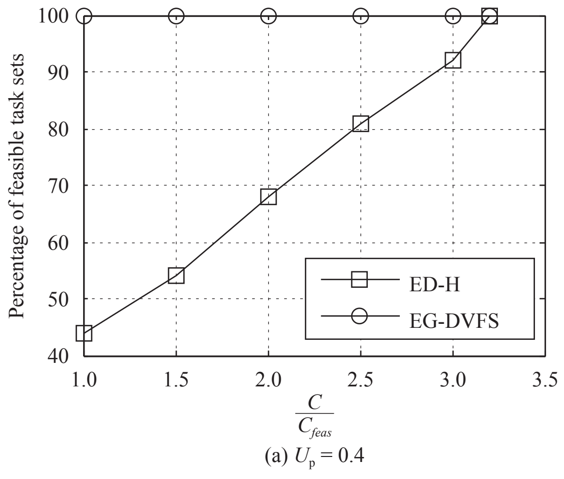


Fig. 5 Task scheduling according to EG-DVFS

Table 3 XScale frequencies, supply voltages and power

Frequency (MHz)	150	400	600	800	1 000
Power (mW)	80	170	400	900	1 600
Voltage (V)	0.75	1.0	1.3	1.6	1.8



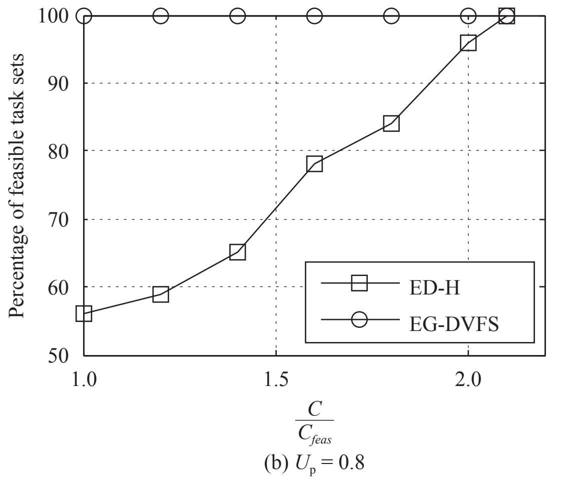


Fig. 6 Percentage of feasible task sets by varying battery capacity

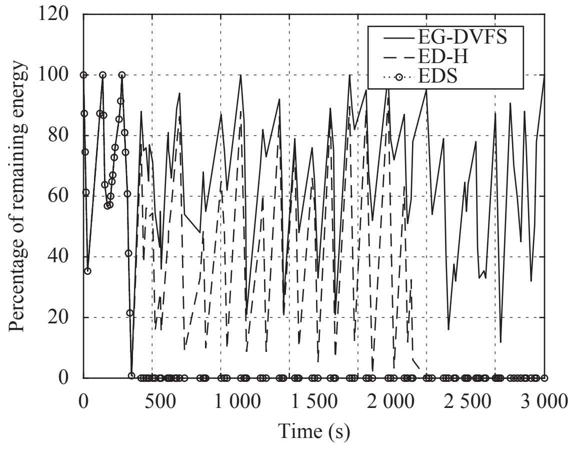


Fig. 7 Variation of the remaining energy level with low utilization

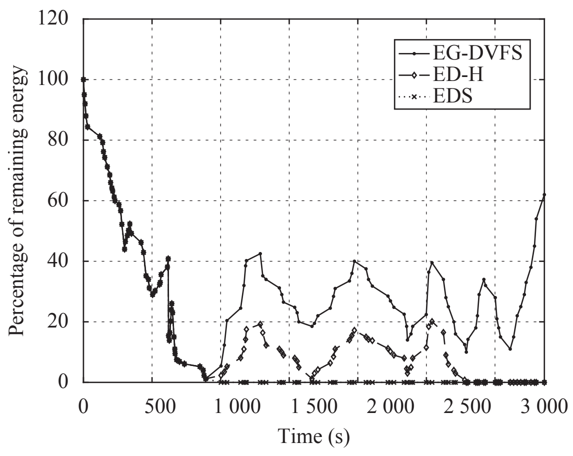


Fig. 8 Variation of the remaining energy level with high utilization

We assume that the energy storage is fully charged at the beginning of the simulation. After a deadline violation is detected, the simulation terminates for ED-H and EG-DVFS.

6.2 Percentage of feasible task sets by varying battery capacity

Our simulation depicts the percentage of feasible task sets by varying the energy reservoir capacity C. Here, we conduct experiments with different processor utilization profiles to test the significant increase in the percentage of task sets with EG-DVFS, compared with the greedy algorithm ED-H.

To evaluate the method that combines ED-H with the DVFS technique, we just set the processor utilization values to 0.4 and 0.8, respectively. We show that EG-DVFS can perform better than the ED-H while considering energy consumption.

For each task set, we find the minimum storage capacity C_{feas} that permits us to schedule all task sets by EG-DVFS without violating deadlines and the energy reservoir is fully replenished at the end of the hyperperiod. After that, we increase the reservoir capacity so that all task sets are feasible with ED-H. Fig. 6 shows the task set feasibility ratio under different desired processor utilizations.

For the case $U_p = 0.4$, we observe that the battery capacity must be more than 3.2 times bigger with ED-H to maintain 100% feasible task sets compared to EG-DVFS. This is due to the fact that EG-DVFS is able to slow down the speed of tasks and save energy so as to obtain a better opportunity to accept a greater number of tasks. However, without DVFS based approach, ED-H algorithm always execute tasks at full processor speed and the energy storage in the reservoir is consumed earlier. Consequently, there is more chance to reject tasks due to energy shortage. In other words, the more stored energy means that more tasks are able to be finished before their deadlines. Hence, the ED-H system incurs a much higher deadline miss rate when compared to EG-DVFS.

Fig. 6(a) shows that the deadline miss rate of ED-H exceeds that of the proposed scheduling algorithm by about 56% and 32% when battery capacity is respectively the same and twice. Hence, the EG-DVFS algorithm is favorable even for small battery capacity.

If we increase the processor utilization to 0.8 and run the simulation again, we observe that the gain in energy savings is decreased but EG-DVFS still beats ED-H by a large margin. EG-DVFS obtains capacity savings of about 45% compared to ED-H. This is because the fundamentals that EG-DVFS outperform are such that the used slack time to slow down task execution such that energy is saved for future tasks. When U_p is high, slack time decreases and most of tasks are just executed at the full speed, as the ED-H algorithm does.

If we consider the case when the processor is always busy, EG-DVFS, ED-H and EDS require exactly the same energy storage. This is because the processor is always active and there is no processor slack time.

6.3 Remaining energy in the battery

In this experiment, we are interested in the remaining energy stored in the system at any current time. The importance of this study comes from the fact that we must always have energy in the storage unit to complete the execution of tasks, otherwise the scheduler will stop.

We aim to illustrate how the energy level in the storage unit changes along time. We only report this information for the three following schedulers: EDS, ED-H, and EG-DVFS. When U_p is set to 0.4, the remaining energy curves in EDS, ED-H, and EG-DVFS schedulers are both plotted in Fig. 7. Under EDS, the remaining energy is decreasing until the storage unit is empty or not sufficient to execute the highest priority instance. It is observed from Fig. 7 that ED-H runs as EDS except that, whenever there is no sufficient energy to execute the highest priority task, then the processor becomes idle. Consequently, whenever a task is required to run, the scheduler compares its energy consumption with the amount of available energy during one unit of time. According to the result of that test, either the task will be authorized to execute or the processor will idle. This mechanism implies that the battery level will decrease systematically when executing a task without necessarily attaining the minimum level, i.e., 0. Then, the storage unit will recharge until being fulfilled as long as the system will be able to meet all the deadlines. Of the three algorithms, EDS and ED-H has the smallest amount of current energy as it operates at the highest processor-supported frequency and does not employ DVFS to save energy.

In details, EDS, ED-H and EG-DVFS are executed as EDS from time $t = 0$ till time $t = 350$. Thus, the current battery energy is the same under the three scheduling policies. EDS and ED-H stop respectively at about 10% and 71% of the total length of the hyperperiod when there is no more energy in the storage unit. As dedicated in Fig. 7, the EG-DVFS scheduler stores significantly more energy than the ED-H scheduler on average. That is because EG-DVFS algorithm slows down the task execution for energy savings. On the contrary, ED-H scheduler always executes the task at the full speed and it consumes more energy to finish an identical task.

When the processor utilization U_p is set to 0.8, we obtain another plot shown in Fig. 8. We observe that as the processor utilization increases, the average energy consumption increases. By observing the variation in the stored energy, we still find that ED-H and EDS have the smallest amount of current energy since it operates at full processor speed and does not employ DVFS to save energy. When the processor utilization is high (say 80%), our proposed approach still has significant energy savings of almost 23% in average energy consumption over existing ED-H . The reason comes from two facts: On one hand, when the processor utilization is high, the processor is almost busy and rarely has chance to slow down the task execution for energy saving. Consequently, most of the tasks will be executed at full speed. On the other hand, the chance for the system to be completely idle is also reduced and thus the system has no chance to harvest energy from the renewable energy source. Hence, the consumed energy can not be supplemented in time.

7 Conclusions

In this work, we presented an energy guarantee scheduling and voltage/frequency selection algorithm targeting real-time systems with energy harvesting capability. The proposed algorithm is an extension of ED-H combined with dynamic voltage and frequency selection. In the case of insufficient energy, we make use of the slack time for energy saving by applying DVFS techniques to stretch the execution time of the ready task with highest priority with lower clock frequency and supply. Here, maximum power savings is achieved since the recharging time increases and the dissipated energy decreases. Compared with the state-of-the-art scheme ED-H, the proposed scheme (EG-DVFS) achieves a comparable percentage of feasible task sets for varying values of the battery capacity. Provided that the average harvested power is sufficient for continuous operation for the EG-DVFS, we are able to determine the minimum battery capacity necessary. Furthermore, achievable capacity savings are demonstrated in a simulative study since the battery capacity must be respectively more than 3.2 and 2.4 times bigger with ED-H to keep the zero deadline miss rate compared to EG-DVFS when the processor utilization U_p is set to 0.4 and 0.8. The results also demonstrate that the proposed scheme is very effective in reducing the battery size when compared to the ED-H scheduler.

The future work will focus on the dynamic task allocation and frequency selection scheme for multiprocessor systems based on the scheduling scheme proposed in this paper.

References

[1] R. Mishra, N. Rastogi, D. K. Zhu, D. Mosse, R. Melhem. Energy aware scheduling for distributed real-time systems. In *Proceedings of International Parallel and Distributed Processing Symposium*, Nice, France, pp. 21–29, 2003. DOI: 10.1109/IPDPS.2003.1213099.

[2] Q. R. Qiu, S. B. Liu, Q. Wu. Task merging for dynamic power management of cyclic applications in real-time multiprocessor systems. In *Proceedings of International Conference on Computer Design*, San Jose, USA, pp. 397–404, 2006. DOI: 10.1109/ICCD.2006.4380847.

[3] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, M. B. Srivastava. Power optimization of variable-voltage core-based systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 1, pp. 1702–1714, 1999. DOI: 10.1109/43.811318.

[4] D. J. Cook, S. K. Das. *Smart Environments: Technologies, Protocols, and Applications*, New York, USA: John Wiley, 2004.

[5] R. Nallusamy, K. Duraiswamy. Solar powered wireless sensor networks for environmental applications with energy efficient routing concepts: A review. *Information Technology Journal*, vol. 10, pp. 1–10, 2011. DOI: 10.3923/itj.2011.1.10.

[6] S. Roundy, D. Steingart, L. Frechette, P. Wright, J. Rabaey. Power sources for wireless sensor networks. In *Proceedings of the 1st European Workshop Wireless Sensor Networks*, Springer, Berlin, Germany, pp. 1–17, 2004. DOI: 10.1007/978-3-540-24606-0_1.

[7] R. Kotz, M. Carlen. Principles and applications of electrochemical capacitors. *Electrochimica Acta*, vol. 45, no. 15–16, pp. 2483–2498, 2000. DOI: 10.1016/S0013-4686(00)00354.

[8] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, M. Srivastava. Design considerations for solar energy harvesting wireless embedded systems. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, Boise, USA, pp. 457–462, 2005. DOI: 10.1109/IPSN.2005.1440973.

[9] X. Jiang, J. Polastre, D. Culler. Perpetual environmentally powered sensor networks. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, Boise, USA, pp. 463–468, 2005. DOI: 10.1109/IPSN.2005.1440974.

[10] A. Kansal, J. Hsu, S. Zahedi, M. B. Srivastava. Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computing Systems*, vol. 6, no. 4, Article number 32, 2007. DOI: 10.1145/1274858.1274870.

[11] A. Kansal, J. Hsu, M. Srivastava, V. Raqhunathan. Harvesting aware power management for sensor networks. In *IEEE Proceedings of the 43rd ACM/IEEE Design Automation Conference*, San Francisco, USA, 2006. DOI: 10.1145/1146909.1147075.

[12] J. Hsu, S. Zahedi, A. Kansal, M. Srivastava, V. Raghunathan. Adaptive duty cycling for energy harvesting systems. In *Proceedings of International Symposium on Low Power Electronics and Design*, Tegernsee, Germany, pp. 180–185, 2006. DOI: 10.1145/1165573.1165616.

[13] C. Moser, D. Brunelli, L. Thiele, L. Benini. Real-time scheduling for energy harvesting sensor nodes. *Real-Time Systems*, vol. 37, no. 3, pp. 233–260, 2007. DOI: 10.1007/s11241-007-9027-0.

[14] C. L. Liu, J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973. DOI: 10.1145/321738.321743.

[15] R. Jayaseelan, T. Mitra, X. F. Li. Estimating the worst-case energy consumption of embedded software. In *Proceedings of the 12th IEEE Real-time and Embedded Technology and Applications Symposium*, San Jose, USA, pp. 81–90, 2006. DOI: 10.1109/RTAS.2006.17.

[16] M. Chetto. Optimal scheduling for real-time jobs in energy harvesting computing systems. *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 2, pp. 122–133, 2014. DOI: 10.1109/TETC.2013.2296537.

[17] H. El Ghor, M. Chetto, R. Hage Chehade. EH-EDF: An on-line scheduler for real-time energy harvesting systems. In *Proceedings of the 18th IEEE International Conference on Electronics, Circuits, and Systems*, Beirut, Lebanon, pp. 776–779, 2011. DOI: 10.1109/ICECS.2011.6122389.

[18] A. Allavena, D. Mosse. Scheduling of frame-based embedded systems with rechargeable batteries. In *Proceedings of Workshop on Power Management for Real-time and Embedded Systems*, Taipei, China, 2001.

[19] H. El Ghor, E. M. Aggoune. Energy efficient scheduler of aperiodic jobs for real-time embedded systems. *International Journal of Automation and Computing*, 2016, published online. DOI: 10.1007/s11633-016-0993-3.

[20] S. B. Liu, Q. Qiu, Q. Wu. Energy aware dynamic voltage and frequency selection for real-time systems with energy harvesting. In *Proceedings of Design, Automation and Test in EUROPE*, Munich, Germany, pp. 236–241, 2008. DOI: 10.1109/DATE.2008.4484692.

[21] S. B. Liu, J. Lu, Q. Wu, Q. R. Qiu. Harvesting-aware power management for real-time systems with renewable energy. *IEEE Transactions on Very Large Scale Integration Systems*, vol. 20, no. 8, pp. 1473–1486, 2012. DOI: 10.1109/TVLSI.2011.2159820.

[22] X. Lin, Y. Z. Wang, S. Y. Yue, N. Chang, M. Pedram. A framework of concurrent task scheduling and dynamic voltage and frequency scaling in real-time embedded systems with energy harvesting. In *Proceedings of International Symposium on Low Power Electronics and Design*, Beijing, China, pp. 70–75, 2013. DOI: 10.1109/ISLPED.2013.6629269.

[23] B. Srbinovski, M. Magno, B. O'Flynn, V. Pakrashi, E. Popovici. Energy aware adaptive sampling algorithm for energy harvesting wireless sensor networks. In *Proceedings of IEEE Sensors Applications Symposium*, Zadar, Croatia, 2015. DOI: 10.1109/SAS.2015.7133582.

[24] Y. H. Tan, X. D. Yin. A dynamic scheduling algorithm for energy harvesting embedded systems. *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, Article number 114, 2016. DOI: 10.1186/s13638-016-0602-8.

[25] H. Z. Xu, R. F. Li, L. N. Zeng, K. Q. Li, C. Pand. Energy-

efficient scheduling with reliability guarantee in embedded real-time systems. *Sustainable Computing*: *Informatics and Systems*, vol. 18, pp. 137–148, 2018. DOI: 10.1016/j.suscom.2018.01.005.

[26] M. L. Dertouzos. Control robotics: The procedural control of physical processes. In *Proceedings of International Federation for Information Processing*, Stockholm, Sweden, pp. 807–813, 1974.

[27] J. W. S. Liu. *Real-time Systems*, New Jersey, USA: Prentice-Hall, 2000.

[28] H. Chetto, M. Chetto. Some results of the earliest deadline scheduling algorithm. *IEEE Transactions on Software Engineering*, vol. 15, no. 10, pp. 1261–1269, 1989. DOI: 10.1109/TSE.1989.559777.

[29] M. Silly. The EDL server for scheduling periodic and soft aperiodic tasks with resource constraints. *Real-time Systems*, vol. 17, no. 1, pp. 87–111, 1999. DOI: 10.1023/A:1008093629946.

[30] M. Chetto, A. Queudet. Clairvoyance and online scheduling in real-time energy harvesting systems. *Real-time Systems*, vol. 50, no. 2, pp. 179–184, 2014. DOI: 10.1007/s11241-013-9193-1.

[31] J. Y. T. Leung, J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation*, vol. 2, no. 4, pp. 237–250, 1982. DOI: 10.1016/0166-5316(82)90024-4.

[32] P. Martineau. Online Scheduling In Real-Time Systems, Ph. D. dissertation, University of Nantes, France, 1994.

[33] Intel Corp. Intel XScale Processor Family Electrical, Mechanical, and Thermal Specification Datasheet, Technical Report, Santa Clara, USA, 2004.

Hussein El Ghor received the B. Eng. degree in engineering from the Lebanese University, Lebanon in 2002, the Ph. D. degree in automatics and applied informatics from University of Nantes, France in 2012. He is currently an assistant professor with Institute of Technology, Lebanese University, Lebanon. He authored many papers in prestigious journals and conferences in the area of real-time systems.

His research interests include scheduling and power management for real-time energy harvesting applications.

E-mail: husseinelghor@ul.edu.lb (Corresponding author)
ORCID iD: 0000-0003-0001-3066

Maryline Chetto received the Ph. D. degree of 3rd cycle doctor in control engineering and the degree of HDR in computer science from the University of Nantes, France in 1984 and 1993, respectively. From 1984 to 1985, she held the position of assistant professor of computer science at the University of Rennes 1, France while her research was with the Institute of Research in Computer Science and Random Systems, France. In 1986, she returned to Nantes and is currently a professor with the Institute of Technology, University of Nantes, France. She is conducting her research at Laboratory of Numerical Sciences of Nantes Institute. She has published more than 100 journal articles and conference papers in the area of real-time operating systems.

Her research interests include scheduling and power management for real-time energy harvesting applications.

E-mail: maryline.chetto@univ-nantes.fr