An Optimal Approach for Minimizing Aperiodic Response Times in Real-Time Energy Harvesting Systems

Conference Paper · October 2018

DOI: 10.1109/AICCSA.2018.8612892

3 authors:

Rola el Osta
University of Nantes
6 PUBLICATIONS **1** CITATION
SEE PROFILE

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

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

An optimal approach for minimizing aperiodic response times in Real-time Energy Harvesting Systems

Rola EL Osta
LS2N Laboratory
University of Nantes
1 Rue de la Noë, Nantes, France
rolaosta@hotmail.com

Maryline Chetto
LS2N Laboratory
University of Nantes
1 Rue de la Noë, Nantes, France
maryline.chetto@univ-nantes.fr

Hussein El Ghor
LENS Laboratory
Lebanese University
Saida, Lebanon
hussain@ul.edu.lb

***Abstract*—This paper explores an energy harvesting-based approach for a jointly execution of hard periodic tasks mixed with occurring soft aperiodic tasks in battery-powered embedded devices. Providing an optimal solution to this scheduling issue is challenging. Accordingly, we propose a new aperiodic task scheduling algorithm in real-time energy harvesting (RTEH) systems that targets at minimizing the response times of aperiodic tasks without compromising the schedulability of periodic tasks. Based on the slack stealing mechanism, the proposed algorithm called Slack Stealing with energy Preserving (SSP) profits whenever possible from available extra processing time and available extra energy so as to service the aperiodic tasks as soon as possible.**

Comparing to other algorithms, SSP achieves better performance in terms of the aperiodic responsiveness under various energy profiles and settings.

***Index Terms*—Real-time scheduling, energy harvesting, soft aperiodic tasks, response time, uniprocessor device.**

I. Introduction

Battery-powered embedded devices are characterized by limited battery lifespan. In most of these devices and in some applications, such as military, health, and environmental, replacing batteries makes them less affordable. Consequently, green solutions based on environmental energy have become economically conceivable for many application fields. Furthermore, energy harvesting for small devices such as wireless sensors offers many technological advantages, such as the continuous replenishment of battery or capacitor, better flexibility and reliability for remote monitoring, complete autonomy and efficiency [1], [2]. As a consequence, energy harvesting technology has been facing a spectacular growth in global interest in the last decade and this development will continue in the years to come.

This paper seeks to settle a crucial scheduling problem in hard real-time systems with energy harvesting considerations often called autonomous real-time systems. In general, such systems consist of hard periodic and soft aperiodic tasks. Periodic tasks are tasks with regular arrival time and deadline and aperiodic tasks may occur and require to be executed as soon as possible while preserving the feasibility of the periodic tasks.

Under fixed priority as well as dynamic priority settings, many research efforts have been made from the eighties in order to propose solutions to solve the underlying scheduling problem [3]. This problem can be simply described as follows: how to jointly execute the hard periodic tasks set mixed with the dynamically occurring soft aperiodic tasks? Many scheduling solutions i.e. aperiodic task servers, have been proposed in the literature, including several ones proved to be optimal. However, all of them have been designed for systems with no energy limitation and they do not take energy into consideration. Providing an optimal solution for the energy harvesting context represents the central objective of this work for dynamic priority-driven systems. The Earliest Deadline for energy Harvesting (ED-H) has been proved optimal in [4] for real-time energy harvesting (RTEH) systems composed of hard deadline tasks. It is dynamic and may achieve maximum processor utilization and maximum energy utilization without restriction on energy production profile.

Accordingly, the ED-H approach represents the bearing of our aperiodic task server in order to minimize the aperiodic response times without compromising the timing and energy constraints on periodic tasks. Before going further, we introduce an illustrative example of a real-time energy harvesting system to show the incentive for the issue addressed in this paper. Consider a real-time forest fire detection system based on wireless sensor networks. The system has to early detect and predict the fires through distributed wireless sensors randomly which have been spread in the forest, in order to minimize the damages and casualties. Suppose now that the system detects a fire in the coverage area, thus transmitting a warning signal to emergency services in order to perform an appropriate action and to mitigate the situation. In this case, the corresponding aperiodic job has to be executed as soon as possible by an adequate scheduler without compromising the whole behavior of the real-time energy harvesting system.

In this paper, we will present a new aperiodic task server called SSP and we will describe the results of an experimental study so as to show the gain of this new server.

The remainder of the paper is constructed as follows.

978-1-5386-9120-5/18/$31.00 ©2018 IEEE

Section II presents the related work. Section III gives a brief explanation about the model and terminology. Section IV presents the background materials: ED-H. Section V describes the proposed Slack Stealing with energy Preserving (SSP) algorithm. Section VI evaluates the new approach. Finally, the paper concludes Section VII.

II. Related work

The problem of scheduling a mixed set of hard periodic tasks and soft aperiodic tasks has been widely considered when there are no energy restrictions. Under that hypothesis, famous aperiodic task servers were proposed [3]. Among them, there are three important families: Background, Slack Stealing and Total Bandwidth [5], [6]. The classical Background servicing is the simplest approach to handle aperiodic tasks in the presence of periodic ones. Any pending aperiodic task is executed only when there are no periodic task ready to be executed. In other terms, the background server can be assimilated to the lowest priority task of the system. The most important problem with this approach is that under high periodic processor utilization, aperiodic response times may be very long and too long for some applications.

The optimal "Slack Stealer" service is used by Chetto and Chetto to yield the famous Earliest Deadline as Late as possible (EDL) server [5], [7], [8]. It consists in stealing all the possible processing time from the periodic tasks to service the aperiodic tasks earlier [9], [10].

The Total Bandwidth Server, proposed by Spuri and Butazzo, supplies good aperiodic response times with maximum simplicity by assigning the total bandwidth of the server to the aperiodic task upon its arrival to the system, and then this task is scheduled by the EDF algorithm, as any other periodic task [6].

ED-H is the optimal uniprocessor scheduling algorithm, in Real-Time Energy Harvesting systems (RTEH), that is responsible for scheduling hard deadline jobs by smartly using both processor time and energy resource so as to meet the deadlines and to not avoid energy starvation.

In the energy harvesting context, we recently proposed two aperiodic servers based on the background principle, respectively named Background with Energy Surplus (BES) and Background with Energy Preserving (BEP) [13]. Assuming that the hard deadline periodic tasks are preemptively scheduled according to the optimal scheduler ED-H, BES server benefits from energy surplus in the storage unit to execute the aperiodic tasks, and BEP server executes the aperiodic tasks as long as no energy starvation is involved for periodic tasks. Although they can provide bounded response time to service aperiodic tasks, the Background approaches have the drawback that the average aperiodic processing time can be long.

As our work aims to provide an optimal solution adapted to the energy harvesting constrained systems, a combination of EDL and ED-H produces our proposal "Slack Stealer with energy Preserving" SSP server for improving aperiodic responsiveness.

III. Model and Terminology

A. System Model

We consider the Energy Harvesting model (mentioned as RTEH) that consists of a computing element, a set of tasks, an energy storage unit, an energy harvesting unit and an energy source [4].

A set of real-time tasks is evaluated and is executed on a monoprocessor system that sustains only one operating frequency and that consumes negligible energy in inactive state. The system contains a set of n independent periodic tasks. We refer to this set by $\tau = \{\tau_i(C_i, D_i, T_i, E_i); i = 1, ..., n\}$, where task τ_i has a worst case execution time of C_i time units, an absolute deadline D_i, a period T_i and a Worst Case Energy Consumption of E_i energy units. We assume that E_i is not necessarily proportional to C_i [11]. The task set τ gives rise to an infinite set of jobs which are scheduled by the optimal monoprocessor scheduler ED-H. The processor utilization of the periodic task set τ is $U_{pp} = \sum_{\tau_i \varepsilon \tau} \frac{C_i}{T_i}$ which is less than or equal to 1. The energy utilization of τ is defined as $U_{ep} = \sum_{\tau_i \varepsilon \tau} \frac{E_i}{T_i}$.

We also consider Ap the stream of m soft aperiodic requests, defined as $Ap = \{Ap_i | 1 \leq i \leq m\}$ and $Ap_i = (a_i, c_i, e_i)$. a_i is the arrival time of a soft aperiodic task, c_i is the worst case execution time and e_i is the worst case energy requirement. The finish time of Ap_i will be denoted by f_i. The parameters of a soft aperiodic task are known when the task arrives.

B. Energy Model

The energy produced by the source $P_p(t)$ is not controllable and not necessarily a constant value. It is given as $E_p(t_1, t_2) = \int_{t_1}^{t_2} P_p(t)dt$ on $[t_1, t_2)$. To avoid short-term energy shortages, we have an accurate estimation of short-term energy within some prediction errors margin. Furthermore, an ideal energy reservoir (e.g. super-capacitor or rechargeable battery) is considered to continue operation even when there is no energy to harvest. The energy reservoir receives power from the harvester and powers the processor. The stored energy at any time t is denoted $E(t)$. The energy reservoir does not leak any energy over time. If it is fully charged at time t and if we continue to charge it, energy is wasted. In contrast, if it is fully discharged at time t (energy depletion), no job can be executed [4].

IV. Earliest Deadline for energy Harvesting systems (ED-H) scheduling

In this section, we present an algorithm which has been presented for scheduling tasks that operate in real-time energy harvesting systems and has been proved optimal [4]. The intuition behind the novel energy-aware ED-H is to run jobs according to Earliest Deadline First (EDF) rules, but this decision is constrained by the use of the notion of slack energy to predict eventual future energy failures. If a job has the potential to miss its deadline in the future due to energy insufficiency, the current jobs are delayed as long as possible by expanding the available slack time to replenish a maximum of energy. In our algorithm, we use the EDL server

[7] to compute the slack time and determine the busy periods of periodic tasks. Furthermore, when the energy reservoir is fully replenished during an idle period, the algorithm resumes executions in order to avoid energy waste. By definition, the slack energy of a job J_i at time t represents the maximum energy that can be consumed to execute jobs from t until the deadline of J_i, while still guaranteeing energy requirements and deadlines [4].In other words, it means the maximum amount of idle time that can be used to delay executions without violating deadlines. Fundamental scheduling concepts required in the energy harvesting system are: slack time and slack energy.

1) The slack time at time t gives the available processor time after executing uncompleted jobs with deadlines at or before d_i. It is given by:

$$ST_{J_i}(t) = d_i - t - h(t, d_i) \tag{1}$$

where $h(t, d_i)$ is the total processing demand of uncompleted jobs at t with deadline at or before d_i. By definition, the slack time of a periodic task set τ at current time t as follows:

$$ST_\tau(t) = \min_{d_i > t} ST_{J_i}(t) \tag{2}$$

The slack time represents the maximum continuous time of processor that could be existing at time t while still meeting the deadlines of all the tasks. It is computed using EDL algorithm [8] and will be used to recharge the battery.

2) The Slack Energy at time t is the maximum quantity of energy that can be used on the time interval [t,d_i) while guaranteeing enough energy for jobs released at or after t and deadline at or before d_i. The slack energy of J_i at current time t by equation 3.

$$SE_{J_i}(t) = E(t) + E_p(t, d_i) - g(t, d_i) \tag{3}$$

where $g(t, d_i)$ represents the total energy required by jobs on the time interval $[t, d_i)$. It is worth noting that if there exists some job J_i such that $SE_{J_i}(t) = 0$, executing any other job with a deadline higher than d_i within $[t, d_i)$ will involve energy starvation for J_i.

Thus, the slack energy of the periodic task set τ at current time t represents the maximum energy surplus that the system could consume instantaneously at t. The slack energy at t is

$$SE_\tau(t) = \min_{t < d_i} SE_{J_i}(t) \tag{4}$$

3) The preemption slack energy at the current time t gives the maximum energy that could be consumed by the active job whilst guaranteeing absence of energy starvation for jobs that may preempt it. Let d be the deadline of this active job. The preemption slack energy at t is

$$PSE(t) = \min_{t < d_i < d} SE_{J_i}(t) \tag{5}$$

V. Slack Stealing with energy Preserving (SSP) algorithm

In this section, we present a new server, namely SSP (Slack Stealing with energy Preserving) which builds upon previous research into slack stealing algorithms including the EDL scheduler. The EDL server determines the maximum processing time which may be stolen from hard deadline periodic tasks, without jeopardizing their timing constraints. Periodic tasks are scheduled according to the EDF scheduler. The EDL server was extended to tasks with synchronization constraints [9]. An approximate version of the EDL server was proposed to reduce the runtime overheads due to high computation costs. The original slack stealer, EDL is greedy since the available slack time is always consumed if there is at least one aperiodic task ready to run.

The main principle of the slack stealer SSP for aperiodic servicing with ED-H is to authorize aperiodic job executions as long as it does not involve a deadline violation for all the jobs generated by the periodic task set τ. Let us recall that a deadline violation occurs either because of processing time starvation (lack of time to complete a task before deadline) or energy starvation (lack of energy to complete a task before deadline).

This leads us to consider the system slack at current time t as a pair of values respectively called slack time and slack energy. The slack time of τ at time t is defined as the maximum processing time which is available at t after executing timely the tasks of τ. Slack time is a dynamic value that expresses variation of processing surplus. Its computation permits to determine whenever necessary for how long time the processor could be let either idle or busy executing additional tasks such as aperiodic ones.
The slack energy of τ at time t is defined as the maximum energy which is available at t after executing timely the tasks of τ. Slack energy is also a dynamic value that expresses variation in energy surplus. Its computation permits to determine whenever necessary how much energy could be either wasted or consumed executing additional tasks such as aperiodic ones.

In summary, the basic idea of the SSP server is to steal as much as possible both processing time and energy. It leads to execute the aperiodic tasks as soon as possible while avoiding energy starvation and deadline missing for periodic tasks. Whenever no aperiodic tasks are present, the periodic tasks are operated classically with the ED-H scheduler.

Whenever new aperiodic task arrives, it uses the collected values of slack time and slack energy to decide to service either an aperiodic task or a periodic one.

The slack stealer SSP can be viewed as a task which is ready for execution whenever the aperiodic queue is

non-empty. This task is suspended when the queue is empty. The slack stealer receives the highest priority whenever there is slack i.e both slack time and slack energy. It receives the lowest priority whenever there is either no slack time or no slack energy. The slack stealer SSP selects the aperiodic tasks in FCFS order.

The framework of the *SSP* scheduling algorithm is as follows:

Algorithm 1 Slack Stealing with energy Preserving (*SSP*)

Input:
Current time
A list of periodic tasks at t L(t)
A list of aperiodic tasks at t Ap(t)
Output: *SSP* Schedule.

```
1: t = 0
2: while (1) do
3:    if Ap(t) is not empty AND energy reservoir is not empty
      AND SlackEnergy(t) > 0 AND SlackTime(t) > 0 then
4:       /*schedule_FCFS (Ap(t))*/
5:    else
6:       /*schedule_ED-H(L(t))*/
7:    end if
8:    t := t + 1
9: end while
```

The property of optimality of SSP is attained if the actual execution time of aperiodic requests is equal to the worst execution time. In that, SSP minimizes the response times of aperiodic requests.

Theorem 1: [12] All periodic tasks meet their deadlines when scheduled according to ED-H with the slack stealer for aperiodic servicing.

Theorem 2: [12] For any periodic task set scheduled according to ED-H and a stream of aperiodic tasks processed in FIFO order, the slack stealing algorithm minimizes the response time of every aperiodic task, amongst all algorithms which are guaranteed to meet all deadlines.

A. Illustrative Example

Consider a periodic task set Γ that is constituted of two tasks, $\Gamma = \{\tau_i \mid 1 \leq i \leq 2 \ \ and \ \tau_i = (C_i, D_i, T_i, E_i)\}$. Let $\tau_1 = (4,9,9,18)$ and $\tau_2 = (3,12,12,18)$. Let us consider also $Ap = \{Ap_j \mid Ap_j = (a_j, c_j, e_j)\}$ the stream of 2 aperiodic tasks where $Ap_1 = (9,1,5)$ and $Ap_2 = (18,3,15)$. We suppose that the energy storage capacity is $E(0) = 10$ and the rechargeable power, P_p , is constant along time and equals 4.

At time 0, the residual capacity of the storage unit is maximum since the storage is full. τ_1 is the highest priority task which finishes at time 4 and consumes 18 energy units. At time 4, the residual capacity is given by $E_{max} - E_1 + P_p * C_1 = 8$. Now, τ_2 has the highest priority. It executes completely until time 7 and consumes 18 energy units. The residual capacity equals 2 energy units.

From $t = 7$ until $t = 9$, the processor remains idle and the energy level at $t = 9$ is $E(9) = 10$ energy units.

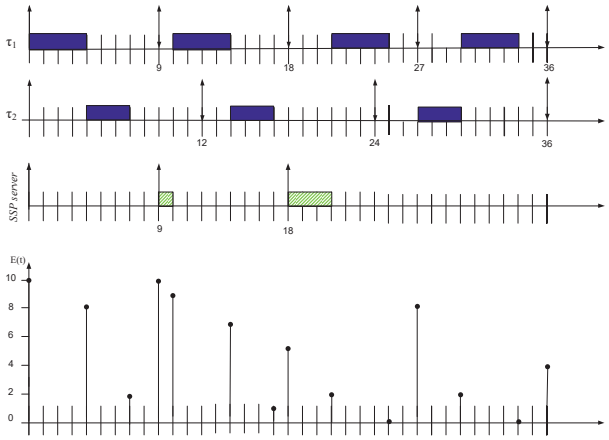

Fig. 1. Aperiodic servicing with the slack stealer SSP.

At time 9, Ap_1 is released. As the storage unit is not empty, the slack time is positive (equal to 5) and the slack energy is positive (equal to 4), Ap_1 is authorized to execute and to consume a maximum of 5 energy units. After its execution, the residual capacity falls at 9. At time 10, according to the ED-H scheduler, we continue to schedule the tasks till time 18 where the aperiodic task Ap_2 is released. Here, we have to check again if we abide by the three conditions: 1) the reservoir is not empty (5 energy units), 2) the slack time is strictly positive, equal to 5 and 3) the slack energy is strictly positive and equal to 23. Consequently, Ap_2 is authorized to execute immediately for executing during 3 units of time and for consuming 15 units of energy. Periodic tasks execute according to ED-H till the end of the hyperperiod where the energy reservoir contains 4 energy units, as illustrated by Figure 1.

We notice that the aperiodic tasks Ap_1 and Ap_2 are executed at the earliest time instant regarding processor and energy availabilities whilst the periodic tasks are deferred as much as possible. We observe that the response times of the aperiodic tasks Ap_1 and Ap_2 are 1 and 3 units of time, respectively, which is a clear evidence of minimizing the aperiodic responsiveness.

VI. Performance Evaluation

In this section, simulation results are presented to demonstrate the variation of the average response times of soft aperiodic tasks under four different energy profiles: constant, sine wave signal of period $\pi = 2$, a rectifier, and a pulse signal with a 20% duty-cycle (Figure 2). The different waveforms produced are periodic.

We assume that the recharging power P_p for the first profile is constant and equal to 5. The output power of the three profiles (Figure 2) is supposed variable between 2 and 17. The minimum size of the reservoir (E_{min}) for the different profiles should not be less than the area of each profile.
The objective is to minimize soft aperiodic responsiveness, while still maintaining the feasibility of periodic tasks. At this prospect, we compared the performance of the SSP algorithm to that of the Background with Energy Surplus (BES) server and the Background with Energy Preserving (BEP) server [13]. Under the BES server, aperiodic tasks should wait for the total replenishment of the storage unit and for the non existence of periodic tasks to execute. Under the BEP server, aperiodic tasks execute only if their energy consumption does not provoke possible energy starvation for any periodic task. This is guaranteed by computing the so-called slack energy of the system which gives at every instant, the maximum energy which could be consumed while still guaranteeing energy feasibility of periodic tasks. We may easily predict that the BEP server will outperform the BES server.

It is worth noting that throughout our simulation studies, we assume that a total energy load U_e includes 50% of the periodic energy utilization U_{ep} and 50% of the aperiodic energy load U_{es}. Similarly, a total processing load U_p incorporates 50% of the periodic processor utilization U_{pp} and 50% of the aperiodic utilization U_{ps}. The periodic task set is composed of 20 tasks with periods, durations and energy requirements randomly generated. Periods and computation times are distributed uniformly, corresponding of U_{pp}. Their energy requirements depend on U_{ep}. Periodic task sets are considered to be schedulable in terms of processing time and energy. For aperiodic tasks, by modeling a poisson aperiodic arrival, we generate a total of 15000 aperiodic jobs with execution times and energy consumption uniformly distributed depending on U_{es} and U_{ps}. Interarrival times are picked and predefined at the start of each experiment so that the aperiodic task arrived and executed before the end of the hyperperiod H.

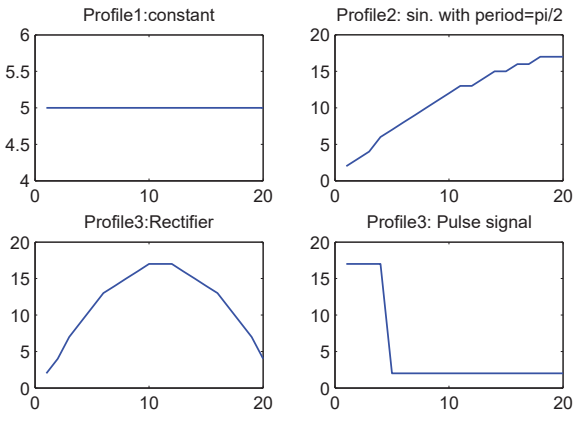

Fig. 2. Energy source profiles under study.

The evaluations are performed for a total energy utilization applied to the system, which varies from 5% to 100%, while the total processing utilization load remains constant ($U_p = 0.6$). For a given U_e/P_p, each point of SSP, BES and BEP is computed over 100 simulations. The four simulation experiments depicted in Figures 3, 4, 5 and 6 refer to the results obtained for constant profile, Sine wave with period $\pi = 2$, Rectifier signal and Pulse signal of 20% duty-cycle respectively. They illustrate the mean aperiodic response time which is normalized with respect to the aperiodic computation time. In this way, a value of 2 on the y-axis means an average response time 2 times longer than the task computation time. The lower the response time curve lies on these graphs, the better the algorithm is for enhancing the aperiodic response time.

The graphs plainly show that SSP outperforms the BEP and BES algorithms under all energy profiles. This confirms our theoretical analysis that has been performed without any restrictive assumption on the production of energy along time. This advantage is all the more significant as the total energy utilization is higher due to smart profit of slack energy. For example, under the constant profile (Figure 3), if we consider the SSP server provides a response time at least 22% lower compared to background servers.
Moreover, BEP performs better than BES with a small deviation and exhibits a significant degradation with respect to the BEP algorithm (Figure 6) under the Pulse signal profile.
It is expected that the results of the Pulse model show that the performance obtained for the three algorithms and in particular for BES is slightly less than ones obtained under the three other models. For example, the responsiveness by BES under the Pulse signal profile is at least 12.5% less than the other profiles. The reason is that the power is only harvested on 20% duty-cycle of the overall signal and BES permits aperiodic tasks to be only executed when the energy reservoir is full, which leads to an increase in aperiodic responsiveness.

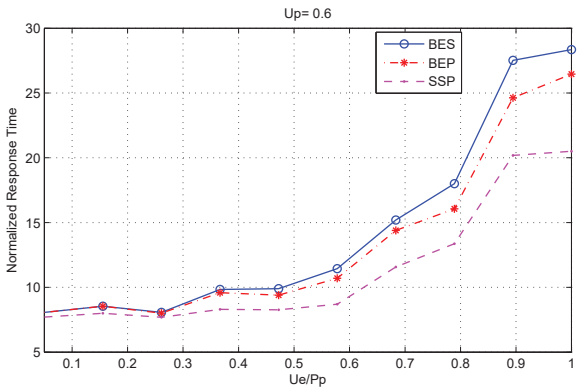

Fig. 3. Normalized aperiodic response time with respect to U_e/P_p, for U_p=0.6 under constant profile.