Real-Time Scheduling of aperiodic tasks in Energy Harvesting Devices

Rola EL Osta and Maryline Chetto
University of Nantes
LS2N Laboratory
1 Rue de la Noë, F-44321
Nantes, France

Hussein El Ghor and Rafic Hage
Lebanese University
IUT Saida, Lebanon
B.P. 813, Saida Lebanon

***Abstract*—This paper deals with the scheduling issue that arises in small energy harvesting devices. Such ones support both soft aperiodic tasks and hard deadline periodic tasks. The scheduler has to produce a sequence without deadline missing and with no energy starvation while reducing the response time of aperiodic tasks. The proposed servers, called *BES* (Background with Energy Surplus server) and *BEP* (Background with Energy Preserving server), are based on the classical background server and on the optimal ED-H scheduler. Experimental results demonstrate that the *BEP* server offers a substantial improvement over the *BES* server in terms of the aperiodic responsiveness under various time and energy settings.**

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

I. Introduction

In Energy Harvesting systems, we have to exploit perpetually the available ambient energy which is strongly dependant on the environment [1]. The energy consumption should be adapted to maximize the performance instead of minimizing it as in the classical battery powered systems. Real-time systems have to simultaneously perform multiple complex tasks under stringent time constraints. So, the operating system has to control properly the occupation of the processing unit in order to have sufficient energy in the storage unit to attain all the constraints at every time. The produced energy may arrive by burstiness and has to be preserved so that the device can still be performed at a later moment. The main challenge for an EH (Energy Harvesting) system resides in its performance improvement while abiding by the quantity of energy varying in time without dissipating or depleting the energy preserved in the storage unit. The system then works in a neutral mode of energy by consuming only as much energy as collected [2].

The problem of jointly scheduling periodic and soft aperiodic tasks has been considered in the literature under fixed and dynamic priority assignments. It consists of scheduling aperiodic jobs as soon as possible, but without compromising the time and energy guarantee performed on periodic tasks. Many approaches with different performance and complexity have been investigated such as the Background server [3], the Deferrable server [4], the Sporadic server [3] [5] [6], the Total Bandwidth server [3] [6] and the Earliest Deadline Late server [7].

In this paper, we further extend the Background mechanism and study it in an energy harvesting environment. We firstly propose "Background with Energy Surplus" (BES) server which executes the aperiodic tasks in idle times and energy surplus. We secondly propose "Background with Energy Preserving" (BEP), a variation of BES. It executes the aperiodic tasks as long as this execution does not involve energy starvation for the future periodic tasks.

We are also reporting the problem of the dynamic joint scheduling of periodic tasks and soft aperiodic tasks with energy constraints. Thus, the objective of a scheduling algorithm is to minimize the response time of aperiodic tasks when periodic tasks are scheduled by Earliest Deadline with energy Harvesting (ED-H) [8].
This paper is constructed as follows. Model and terminology are explained in Section 2. Background materials are presented in Section 3. Scheduling of aperiodic tasks with the two algorithms are discussed in Section 4. Simulation results are illustrated in Section 5. Finally, the conclusion is given in Section 6.

II. 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 [8].

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 [9]. 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 $U_{pp} = \sum_{\tau_i \epsilon \tau} \frac{C_i}{T_i}$ which is less than or equal to 1. The energy utilization of τ is defined as $U_{ep} = \sum_{\tau_i \epsilon \tau} \frac{E_i}{T_i}$.

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.

The energy source is characterized by $P_p(t)$, the instantaneous charging rate produced by the source of power that includes all losses. The energy produced on $[t_1, t_2)$ is given as $E_p(t_1, t_2) = \int_{t_1}^{t_2} P_p(t)dt$. Our system utilizes an ideal energy storage unit with a nominal capacity C of units of energy. The energy level at time t is denoted $E(t)$. Initially, the storage unit is completely charged (i.e. $E(0) = C$). The reserved energy does not leak any energy over time and may be used at any time later [8].

III. Background Materials

The ED-H [8] is a novel energy-aware scheduling algorithm; which is a variant of EDF, and proved to be optimal for real time tasks scheduling. All tasks are executed as early as possible while there is enough residual energy capacity in the battery to guarantee the energy-feasibility of all future arriving jobs, i.e. ED-H enables a job to execute just after having checked that a time slot execution will not lead to an energy shortage either for itself or for a job whose occurrence will arrive in the future. This scheduler relies on the time and energy characteristics of jobs as well as the charging level of the storage unit. The processor is in idle state if one of these conditions is not respected. Otherwise, it becomes busy and a job is processed as long as the slack energy (the maximum amount of energy that a job can consume while providing sufficient energy for highest priority tasks) is positive and the storage unit is filled. The system cannot be active as long as the slack time (the maximum continuous time of processor while still meeting the deadlines of all the jobss) is positive and the storage unit is deplenished. The time and energy concepts are deeply explained in [8].

IV. Background analysis under ED-H

In this section, we first introduce an algorithm, called Background with Energy Surplus (BES) which executes an aperiodic task only if no periodic task is pending for execution and the energy reservoir is fully replenished. A similar algorithm, called Background with Energy Preserving (BEP), is also presented. The algorithm significantly improves the performance of the previous server since aperiodic tasks can be executed as long as this does not involve energy starvation for the future periodic ones.

A. The Background with Energy Surplus (BES) algorithm

In BES algorithm, periodic tasks are scheduled with ED-H. If an aperiodic task arrives, it is executed only if there is no ready periodic task and the energy storage unit is fully replenished. This means that the aperiodic task consumes only the energy which should be wasted if there was no aperiodic task (i.e. in remaining idle times with remaining energy). The aperiodic task is selected in FIFO order. However, this mechanism does not compromise the performance of future periodic tasks.

The procedure of producing the BES schedule is summarized in the following pseudo-code (Algorithm 1).

Algorithm 1 BES

```
Require:
    t: current time
    $L(t)$: list of periodic tasks at t
    $J(t)$: list of aperiodic tasks at t
 1: while TRUE do
 2:     if L(t) not empty then
 3:         schedule_EDH(L(t))
 4:     else if J(t) not empty AND reservoir is FULL then
 5:         schedule_FCFS (one time unit of J(t))
 6:     else
 7:         let processor idle
 8:     end if
 9:     $t := t + 1$
10: end while
```

B. The Background with Energy Preserving (BEP) algorithm

In this algorithm, an aperiodic task is executed if a) the energy reservoir is not empty, b) no periodic task is pending for execution and c) this execution does not involve energy starvation for the future periodic ones (i.e, the system slack energy is positive). BEP selects the aperiodic task in FIFO order. Periodic tasks are also scheduled according to ED-H.

The procedure of generating the BEP schedule is summarized in the pseudo-code (Algorithm 2).

Algorithm 2 BEP

```
Require:
    t: current time
    $L(t)$: list of periodic tasks at t
    $J(t)$: list of aperiodic tasks at t
 1: while TRUE do
 2:     if L(t) not empty then
 3:         schedule_EDH(L(t))
 4:     else if J(t) not empty AND reservoir not empty AND
            $SlackEnergy(t) > 0$ then
 5:         schedule_FCFS (J(t))
 6:     else
 7:         let processor idle
 8:     end if
 9:     $t := t + 1$
10: end while
```

C. 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 $\tau_1 = (3, 6, 6, 7)$ and $\tau_2 = (2, 8, 8, 5)$. Let us consider also $Ap = \{Ap_j \mid Ap_j = (a_j, c_j, e_j)\}$ the stream of 2 aperiodic tasks where $Ap_1 = (6, 1, 3)$ and $Ap_2 = (13, 2, 6)$. We suppose that the energy storage capacity is $E(0) = 8$ and the rechargeable power, P_p , is constant along time and equals 2.

Figs. 1 and 2 illustrate the schedules produced by BES and BEP respectively during an hyperperiod $H = 24$. At time $t = 0$, all tasks are released (Fig. 1) τ_1 is the task with highest priority and is executed until $t = 3$ where $E(3) = E(0) - E_1 + P_p * C_1 = 7$ energy units. At time $t = 3$, τ_2 is the task with highest priority and is executed until $t = 5$ where $E(5) = 6$ energy units.

From time $t = 5$ up to $t = 6$, the processor remains idle because there are no pending tasks. During that time interval, the energy storage will recharge and the energy level at $t = 6$ is given by $E(6) = 8$ energy units.

At $t = 6$, the aperiodic task Ap_1 is released. As τ_1 is ready to execute, it is processed till $t = 9$, where $E(9) = 7$ energy units. τ_2 is now executed until $t = 11$ where the energy storage capacity becomes 6 energy units.

At $t = 10$, Ap_1 can't be processed as the storage unit is not fully charged. It is authorized to execute at $t = 22$, where the periodic list is empty and the energy storage is completely charged ($E(22) = 8$ energy units). Ap_1 is executed till $t = 23$ where $E(23) = 7$ energy units. Its response time is 17 time units.

Under BEP (Fig. 2), when the aperiodic task Ap_1 arrives at time $t = 6$. But as its execution conditions are not all verified (i.e. the periodic list is not empty), Ap_1 cannot be executed. It is processed at time $t = 11$ where all conditions are applied (storage unit is not empty, no ready periodic task and by applying equation 13 in [8], we have $SE(11) = E(11) + \int_{11}^{18} P_p dt - E_1 = 7 > 0$). Ap_1 is completely finished at $t = 12$, where $E(12) = 5$ energy units. Its response time is 6 time units.

We continue to schedule periodic tasks and the soft aperiodic task Ap_2 in the same way, either under BES server or under BEP server. It can be concluded from this example that the BES schedule results in an aperiodic execution delay of Ap_1 and in a non-execution of Ap_2 during the Hyperperiod. On the other hand, the BEP schedule shows a much greater performance in terms of aperiodic responsiveness of each of Ap_1 and Ap_2.

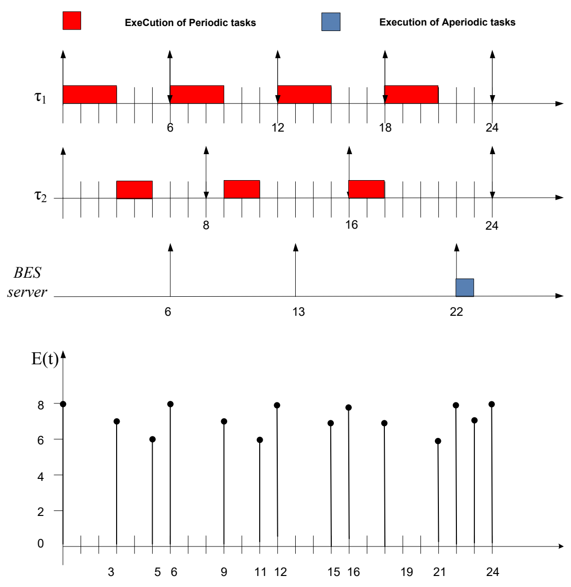

Fig. 1. Scheduling under BES server

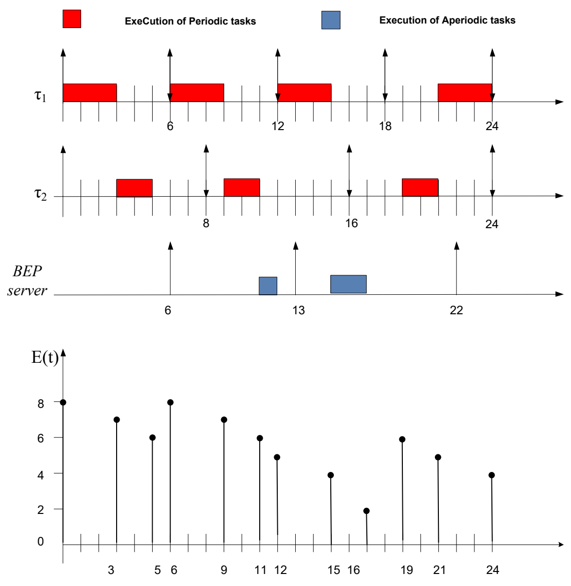

Fig. 2. Scheduling under BEP server

V. Performance Evaluation

In this section, simulations were reported to compare the performance of BES against BEP. We shall measure the average response times of soft aperiodic tasks as a function of the total processing load U_p. 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}. For a given U_p, the average response time is computed over 100 simulations. The periodic task set is composed of 30 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}. For aperiodic tasks, by modeling a poisson aperiodic arrival, we generate a total of 15000 aperiodic jobs depending on U_{es} and U_{ps}. In the results illustrated below, each point corresponds to the mean response time normalized with respect to the average aperiodic execution time.

Figs. 3 and 4 show that BEP offers lower response times than BES. When the system has low energy constraints (i.e. $U_e/P_p = 0.2$), the response times achieved by BES and BEP are close to one for a lower total load, meaning that aperiodic tasks execute immediately. When the total load increases, the two background servers perform poorly and have a similar performance (Fig. 3). Fig. 4 shows the case in which $U_e/P_p = 0.8$. The normalized response times of BES and BEP increase with the increase of U_p. BEP services the aperiodic tasks 1.5 times faster than BES and converges gracefully to BES as U_p increases as the execution of aperiodic tasks is done as long as there is always sufficient energy for future periodic tasks.

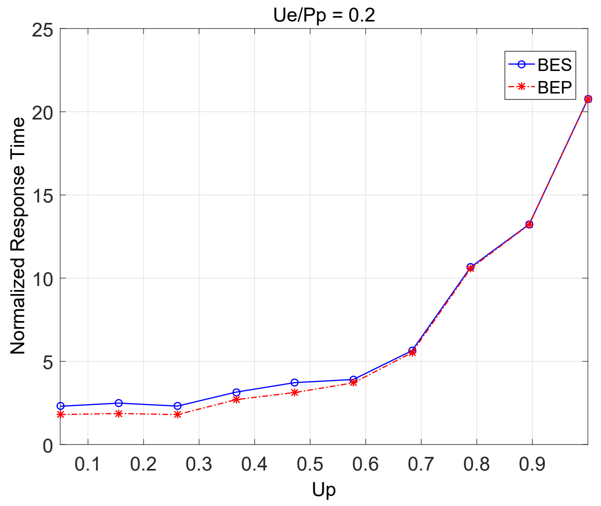

Fig. 3. Normalized response time with respect to U_p, for U_e/P_p=0.2.

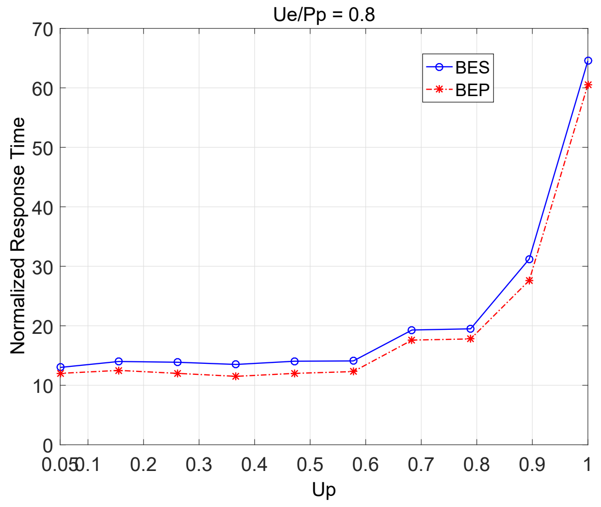

Fig. 4. Normalized response time with respect to U_p, for U_e/P_p=0.8.

VI. Conclusion

In this paper, we studied the case when we have to execute both hard deadline periodic tasks and soft aperiodic tasks in a processing and energy constrained environment. The objective is to guarantee that no deadline is missing for the periodic tasks and that a good responsiveness is achieved for the aperiodic tasks. Periodic tasks are scheduled according to the ED-H scheduling algorithm. We proposed two algorithms based on background servicing, respectively called BES and BEP. These servers are simple to implement. The simulation results show that the BEP server performs better than BES in terms of responsiveness because the aperiodic tasks in BEP are executed with the fact that the future periodic are certainly executed. As a future work, we are considering to use the background-based servers to handle hard aperiodic tasks.

References

[1] S. Sudevalayam and P. Kulkarni.*Energy Harvesting Sensor Nodes: Survey and Implications*, IEEE Communications Surveys and Tutorials, Vol. 13 ,no.3, pp. 443-461, 2010.

[2] A. Kansal, J. Hsu, M. Srivastava, and V. Raghunathan. *Harvesting aware power management for sensor networks*, On DAC'06. pp. 651-656.

[3] M. Spuri and G. Buttazzo.*Scheduling Aperiodic Tasks in Dynamic Priority Systems*, Journal of Real-Time Systems, 1996.

[4] J.K. Strosnider, J.P. Lehockzy, and L. Sha. *The Deferrable Server Algorithm for Enhanced Aperiodic Responsiveness in Hard Real-Time Environments*, IEEE Transactions on computera, Vol. 44, No.1, January 1995.

[5] J. A. Stankovic, M. Spuri, K. Ramamritham, and G. C. Buttazzo. *Deadline Scheduling for Real-Time Systems, EDF and related algorithms*, Kluwer Academia Publishers, 1998.

[6] M. Spuri and G.-C. Buttazzo. *Efficient aperiodic service under earliest deadline scheduling*, In proceedings of the IEEE Real-Time Systems Symposium, 1994.

[7] H. Chetto and M. Chetto. *Some results of the earliest deadline scheduling algorithm*, In IEEE Transactions on Software Engineering, 15(10): 1261-1269, 1989.

[8] M. Chetto. *Optimal Scheduling for Real-Time Jobs in Energy Harvesting Computing Systems*, IEEE Transactions on Emerging Topics in Computing, DOI: 10.1109/TETC.2013.2296537, 2014.

[9] R. Jayaseelan, T. Mitra, and X. Li. *Estimating the Worst-Case Energy Consumption of Embedded Software*, 12th IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 81-90, 2006.