Energy Management and Real-Time Scheduling for Self-Powered Sensors

Rola El Osta, Maryline Chetto, Hussein El Ghor

Energy Management and Real-Time Scheduling for Self-Powered Sensors

Rola El Osta
LS2N Laboratory - UMR CNRS 6004
University of Nantes
Nantes, France
rola.elosta@ul.edu.lb

Maryline Chetto
LS2N Laboratory - UMR CNRS 6004
University of Nantes
Nantes, France
maryline.chetto@univ-nantes.fr

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

***Abstract*—Most of wireless sensors are dynamically changing and execute two types of tasks: hard deadline periodic ones and aperiodic ones with no deadline. The emergence of energy harvesting technologies makes it possible to design self-powered sensors through environmental energy. Nevertheless, classical scheduling techniques need to be reconceived so as to take into account the fluctuating energy source and energy consumptions of tasks. Hence, we firstly describe the called TB-H and TB*-H aperiodic task servers which are extensions of the famous Total Bandwidth server. We show how TB-H and TB*-H permit to provide short response times for the aperiodic tasks by computing adequate virtual deadlines taking into account both time and energy constraints. Our second contribution lies in extensive simulations to bring to light the effectiveness of these two novel Bandwidth based servers in comparison to basic background approaches.**

***Index Terms*—energy harvesting, aperiodic task servicing, preemptive scheduling, energy management, Earliest Deadline First.**

I. Introduction

Wireless sensor networks (WSNs) have the ability to change our world with online machine condition monitoring because they are cost-effective. They can communicate wirelessly to gather environmental data and provide more convenient and swift solution in industry, which leads to significant savings. One of the most important challenges on sensor nodes is to power them and to keep them functional as long as possible, especially when the design complexity and the topology are not guaranteed.

To overcome this severe challenge, energy harvesting wireless sensor nodes are gaining more the attention of engineers. They convert microwatt or milliwatt level power from the environment. They also support a true wireless and maintenance-free machine condition monitoring [1] [2].

A wireless sensor node generally includes four key units: a sensing unit, a processing unit, a communication unit and a power unit as described in Figure 1. Since traditional batteries have a finite lifetime, restricted energy density, and capacity, the power unit poses a significant problem. Furthermore, battery efficiency has not improved dramatically in relation to the massive rise in power consumption in electronic devices.

When the batteries run out, replacing or recharging them can be a costly and time-consuming process, particularly if the nodes are located in remote or inaccessible locations. Fortunately, harvesting the wasted energy from machines or its surrounding environments, such as thermal energy, magnetic and electric fields and mechanical energy for use in powering sensor nodes by means of energy harvesting (EH) technologies, is the conversion of ambient energy present in the environment into electrical energy. This method significantly extends the life of sensing nodes, as well as lowering monitoring device maintenance costs and avoiding battery leakage in the field (see Figure I).

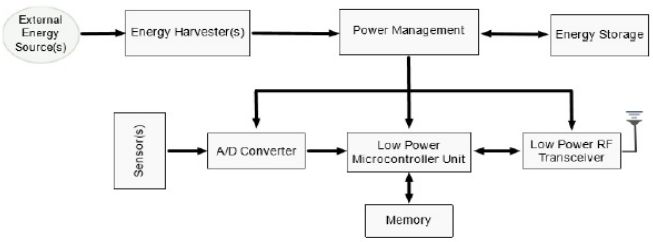

Fig. 1. Framework of an energy harvesting sensor.

Task scheduling in a self-powered sensor should consider the properties of the energy source, capacity of the energy storage as well as the deadlines of the tasks [3]. The real-time operating system should be able to handle not only hard deadline periodic tasks but also soft aperiodic tasks with irregular arrival times and no deadlines. Periodic tasks are generally used to implement activities such as sensory acquisition or control loops which need to be executed at constant rates to insure system stability. Hence periodic tasks often have hard deadlines that must be met under all anticipated circumstances. On the other hand, aperiodic tasks are usually employed to implement non critical activities such as external event task, human interaction task, etc.. Contrary to periodic tasks, most of those aperiodic tasks are required to be completed as early as possible rather than to be completed by deadlines. The goal of the scheduler is to guarantee the deadlines of periodic task while providing good response times for soft aperiodic tasks. We have proposed a novel scheduling algorithm, namely Energy Deadline with energy Harvesting (ED-H) and we have proved it to be optimal [4]. If a task set is feasible on a

platform composed of given processor, energy harvester and energy reservoir, then it can be feasibly scheduled using the ED-H algorithm on the same platform. ED-H assigns priorities based upon the deadlines of the tasks as the non energy aware scheduler EDF (Earliest Deadline First) [5]. The scheduler ED-H requires clairvoyance since its uses accurate prediction on the energy coming from the environmental source. The study of ED-H presented in [4] assumes that all the tasks are deadline constrained. Real-time applications exhibit aperiodic events leading to execute sporadically soft real-time tasks i.e. tasks with no deadline. In this paper, we consider mixed task sets in energy harvesting sensors where the application software has hard deadline periodic tasks and soft aperiodic tasks. The scheduling issue then consists in guaranteeing deadlines of the periodic tasks while executing the aperiodic ones with minimal responsiveness.

There have been extensive studies about scheduling hard deadline periodic tasks and soft aperiodic tasks for several decades. Priority driven algorithms can be found in the literature [6]. Spuri and Buttazzo proposed the Total Bandwidth (TB) servicing approach. TB provides optimal responsiveness in dynamic priority systems [7]. Under this method, suitable deadline is assigned to every occurring aperiodic task which is scheduled according to the EDF rule, jointly with the periodic tasks.

Our work is to extend the Total Bandwidth technique so as to take into account energy harvesting constraints.

The rest of this paper is organized as follows: Background materials are given in the subsequent section. Section 3 describe the task model under study. The novel energy harvesting aware schedulers for aperiodic tasks are presented in section 5 and the effectiveness of simulation are illustrated in section 6. Finally, section 7 concludes the paper.

II. Background materials

A. Scheduling periodic tasks with ED-H

As EDF, the energy harvesting aware ED-H scheduler selects for execution the ready task with the closest deadline. ED-H includes dynamic power management facilities so as to decide when the processor should be in the busy mode and when the processor should be in the idle mode. At every time instant, the decision depends on two dynamic variables respectively called *slack time* and *preemption slack energy*. The slack time represents the maximum length of the time interval where the processor could be let idle while guaranteeing no deadline violation. The preemption slack energy represents the maximum energy which could be consumed by the active task while no task can suffer from energy starvation. Optimality of ED-H was established in [4].

B. Aperiodic servicing with no energy consideration

Aperiodic tasks can be simply executed by allocating them in time slots left unused by periodic tasks. Such a background approach does not lead to efficient responsiveness in contrast to server based techniques. An aperiodic server is defined as an additional periodic task, characterized by a period and a fixed execution time called server capacity. Once active, the server uses its capacity to execute the aperiodic tasks. The so-called Total Bandwidth server (TB) was introduced by Spuri and Buttazzo in [8]. TB showed excellent performance in response time, while its design is fairly simplified. Every time an aperiodic task arrives, the TB algorithm assigns the possible earliest deadline to the aperiodic task. Since the aperiodic task assumed in this paper does not have a deadline, it is a virtual deadline. Once the task is assigned the virtual deadline, it is scheduled by EDF jointly with the periodic tasks. We recall how to compute the virtual deadline of any occurring aperiodic task so that the fraction of processing time demanded never exceeds the server utilization U_{ps}. The formula is as follows [8]:

$$d_k = max(r_k, d_{k-1}) + \frac{c_k}{U_{ps}} \tag{1}$$

where c_k is the execution time of the newly occurring aperiodic task and U_{ps} is the server utilization factor (i.e. its time bandwidth). By definition, $d_0 = 0$. Moreover, when a new deadline d_k is assigned, the bandwidth already allocated to the previous requests is taken into account by the value d_{k-1}. Once this fictive deadline is assigned to the aperiodic request, it is scheduled jointly with the periodic tasks according to the EDF algorithm.

The following lemma gives the schedulability condition for applying TB:

Lemma 1: In any time interval $[t_1, t_2]$, if C_{ape} is the total execution time demanded by aperiodic tasks arrived at t_1 or later and served with deadlines less than or equal to t_2, then

$$C_{ape} \leq (t_2 - t_1)U_{ps} \tag{2}$$

Proof: see [7]

Theorem 1: Let a set of n periodic tasks with processor utilization U_{pp} and a set of aperiodic tasks served by TB with processor utilization U_{ps}. The application is feasible if and only if

$$U_{pp} + U_{ps} \leq 1. \tag{3}$$

Proof: see [7]

Theorem 1 will be extended so as to consider energy consumption of the tasks and energy production of the environmental source.

III. Assumptions and Terminology

A. Task model

A periodic task set τ is defined as follows: $\tau = \{\tau_i \mid 1 \leq i \leq n\}$. Task τ_i has period T_i, relative deadline D_i, worst case execution time C_i (normalized to processor computing capacity) and worst case energy requirement E_i. So, the four-tuple (C_i, E_i, D_i, T_i) is associated to τ_i. We assume that $0 < C_i \leq D_i \leq T_i$. Task τ_i generates an infinite set of jobs, released at times 0, T_i, $2T_i$. The hyper-period H of τ is the least common multiple (LCM) of the periods T_i. So, $H = LCM(T_1, T_2, ..., T_n)$. The processor utilization of τ is $U_{pp} = \sum_{\tau_i \epsilon \tau} \frac{C_i}{T_i}$ which is less than or equal to 1. Similarly,

$U_{ep} = \sum_{\tau_i \epsilon \tau} \frac{E_i}{T_i}$, namely the energy utilization, characterizes the average energy consumption of τ per time unit.

The task set τ gives rise to an infinite set of jobs which are scheduled by the optimal uniprocessor scheduler ED-H. A four-tuple (r_j, C_j, E_j, d_j) is associated with a job J_j and gives its release time, worst case execution time (WCET), worst case energy consumption (WCEC) and (absolute) deadline, respectively [9].

Aperiodic tasks arrive in the system irregularly. Each aperiodic task has worst case execution time and worst case energy requirement both considered to be known at arrival time. A stream of aperiodic tasks is defined by $Ap = Ap_i(a_i, c_i, e_i), i = 1..m$. a_i is the arrival time, 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 overhead due to context (processor) switching is negligible compared to computational time of the tasks. Otherwise, it is included in it. Tasks do not suspend themselves or synchronize with other ones.

B. Energy assumptions

At every time t, the harvester (e.g. solar panel) draws energy from the environment. It converts energy into electricity with instantaneous charging rate $P_p(t)$ that incorporates all losses. The energy harvested in the interval $[t_1, t_2)$ is given by $E_p(t_1, t_2) = \int_{t_1}^{t_2} P_p(t)dt$. The energy consumed in any unit time-slot is never lower than the energy produced in the same unit time-slot. Consequently, the residual capacity of the energy storage never increases whenever a job executes. The source energy is not controllable and is not necessarily constant along time.

We may predict incoming energy accurately for near future with negligible time and energy cost.

An ideal energy reservoir (e.g. super-capacitor or rechargeable battery) permits to continue operation when there is no production of energy. The nominal capacity C of the reservoir gives the maximum amount of energy that can be stored at any time. The energy at given time t available in the storage is $E(t)$. If the reservoir is fully charged at time t and we continue to charge it, energy is wasted. If the reservoir is fully discharged at time t (energy depletion), the processor is passive and executes no task.

In subsequent sections, we assume the periodic task set is feasible. In other terms, the optimal deadline driven scheduler ED-H guarantees to execute the periodic tasks with no deadline violation and no energy starvation.

IV. The TB-H Server

A. Total Bandwidth for energy harvesting settings

Let us show how to assign the virtual deadlines to the aperiodic tasks when the energy availability is limited in each time interval. We have to consider that the fraction of energy consumed by the aperiodic tasks should not exceed the aperiodic energy utilization of the server, say U_{es}.

Lemma 2: Let a periodic task set τ with average energy consumption per time unit U_{ep}. Let a stream of aperiodic tasks processed in FCFS order by a Bandwidth based server with energy utilization U_{es}. Let E_{ape} the total energy demanded by aperiodic tasks arrived at t_1 or later and serviced with deadlines less than or equal to t_2. Then, we should have

$$E_{ape} \leq (E(t_1) + (t_2 - t_1)P_p)\frac{U_{es}}{P_p} \tag{4}$$

Proof: See [12]

Let us prove that the aperiodic energy utilization does not exceed $\frac{U_{es}}{P_p}$.

Theorem 2: Let a periodic task set τ with average energy consumption per time unit U_{ep}. Let a stream of aperiodic tasks processed in FCFS order by a Bandwidth based server with energy utilization U_{es}. The task set is schedulable only if:

$$U_{ep} + U_{es} \leq P_p. \tag{5}$$

Proof: See [12]

The energy required by the aperiodic tasks in a certain interval of time should never exceed the energy available in that interval of time. Lemma 1 states that there exists some interval $[t_1, t_2]$ where the aperiodic energy demand E_{ape} is lower than the energy equal to $(E(t_1) + (t_2 - t_1)P_p)U_{es}$ that could be available in $[t_1, t_2]$. We may draw Theorem 3, the new deadline assignment method for an aperiodic task under energy harvesting constraints.

Theorem 3: The k-th aperiodic task that arrives at time $t = r_k$ should be assigned a virtual deadline as follows:

$$\tilde{d_k} = max(r_k, \tilde{d_{k-1}}) + \lceil \frac{\frac{E_k}{U_{es}} - E(r_k)}{P_p} \rceil \tag{6}$$

Proof: See [12]

Hereafter, the deadline assignment with TB-H (Total Bandwidth for energy Harvesting systems) is presented.

Theorem 4: Let a periodic task set τ with average energy consumption per time unit U_{ep}. Let a stream of aperiodic tasks processed in FCFS order by the TB-H server with energy utilization U_{es}. The virtual deadline for the aperiodic task Ap_k should be computed as follows:

$$d_k^f = max(d_k, \tilde{d_k}) \tag{7}$$

Proof: From formula 1 and formula 6 in Theorem 3, it comes that formula 7 is satisfied. □

The TB-H aperiodic task server is illustrated by the following pseudo-code:

Require:
t: current time
Ap_k: Aperiodic task that occurs at time t
while True **do**
  if $Ap(t)$ is not empty **then**
    $d_k = max(t, d_{k-1}) + \lceil c_k/U_{ps} \rceil$
    $\tilde{d_k} = max(t, d_{k-1}) + \lceil e_k/U_{es} \rceil$
    $d_k^f \leftarrow max(d_k, \tilde{d_k})$
    assign d_k^f to Ap_k
    insert Ap_k in the ready queue
  end if
  execute the ED-H scheduler
end while

Example 1:
The following example illustrates the TB-H deadline assignment procedure when any aperiodic task occurs. A set of three periodic tasks is considered (see Table I).

TABLE I
PARAMETERS OF THE PERIODIC TASKS

Task	C_i	E_i	D_i	T_i
τ_1	4	9	9	18
τ_2	3	12	12	18

We assume that the reservoir capacity is $C = 10$ energy units at $t = 0$. The rechargeable power is constant along the hyperperiod, here equal to 4 ($P_p = 4$).

The first aperiodic task Ap_1 has computation time equal to 1, energy consumption equal to 5, and release time equal to 9. The second aperiodic task with computation time equal to 3 and energy consumption equal to 15 energy units arrives at $t = 18$.

The periodic processor utilization is $U_{pp} = 0.7$ and the periodic energy utilization is $U_{ep} = 0.875$. This leads to get a bandwidth of processing time $U_{ps} = 0.3$ and bandwidth of energy $U_{es} = 0.125$ dedicated to the aperiodic tasks.

At time 0, the residual capacity is the greatest one since the storage unit is full. τ_1, the highest priority periodic task, runs and completes at time 4, consuming 18 energy units. At time 4, the residual capacity is given by $E_{max} - E_1 + P_p * C_1 = 8$. τ_2 gets the highest priority, and executes completely until time 7 consuming 18 energy units. The residual capacity equals 2 energy units. At time 7, the storage unit is recharging since the processor is let idle. At time 9, Ap_1 arrives. The virtual deadline $d_1 = 13$ is computed through equation 1 (due to processing time bandwidth of the server). A second deadline $\tilde{d_1} = 17$ is computed through equation 6 (due to energy bandwidth of the server). Finally, the deadline $d_1^f = max(d_1, \tilde{d_1}) = 17$ is assigned to Ap_1.
Ap_1 will be jointly scheduled with the periodic tasks under the ED-H scheduler using the deadline newly assigned to Ap_1. As time 17 is the earliest deadline to be met in the system, Ap_1 is executed immediately, consuming 5 energy units. At time 10, the highest priority task τ_1 executes completely according to ED-H where the residual capacity equals 7. Periodic tasks run till time 18 where the second aperiodic task Ap_2 arrives. Ap_2 is assigned a deadline $d_2^f = 47$. Ap_2 is not executed immediately, since at time $t = 18$, there is an active periodic task with a shorter deadline equal to 27. Tasks of the system execute timely according to ED-H till the end of the hyperperiod where the energy reservoir has 8 energy units.

Let us notice that the response times of the aperiodic tasks Ap_1 and Ap_2 are respectively 1 unit of time (which is the lowest possible response time) and 16 units of time.

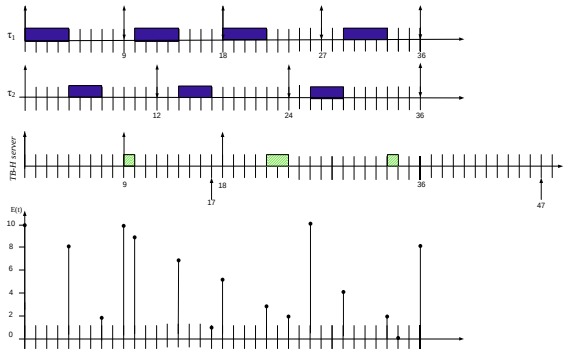

Fig. 2. Illustrative example for the TB-H server.

We have performed a simulation study to show that TB-H exhibits better performance in terms of responsiveness compared to basic background approaches. The following section aims to propose an improved version of TB-H, namely TB*-H. We will demonstrate how TB*-H could optimize responsiveness through additional online computations in comparison to TB-H.

V. THE TB*-H SERVER

The idea of the TB*-H Server lies in finding a virtual deadline which is shorter than the deadline obtained by TB-H while still guaranteeing the feasibility of the periodic tasks. Clearly, the TB*-H Server is an improvement of the TB-H server as TB* was an improvement of TB [10].
The deadline assignment of TB*-H works as follows: the formula given by TB-H serves to determine the deadline of any aperiodic task which arrives while there is no aperiodic task present in the system. A recurring formula is then applied to shorten as much as possible the initial deadline so as to enhance aperiodic responsiveness. The new deadline leads to the worst case finishing time of the aperiodic task taking into account the interferences with the periodic tasks. This procedure assumes that the periodic tasks are scheduled by ED-H.
Let us recall from [10] that the formula is:

$$d_k^{s+1} = t + c_k + I_p(t + d_k^s) \quad (8)$$

Where t is the current time, c_k is the worst case execution time of the aperiodic task and $I_p(t + d_k^s)$ represents the

interference on the aperiodic task due to the jobs of the periodic tasks between t and the deadline. Here the tasks are scheduled by ED-H. As ED-H is an idling variant of EDF they use the same rule to select the future active task.

The final deadline assigned to an aperiodic task Ap_k by TB*-H is computed as follows:

$$d_k^f = max(d_k^{s+1}, \tilde{d_k}) \tag{9}$$

The pseudo-code of TB*-H is similar to that of TB-H with a difference in the computation of the virtual deadline. The formal demonstration of optimality of TB*-H server has not been stated yet. Sub-optimal versions of TB*-H can be implemented according to the number of iterations which are performed to determine the sub-optimal virtual deadline. Higher is the number of iterations, better will be the performance of the server and higher will be the computing overhead.

Example 2: Let us consider the previous example. The periodic processor utilization is $U_{pp} = 0.7$ and the periodic energy utilization is $U_{ep} = 0.875$. We have $U_{ps} = 0.3$ and $U_{es} = 0.125$ dedicated to the aperiodic tasks. By applying equation 8, we obtain $d_1 = 13$ for Ap_1 and $d_2^* = d_2^1 = 25$ for Ap_2. Through equation 6, we got $\tilde{d_1} = 17$ and $\tilde{d_2} = 47$ for Ap_1 and Ap_2, respectively. Finally, 9 gives us the final deadlines $d_1^f = max(d_1, \tilde{d_1}) = 17$ and $d_2^f = max(d_2^*, \tilde{d_2}) = 47$. Hence, these values lead to the same schedule as illustrated for the TB-H server in Figure 2.

VI. Performance Evaluation

In order to assess the performance of TB-H and TB*-H, we performed a simulation study under Matlab and compared them with the two servers BEP and BES [11]. Background with Energy Surplus (BES) serves the aperiodic tasks whenever there are no periodic tasks ready for execution and the energy storage is fully replenished. Under Background with Energy Preserving (BEP), any aperiodic task may execute only if that does not lead to energy shortage for future requests of periodic tasks.

The input data of the generator of periodic tasks are number *n* of tasks, hyper-period *H*, processing utilization U_{pp}, and energy utilization U_{ep}. Periods and computation times of tasks are uniformly distributed, in dependance of $U_{pp} = \sum_{i=1}^{n} \frac{C_i}{T_i}$. Every task has energy consumption proportional to period. Periodic task sets are generated so as to guarantee feasibility in terms of processing time and energy consumption i.e. $U_{pp} \leq 1$ and $U_{ep} \leq P_p$ where P_p is the average recharging power.

The input data of the aperiodic tasks are number of desired tasks *m*, processing utilization factor U_{ps} and energy utilization U_{es}. A stream of aperiodic tasks with uniform distribution is generated trough a Poisson arrival pattern.

A simulation run consists of one task set composed of 20 periodic tasks. Simulations are performed on 10 hyperperiods and every point corresponds to 100 runs. The total processing load U_p incorporates 50% of the periodic processor utilization U_{pp} and 50% of the aperiodic utilization U_{ps}. The total energy utilization U_e includes 50% of the periodic energy utilization U_{ep} and 50% of the aperiodic energy utilization U_{es}. In this work, we assume that the recharging power P_p does not vary along time.

A. Experiment 1: Average response time of aperiodic tasks

In this first set of experiments, $TB^* - H$, TB-H, BEP, and BES servers are compared based on the average response times of soft aperiodic tasks with respect to the total energy load.

Simulation results are reported in Figure 3.

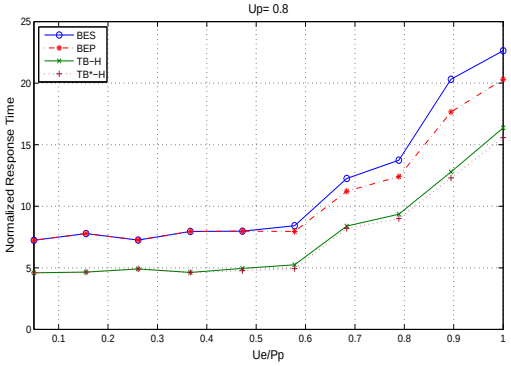

Fig. 3. Aperiodic response time with respect to U_e/P_p, for U_p=0.8.

The Total Bandwidth servers offer better performance compared to the two Background servers. The gain is more significant as the energy load U_e is higher. The major difference in the performance between $TB^* - H$ and the naive servers (BEP and BES) appears for heavy energy loads. TB-H and $TB^* - H$ have identical responsiveness when the energy load is low. They show a slightly different behavior for high energy conditions. The Total Bandwidth servers provide the best response time performance still with a highly processing constrained system (U_p = 0.8). For example, if we consider the performance of $TB^* - H$ when the total energy load equal to 90%, the aperiodic response time offered is reduced by at least 28% in comparison with the background strategies. The optimality of $TB^* - H$ server has to be paid with the increasing number of shortening steps.

B. Experiment 2: Impact of harvested power and reservoir capacity on responsiveness

We assessed the impact of the harvested power and the reservoir capacity on the aperiodic responsiveness of TB-H. The aperiodic response times on the y-axis are normalized: a value of 1 on the y-axis corresponds to the shortest response time, and a value of zero to the worst response time. Figure 4 illustrates the three-dimensional plots of normalized response time of the TB-H server by sweeping both harvest power and reservoir capacity for $U_e/P_p = 0.8$. Five different storage capacities sweeping from E_{min} to 9*E_{min} are considered.

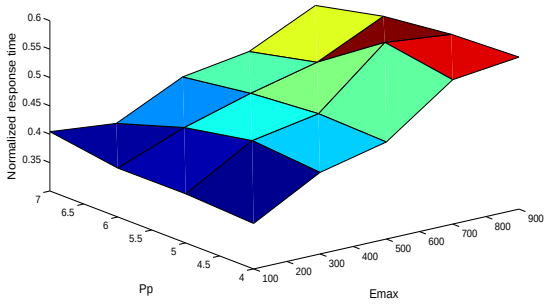

Fig. 4. Impact of storage capacity and harvested energy on Responsiveness under the TB-H server.

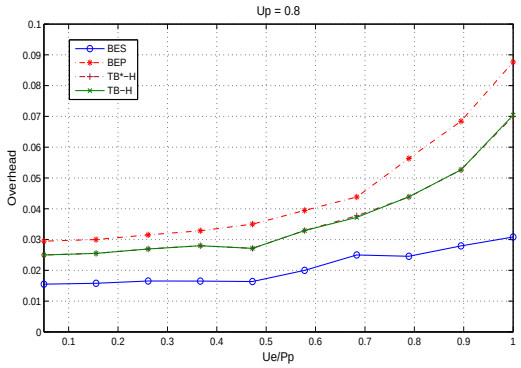

Fig. 5. Overhead for high processing utilization.

Figure 4 shows that, in addition to the total energy utilization, the harvested power P_p and the reservoir capacity E_{min} also affect the performance of TB-H. For weakly energy constrained systems, TB-H exhibits good aperiodic responsiveness. For highly energy constrained system, its performance degrades by at least 20%. In all experiments, the higher the harvested power and/or the reservoir capacity, the better the response time achieved by TB-H. This appears because of the excess of energy introduced by the power or the reservoir. As shown in the plots, the reservoir capacity has the most impact on responsiveness since the improvement in reducing the normalized response time is significant with the increase of the reservoir capacity, e.g. the response time is reduced by 49% when rising from E_{min} to 9*E_{min} and by only 6% when sweeping from P_p to $8 * P_p$ in Figure VI-B.

C. Experiment 3: Time overhead of the different servers

Implementation overheads are evaluated as a function of total processing utilization U_e/P_p. Figure 5 shows the normalized overhead introduced by BEP, BES, TB-H, and $TB^* - H$ algorithms with U_p = 0.8.

The Total Bandwidth servers are more efficient than BEP by 19.5%. The BES server incurs less overhead than all other servers.

VII. Conclusion

Wireless sensors are used for real-time monitoring and control. They implement both aperiodic and periodic tasks. Periodic tasks have regular arrival times and hard deadlines whereas aperiodic tasks have irregular arrival times and no deadline. As a consequence, we have to provide a flexible scheduler able to satisfy deadlines of the periodic tasks and minimal responsiveness for the aperiodic tasks. In this paper, we presented a new family of task schedulers for enhancing aperiodic responsiveness in energy harvesting sensors [12].
The first server, namely TB-H is an extension of the Total Bandwidth server introduced by Spuri and Buttazzo in 1994. It consists first in computing a virtual deadline to any occurring aperiodic task so as to guarantee no deadline violation and no energy starvation and second in inserting the aperiodic task into a single ready queue. A refinement of TB-H called TB^*-H has been proposed. Even if not yet proved, optimality of $TB^* - H$ is intuitive as regard the optimality of the server TB^* stated in 1997 with no consideration of energy. Simulation results bring to light that the new servers TB-H and $TB^* - H$ allow significant reductions in aperiodic response times compared to classical Background approaches. This paper also outlined the advantages of the ED-H scheduler which is the energy harvesting aware version of the famous EDF scheduler.
Scheduling dependent tasks with resource access under energy harvesting settings is currently under study which also shows the potential of the dynamic priority scheduler ED-H for designing flexible self powered sensors.

References

[1] S. Chalasani and J. M. Conrad. *A survey of energy harvesting sources for embedded systems*. In Proceedings of the IEEE Southeastcon 2008, pp. 442447, April 2008.

[2] F. Yildiz. *Potential ambient energy-harvesting sources and techniques*. The Journal of Technology Studies, 35(1), pp. 4048, 2009.

[3] M. Chetto, A. Queudet. *Energy Autonomy of Real Time Systems*, Elsevier, 142 pages, ISBN: 9780081011577, November 2016.

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

[5] C. l. Liu and J. w. Layland. *Scheduling algorithms for multiprogramming in a hard real-time environment*. Journal of the Association for Computing Machinery, 20(1), pp. 4661, 1973.

[6] J. Liu. *Real-Time Systems*, Prentice Hall, United States, 2000.

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

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

[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.

[10] G.-C. Buttazzo and F. Sensini. *Optimal deadline assignment for scheduling soft aperiodic tasks in hard real-time environments*. IEEE Transactions on Software Engineering, pp. 2232,1999.

[11] R. El Osta, M. Chetto, H. El Ghor and R. Hage. *Real-Time Scheduling of aperiodic tasks in Energy Harvesting Devices*. International Conference on Sensors, smart and Emerging Technologies, 12-14 September 2017.

[12] R. EL Osta. Contribution to real time scheduling for energy autonomous systems, PhD thesis, University of Nantes, France, October 26, 2017.