ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/337974318

An Efficient Aperiodic Task Server for Energy Harvesting Real-Time Systems

Conference Paper · November 2019

CITATIONS
0

READS
18

3 authors:

Rola el Osta
University of Nantes
12 PUBLICATIONS **12** CITATIONS
SEE PROFILE

Maryline Chetto
University of Nantes
122 PUBLICATIONS **890** CITATIONS
SEE PROFILE

Hussein El Ghor
Lebanese University
34 PUBLICATIONS **120** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Real time systems View project

Project Real-Time Systems View project

All content following this page was uploaded by Hussein El Ghor on 17 December 2019.

The user has requested enhancement of the downloaded file.

An efficient aperiodic task server for energy harvesting embedded systems

Rola El Osta
LS2N Laboratory - UMR CNRS 6004
University of Nantes
Nantes, France
rolaosta@hotmail.com

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*—The energy existing in our environment can be converted into electricity to supply a wireless device such as sensor node. In this paper, we address the scheduling problem for such a device that executes a mixed set of real-time tasks, composed of aperiodic and hard deadline periodic tasks. High responsiveness of the aperiodic tasks and timeliness of the periodic tasks can be performed through an aperiodic task server that takes into account both time and energy limitations. This paper describes an extension of the well known TBS (Total Bandwidth Server) which is energy harvesting aware. The efficiency of the so-called TB-H aperiodic server is evaluated and compared to background approaches through simulation experiments.**

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

I. Introduction

Energy harvesting, or energy scavenging, is a process that captures small amounts of energy that would otherwise be lost as heat, light, sound, vibration or movement. EH permits to replace batteries for small, low power electronic devices. This technology has several benefits: devices are maintenance free since there is no need to replace batteries. Devices are environmentally friendly since batteries contain chemicals and metals that are harmful to the environment and hazardous to human health. In addition, EH opens up new applications where EH sensors can be deployed in remote or underwater locations [15]. Consequently EH enables us to design autonomous embedded systems which are supplied perpetually. In comparison with energy stored in classical storage units as batteries, the environment represents an infinite source of available energy. Many environmental sources can be exploited to supply autonomous small devices, including solar energy, electromagnetic waves, thermal energy, mechanical, etc. The energy source is selected based on the application characteristics. In this paper, we consider an EH system which is composed of three parts (Figure 1): the processing unit with unique voltage and frequency, the energy harvester and a rechargeable energy storage such as super-capacitor.

Most of wireless sensors implement software which have hard real time constraints. They are provided with a specific operating system called RTOS (Real Time Operating System). The difference between a RTOS and a conventional OS is the response time to external events. OS's typically provide non-deterministic responses. There are no guarantees as to when each task will complete. An RTOS typically provides a hard real time response, providing a fast, highly deterministic reaction to events including the periodic ones from the real-time clock. When switching between tasks the scheduler of the RTOS has to choose the most appropriate task to execute next. There are many possible scheduling algorithms available, including Round Robin, SPT (Shortest Processing Time first), etc. However, to provide a bounded responsive system, most RTOS's use a preemptive priority driven scheduling algorithm.

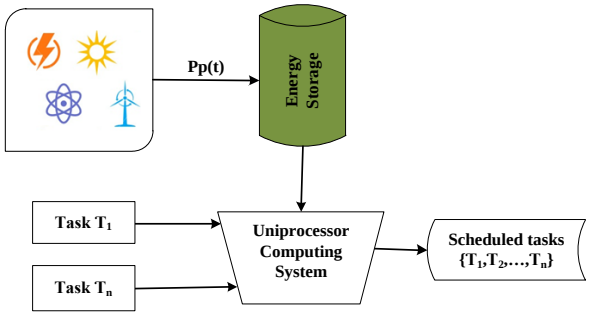

Fig. 1. A Real-Time Energy Harvesting System

In a fixed priority system, each task is given an individual priority value which is constant along time. Under the Rate Monotonic (RM) scheduler, the shorter the task period, the higher the priority level assigned. RM may achieve a 88% processor utilization [8]. In a dynamic priority system, the jobs of a given periodic task have distinct priorities. Earliest Deadline First (EDF) scheduling executes first the job with the closest deadline [9]. EDF is the optimal scheduler and may achieve up to 100% processor utilization while guaranteeing no deadline violation.

Every RTOS which is commercialized now-days uses a non-idling (also said work-conserving) scheduling strategy. If there is at least one task which is pending for execution, the scheduler cannot let the processor in the sleep mode. It systematically executes the highest priority task which is waiting for execution. The most important consideration

when designing a real-time application is what types of timing constraints for the tasks should be considered. Most of tasks are hard real-time ones i.e. they should be executed completely before specified deadlines. if the deadline is not met, this will cause the system to fail. In contrast, any application has soft aperiodic tasks that should be executed with minimal response time. They have no strict deadline to guarantee. In that paper, we consider a real-time software composed of a mixed set of tasks: hard deadline periodic tasks in one hand and soft aperiodic tasks in the other hand.

This paper tackles a central scheduling problem for a hard real-time system which is supplied through energy harvesting from an environmental source. The question is: how to guarantee deadlines of periodic tasks while providing a minimal response time for any occurring aperiodic task with unpredictable arrival time. This scheduling problem has been extensively studied from about the last three decades under the hypothesis of no energy limitation. A survey can be found in [1] [10].

The well known EDF scheduler is preemptive and non idling. It behaves very poorly under energy harvesting considerations because optimal scheduling requires clairvoyance and idling capabilities of the scheduler as proved in [5]. This has motivated additional research works to propose novel efficient schedulers that adapt to energy harvesting settings. In 2014, an idling variant of the EDF, named ED-H was proved to be the optimal one [4]. Optimality of ED-H signifies that any set of hard real-time tasks which is feasible on a given platform, will be feasibly scheduled according to ED-H. The platform is here precisely characterized by given single computing unit, energy harvester with given power production and energy storage unit with given energy capacity as shown in Figure 1.

The contribution of this paper is a new scheduling algorithm that permits to jointly schedule soft aperiodic tasks and hard periodic tasks under energy harvesting constraints. The so-called TB-H (Total Bandwidth with energy Harvesting) server consists of an extended version of the Total Bandwidth server proposed by Spuri and Butazzo [13]. TBS (Total Bandwidth Server) provides optimal responsiveness with very little overhead. However, TBS does not consider energy constraints. According to this approach, a virtual deadline is suitably assigned to every occurring aperiodic task so as to process it as soon as possible. It guarantees no deadline missing for the periodic tasks. All the tasks, periodic and aperiodic ones are jointly scheduled according to the preemptive EDF scheduler. We show here how to modify the TBS scheduler so as to adapt to the energy harvesting context. Each running task is now assumed to consume both processor time and energy. All the tasks, periodic and aperiodic ones, are now scheduled according to the optimal scheduler ED-H. We will demonstrate the efficiency of the new aperiodic task server TB-H through a set of experiments.

The paper is organized as follows. The model under study is presented in the next section. Section III gives background materials. Principles of the TB-H aperiodic task server are described in Section IV. Section V reports the main results of experiments so as to illustrate efficiency of the TB-H server. Finally, Section VI concludes the paper.

II. System Model

We consider a platform composed of energy storage unit, energy harvester and uniprocessing unit as described above. The processor sustains one operating frequency. A four-tuple (C_i, E_i, T_i) is associated with a periodic task τ_i and gives its Worst Case Execution Time (WCET), Worst Case Energy Consumption (WCEC) and period respectively. We assume that E_i is not necessarily proportional to C_i [6]. A job is any request that a task makes. The first job of τ_i is released at time 0 and the subsequent ones at times $kT_i, k = 1, 2, ...$ called release times. H is the least common multiple of the request periods T_i, called the hyper-period. 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.
In addition, we 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 actual finishing time of Ap_i will be denoted by f_i. The energy source is characterized by an instantaneous charging rate $P_p(t)$ that incorporates all losses. We define $E_p(t)$ as the energy produced by such a power source from time 0 to time t. We assume that the energy production times can overlap with the consumption times and the instantaneous power consumed by any task is not less than the instantaneous power drawn from the source. The energy produced on the time interval $[t_1, t_2)$ is denoted $E_p(t_1, t_2)$ while the energy consumed by tasks on the same interval is denoted $E_c(t_1, t_2)$. In our work, we deal with solar energy which is harvested by solar panels and we consider that the energy produced by the source is scavenged from small time slot of the energy harvesting profile in a day i.e. constant energy production power equal to P_p is assumed. Consequently, we may define the energy utilization of τ as $U_{ep} = P_p \times \sum_{\tau_i \epsilon \tau} \frac{E_i}{T_i}$. We assume that U_{ep} is less than or equal to 1. In other terms, the average power consumed by the periodic tasks is less than or equal to the power of the environmental source. We assume that the application is feasible regarding the set of periodic tasks. Consequently, if no aperiodic task occurs, every task τ_i cannot miss its deadline either due to time insufficiency or energy insufficiency.

Our system uses an ideal energy storage unit that has a nominal capacity, namely E. $E(t)$ is defined as the energy level of the system at time t. The stored energy may be used at any time later and does not leak any energy over time. Energy is wasted if the storage is fully charged at time t (i.e. $E(t) = E$) and we continue to charge it. In contrast, no task

can be executed and the application definitively stops if the storage is fully discharged at time t (i.e. $E(t) = 0$). We assume that the energy level of the storage is never increasing every time a task executes.

III. Background Materials

A. ED-H: a variant of Earliest Deadline First

ED-H has been stated as the optimal scheduler to support energy harvesting settings [4]. As EDF, ED-H is a dynamic priority scheduler which selects the next task to execute with the closest deadline. However, ED-H may deliberately postpone the execution of such task in order to avoid energy starvation for future occurring jobs. Consequently, ED-H uses a Dynamic Power Management technique so as to put the processor in the busy mode v.s. the idle mode whenever necessary. At every time instant, the decision depends on two dynamic variables respectively called *slack time* and *preemption slack energy*. The slack time gives the maximum duration of the interval where the processor could be inactive while ensuring no deadline violation. The preemption slack energy gives the maximum energy that could be consumed by the highest priority task while preventing energy starvation for periodic requests that may preempt it.

We are now prepared to present the ED-H scheduler. Let us use the following notations:

- t: current time
- $L_r(t)$: list of periodic jobs ready to be processed
- $A_r(t)$: list of aperiodic jobs ready to be processed
- $E(t)$: residual capacity of the energy reservoir
- $ST(t)$: slack time of the periodic task set
- $SE(t)$: slack energy of the periodic task set
- $PSE(t)$: preemption slack energy of the periodic task set

The operations of the ED-H scheduler are as follows.

- **Rule 1:** The EDF priority order is used to select the future running job in $L_r(t)$.
- **Rule 2:** The processor is imperatively idle in $[t, t+1)$ if $L_r(t) = \emptyset$.
- **Rule 3:** The processor is imperatively idle in $[t, t+1)$ if $L_r(t) \neq \emptyset$ and either $E(t) = 0$ or $PSE(t) = 0$.
- **Rule 4:** The processor is imperatively busy in $[t, t+1)$ if $L_r(t) \neq \emptyset$ and either $E(t) = C$ or $ST(t) = 0$
- **Rule 5:** The processor can equally be idle or busy (a tie breaking rule should be defined) if $L_r(t) \neq \emptyset$, $0 < E(t) < C$, $ST(t) > 0$ and $PSE(t) > 0$.

Optimality of the ED-H scheduler has been established in [4]. If a hard real-time task set is schedulable by any algorithm on a platform composed of given processor, energy harvester and energy reservoir, then it is schedulable using the ED-H algorithm on the same platform.

B. Aperiodic task servers

Let us consider a real-time system that consists of both aperiodic and periodic tasks. Scheduling algorithms for aperiodic tasks have to guarantee the deadlines for the periodic tasks and provide good average response times for the aperiodic tasks even though the aperiodic tasks occur in a non deterministic manner.

The simplest approach for servicing soft aperiodic tasks is background processing. Background servicing of aperiodic tasks occurs whenever the processor is not executing any periodic task and no periodic tasks are pending for execution. If the processor utilization of the periodic task set is high, then the processing times left for background service is low and the responses times of the aperiodic tasks will be prohibitive. In order to reduce the response time of aperiodic tasks, an aperiodic server is used. The aperiodic server is a periodic task which services the aperiodic tasks. Such server is characterized by a period and a fixed execution time called *server capacity*. The server is scheduled with the same algorithm used for the periodic tasks. It serves the aperiodic tasks respecting the range of its capacity. A lot of efficient aperiodic task server algorithms have been developed such as Priority Exchange (PE) and Deferrable Server (DS) algorithms, introduced in [7] to improve aperiodic responsiveness over traditional background and polling approaches. Another aperiodic task servicing approach for dynamic priority systems was proposed in [3]. The so-called EDL (Earliest Deadline Late) algorithm is based on Slack Stealing. It consists in postponing as much as possible the execution of the periodic tasks so as to execute the aperiodic ones as soon as possible.

The Total Bandwith Server (TBS) is a mechanism for servicing aperiodic tasks in conjunction with periodic tasks in a dynamic priority environment using EDF [13] [14]. Every time an aperiodic task arrives, the TBS algorithm assigns a virtual deadline to the aperiodic task. Once the task is assigned the deadline, it is scheduled according to EDF together with the periodic tasks. The virtual deadline is determined based on the server bandwidth which depends on available processor utilization for the aperiodic tasks. An improved version of TBS called TB^* was proved to be optimal [2]. Each aperiodic task gets a shorter virtual deadline than that provided by TBS. Thus, whenever an aperiodic task arrives, the server TB^* first assigns to it a virtual deadline according to the TBS algorithm. This one will then try to shorten this deadline to the maximum so as to improve the response time of the aperiodic tasks without compromising the execution of the periodic tasks. The process of shortening this deadline is applied iteratively until no improvement is possible, while guaranteeing the schedulability of the periodic task set.

C. Illustration of the Total Bandwidth server

The following example shows the sequence which results from the EDF scheduling algorithm and the TBS aperiodic servicing algorithm (see Figures 2). Let us consider two periodic tasks and two aperiodic tasks as imparted in Tables I and II respectively. The periodic tasks are scheduled according to EDF. It is worth noting that the periodic utilization is $U_{pp} = 0.7$ leaving a low bandwidth for the aperiodic tasks. When the aperiodic task Ap_1 arrives at time 9, it receives

a virtual deadline ($d_1 = 13$). As time 13 is the earliest deadline, Ap_1 is immediately serviced. A second aperiodic task Ap_2 arrives at time 18. Identically, it receives a virtual deadline ($d_2 = 28$) and is executed at time 22 as τ_1 with closest deadline is active at this time. We note in Figure 2 that the response time of Ap_1 and Ap_2 is 1 and 7 units of time respectively.

TABLE I
PARAMETERS OF PERIODIC TASKS

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

TABLE II
PARAMETERS OF APERIODIC TASKS

Task	a_i	c_i
Ap_1	9	1
Ap_2	18	3

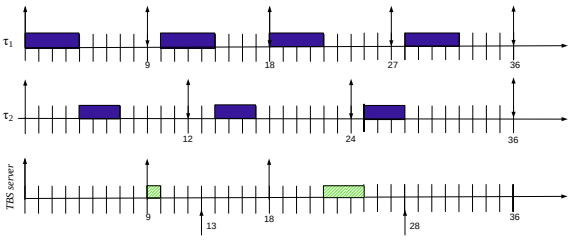

Fig. 2. Illustration of TBS

IV. THE TB-H APERIODIC SERVER

Looking at the characteristics of the Background Server, we observe the following: when the load of the periodic task set is high, the execution of the aperiodic jobs can be delayed significantly. It is natural to ask whether a variant of the TBS algorithm could improve responsiveness over background solutions in energy harvesting applications. The assignment of the virtual deadline for any occurring aperiodic task should take into account energy limitation so as to prevent energy starvation for all the periodic tasks.

A. Definition of TB-H

In this section, we present an extension of TBS for energy harvesting settings. The so-called TB-H (Total Bandwidth for energy harvesting settings) assigns a virtual deadline based on both energy and processing time considerations. Once the aperiodic task is assigned the virtual deadline, it is scheduled by ED-H with periodic tasks, jointly. We assume that the aperiodic tasks are served in FCFS order. Thus no aperiodic task can be preempted by another one. A first virtual deadline d_k is computed according to the processing bandwidth as in TBS ($U_{ps} = 1 - U_{pp}$). Then, a second virtual deadline $\tilde{d_k}$ is computed according to the energy bandwidth ($U_{es} = 1 - U_{ep}$).

Under TBS, the virtual deadline assigned to each aperiodic job guarantees that the fraction of processor demanded by aperiodic tasks never exceeds the processor utilization of the server, U_{ps}. In the same idea, we have demonstrated that the deadline should be assigned so that the fraction of energy consumed by the aperiodic tasks should never exceed the energy utilization of the server, U_{es}.

Let us assume that $U_{ep} + U_{es} \leq 1$. For the k-th aperiodic task that arrives at time $t = r_k$ a virtual deadline is computed as follows:

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

Finally, the deadline assignment process under energy harvesting settings is given in theorem 4.1.

Theorem 4.1: [12] Given a set of n periodic tasks and a stream of m aperiodic tasks served by TB-H, the virtual deadline of the aperiodic task Ap_k is computed as follows:

$$d_k^f = max(d_k, \tilde{d_k}) \quad (2)$$

We proved in [12] that no periodic task can miss a deadline using the TB-H aperiodic task server. The scheduling framework of the TB-H server can be described by the following pseudo-code:

Algorithm 1 ED-H with the Total Bandwidth server

```
Require:
  t: current time
  Ap_k: Aperiodic task that occurs at t
  while True do
    if Ap(t) is not empty then
      d_k = max(t, d_{k-1}) + ⌈c_k/U_s⌉
      d̃_k = max(t, d_{k-1}) + ⌈e_k/P_p.U_es⌉
      d_k^f ← max(d_k, d̃_k)
      assign d_k^f to Ap_k
      insert Ap_k in the ready queue
    end if
    execute the ED-H scheduler
  end while
```

Overheads of the TB-H server are clearly the cost for computing the virtual deadline whenever one aperiodic task occurs. As with ED-H, the RTOS keeps a ready queue, ordered by absolute deadline of all the uncompleted periodic or aperiodic tasks pending for execution.

B. Illustration of TB-H

The following example illustrates the TB-H deadline assignment. We consider the set of periodic and aperiodic tasks given

in the previous section. The periodic processor utilization is $U_{pp} = 0.7$. The periodic energy utilization is $U_{ep} = 0.875$, which leads to available processor utilization, $U_{ps} = 0.3$ and available energy utilization $U_{es} = 0.125$ for the aperiodic tasks. The energy production power is constant with $Pp = 4$.

At time 0, the residual capacity is maximum since the storage unit is fully replenished. τ_1 with the highest priority, runs and finishes at time 4, consuming 18 energy units. At time 4, the residual capacity is given by $E_{max} - E_1 + P_p * C_1 = 8$. Now, τ_2 gets the highest priority. It executes completely until time 7 and consumes 18 energy units. The residual capacity equals 2 energy units. From time 7, the battery is recharging as the processor is idle. Ap_1 arrives at time 9. $d_1 = 13$ and $\tilde{d_1} = 17$. The virtual deadline $d_1^f = max(d_1, \tilde{d_1}) = 17$ is assigned to Ap_1. As 17 is the earliest deadline, the aperiodic job is executed immediately and consumes a maximum of 5 energy units. At time 10, the highest priority task τ_1 executes completely according ED-H where the residual capacity equals 7. Tasks execute till time 18 where Ap_2 occurs. Ap_2 receives a virtual deadline $d_2^f = 47$. However, Ap_2 does not start execution immediately, since there is a ready periodic task with shorter deadline equal to 27. Tasks are executed according to ED-H until the end of the hyperperiod where the energy reservoir contains 8 energy units.

Let us note that the response times of the aperiodic tasks Ap_1 and Ap_2 are respectively, 1 (which is the optimal value) and 16 units of time.

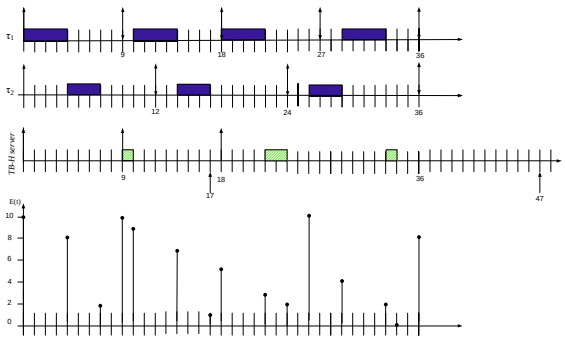

Fig. 3. Tasks scheduled according to TB-H

V. Performance Evaluation

This section presents a set of experiments carried out to evaluate the performance of the new proposed algorithms TB-H for different configuration parameters. Our performance evaluation is also compared with three algorithms with different performances and implementation overheads: Background with Energy Surplus (BES) and Background with Energy Preserving (BEP). BES serves aperiodic tasks when no periodic task is present in the system and when the energy reservoir is fully replenished. Under BEP, the enhanced version of BES, aperiodic task is authorized to execute when its execution does not involve energy shortage for all future occurring periodic tasks. It is worth noting that periodic tasks are scheduled according to the ED-H rules. In all the experiments, the performance of the various algorithms is evaluated by measuring the average aperiodic response time normalized with respect to the computation time. The average is computed over 100 runs, in which a total of 15000 aperiodic jobs are generated. We assume that the storage capacity is initially full and the recharging power P_p is constant.

A. Task Set Generation

In all simulations, a set of 30 periodic tasks with periods, execution times and energy consumptions are randomly generated. Periods and computation times are distributed uniformly in discrete time steps, depending on U_p. Energy consumption of every task is proportional to its period and depends on the setting of U_e. Periodic task sets are assumed to be feasible. The aperiodic load is made varying across the margin of processor utilization left unused by periodic tasks. They are generated according to desired values for U_{ps} and U_{es} by modelling a poisson aperiodic arrival. The aperiodic load is denoted by U_{ps}. Throughout our simulation results, 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}.

B. Relative performance under various time and energy conditions

In this first set of experiments, TB-H, BEP and BES algorithms have been simulated to compare the average response times of soft aperiodic tasks with respect to the total energy load.

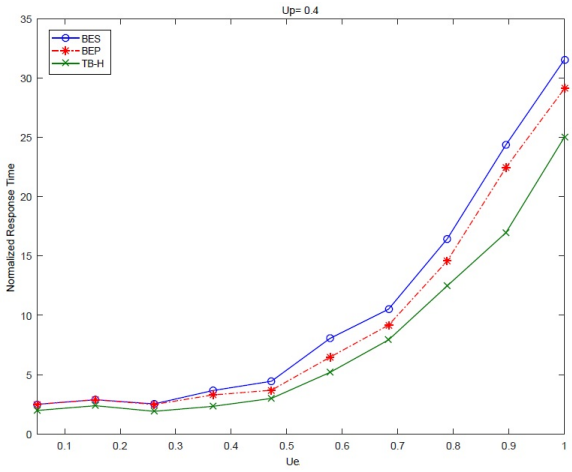

Fig. 4. Normalized aperiodic response time with respect to U_e, for U_p=0.4.

Simulation results reported in Figure 4 are carried out for a processing load equal to 0.4 varying the energy load ($5\% \leq U_e \leq 100$). From the graphs, the TB-H server clearly offers better performance compared to the two Background servers. Moreover, this advantage is more significant as the energy load U_e is higher.

C. Relative performance with different reservoir sizes

In this set of experiments, we evaluate the performance of the servers by varying the reservoir size with E_{min}, 5*E_{min}, and 9*E_{min}. E_{min} is the minimum size of the reservoir that guarantees time and energy feasibility, for given U_p, U_e and P_p. Here, we report the results for systems which are lightly time-constrained with $U_p = 0.2$. The 3rd, 4th and 5th columns of Table III give the aperiodic responsiveness of the BEP, BES, and TB-H servers, respectively, for two profiles in terms of energy constraints. Table III shows that the Total bandwidth server achieves significant reduction in aperiodic responsiveness, compared to the Background servers under all parameter settings.

TABLE III
RELATIVE PERFORMANCE WITH DIFFERENT RESERVOIR SIZES

Capacity	U_e/P_p	BES	BEP	TB-H
E_{min}	0.2	2.4	2.1	1.8
	0.8	37.4	35.2	29.0
$5 * E_{min}$	0.2	2.0	1.7	1.5
	0.8	23.0	15.8	14.9
$9 * E_{min}$	0.2	1.5	1.3	1.2
	0.8	13.4	8.7	7.3

When the system uses 20% of available energy with minimum reservoir size, the response time under TB-H is 14% and 25% lower compared to BEP and BES respectively. If the energy requirement is set to 80%, all servers have similar high response times.

For each of the three strategies, higher is the size of the reservoir, lower is the normalized aperiodic response time for a given energy setting. For example, if the reservoir size is set to E_{min} and the system consumes up to 80% of available energy, the TB-H server has aperiodic response time equal to 29.0. When increasing the reservoir size to $9 * E_{min}$, the response time is reduced by 75%. Such a significant improvement in aperiodic responsiveness comes from immediate service which is made possible by TB-H and extra energy available in the reservoir.

VI. DISCUSSION AND CONCLUSIONS

This paper has presented a scheduling technique for reducing the response time of aperiodic tasks which runs on a processor supplied by environmental energy through an harvester such as solar cell. In the model, each running task consumes both processor time and energy. We proposed an algorithm which is an extension of the TB server that initially did not consider energy limitation. The TB-H server consists in assigning a virtual deadline to each newly occurring aperiodic task based on both processing and energy constraints. This permits to execute any aperiodic task with good response time while not jeopardizing the schedulability of the periodic tasks. Compared to basic background approaches, TB-H appears as an interesting energy aware aperiodic task server looking at both responsiveness and implementation overheads.

REFERENCES

[1] G. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications 2nd edn.*, Springer, Berlin, 2005.

[2] G.C. Buttazo and F. Sensini. *Optimal Deadline Assignment for Scheduling Soft Aperiodic Tasks in Hard Real-Time Enviroments*, Proceedings of the 3rd IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'97), Como, Italy, pp. 39-48, September 1997.

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

[4] M. Chetto. *Optimal Scheduling for Real-Time Jobs in Energy Harvesting Computing Systems*, IEEE Transactions on Emerging Topics in Computing, DOI.ieeecomputersociety.org/10.1109/TETC.2013.2296537, Vol. 2, No. 2, pp. 122-133, 2014.

[5] M. Chetto, A. Queudet, *A Note on EDF Scheduling for Real-Time Energy Harvesting Systems*, IEEE Transactions on Computers, 63(4): 1037-1040, April 2014.

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

[7] J. P. Lehoczky, L. Sha, J. K. Strosnider, *Enhanced Aperiodic Responsiveness in Hard Real-Time Environments*, in: Proceedings of theReal-TimeSystems Symposium, IEEE Computer Society, San jose, California, ISBN0-8186-0815-3, 110123, 1987.

[8] J.P. Lehoczky, L. Sha., and Y. Ding. *The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behaviour*, Proc. of Real-Time Systems Symposium, pp. 166-171, 1989.

[9] C.-L. Liu, J.-W. Layland. *Scheduling algorithms for multiprogramming in a hard real-time environment*, Journal of the Association for Computing Machinery, Vol. 20, No. 1, pp. 46-61, 1973.

[10] J. Liu. *Real-Time Systems*. Prentice Hall, 2000.

[11] 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.

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

[13] M. Spuri, and G. Buttazzo. *Scheduling aperiodic tasks in dynamic priority systems*, Real-Time Systems, Vol. 10, No. 2 , pp. 179-210, March 1996.

[14] M. Spuri and G. C. Buttazzo. *Efficient aperiodic service under earliest deadline scheduling*, Proc. of Real-Time Systems Symposium, pp. 2-11, 1994.

[15] T. tanaka, T. Suzuki and K.Kurihara. *Energy Harvesting Technology for Maintenance-free Sensors*, FUJITSU Sci. Tec.J., Vol. 50, No. 1, January 2014.

View publication stats