Online Admission of Hard Aperiodic Tasks in Real-Time Energy Harvesting Systems

Rola El Osta, Maryline Chetto

Online Admission of Hard Aperiodic Tasks in Real-Time Energy Harvesting Systems

Rola El Osta
LENS Laboratory
Lebanese University
Saida, Lebanon
rola.elosta@ul.edu.lb

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

***Abstract*—It is expected that energy harvesting technology will increasingly be deployed to realize autonomous embedded real-time systems such as wireless sensors. In this paper, we consider the problem of jointly scheduling periodic tasks and dynamically arriving hard deadline aperiodic tasks in a monoprocessor system powered with regenerative energy. Our result is an exact admission algorithm in that sense that it accepts every schedulable task based on the amount of available energy and computing capacity. It is assumed that all the accepted tasks are preemptively scheduled according to the optimal algorithm, ED-H which is an energy aware extension of Earliest Deadline First. We also discuss issues concerning the implementation of the admission test.**

***Index Terms*—Real-time system, Energy harvesting, Schedulability analysis, Online scheduling, Aperiodic task, Time laxity, Energy laxity**

I. Introduction

Nowadays, batteries are the dominant source of energy for small electronic devices such as wireless sensor nodes. However, besides their impact on the environment, their limited energy storage capacity and their finite lifetime make their use restricted. For these reasons, alternative energy sources have been sought. Energy harvesting (EH), or energy scavenging, is a technology that captures unused ambient energy and convert it into usable electrical energy that can be used immediately or later thanks to a storage unit [1]. Generally, we use this terminology for low power and small autonomous devices such as wireless sensor networks and portable electronic equipments. Hence, ambient energy scavenging has engendered a lot of interest for researchers. It permits that autonomous embedded systems be supplied perpetually.

In comparison with energy stored in classical storage unit as batteries, the environment represents an infinite source of available energy. Many environmental sources can be exploited, including solar energy, electromagnetic waves, thermal energy, mechanical, etc. and must be selected based on the application characteristics. An EH system consists of three components (see Fig. 1): a single processing unit, an energy harvester and a rechargeable energy storage. The energy harvester scavenges energy from the environment and converts it to usable electrical power. The rechargeable energy storage (e.g battery or capacitor) stores energy. And after that, the processor unit uses this energy to execute the programs.

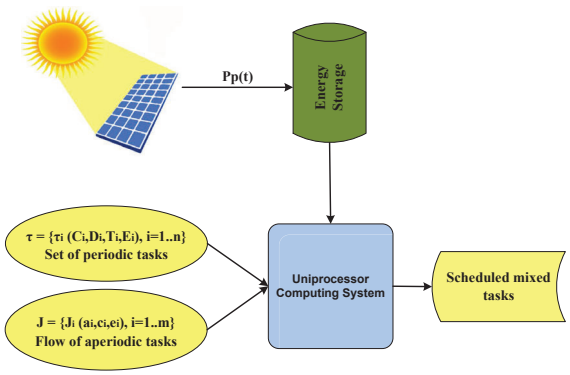

Fig. 1. A typical Real-Time Energy Harvesting System

In EH systems, we have to exploit perpetually the available ambient energy which is strongly dependent on the environment. The consumed energy of the system should be adapted to maximize the performance instead of minimizing it as in the classical battery powered systems. So, the operating system has to manage properly the activity of the processing unit such that there is sufficient energy in the storage unit to satisfy all the constraints at every time. Most environmental energy sources like the electromagnetic waves, do not deliver a constant power over time. The energy generated may arrive in bursts and has to be stored so that the device can still be operated at a later moment. The main challenge for an EH system lies in the optimization of its performance while respecting the time-varying amount of energy without wasting or exhausting the energy stored in the battery or capacitor. We then say that the system operates in an energy neutral mode by consuming only as much energy as harvested [2].

Most of real-time applications consist of a set of hard periodic tasks. Hard aperiodic tasks can arrive at a certain time from alert conditions or from failures of hard periodic tasks which fail to check and validate their results and must be retried and completed before the original deadline [3]. Unlike the problem of scheduling periodic tasks only, the joint scheduling problem is difficult because the scheduler

has to choose which tasks to accept for processing when not all of the timing requirements can be met due to processing overload or energy shortage. The operating system should be provided with a procedure called acceptance test which dynamically decides whether a newly arriving aperiodic task could be accepted [4]. As aperiodic tasks are not known before run-time, the acceptance test is performed online and at unpredictable times so as to guarantee the deadlines of all periodic tasks and any already accepted. If a hard aperiodic task cannot be guaranteed, it is rejected.
The mechanism of the acceptance test was also presented in [5] based on the the APS (Alternative Priority Algorithm).
In this paper, we specifically focus on hard deadline periodic tasks with hard deadline aperiodic tasks that execute on a uniprocessor platform which is powered by regenerative energy. We present the theoretical basis of our approach to solve the acceptance problem. This method is based on the assumption that all periodic and aperiodic tasks are scheduled according to the optimal algorithm ED-H [6] which is an energy aware extension of Earliest Deadline First [7]. We will answer to the acceptance question by calculating, for each newly arriving aperiodic task, two key values called time laxity and energy laxity. The time laxity is a measure of processing time surplus. The energy laxity is a measure of energy surplus. Both will be assessed and available after accepting the task. Clearly, these two values should be non negative to admit the newly arriving task.
The remainder of the paper is organized as follows. Section II presents background materials about Earliest Deadline First scheduling and online admission of aperiodic tasks with no energy considerations. Section III introduces the model and terminology relative to the scheduling issue in real time energy harvesting systems.Section IV describes the ED-H scheduler. Our admission test is presented in section V. In Section VI, we give some suggestions for an efficient implementation of the test. Section VII summarizes the paper and describes future work.

II. Background material

Hereafter, we give necessary background about scheduling real-time tasks considering that energy is unlimited. We only focus on preemptive scheduling for deadline constrained and independent tasks.

A. Scheduling periodic tasks with EDF

The most common dynamic priority scheduling algorithm for real-time systems is the Earliest Deadline First (EDF) which was introduced by Liu and Layland [7] in 1973. EDF is clearly a dynamic priority driven scheduler since closer is the deadline, higher is the priority. The EDF algorithm has been proven by Dertouzos to be optimal among all scheduling algorithms on a uniprocessor in the sense that if a task set cannot be feasibly scheduled by EDF, then this task set cannot be feasibly scheduled by any other algorithm [8]. The classical implementation of EDF consists in executing the tasks under the ASAP (As Soon As Possible) version. We then say that EDF is greedy, consuming the available energy whenever at least one task is pending for execution and ignoring the future energy requirements. Let us notice that the ALAP (As Late As Possible) version of EDF is no more appropriate to schedule deadline constrained tasks in energy harvesting systems.

B. Admission test for aperiodic tasks and EDF

Most of real-time software consist of a set of hard periodic tasks with aperiodic tasks that may be authorized for execution as long as they do not cause any periodic task to violate its deadline. A hard aperiodic task consists of one computation that responds to an internal or external event. The admission test considers the processing capacity required by future periodic invocations and aperiodic tasks. The optimal admission test is based on the exact computation of the largest amount of processor idle time available during the interval between the arrival time and the deadline to complete the processing, while still ensuring that all accepted tasks meet their deadline. The fact that the utilization of the periodic tasks is less than 1 means that over the hyperperiod given by the least common multiple of the periods they are a positive number of slack time units remaining to be used by aperiodic tasks. However, to guarantee the deadlines of the periodic tasks while executing the aperiodic tasks as soon as possible before deadlines, this slack should be distributed adequately. In that objective, we have to exploit the central property of the earliest-deadline-late (EDL) algorithm which schedules periodic tasks as late as possible. Whenever a hard aperiodic task occurs, the idle times of the EDL schedule are computed and compared to the execution requirements of the aperiodic tasks not yet completed, checking that the aperiodic tasks will not use more idle time than available. The optimality of this test was stated in [9]. Nonetheless, such a test considers that there is no limitation on energy availability and should be revisited to take into account, in one part the energy required by any task in execution and in the other part both the profile of the energy source and the size of the energy storage unit.

III. Model under study

The Energy Harvesting model (hereafter, referred to as RTEH) consists of a computing element, a set of tasks, an energy storage unit, an energy harvesting unit and an energy source.
We consider a set of real-time tasks that are executed on a uniprocessor system that supports only one operating frequency and that consumes negligible energy in the idle state when it does not execute any task. 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, a relative deadline D_i, a period T_i (with $D_i \leq T_i$) and a Worst Case Energy Consumption of E_i energy units. We assume that E_i is not necessarily proportional to C_i [10]. Each task gives rise to an infinite sequence of invocation requests that starts at the time origin, here equal to zero. Let H, the hyperperiod, be equal to the least common multiple of the periods T_1, T_2,. . . ,T_n.

Let t_c denote the current time.
We next introduce the aperiodic tasks. It is assumed that there is no a priori knowledge about which set of aperiodic task requests will be encountered. Furthermore, it is supposed that any aperiodic task is ready to execute as soon as it arrives and does not interact with periodic tasks in the sense that it does not share resources. Let t_c the current time that coincides with the arrival of a new aperiodic task. Let $J = \{J_i | 1 \leq i \leq m\}$ the stream of m hard aperiodic tasks already admitted at t_c. Every uncompleted aperiodic task at t_c is characterized by c_i, e_i and d_i that respectively denote its remaining execution time, remaining energy consumption and absolute deadline. In what follows, $J_c(c, e, d)$ denotes the arriving aperiodic task to be tested.

Let $P_p(t)$ be the instantaneous charging rate produced by the power source that incorporates all losses. The energy produced on $[t_1, t_2)$ is given as $E_p(t_1, t_2) = \int_{t_1}^{t_2} P_p(t)dt$. We assume that the energy produced in any unit of time can overlap with the energy consumed in one unit of time. We also assume that a short term prediction of the energy harvesting rate is possible. Our system uses an ideal energy storage unit with a nominal capacity C of units of energy. The energy level at time t is denoted $E(t)$. The stored energy may be used at any time later and does not leak any energy over time. Finally, we also suppose that the available average power from the energy source is sufficient to support the long term power demand of the periodic task set τ.

IV. The energy aware ED-H scheduling algorithm

A. Principles

In [6], the author has introduced a novel energy-aware scheduling algorithm, namely ED-H. ED-H is an energy harvesting aware variant of EDF, proved to be optimal for real time tasks scheduling under energy harvesting settings. ED-H answers the question: when should the processing unit use energy to execute a deadline constrained task, and when should the processing unit idle and recharge the energy storage unit. ED-H allows a task to execute only if its execution cannot cause energy starvation for a future occurring task. In that sense, ED-H is a clairvoyant algorithm since its decision necessarily lies on the short term prediction of the incoming energy and the pattern of future task arrivals.

B. Implementation

The implementation of ED-H needs the online computation of the following values:

- *The slack time of the task set τ at current time t*, denoted $ST_\tau(t)$ and defined as the maximum continuous processor idle time that could be made available from time t while still guaranteeing the feasibility of all the tasks in the set τ. Whenever the energy storage unit is completely deplenished, the system enters the idle state so as to recharge the energy storage during at most $ST_\tau(t)$ unit of time.
- *The slack energy of the task set τ at current time t*, denoted $SE_\tau(t)$ and defined as the maximum energy that could be made available and consumed from time t while still guaranteeing the feasibility of all the task set τ. Clearly, no energy should be wasted so as to guarantee that $SE_\tau(t)$ be always non negative.
- *The preemption slack energy of the task set τ at current time t*, denoted $PSE_\tau(t)$ and defined as the maximum energy that could be consumed from time t by the current active task while still guaranteeing the feasibility of all the future tasks with a higher priority i.e. the tasks that may preempt the current active one.

C. Illustrative Example

Consider a periodic task set τ that is composed of three periodic tasks. $\tau = \{\tau_i \mid 1 \leq i \leq 3 \quad and \ \tau_i = (C_i, D_i, T_i, E_i)\}$. Let $\tau_1 = (1, 5, 6, 12)$, $\tau_2 = (2, 8, 10, 15)$ and $\tau_3 = (4, 11, 15, 22)$. We assume that the energy storage capacity is $E = 40$. For simplicity, the rechargeable power, P_p , is constant along time and equals 5.
First of all, we have to schedule the periodic task set τ according to ED-H. We verify that τ is schedulable since all the tasks are executed before their deadline and without depleting the energy reservoir (Fig. 2). At time $t = 0$, all tasks are released. τ_1 is the highest priority task and is executed until $t = 1$ where $E(1) = 33$ energy units. At time $t = 1$, τ_2 is the highest priority task and is executed until $t = 3$ where $E(3) = 28$ energy units. τ_3 is now the highest priority task and is executed until $t = 7$ where the energy storage capacity becomes 26 energy units. At time $t = 7$, τ_1 as the highest priority task, ready to be processed, runs and finishes at $t = 8$. $E(8) = 19$ energy units.
From time $t = 8$ up to $t = 10$, the processor remains idle because there are no pending tasks. During that time interval, the energy storage will recharge and the energy level at $t = 10$ is given by $E(10) = 29$ energy units.
At $t = 10$, τ_2 the highest priority task, is ready to be processed. Here, computation of the slack energy is necessary because task τ_1 with deadline less than deadline of τ_2 that will be released after time $t = 10$. $SE_{\tau_1}(t = 10) = E(10) + \int_{10}^{17} P_p dt - E_1 - E_2 = 37 > 0$.

As $E(10) = 29$ and as the slack energy is greater than zero, τ_2 is authorized to execute immediately. At time $t = 12$, $E(12) = 24$. Now, τ_1 has the highest priority. It is executed till $t = 13$, where $E(13) = 17$ energy units. Again, the processor is idle from $t = 13$ up to $t = 15$ where the energy reservoir will recharge leading to $E(15) = 27$ energy units.
At $t = 15$, τ_3 is ready and has the highest priority. Slack energy is requires to be computed: $SE_{\tau_3}(t = 15) = E(15) + \int_{15}^{23} P_p dt - E_1 - E_3 = 33 > 0$. Consequently, the slack energy is 33 energy units. τ_3 is executed till $t = 18$, where preemption occurs since τ_1 has a higher priority. At $t = 18$, we can evaluate the maximum energy that remains to be consumed by τ_3 as follows: we memorize the energy level of the storage

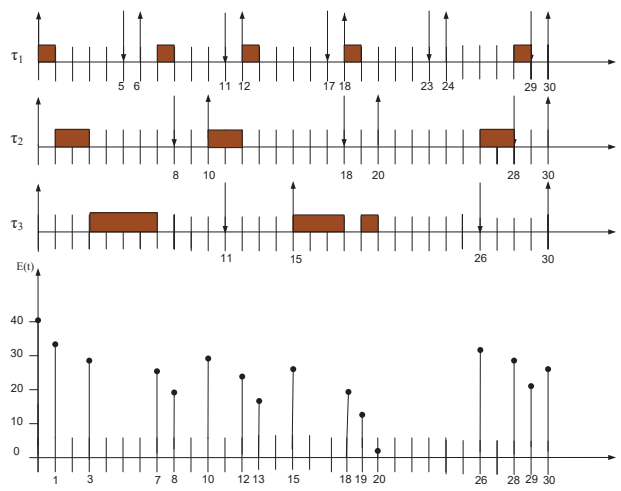

Fig. 2. ED-H scheduling on a periodic task set

unit at every scheduling point i.e. whenever a preemption occurs or a task begins execution. Thus, we deduce that: $E_3^{rem} = E_3 - (E(15) - E(18)) = 15$.

At $t = 18$, τ_1 is the highest priority task ready to be processed, runs and finishes at $t = 19$. $E(19) = 13$ energy units. τ_3 will now resume its execution until $t = 20$. At $t = 20$, τ_2 is the highest priority task ready to be processed and $E(20) = 3$. But there is no sufficient energy in the energy storage unit for execution. So, we have to insert an idle time to let the processor inactive as long as the energy storage has not filled completely and the latest start time of the next periodic task has not been attained. The slack time is equal to 6. Hence, the processor is let idle till $t = 26$ permitting the energy reservoir to recharge so as $E(26) = 33$ energy units. τ_2 can be executed until $t = 28$, where $E(28) = 28$ energy units. At $t = 28$, τ_1 is the highest priority task ready to be processed. It executes and finishes at $t = 29$ with $E(29) = 21$ energy units. All invocations of periodic tasks are feasibly scheduled until the end of the hyperperiod where the energy reservoir contains 26 energy units.

V. Acceptance Test

This section is concerned with the dynamic acceptance test that answers the following decision question relative to a given arriving aperiodic task J_c: "Can J_c be accepted?". The approach is based on the assumption that all periodic and aperiodic tasks are executed according to the optimal scheduling algorithm ED-H. The main principle of the admission test 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 and the previously accepted aperiodic tasks. 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). The basic idea of our acceptance test is to determine both processing time and energy that could be stolen for the newly arriving task. In order to provide an exact admission test, we will separate the constraints in timing and energy domains.

A. Time schedulability analysis

Let $J_c(c, e, d)$ be the arriving aperiodic task. The acceptance test of J_c is executed once at the time of its arrival, t_c. The idea is to postpone the execution of periodic activities, making any spare processing time available as soon as possible for the aperiodic activities. Our approach is based on the exact computation of the largest amount of processor idle time available during the interval between the arrival time and the deadline to complete the processing, while still ensuring that all guaranteed tasks meet their deadline. Consequently, when J_c occurs, the idle times of the EDL schedule are computed and compared to the execution requirements of the aperiodic tasks not yet completed [9].

Let us introduce $\delta_i^t(t_c)$ called the dynamic *time laxity* of task J_i at time t_c. By definition, $\delta_i^t(t_c)$ gives the highest quantity of processing time that could be consumed between t_c and d_i by aperiodic tasks with priority higher than or equal to J_i while guaranteeing the deadlines of all the tasks. Let $\Omega^{EDF}_{\tau_{(t_c)}}(t_c, d_i)$ be the processing time which may be stolen from the periodic task set between t_c and d_i. As proved in [9] it is derived from the EDL schedule produced at current time t_c. Consequently, we have $\delta_i^t(t_c) = \Omega_\tau^{EDF}(t_c, d_i) - \Sigma_{j=1}^i c_j$. We may draw the following necessary condition for the admission test [11]:

Lemma 1: Aperiodic task J_c is accepted only if for every aperiodic task J_i such that $d_i \geq d$,

$$\delta_i^t(t_c) \geq 0 \tag{1}$$

Note that $\Sigma_{j=1}^i c_j$ gives the total remaining execution time of all the aperiodic tasks with a priority higher than J_i.

The value of $\Omega_\tau^{EDF}(t_c, d_i)$ is dynamically computed from two arrays which respectively give the start times and the durations of all idle time intervals in the EDL schedule starting at current time t_c and finishing at the end of the current hyperperiod.

B. Energy schedulability analysis

Let us introduce $\delta_i^e(t_c)$ the *energy laxity* of task J_i at time t_c. $\delta_i^e(t_c)$ gives the maximal energy which may be stolen from the periodic task set between t_c and d_i and consumed by aperiodic tasks with priority higher than or equal to J_i. Let $\Pi^{EDF}_{\tau_{(t_c)}}(t_c, d_i)$ give the maximal energy which remain after executing the periodic tasks between t_c and d_i. Consequently, $\delta_i^e(t_c) = \Pi_\tau^{EDF}(t_c, d_i) - \Sigma_{j=1}^i e_j$. $\Pi^{EDF}_{\tau_{(t_c)}}(t_c, d_i)$ is computed through the following equation:

$$\Pi_\tau^{EDF}(t_c, d_i) = E(t_c) + E_s(t_c, d_i) - g(t_c, d_i) \tag{2}$$

where

- $E(t_c)$ is the residual capacity of the energy reservoir at t_c,

- $E_s(t_c, d_i)$ is the total energy produced by the environmental source between t_c and d_i,
- $g(t_c, d_i) = \sum_{j=1}^{n} \lceil \frac{d_i - t_c}{T_j} \rceil E_j + \sum_{k=1(d_k > d_i)}^{n} E_k(t)$ and refers to the energy demand issued from the periodic tasks between t_c and d_i. Here, $E_k(t)$ denotes the remaining amount of energy of the current invocation of the periodic task τ_k at t_c.

We may draw the following necessary condition for the admission test [11]:

Lemma 2: Aperiodic task J_c is accepted only if for every aperiodic task J_i such that $d_i \geq d$,

$$\delta_i^e(t_c) \geq 0 \tag{3}$$

C. Exact admission test

From the previous lemmas, we prove the following exact admission test [11]:

Theorem 1: Aperiodic task J_c is accepted if and only if for every aperiodic task J_i such that $d_i \geq d$,

$$\delta_i^t(t_c) \geq 0 \text{ and } \delta_i^e(t_c) \geq 0 \tag{4}$$

The purpose of this admission test is to accept or reject any arriving task based on first, the amount of processing time and energy required for its execution and second, the amount of processing time and energy available. Theorem 1 tells us that energy and time constraints can be decoupled if we have to decide whether a newly arriving task is admitted or not.

D. Illustrative example

Consider the previous periodic task set τ and suppose now that some aperiodic tasks arrive to be jointly executed with τ. Let us consider $J = \{J_j \mid J_j = (a_i, c_i, d_j, e_j)\}$ the stream of 3 aperiodic tasks where $J_1 = (7, 4, 11, 17)$, $J_2 = (18, 3, 26, 20)$ and $J_3 = (20, 2, 28, 10)$. Periodic task set τ is scheduled according to ED-H till $t = 7$ where $E(7) = 26$ energy units (Fig. 3). At time 7, J_1 is released. The admission test requires to compute energy laxity and time laxity of J_1 at $t = 7$. $\delta_1^e(7) = E(7) + \int_7^{11} P_p dt - E_1 - E_2 - e_1 = 7 > 0$ and $\delta_1^t(7) = 3 - 4 = -1 < 0$. As the slack time is less than zero, from lemma 1, J_1 is rejected and not authorized for execution.

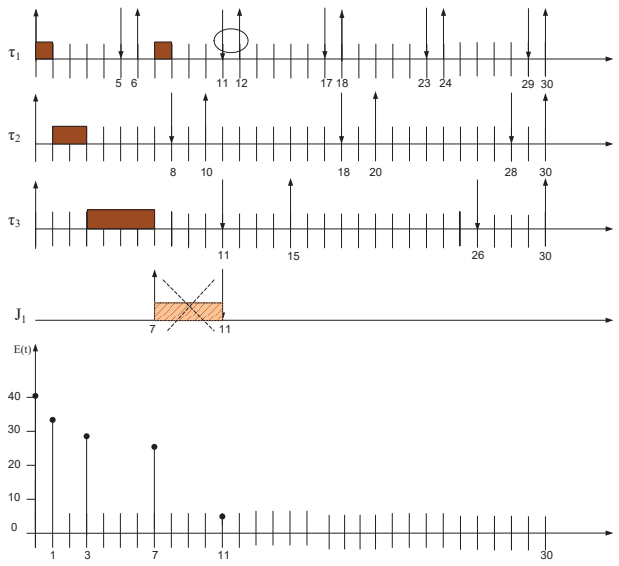

Fig. 3. Rejection of J_1 due to time starvation; $\delta_1^t(7) = -1 < 0$

J_2 arrives at $t = 18$ (Fig. 4). J_2 is rejected since $\delta_2^e(18) = -2 < 0$.

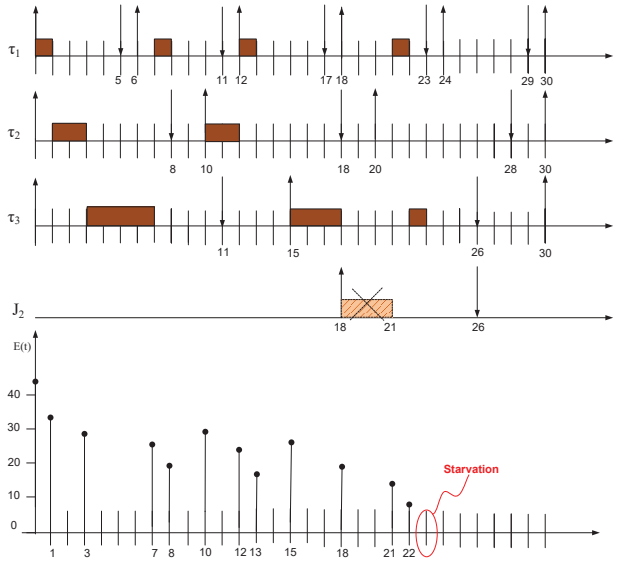

Fig. 4. Rejection of J_2 due to energy starvation; $\delta_2^e(18) = -2 < 0$

At $t = 20$, J_3 arrives (Fig. 5). $\delta_3^e(20) = 6$ and $\delta_3^t(20) = 4$. As both the time laxity and energy laxity of J_3 are positive, the test leads to admit J_3 . As the absolute deadline of J_3 is 28, J_3 is the highest priority task in the ready list. At $t = 20$, we have $E(t) = 3$ and $PSE(t) = 18$. J_3 may execute immediately to be finished at $t = 22$.

τ_2 is the highest priority task ready for execution at $t = 22$. Nonetheless, there is no sufficient energy in the energy storage unit for execution since $E(t) = 3$. The energy storage unit should be recharged. The slack time has to be computed in order to determine the maximal possible idle time. We have $ST(22) = 4$. The processor is let idle as long as the energy storage has not filled completely and for at most 4 time units. The energy storage recharges up to $t = 26$ where $E(26) = 23$ energy units. Finally, τ is scheduled according to the ED-H algorithm till the end of the hyperperiod where $E(30) = 16$ energy units.

VI. Implementation aspects

Clearly, the performance of our admission test is strongly dependent on the accuracy of the predicted power of the harvesting unit. Every time a new aperiodic task occurs,

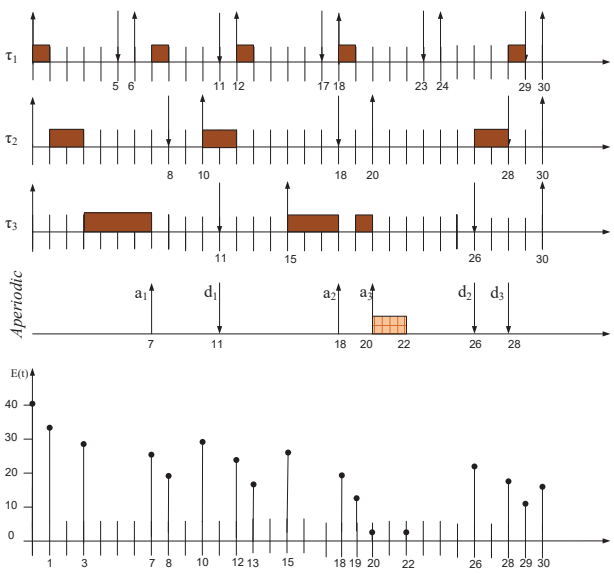

Fig. 5. Admission of J3; $\delta_3^e(20) = 6 > 0$ and $\delta_3^t(20) = 4 > 0$

computation of its energy laxity requires to estimate future incoming energy. The better the approximation, the better the acceptance algorithm performs in terms of exactness. During short time intervals, the produced power can be considered as constant. Let us consider an energy harvester composed of a solar panel in order to draw energy from outdoor light. A sudden change of the light intensity is improbable during a time interval with a duration in the order of the second. Let us reminder that a real time periodic task generally consists in sampling so as to control a physical process. As a consequence, we may consider that the incoming power is constant over time. This simplifies online computations since sophisticated prediction methods may not be necessary.

There are mainly two techniques to store energy in a small device such as a wireless sensor node. The most common one uses rechargeable batteries. The second technique uses supercapacitors or ultra-capacitors which support higher power flows than batteries. In addition, they offer very long lifetimes with very high number of charge/recharge cycles without performance degradation.

An important feature of energy harvesting systems is the capability to estimate the remaining energy in the storage unit. Unlike batteries, the energy of supercapacitors can be measured in a straightforward way. So, we may assume that at every time instant, the operating system may determine precisely the energy level in the storage unit and the energy harvested on a short term time interval in order to compute the energy laxity with accuracy according to equation 3. It follows that the implementation complexity of the acceptance test is O(k.n + m) in the worst case where n is the number of hard periodic tasks, m is the number of previously guaranteed but yet uncompleted hard aperiodic tasks and k is an integer which depends on periods and deadlines.

If the technique used to store energy considers a non-ideal energy storage unit, consuming the stored energy as soon as possible is recommended to avoid energy leakage and task execution failure. However, when the leakage rate of the ultra-capacitor is very high during charging and discharging, we may calculate the amount of energy leakage of ultra-capacitor for each periodic task and each arriving aperiodic task at charging and discharging phases. We take into account energy leakage by modifying the formula for the computation of the energy laxity.

VII. Conclusion and future works

This paper has presented an exact admission test to prevent processing overload and energy shortage in energy harvesting systems such as sensor nodes in the IoT. Such systems consist of a baseload of hard deadline periodic tasks where aperiodic tasks may be accepted for execution as long as they do not cause any periodic task to miss its deadline. We assume that all the deadline constrained tasks are scheduled according to the optimal scheduling algorithm ED-H [6]. Upon its arrival, any aperiodic task is tested for admission calculating two key values: the time laxity and the energy laxity. Provided that precise information about the future energy generation is available, the proposed online admission test is exact: it enables us to continuously optimize the capacity of the system in the two dimensions: processing time and energy.

Work is in progress to extend our analysis to more general applicative software composed of deadline constrained tasks that may share resources and exhibit precedence constraints.

References

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

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

[3] S. R. Thuel, "Enhancing Fault Tolerance of Real-Time Systems through Time Redundancy", PhD thesis, Carnegie Mellon University, May 1993.

[4] T.-S. Tia, J.W.-S. Liu, and J. Sun, R. Ha, "A linear-time optimal acceptance test for scheduling of hard real-time tasks", Technical report, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL, 1994; available at the URL: http://wwwrtsl.cs.uiuc.edu/papers/TiLS94c.ps.

[5] H.Kim, S. Lee and J.Lee, "Scheduling of Hard-Aperiodic Requests in Dynamic Priority Systems", Proceedings of 3rd International Workshop on Real-Time Computing Systems and Applications, 1996.

[6] 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.

[7] 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.

[8] M.-L. Dertouzos, "Control Robotics: The Procedural Control of Physical Processes", Proceedings of International Federation for Information Processing Congress, 1974.

[9] H. Chetto and M. Chetto, "Scheduling Periodic and Sporadic Tasks in a Real-Time System", Elsevier Science Publishers B.V. (North-Holland), 1989.

[10] 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.

[11] R. El Osta and M. Chetto, "An exact admission test for hard deadline aperiodic tasks with energy harvesting considerations", Technical Report, LS2N Laboratory, University of Nantes, 2021.