Real-Time Scheduling of DAG Tasks in Self-Powered Sensors with Scavenged Energy

Maryline Chetto, Rola El Osta

Real-Time Scheduling of DAG Tasks in Self-Powered Sensors with Scavenged Energy

Maryline Chetto
LS2N, UMR 6004, F44000
Nantes University, Ecole Centrale de Nantes, CNRS
Nantes, France
maryline.chetto@ls2n.fr

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

***Abstract*— Energy harvesting sensors move into the mainstream because they are regarded as a perfect match for the monitoring applications. Nonetheless, their implementation is only feasible if the computing unit, generally a microcontroller, is capable of running the real-time tasks using sophisticated scheduling and power management techniques. In this paper, we address the hard real-time scheduling problem of Directed Acyclic Graph (DAG) tasks in a uniprocessor RTEH (Real Time Energy Harvesting) computing system. In this model, a task is defined as a set of dependent sub-tasks that execute under precedence constraints with processing time and energy requirements. We demonstrate that the energy aware ED-H algorithm provides an optimal solution. We show that the scheduling policy directly includes a feasibility analysis by enforcing the precedence constraints, thus making it possible to get rid of semaphore synchronization.**

Keywords—Real-time, Energy harvesting sensor, Operating system, Scheduling, Precedence constraints, Slack energy, Slack time

I. Introduction

There is now a range of new opportunities for connecting a lot of diverse equipments to the Internet of Things (IoT). Energy harvesting technology and ultra-low-power radio transmission provide a flexible infrastructure for monitoring such equipments, without the need for hard-wired connections [1] [2]. Energy harvesting wireless sensors and switches are well suited for monitoring applications. Sensors which are powered from their environment thanks to heat, light or motion, require no batteries to acquire, process and send data. This permits to sensors as well as actuators, to be placed practically everywhere. The wireless communication protocols allow sensors to communicate with each other. And this controlling framework can be connected to the cloud to enable the so-called 'smart' paradigm. This allows the monitoring and control homes, offices, workshops, cities, etc. from anywhere with an internet connection. Consequently, technology for energy harvesting (EH) has advanced quickly over the past ten years [3] [4].

Developing a sensor that runs from harvested energy imposes some specific design requirements. As shown in Fig. 1 the sensor node is equipped with three major units: the energy source harvester, the energy storage unit (battery or capacitor) with limited capacity and the computing unit (one microcontroller) in charge of executing the application software. In this work, we do not focus on how energy is harvested and temporarily stored before consumption.

We focus on an on-line processor management so as to guarantee an energy neutral behavior of the computing system. Priority-driven schedulers take into account both both fluctuations and limitations of environmental energy and timing constraints of the jobs in the energy harvesting systems. They are different from each other first by how priorities are assigned to the jobs, second by how to determine when a job should be executed and third how to decide when the processor should be let idle [5] [6] [7].

In this paper, we are interested in dependent constrained deadline jobs. We need an optimal priority-based preemptive scheduling policy, and an associated schedulability test [8] [9]. There are mainly two different approaches to deal with this issue. The first one relies on the use of binary semaphores for synchronization. In the second one, the respect of precedence constraints is guaranteed through the way priorities and timing parameters are assigned to jobs. In [10], precedences are encoded in the real-time attributes of jobs by adjusting their release times and deadlines. The adjusted job set is then scheduled according to the EDF policy [11]. This technique is optimal in the sense that feasibility can be tested by applying a schedulability test said to be exact. It directly applies to periodic task sets with constrained deadlines (deadlines less than periods) and simple precedences (precedence relations only between tasks that share the same period). In this paper, we propose extensions for jobs with energy demands that execute on an energy harvesting platform. Our main contribution is to show how the optimal energy harvesting aware scheduler ED-H [7] can be extended to deal with task sets that have both timing, energy and precedence constraints.

The remainder of the paper is organized as follows. In Section II, the analysis model is introduced. We give background materials in Section III. Section IV includes our contribution on scheduling dependent jobs with energy harvesting considerations. Section V shows most important related works. Section VI provides summary and conclusion.

II. Assumptions and Terminology

The real-time system under study is defined by an instance of jobs with hard real-time restrictions and by the RTEH platform in which the jobs will run. The target RTEH platform is composed of processing module (PM), energy storage

This work has been supported in part by the European Regional Development Fund under the project "Optimization and real-Time management of the energy consumption of a cobot (OTREC)" and in part by the Isite NExT within the New Partnerships framework.

978-1-6654-6697-4/22/$31.00 ©2022 IEEE

module (SM) and energy harvesting module (HM) (see Fig. 2). The PM has only one operating frequency. It consumes energy which is considered as negligible when it executes no job.

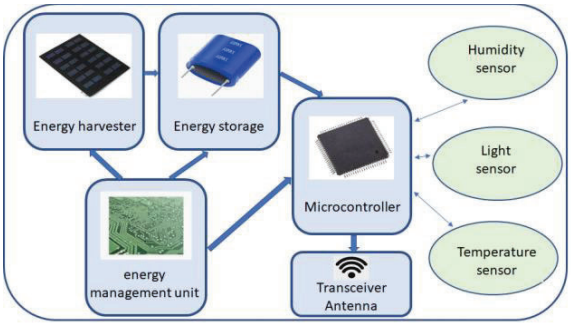

Fig. 1. A typical Energy Harvesting Sensor.

We consider the job set denoted by $J = \{J_1, J_2, \ldots J_n\}$ associated with its related precedence graph G. We respectively denote by r_i, C_i, E_i and d_i, arrival time, Worst Case Execution Time (WCET), Worst Case Energy Consumption (WCEC) and absolute deadline of the job J_i, respectively. The energy which is actually consumed by a job is not necessarily proportional to its WCET. J_i must receive C_i units of execution and E_i units of energy in the time interval $[r_i, d_i)$ called scheduling window of J_i. The response time of J_i corresponds to the time from its arrival to the finishing time, f_i. J_i has consumed C_i units of processing time and E_i units of energy at f_i.

A job can be preempted at any time and later resumed, with no processing cost and no energy cost. We assume that each job consumes energy with arbitrary power consumption rate.
J also refers to the set of nodes in the graph G. The graph G represents a partial order $\prec$ on J. $J_i \prec J_j$ if and only if G contains a directed path from the node J_i to the node J_j. We can say that J_i is a predecessor of J_j (or J_j is a successor of J_i). We can write $J_i \rightarrow J_j$ so as to describe a precedence relation between J_i and J_j. In such a case, J_i is an immediate predecessor of J_j (J_j is an immediate successor of J_i). A job set may have one or more beginning jobs and ending jobs. A beginning job has no predecessor job. An ending job has no successor job.

Let denote by $E_p(t)$ the global energy produced by the power source over the time interval $[0, t]$. It contains all losses due to power conversion and charging. $E_p(t_1, t_2)$ gives the total energy produced on $[t_1, t_2)$. We assume that, in any unit of time, the energy generated by the harvester never crosses the energy which is expended in the same unit of time (i.e. all the jobs are said to be discharging). $E_c(t)$ represents the amount of energy used by all jobs from time 0 to time t. We consider an ideal energy storage unit characterized by its nominal capacity E. The energy level in the SM at time t is denoted $E(t)$. We assume that the stored energy does not suffer from leak over time. A schedule Γ for the job set J is said to be *valid* if the deadlines of all jobs in J are met on the PM, assuming that we start with an SM fully charged.

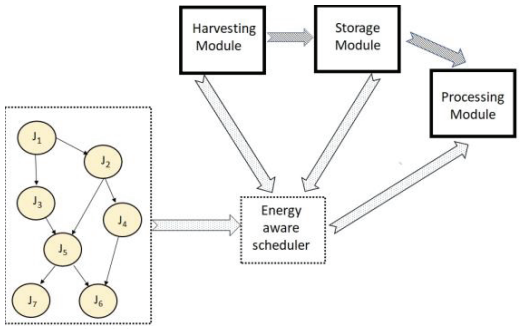

Fig. 2. Framework of the RTEH system.

III. BACKGROUND MATERIALS

A. *Scheduling dependent jobs with no energy considerations*

The precedence encoding technique presented in [10] is based on the assumption that "the relative urgency of a task depends both on its deadline and on the deadlines of its successors" and "the starting time of a task depends both on its release time and on the completion time of its predecessors". The scheduling policy then applies the EDF policy [11] on the adjusted job set. According to [10], a topological sort is used to modulate the job set's deadlines: we begin by modulating the deadlines of jobs without successors. We gradually adjust the deadlines of the jobs assigned to the successors that have previously been adjusted. For every ending job J_i of the partial order, d_i^* is set to d_i. For the other ones, the deadline is computed as follows:

$$d_i^* = min\left(d_i, d_j^* - C_j | J_i \rightarrow J_j\right) \qquad (1)$$

Based on the same idea, for every beginning job J_i of the partial order, r_i^* is set to r_i. For any other job J_i, the earliest time when it can possibly start is given by

$$r_i^* = max\left(r_i, r_j^* + C_j | J_j \rightarrow J_i\right) \qquad (2)$$

Therefore, we can replace r_i by r_i^* and d_i by d_i^* without changing the problem. It clearly comes that for two jobs J_i and J_j such that $J_i \rightarrow J_j$, $r_i^* \leq r_j^*$ and $d_i^* \leq d_j^*$.

The idea is to use these new values, first for the EDF schedulability test and second for the online scheduling of the jobs using the precedence-unaware dynamic priority scheduler EDF.

B. *Scheduling independent jobs with energy harvesting considerations*

We proved in [20] that EDF is no longer optimal and has a zero competitive factor under RTEH settings. The paper [7] extends the popular EDF algorithm. It evinces the optimality of ED-H, an idling lookahead-D scheduler where D is the jobs' longest relative deadline in the application. Identically to EDF, ED-H scheduler takes decisions at run time. Nevertheless, it involves clairvoyance at any time instant for

at least D time units and it is not work-conserving. In a RTEH platform, the available energy may not sometimes allow to execute all the jobs on time (including future ones) unless the processor deliberately enters the sleep mode. When the goal of the scheduler is to meet deadlines of the jobs only, there is no advantage to complete any job sooner than necessary. The idling capabilities of ED-H permit to anticipate energy depletion that would otherwise involve deadline missing with the work-conserving scheduler EDF.

Before describing the principles of ED-H, let us introduce the following necessary definitions:

- The slack time of the job set J at t_c denoted by $ST_J(t_c)$ represents the largest time during which the processor may stay continuously idle from t_c. Methods for computing $ST_J(t_c)$ can be found in [8].
- The preemption slack energy of the job set J at t_c denoted by $PSE_J(t_c)$ represents the largest energy that may be consumed by the currently active job while still preserving energy feasibility for jobs that may preempt it. $PSE_J(t_c) = \min_{t_c<r_i<d_i<d} SE_{J_i}(t_c)$.
- The slack energy of the job J_i at t_c, denoted by $SE_{J_i}(t_c)$ is the largest energy that may be consumed within $[t_c, d_i)$ while guaranteeing enough energy for jobs released at or after t_c with deadline at or before d_i. $SE_{J_i}(t_c) = E(t_c) + E_p(t_c, d_i) - g(t_c, d_i)$ where $g(t_c, d_i)$ is the energy demand of J on $[t_c, d_i)$.

Let denote $L_r(t_c)$ the list of jobs ready for execution at t_c and not completely executed. The following rules define the ED-H scheduling algorithm:

- *Rule 1:* The future running job in $L_r(t_c)$ is selected through the EDF priority order.
- *Rule 2: If* $L_r(t_c) = \emptyset$, *then let the processor idle in* $[t_c, t_c + 1)$.
- *Rule 3: If* $L_r(t_c) \neq \emptyset$ *while* $E(t_c) = 0$ *or* $PSE_J(t_c) = 0$, *then let the processor idle in* $[t_c, t_c + 1)$.
- *Rule 4: If* $L_r(t_c) \neq \emptyset$ *while* $E(t_c) = C$ *or* $ST_J(t_c) = 0$, *then let the processor busy in* $[t_c, t_c + 1)$.
- *Rule 5: If* $L_r(t_c) \neq \emptyset$, $0 < E(t_c) < C$, $ST_J(t_c) > 0$ *and* $PSE_J(t_c) > 0$, *let the processor idle or busy in* $[t_c, t_c + 1)$ *according to a pre-specified breaking rule.*

C. Feasibility testing with energy harvesting considerations

As ED-H is an optimal scheduler, any job set J is feasible if and only if at least one ED-H schedule exists for which all the deadlines can be satisfied, be given the capacity C of the energy storage, and the source power $P_p(t)$, for $0 \leq t \leq d_{Max}$. This feasibility decision problem is addressed in [7]. The limiting factors for feasibility may be the processing time or/and energy. In addition to the processor demand analysis used in real-time systems with no energy limitations, the energy demand analysis should be considered. Consequence, we define time starvation and energy starvation as follows:

- time starvation: the time required to process a job by its deadline is not sufficient while there is available energy when the deadline violation occurs.
- energy starvation: the time required to process a job by its deadline is sufficient but the energy is exhausted when the deadline violation occurs.

In one hand, the time feasibility test checks whether there is time starvation and in the other hand the energy feasibility test checks whether there is energy starvation, for any interval of finite length. Let us introduce the static slack time of the job set J. The processor demand of J on the time interval $[t_1, t_2)$ is $h(t_1, t_2) = \sum_{t_1 \leq r_k, d_k \leq t_2} C_k$. We define the static slack time of J on $[t_1, t_2)$ as $SST_\tau(t_1, t_2) = t_2 - t_1 - h(t_1, t_2)$. $SST_J(t_1, t_2)$ gives the longest time available within $[t_1, t_2)$ after executing jobs of J with release time at or after t_1 and deadline at or before t_2. Finally, the static slack time of J is defined as

$$SST_J = \min_{0 \leq t_1 < t_2 \leq d_{Max}} SST_J(t_1, t_2) \qquad (3)$$

Let us introduce the static slack energy of the job set J. The energy demand of J on the time interval $[t_1, t_2)$ is $g(t_1, t_2) = \sum_{t_1 \leq r_k, d_k \leq t_2} E_k$. The static slack energy of J on $[t_1, t_2)$ is defined as $SSE_J(t_1, t_2) = C + E_p(t_1, t_2) - g(t_1, t_2)$. Consequently, $SSE_J(t_1, t_2)$ gives us the largest energy available within $[t_1, t_2)$ after executing jobs of J with release time at or after t_1 and deadline at or before t_2. Finally, the static slack energy of J is obtained by

$$SSE_J = \min_{0 \leq t_1 < t_2 \leq d_{Max}} SSE_J(t_1, t_2) \qquad (4)$$

Intuitively, the static slack time of J is a lower bound on the processing surplus that could be accepted by J at any instant. The static slack energy of J is a lower bound on the additional energy that could be consumed at any instant. The following theorem proved in [7] can therefore be deduced intuitively:

Theorem 1. *J is feasible if and only if*

$$SST_J \geq 0 \text{ and } SSE_J \geq 0 \qquad (5)$$

Theorem 1 says that (5) ensures that both energy and real-time constraints can be met. Since there exists an energy-feasible and time-feasible schedule, the optimal ED-H algorithm will discover it.

IV. Scheduling Dependent Jobs with Energy Harvesting Considerations

We are now prepared to focus on a set of jobs with both precedence constraints and energy harvesting considerations.

A. Optimality analysis

The approach reported in [10] assumes that the work conserves scheduling policy EDF, with no energy limitation. We show that this result also holds the idling scheduling policy ED-H under the RTEH model described in section 2.

Theorem 2. *Let* $J = \{J_i(r_i, C_i, E_i, d_i), i = 1, \dots, n\}$ *and* $\prec$ *be a partial order on* J. *Let* $J^* = \{J_i(r_i^*, C_i^*, E_i^*, d_i^*), i = 1, \dots, n\}$ *be a set of independent jobs such that* $C_i^* = C_i$, $E_i^* = E_i$, r_i^* *and* d_i^* *are given by formulas 1 and 2.* J *is feasible on a*

given RTEH platform if and only if J^* is feasible on the same RTEH platform.

Proof: (If part) Suppose that the dependent job set J is feasible and the associated independent job set J^* is not feasible. Then there exists at least one valid schedule, say S, such that for every job J_i, its finishing time f_i in S satisfies: $f_i \leq d_i^*$ and all the precedence relations are satisfied. As $d_i^* \leq d_i$, then the valid schedule S where the precedence constraints are satisfied can be applied to the independent set of jobs J^* with identical timing and energy characteristics. As there is at least one valid schedule for J^*, J^* is feasible which contradicts the hypothesis.

(Only If part) Suppose the independent job set J^* is feasible. Let us prove that the dependent job set J is feasible too. As ED-H is optimal, there is at least one ED-H valid schedule where all deadlines of J^* are satisfied i.e there is no energy starvation and no time starvation. Jobs in J have the same energy and timing requirements but from Formulas 1 and 2, they have longer deadlines. Thus, there is at least one valid schedule for J.

Let us prove that the precedence constraints are satisfied too. We have to prove that every job J_i starts after each of its predecessors finishes. Let us consider two jobs J_i and J_j such that $J_i \rightarrow J_j$. Formulas 1 and 2 imply that $r_i^* \leq r_j^*$ and $d_i^* \leq d_j^*$. From rule 1 of the scheduling algorithm ED-H, the ready job with closest deadline gets the highest priority for execution. Consequently, J_j^* will start execution no sooner that all ready higher priority jobs have been finished i.e. jobs with earlier deadline and earlier release time. These jobs include the predecessors of J_j. We have $s_j^* \geq f_i^*$ with s_j^* the start time of J_j^* and f_i^* the finishing time of J_i^*. Thus, for two jobs J_i and J_j such that $J_i \rightarrow J_j$, the precedence constraint is met. □

Theorem 2 enables us to replace r_i by r_i^* and d_i by d_i^* without changing the problem by scheduling the jobs using the optimal precedence-unaware scheduler ED-H.

B. Schedulability analysis

Let us extend the ED-H schedulability test given in theorem Theorem 1 to precedence constrained job sets.

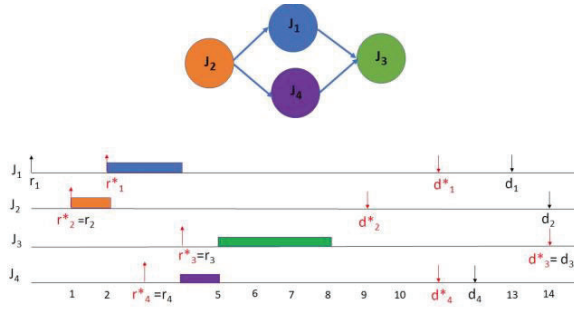

Fig. 3. EDF scheduling disregarding energy constraints.

Corollary 1. *Let $J = \{J_i(r_i, C_i, E_i, d_i), i = 1, \ldots, n\}$ and $\prec$ be a partial order on J. Let $J^* = \{J_i(r_i^*, C_i^*, E_i^*, d_i^*), i = 1, \ldots, n\}$ be a set of independent jobs such that $C_i^* = C_i$, $E_i^* = E_i$, r_i^* and d_i^* are given by formulas 1 and 2. J is feasible if and only if*

$$SST_J^* \geq 0 \text{ and } SSE_J^* \geq 0 \qquad (6)$$

Proof: From Theorem 2, J is feasible if and only if J^* is feasible (that is, feasibly scheduled according to ED-H). From Theorem 1, we immediately deduce that relations (6) provide a feasibility test for J. □

C. Illustrative example

The following is an example to illustrate the RTEH computing model and the associated scheduling issue. Let us consider an application composed of the job set J whose time and energy parameters are shown in Table I. The hardware platform has an energy storage unit whose capacity $C = 20$ mJ. The harvesting power is set to 20 mW from initial time instant 0 during 4 seconds, 10 mW from time 4 to time 7, 0 mW from time 7 to time 8 and then 10 mW (see Fig. 4).

TABLE I.
TIME AND ENERGY PARAMETERS OF THE JOBS

Job J_i	**C_i (s)**	**r_i (s)**	**d_i (s)**	**E_i (mJ)**
J_1	2	7	13	30
J_2	1	5	14	60
J_3	3	6	14	60
J_4	1	0	12	30

Firstly, let us show the EDF schedule for the energy non-constrained case where all the jobs execute in ASAP mode (see Fig. 3). The precedence constraints are considered through a suitable modification of the release times and deadlines with (1) and (2) (see Table II). This allows us to schedule independent jobs instead of the DAG using the Earliest Deadline First scheduler.

TABLE II.
TIME PARAMETERS OF THE JOBS

J_i	J_1	J_2	J_3	J_4
(r_i, d_i)	(0,13)	(1,14)	(4,14)	(3,12)
(r_i^*, d_i^*)	(2,11)	(1,9)	(4,14)	(3,11)

Secondly, let us consider the energy constraints. Through theorem 2, we are now able to state that the DAG job set can be feasibly scheduled by ED-H respecting its precedence constraints. For every scheduling interval $[r_i^*, d_j^*)$, $SST_J(r_i^*, d_j^*)$ and $SSE_J(r_i^*, d_j^*)$ are computed in order to verify that (6) is satisfied. In other terms, the job set has no processing overload and no energy starvation, since in every interval the required processing time does not exceed the length of the interval and the total energy consumption of the jobs is no more than the available energy i.e. the energy stored in the reservoir plus the energy harvested from ambient. Fig. 4 depicts the ED-H schedule produced on the resulting independent job set with the suitable modified release times and deadlines given in Table II. Note that this schedule is identical to the ED-H schedule produced on the DAG with initial release times and deadlines.

The following is an example to illustrate the RTEH computing model and the associated scheduling issue. Let us consider an application composed of the job set J whose time and energy parameters are shown in Table I. The hardware platform has an energy storage unit whose capacity $C = 20$ mJ. The harvesting power is set to 20 mW from initial time instant 0 during 4 seconds, 10 mW from time 4 to time 7, 0 mW from time 7 to time 8 and then 10 mW (see Fig. 4).

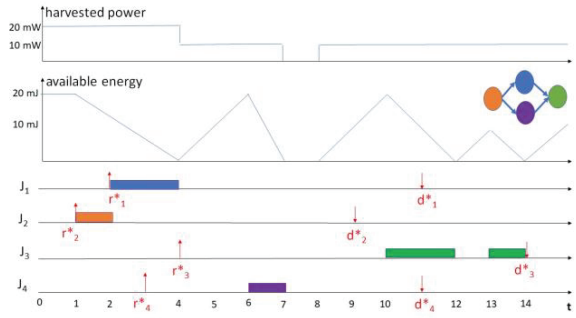

Fig. 4. ED-H scheduling for precedence constrained jobs.

V. Related work

The problem of scheduling sets of periodic tasks is present in majority of real-time embedded systems including wireless sensors of the Internet of Things [8]. Up until now, most of work on real-time scheduling typically focused on models that include tasks which must be completed before a deadline and have computation time requirements only, expressed with respect to Worst Case Execution times (WCET) [9]. Liu and Layland [11] studied fixed-priority and dynamic-priority task scheduling [8]. Through the use of Rate-Monotonic (RM) priority assignment, tasks can be statically guaranteed. The RM technique utilizes periods to determine priority: the smaller the period of a task, the higher the priority for all jobs associated with that task. EDF (Earliest Deadline First) is based on a dynamic task-level priority assignment and fixed job-level priority job assignment such as the earlier the deadline of the job, the higher the priority. Also called non idling, these online schedulers are work conservative. Since idle processing does not enhance schedulability, they never permit the processor to be inactive when at least one job is waiting to be executed. In other words, they consume the available energy greedily.

Multiple studies that concern different scheduling strategies on parallel task model, for DAG tasks models in uniprocessor systems particularly, have been documented in the literature. In 2005, Zhao et al. [12] provided an approach to schedule a set of sporadic graph tasks with fixed priority, for different scheduling contexts. This research had been lately improved in [13] to assume Fixed Job Priority (FJP) scheduling of graph tasks under Earliest Deadline First (EDF) algorithm. By using priority-based resource scheduling and resource partitioning, Jayachandran and Abdelzaher [14] focused on uniprocessor scheduling in distributed systems where jobs are depicted by Directed Acyclic Graphs (DAGs) by employing a Delay Composition theorem. The scope of this work was then enhanced to include tasks whose subtasks execute on various resources and build a DAG with internal precedence constraints. Recently, Stigge et al. [15] explored directed graphs to model our tasks in order to express the release situation of jobs in terms of timing and order. The main contribution of this study is the demonstration of the pseudopolynomial-time solution to the feasibility problem. The authors of [16] investigated the computational difficulty of the model modification that enables the expression of global inter-release time limits between non-adjacent activity releases. None of these works has considered energy-aware scheduling for DAG tasks in a uniprocessor platform. In the other hand, the existing literature explore task scheduling in uniprocessor RTEH systems that integrate deadline constraints. The problem was early addressed by Allavena and Mossé in [5] for frame-based tasks with voltage and frequency scaling capabilities. An optimal scheduler is presented, assuming that the energy storage unit is replenished with a constant power rate. The Lazy Scheduling Algorithm called LSA is an EDF-based algorithm proposed by Moser et al. [6]. LSA is optimal, based on as late as possible policy. It may apply to any set of deadline constrained tasks, periodic or not. The harvested power is estimated as a time-varying variable but all tasks consume energy with the same rate. Liu et al. extend LSA with different DVFS-based algorithms, called EA-DVFS and HA-DVFS [17]. They use slack time to slow down the processor whenever possible to save energy. In case of surplus harvested energy, they accelerate the task execution. Abdeddaim et al. [18] propose PFP_{ASAP} a preemptive fixed-priority scheduling algorithm. They assume a constant power source and periodic tasks that consume energy linearly. They derive a schedulability condition and prove optimality of PFP_{ASAP}. In [19], it is proved that only lookahead-D schedulers can be optimal where D denotes the maximum relative deadline of jobs in the application. This signifies that any optimal scheduler requires forecast of the incoming energy and model of task arrivals on at least D time units. We also prove in [20] that EDF remains the best work-conserving scheduler under RTEH settings, even if with zero competitive factor. Recently, an energy-aware framework called AsTAR prioritizes tasks using their importance [21]. It adapts the computing service to the available environmental energy. It does not require modeling the hardware platform and predicting the harvested energy to take decisions. Nevertheless, it cannot suit to critical real-time constraints. Clearly, most of the works above are based on EDF scheme. EDF has the advantage to support higher processor utilization than fixed-priority schedulers. As previously mentioned, no existing work takes into account the DAG task workload scheduling in self-powered sensors in a uniprocessor RTEH computing systems and avoids semaphore synchronization.

VI. Conclusion

Energy harvesting is anticipated to be crucial in the Internet of Things by allowing sensors to be placed almost anywhere, forming part of an autonomous and maintenance-free wireless network to provide remote monitoring of assets. The ability to harvest enough energy to run a wireless sensor in a mesh network will create a wide range of applications, no longer limited by the absence of power. However, the design of self-powered sensors involves solving various complex technological problems.

This paper addressed the scheduling problem of a set of jobs with precedence, deadline and energy constraints. The schedulability analysis, in the design of RTEH systems, utilises an abstract model as an input. Along with describing the characteristics of the energy harvester and the energy storage unit, it also describes the scheduling and energy demands for the jobs. All the constraints directly affect the schedulability and need to be concerned in the system model.

In this paper, we presented a technique to enforce precedence constraints through a deadline assignment procedure so as to fit the optimal energy-aware scheduler ED-H. In [7], Chetto demonstrates that ED-H is a strongly optimal scheduler, for the general RTEH model with no restriction on task arrival profile and energy source profile. Our approach for scheduling precedence constrained jobs permits to prevent semaphore synchronizations and results in a smaller operating system, which is a crucial requirement in wireless sensors. Numerous RTEH systems run on a cyclic basis where jobs repeatedly run (e.g. for sensing) with prospective arrival times and deadlines. Although the scope of the schedulability analysis is a finite set of jobs, our results extend to most of practical applications where an infinite set of jobs is generated by a finite set of periodic tasks. A finite graph that depicts one hyperperiod and allows for simple static analysis can be used to model them. The deadlines can be constrained (i.e.,less or equal to the period) or implicit (i.e.,equal to the period). Typical task sets observed in sensors mostly only show a maximum of one hundred jobs in a hyperperiod, however in the worst case the number of jobs in the hyperperiod H is exponential in the number of tasks. The offsets between the first occurrence of several periodic tasks are often statically known. For precedence constrained systems, our work is adapted to schedule the hyperperiod of periodic implicit or constrained deadline tasks. As a conclusion, this paper emphasizes that the design of a real time energy harvesting system can be addressed in an integrated manner.

REFERENCES

[1] S. Chalasani and J. M. Conrad., " survey of energy harvesting sources for embedded systems", IEEE SoutheastCon. pp. 442–447, 2008.

[2] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications", Communications Surveys and Tutorials, vol. 13, no. 3, pp. 443–461, 2011.

[3] K. S. Adu-Manu, N. Adam, C. Tapparello, H. Ayatollahi, and W. Heinzelman, "Energy-harvesting wireless sensor networks (ehwsns): A review", ACM Transactions on Sensor Networks (TOSN), vol. 14, no. 2, p. 10, 2018.

[4] D. Ma, G. Lan, M. Hassan, W. Hu, and S. K. Das, "Optimizing sensing, computing, and communication for energy harvesting iots: A survey", IEEE Communications Surveys and Tutorials, vol. 22, no 2, pp. 1222-1250, 2019.

[5] A. Allavena and D. Mosse, "Scheduling of Frame-based Embedded Systems with Rechargeable Batteries", Workshop on Power Management for Real-Time and Embedded Systems, 2001.

[6] C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Real-time scheduling for energy harvesting sensor nodes", Real-Time Systems, vol. 37, no 3, pp. 233-260, 2007.

[7] M. Chetto, "Optimal Scheduling for Real-Time Jobs in Energy Harvesting Computing Systems", IEEE Transactions on Emerging Topics in Computing, vol. 2, no. 2, pp. 122-133, 2014.

[8] J. W. S. Liu. Real-Time Systems, Prentice Hall, 592 pages, 2000.

[9] Robert I. Davis, Liliana Cucu-Grosjean, Marko Bertogna, and Alan Burns, "A review of priority assignment in real-time systems", Journal of Systems Architecture, vol. 65, pp. 64-82, 2016.

[10] H. Chetto, M. Silly and T. Bouchentouf, "Dynamic scheduling of real-time tasks under precedence constraints", The Journal of Real-Time Systems, vol. 2, pp. 181, 1990.

[11] C.-L. Liu and 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.

[12] H. X. Zhao, S. Midonnet, and L. George, "Worst-Case Response Time Analysis of Sporadic Graph Tasks with Fixed Priority Scheduling on a Uniprocessor", In Proceedings of Embedded and Real-Time Computing Systems and Applications (RTCSA), 2005.

[13] H.-X. Zhao, L. George, and S. Midonnet, "Worst-Case Response Time Analysis of Sporadic Graph Tasks with EDF Scheduling on a Uniprocessor", In Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2006., pp. 271–278, 2006.

[14] P. Jayachandran and T. Abdelzaher, "Transforming Distributed Acyclic Systems into Equivalent Uniprocessors under Preemptive and Non-Preemptive Scheduling", In Proceedings of the 20th Euromicro Conference on Real-Time Systems (ECRTS), pp. 233–242, 2008.

[15] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "The Digraph Real-Time Task Model", In Proceedings of the 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 71–80, April 2011. doi: 10.1109/RTAS.2011.15.

[16] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "On the Tractability of Digraph-Based Task Models", In Proceedings of the 23rd Euromicro Conference on Real-Time Systems (ECRTS), pp. 162–171, July 2011. doi: 10.1109/ECRTS.2011.23.

[17] S. Liu, J. Lu, Q. Wu and Q. Qiu, "Harvesting-Aware Power Management for Real-Time Systems with Renewable Energy", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, pp. 1-14, 2011.

[18] Y. Abdeddaïm, Y. Chandarli and D. Masson, "The Optimality of PFPASAP Algorithm for Fixed-Priority Energy-Harvesting Real-Time Systems", Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS), 2013.

[19] M. Chetto, A. Queudet, "Clairvoyance and Online Scheduling in Real-Time Energy Harvesting Systems", Real-Time Systems Journal, March 2014, vol. 50, pp. 179-184, March 2014.

[20] M. Chetto and A. Queudet., "A Note on EDF Scheduling for Real-Time Energy Harvesting Systems", IEEE Transactions on Computers, vol. 63, no 4, pp. 1037-1040, April 2014.

[21] F. Yang, A. S. Thangarajan, G. S. Ramachandran and W. Joosen, "AsTAR: Sustainable Energy Harvesting for the Internet of Things through Adaptive Task Scheduling", ACM Transactions on Sensor Networks, vol. 18, no 1, February 2022.