Feasibility Analysis of Periodic Real-Time Systems with Energy Harvesting Capabilities

Maryline Chetto, Audrey Queudet

LS2N laboratory, University of Nantes, 1 Rue de la Noë, F-44321 Nantes FRANCE

Abstract

Energy harvesting now appears as a feasible solution to power wireless devices. Such ones including sensor nodes have to be energy neutral i.e. be able to never consume more energy than available. Energy can be drawn from many ambient sources including solar, vibration, etc. Most of wireless devices perform periodic sampling and computations using a microprocessor for real-time monitoring. The software tasks that typically take place in these systems are time critical and cyclic with specified deadlines attached to finishing times. In this paper, we explore the problem of feasibility analysis of hard real-time periodic task sets on a single processor powered by renewable energy. Firstly, we investigate the notion of robustness as the capability of the system to remain *stable* even in case of positive changes in its operating environment. We prove that our feasibility test is robust with respect to the profile of the energy source. Secondly, we prove that for asynchronous tasks whose initial release times are not known in advance, feasibility can be checked considering the synchronous activation pattern which corresponds to the worst-case scenario. Finally, whenever accurate prediction of the incoming environmental energy is impossible, a sufficient feasibility test with high implementation efficiency is proposed.

Keywords: Energy harvesting, Self-sustaining real-time systems, Periodic tasks, Uniprocessor scheduling, Earliest Deadline First

1. Introduction

1.1. Background

The research area of real-time scheduling has received considerable attention from the academia and industry over the past four decades [1]. Most of the work in the literature assumes that there is no limitation on energy consumption, that is, the tasks have processing time requirements only. In reality, energy is an essential resource in addition to the processor. Therefore, it is of crucial importance to address the problem of scheduling under both timing and energy constraints.

Energy harvesting is an alternative to a battery-run device in order to extend its lifetime or even to achieve perpetual and self-sustaining operation. It permits to power a wireless electronic device by scavenging renewable energy and converting it into useable electrical energy. Renewable energy is generated from many natural resources (e.g. environmental vibrations, human power, sunlight, wind) which continuously replenish. Energy harvesting can make a device work perpetually in total autonomy. Nonetheless, this requires an effective utilization of available and future energy resources so as to avoid energy shortage. Notably, two prototypes have demonstrated efficiency of energy harvesting technology for electronic devices: Heliomote [2] and Prometheus [3].

In contrast to a traditional battery-power system, an energy harvesting one has to face several important technical and scientific challenges so as to work effectively with complete autonomy. Many environmental energy sources are unstable with fluctuations in time which makes the harvested power a time-varying function. Whereas some energy sources have a periodic profile such as the sunlight, other sources have a stochastic profile which makes uncertain energy availability. As a consequence, the traditional real-time scheduling and power management techniques should be revisited in order to be flexible enough and adapt to dynamic operational conditions with energy awareness.

1.2. Motivation

One concern in the analysis and development of a hard real-time system is the question of predictability of the system's behaviour with respect to timing requirements. In other words, the objective is to guarantee prior to system run-time that deadlines (i.e. latest acceptable finishing times) will be always respected. Considering RTEH (Real Time Energy Harvesting) systems, the challenges in designing them are not limited to timing constraints. We can represent these challenges as an intersection of three distinct problems as shown in Figure 1.

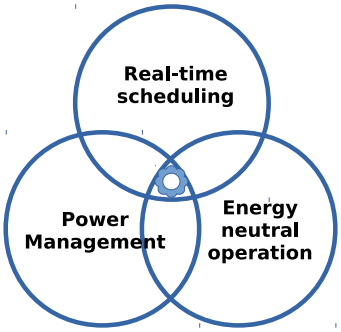

Figure 1: The challenges involved in scheduling RTEH systems

The concept of *energy neutral operation* derives from the fact that RTEH systems potentially sustain their operation perpetually. They scavenge for their own energy once they are deployed and operate for as long as they are installed. A key factor of RTEH systems is that they must provide *real-time guarantees*. Hence, they must jointly cope with timing and energy constraints. This means that these systems must always maintain a suitable energy storage level (even possibly zero over some time slots) to avoid energy starvation which result in deadline missings.

Traditional real-time power management mainly focuses on minimizing the power consumptions under given timing requirements [4, 5, 6, 7, 8]. Combined with the purpose of energy neutral operation, the goal of *power management* is no longer to minimize the power consumption but to make sure that the execu-

tion of the tasks within their deadlines copes with uncontrollable but predictable environmental power so as to make sustainable harvested energy.

Google's Project Loon [9] launched in 2013 is a typical example of RTEH system. It is a network of stratospheric balloons designed to provide Internet access to rural and remote areas of the globe. Each balloon harvests energy from an array of solar panels in order to power its various components including wireless connectivity devices, GPS and robust high throughput link to the existing internet.

RTEH systems have also invaded the market of wearable devices, with a vast number of applications ranging from health monitoring, activity recognition, to gait analysis. For example, a pedometer embedded in a piezoelectric-based energy harvesting shoe has been proposed in [10] as an effective wearable device with an indefinite lifetime.

Traditional real-time scheduling algorithms are not suitable for scheduling RTEH systems because they do not account for energy constraints in taking scheduling decisions. In this work, we present a feasibility analysis using which it will be possible to determine, at the time a designer sets its RTEH system, whether the system can operate perpetually respecting all the requirements. The choice of the real-time periodic model is motivated by the fact that it is widely adopted in the real-time development environment. Consequently, by considering the real-time periodic model, our findings and results are applicable to most real-time embedded applications which mainly perform cyclic controlling/monitoring functions.

In the context of designing RTEH platforms, one has to dimension three fundamental system parameters which are: (i) the size of the energy storage (e.g. battery, capacitor, etc.), (ii) the size of the harvester (e.g. solar panel), and (ii) the speed of the processor. Hardware dimensioning is constrained by the application software with both timing and energy parameters (e.g. processing times, periods, and energy consumptions of the tasks). A timing and energy verification process must take place off-line in order to prevent any deadline violation and any energy starvation at run-time.

1.3. Contributions

In this paper, we consider the problem of determining the feasibility of preemptable periodic task sets on a single processor RTEH platform. In that system, an energy reservoir with given capacity replenishes with renewable energy. It serves to power the processor on which periodic tasks have to complete before specific deadlines (see Figure 2).

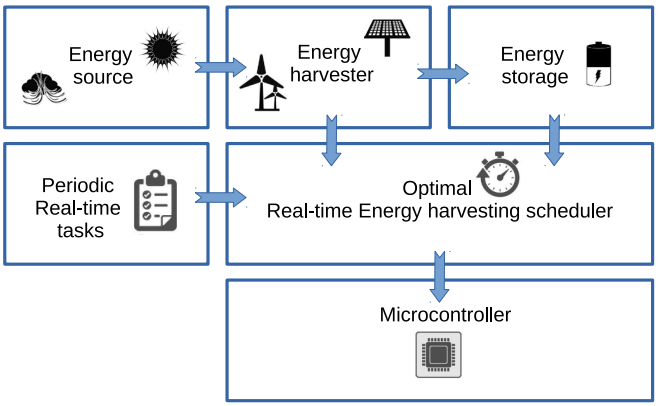

Figure 2: RTEH system overview

Our contributions are the following: Firstly, we address *robustness* for real-time energy harvesting systems, namely their capability to remain *stable* even in case of positive changes in their parameters (e.g. variation in worst-case execution times and/or harvested energy) [11].

Based on this robustness prerequisite, we explore the design of a suitable feasibility test for use in uniprocessor RTEH platforms. By considering the particular case of a constant power source, we show how the problem reduces to verify whether a simple sufficient condition is satisfied. Under this assumption, the feasibility could be only tested (off-line) on one time interval whose duration is equal to the hyperperiod (i. e. LCM of tasks' periods). On the contrary, under a non-uniform source profile, the feasibility test should be implemented on-line on sliding time windows then requiring using prediction of the incoming environmental energy. Consequently, this leads to both a higher overhead and a

far more difficult implementation from a computational complexity perspective. Note that if it is possible to determine a lower bound on the power source then the feasibility problem reduces to the constant power source case. Even if based on pessimistic assumptions on energy production, the feasibility test permits to guarantee once and for all both the energy requirements and the real-time constraints by its deterministic nature.

The practical relevance of the feasibility test that will be presented hereafter is threefold:

- it can be implemented off-line in a pre-operational phase;
- its computational complexity is low because it can pass over the variability of the incoming source power;
- it can easily be used in the design process to choose the suitable hardware platform characteristics.

The remainder of this paper is organized as follows. The next section briefly reviews the literature in real-time scheduling for energy harvesting systems. Section 3 presents our system model with related terminology. In Section 4, we recall basic real-time scheduling concepts with a focus on the Earliest Deadline First (EDF) scheduling properties. Section 5 introduces novel concepts for RTEH systems and presents existing results on ED-H, a variant of EDF for the real-time scheduling with energy considerations. Robustness concerns are also discussed. In Section 6, we study real-time task feasibility under the RTEH model. Section 7 restricts the analysis to systems powered by a constant source of energy. Finally, we conclude the paper in Section 8.

2. Related work

2.1. Real-time scheduling with no energy considerations

Real-time scheduling theory has traditionally been interested in: (i) designing algorithms that tell us whether a given task set can be feasibly scheduled

(i.e. determining whether all jobs generated by the tasks can complete within their timing constraints) and (ii) developing run-time schedulers (i.e. establishing schedules at run-time for a task set deemed feasible). In [1], the authors survey selected prior work that addresses these issues.

Within the context of preemptive uniprocessor scheduling, EDF is optimal for scheduling sets of independent real-time jobs, possibly generated from requests of periodic tasks [12] [13]. EDF is a non-idling (i.e. work-conserving) dynamic-priority scheduling algorithm which at each instant chooses for execution the currently-ready job with the closest absolute deadline (with ties broken arbitrarily).

The problem of providing fast exact schedulability analysis for EDF scheduling for a synchronous periodic task set (i.e. tasks having the same initial release times) has been adressed by [14, 15, 16]. Upper bounds were proved in order to reduce the number of times the processor demand function [14] has to be calculated. Each approach consists of checking all the absolute deadlines in the upper bound when a task set is feasible and stopping when a deadline missing occurs for an unfeasible task set.

More recently, in [17], a tighter upper bound is obtained by the so-called Quick convergence Processor-demand Analysis (QPA) algorithm.

2.2. Scheduling for real-time energy harvesting systems

Initial research works on real-time scheduling under energy harvesting conditions dates from the beginning of the 2000s. It was first assumed that the harvested power is constant along time for periodic tasks with identical periods [18]. In 2007, Moser et al. have considered a more realistic energy model where the harvested energy may be non uniform but predictable on near future. When accurate prediction of the harvested energy is assumed, Lazy Scheduling algorithm (LSA) provides an optimal solution [19]. LSA is based on idling EDF but requires clairvoyance on the energy incoming. Consequently, the effective performance of LSA in a real application depends on the accuracy of the energy harvesting prediction technique.

DVFS (Dynamic Voltage Frequency Scaling) extensions of LSA have been studied in [20] and [21]. And several prediction methods have been proposed to predict solar energy, namely Exponentially Weighted Moving Average (EWMA) in [22] and Weather Conditioned Moving Average (WCMA) in [23]. Run-time prediction of future harvested energy was also addressed in [24]. One central hypothesis on the LSA scheduler is that the energy demand and the execution time of any job behave proportional. However, instantaneous power consumed by a job in execution may vary along time depending on circuitry and devices required by its execution. In 2014, we have based our research works on a more realistic energy model by relaxing this restrictive hypothesis. We have proposed an optimal scheduling algorithm, namely ED-H [25]. As LSA, ED-H is a variant of EDF thus a dynamic priority driven scheduler which takes its decisions online. It also requires an accurate prediction on near future of both the harvested energy and the energy consumed by the tasks.

Energy requirements are not considered by the classical EDF scheduler which is a greedy one by executing the tasks as soon as possible. This approach, when applied under energy harvesting considerations, can lead to deadline violations for future jobs that could experience energy starvation. The ED-H scheduler provides a suitable scheduling solution for simultaneously exploiting the processing resource, which is either a processor or a microcontroller, and the energy resources present in the environment. The main advantage of ED-H compared to a classical scheduler such as EDF is to adapt the processing activity, thus postponing the execution of some jobs whenever the available energy in the storage unit is not sufficient. In contrast with EDF which executes jobs as soon as possible, ED-H provides a dynamic and adaptive power management strategy that permits to avoid energy shortage while respecting the time constraints of the jobs.

In [26], we proved that EDF is non-competitive for the RTEH model whatever the profile of the energy source and the profile of task arrival. Nevertheless, EDF was proved to be the best non-idling scheduling algorithm. We also demonstrated in [27] that no nonclairvoyant scheduler can be optimal in this

case. Optimality can be reached only on the condition that the algorithm has a clairvoyance over a time interval equal to at least the greatest relative deadline among all the tasks. The authors proved that the ED-H scheduler needs clairvoyance exactly on this time interval. Consequently, ED-H can be defined as the best candidate for real-time scheduling associated to an accurate energy predictor such as those cited previously.

3. Problem Statement

In this section, we explain the RTEH model and give necessary terminology.

3.1. System model

The target system consists of one processing module (PM), an energy harvesting module (HM) and an energy storage module (SM), as illustrated in Figure 3.

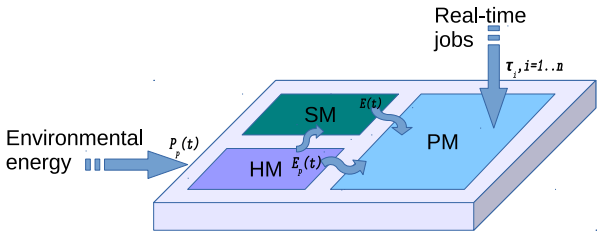

Figure 3: System architecture

The PM has one operating frequency and its energy consumption is only due to dynamic switching energy. We assume that it consumes negligible energy in idle state. The PM is supplied exclusively by the energy generated by the environmental source. Real-time jobs are units of work that need to be executed on the PM while guaranteeing their deadlines.

3.2. Task model

In this paper, we consider the periodic task model in which five-tuple $(r_i, C_i, E_i, D_i, T_i)$ is associated with periodic task τ_i. It respectively gives initial release

time (often called the offset of the task), worst-case execution time (normalized to processor computing capacity), worst-case energy consumption, relative deadline and period of jobs of τ_i. The worst-case energy consumption of τ_i, namely E_i, is expressed in energy units. The actual energy consumed by a job is not necessarily proportional to its execution time. We consider constrained-deadline task sets in which every τ_i has its relative deadline no larger than its period (i.e. $D_i \leq T_i$).

For the remainder of this paper, let τ denote a periodic task set comprised of n tasks: $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$. τ is completely-specified since all parameters of each job can be determined prior to system run-time. The periodic task set τ generates an infinite collection of real-time jobs, denoted by ω. Jobs arrive at each instant $(r_i + kT_i)$ for all integers $k \geq 0$. Each job J_i in ω is characterized by three parameters: arrival time a_i, worst-case execution time C_i and absolute deadline d_i with the interpretation that job J_i must complete within D_i time units from the time it was issued, i.e. it must receive C_i units of execution over the interval $[a_i, d_i)$. Let d_{Max} be the greatest absolute deadline in the job set (d_{Max} equals infinity if ω is an infinite set of jobs).

Let H_τ denote the least common multiple of the periods of the tasks in τ: $H_\tau = lcm\{T_1, T_2, ..., T_n\}$. If all offsets are equal to zero (i.e. $\forall \tau_i,\ r_i = 0$), τ is said to be *synchronous*, otherwise it is *asynchronous*. Jobs are independent of each other and can be preempted and later resumed at no time and energy cost.

3.3. Key Concepts and Assumptions

The processor utilization of τ is defined as $U_\tau^p = \sum_{i=1}^{n} \frac{C_i}{T_i}$. U_τ^p characterizes the busy time ratio of the PM. The energy utilization U_τ^e characterizes the average energy consumption of τ per time unit. It is defined as $U_\tau^e = \sum_{i=1}^{n} \frac{E_i}{T_i}$. Thus U_τ^p cannot be greater than 1 and U_τ^e cannot be greater than the average energy drawn from the source during one time unit. The energy consumed by τ on the time interval $[t_1, t_2)$ is denoted by $E_c(t_1, t_2)$.

At every time t, the HM (e.g. solar panel) draws energy from ambient and converts it into electrical power with instantaneous charging rate $P_p(t)$ that

incorporates all losses. The energy harvested in the time interval $[t_1, t_2)$ is given by $E_p(t_1, t_2) = \int_{t_1}^{t_2} P_p(t)dt$. The energy produced by the ambient source is assumed to be uncontrollable. Consequently, it may be zero in some time slots. We assume that the energy production times can overlap with the consumption times. While the source power is not necessarily a constant value, we assume that we can predict it accurately for near future with negligible time and energy cost. The electrical energy produced by the HM in any unit of time never exceeds e_m, the energy consumed in that unit of time (i.e. executing any job of τ consumes at most e_m units of energy during one unit time-slot and the actual energy consumption in the slot is not known beforehand). Consequently, the remaining energy needed by any job executing on the PM has to be drawn from the SM. We say that the jobs are discharging.

Our system uses a SM (e.g. supercapacitor or rechargeable battery) that enables it to continue operation even when there is no sufficient energy to harvest. Its nominal capacity C corresponds to the maximum amount of energy (expressed in energy units) that can be stored at any time. The SM receives power from the HM and delivers power to the PM. The stored energy at any time t is denoted $E(t)$. The SM does not leak any energy over time. If the SM is fully charged at time t and we continue to charge it, energy is wasted and we say that an energy overflow occurs. In contrast, if the SM is fully discharged at time t (energy depletion), no job can be executed and the PM enters idle state.

We focus on an autonomous embedded system such as a wireless sensor node. Our objective is to guarantee before deployment that it can achieve a mode of operation described as energy-neutral. It must never consume more energy than it gathers from the ambient source. In the energy harvesting context, the objective of the scheduler is *not* to minimize the energy consumption but to guarantee that jobs never consume more energy than available. This is what is called *energy neutrality* and ensures the perpetual autonomy of the system.

3.4. Definitions

Various definitions used in the subsequent sections are detailed here. A schedule Γ for τ on the PM is *valid* if the deadlines of all jobs of τ are met in Γ with the energy generated by a given HM and stored in a given SM. A RTEH system is *feasible* if there exists at least one valid schedule for task set τ with given PM, HM and SM. Otherwise, it is *unfeasible*. We say that job misses its deadline because of *time starvation* if there is available energy in the SM when the deadline violation occurs (i.e. $E(t) > 0$). If the energy in the SM is exhausted when the deadline violation occurs (i.e. $E(t) = 0$) the system is referred to as *energy starvation*. A job is referred to as schedulable according to a given scheduling algorithm if its worst-case response time under that scheduling algorithm is less than or equal to the relative deadline of the associated task. A task set is referred to as schedulable according to a given scheduling algorithm if all of its jobs are schedulable.

A schedulability test is termed sufficient, with respect to a scheduling algorithm and a RTEH system if all of the task sets that are deemed schedulable according to the test are in fact schedulable. A schedulability test is termed necessary if all of the task sets that are deemed unschedulable according to the test are in fact unschedulable. A schedulability test that is both sufficient and necessary is referred to as exact.

A scheduling algorithm is optimal if it can schedule all of the task sets that can be scheduled by any other algorithm, that is, all of the feasible task sets. An online algorithm makes its decision at run-time. It can be idling if it is allowed to keep the processor idle even when there are pending jobs. Otherwise, it is non-idling or work-conserving. A clairvoyant algorithm has an a priori knowledge of both all the jobs arrivals and the entire profile of energy produced by the source.

A task is said to be ready at time instant t if one of its jobs has been released by time t, its service is not complete and its absolute deadline has not expired.

4. Scheduling with no energy consideration

In this section, we recall basic definitions for real-time scheduling concepts used for checking the schedulability of a jobset under EDF.

4.1. Processor demand and slack time

Definition 1. *The processor demand of jobset ω on the time interval $[t_1, t_2)$ is*

$$h_\omega(t_1, t_2) = \sum_{t_1 \leq a_k, d_k \leq t_2} C_k \tag{1}$$

When the jobs in ω utilize the processor with a ratio less than 100%, there is unused processor time, hence the notion of slack time.

Definition 2. *The static slack time of jobset ω on the time interval $[t_1, t_2)$ is*

$$SST_\omega(t_1, t_2) = t_2 - t_1 - h_\omega(t_1, t_2) \tag{2}$$

The schedulability analysis for EDF needs to compute the processor demand for every time interval starting with an arrival time and finishing with a deadline according to formula (1). We must check if the processor demand exceeds the length of the interval. We may define the static slack time $SST_\omega(t_1, t_2)$ with formula (2) that gives the longest time that could be made available within $[t_1, t_2)$ after executing jobs of ω. In other words, we check if the static slack time in any interval is no less than zero.

Definition 3. *The static slack time of jobset ω is*

$$SST_\omega = \min_{0 \leq t_1 < t_2 \leq d_{Max}} SST_\omega(t_1, t_2) \tag{3}$$

Thus, Theorems 1 and 2 provide necessary and sufficient feasibility tests for jobsets with no energy consideration [16].

Theorem 1. *[16] Jobset ω is feasible if and only if for any t_1, t_2 satisfying $t_1 < t_2$,*

$$h_\omega(t_1, t_2) \leq t_2 - t_1 \tag{4}$$

Theorem 2. *[16] Jobset ω is feasible if and only if $SST_\omega \geq 0$*

Example 1. *To understand how feasibility is checked, let us consider a jobset whose attributes are shown in Table 1.*

Table 1: Jobset ω

Job	a_i	C_i	d_i
J_1	4	1	6
J_2	0	1	2
J_3	0	4	9

Table 2 summarizes the different time intervals for which both the processor demand and the static slack time are computed. Observe that we do evaluate equation (1) and (3) only in those intervals whose starting time corresponds to a job release and whose finishing time corresponds to a deadline end point, thus assuming jobs will be scheduled under EDF. Jobset ω is feasible since its static slack time is always greater than or equal to 0. Figure 4 depicts the schedule produced by EDF. Over time intervals $[0, 2)$ and $[4, 6)$, we can see that the processor could stay idle during only one unit of time without jeopardizing the deadlines of jobset ω, as reflected by the value of the slack time (equal to 1) computed over these intervals.

Table 2: Static slack time computations for ω

$[t_1, t_2)$	$[0, 2)$	$[0, 6)$	$[0, 9)$	$[4, 6)$	$[4, 9)$
$h_\omega(t_1, t_2)$	1	2	6	1	1
$SST_\omega(t_1, t_2)$	1	4	3	1	4

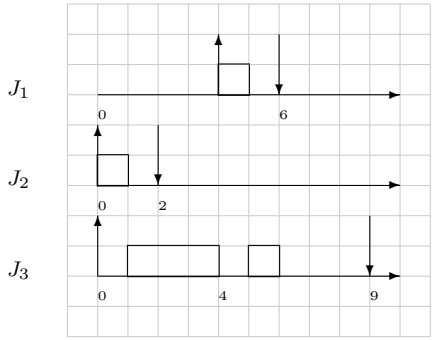

Figure 4: EDF Schedule

4.2. Schedulability test for periodic tasks

As a consequence of its optimality, any feasible periodic task set is guaranteed to be successfully scheduled by EDF. The special case of implicit-deadline task sets (where $D_i = T_i$ for all tasks τ_i) is tractable since a necessary and sufficient condition for τ to be feasible is that $U_\tau^p \leq 1$ [13]. Let $Dens(\tau_i) = \frac{C_i}{\min(T_i, D_i)}$ be the density of task τ_i. Checking that total density is at most one, i.e., $\sum_{i=1}^{n} Dens(\tau_i) \leq 1$, is also a widely used test with linear-time complexity even if it is only sufficient. A linear sufficient test, more accurate than the density test, was shown in [28].

Determining whether an arbitrary periodic task set is feasible has been proved to be intractable in 1980 [29]. Leung and Merril prove that there exists a valid schedule if and only if there exists a valid schedule which is cyclic with a period called *hyperperiod* and equal in length to the least common multiple of the periods of the individual tasks, here denoted by H_τ. The processor does exactly the same thing at time t that it does at time $t + H_\tau$. If the processor is idle at t, it is idle at $t + H_\tau$ and if it works on a job of task τ_j at time t, it works on another job of task τ_j, released H_τ times units later, at time $t + H_\tau$.

A feasibility test can consist in simulating the behavior of EDF upon the periodic task set from time 0 to time $2H_\tau + \max_{1 \leq i \leq n} r_i + \max_{1 \leq i \leq n} D_i$. Thus, this exact test is exponential in the number of jobs issued from the periodic

requests.

For constrained-deadline task sets, the processor demand criterion was initially proposed for exact schedulability analysis by Baruah et al. [14]. The processor demand of the jobs generated by requests of periodic tasks in τ is given by Lemma 1 [14].

Lemma 1. *A periodic task set τ is feasible up to time t if and only if for any t_1, t_2 satisfying $t_1 < t_2 < t$,*

$$h_\tau(t_1, t_2) \leq t_2 - t_1 \tag{5}$$

with $h_\tau(t_1, t_2) = \sum_{i=1}^{n} N(\tau_i, t_1, t_2).C_i$

$N(\tau_i, t_1, t_2)$ is the number of requests of task τ_i which occur in the interval $[t_1, t_2)$ with a deadline less than or equal to t_2, namely $max\left(0, \lfloor \frac{t_2 - r_i - D_i}{T_i} \rfloor - \lceil \frac{t_1 - r_i}{T_i} \rceil + 1\right)$ requests over $[t_1, t_2)$. Any feasible scheduling algorithm must give at least $N(\tau_i, t_1, t_2)C_i$ processor time units to τ_i in this interval; this is a necessary condition for the schedulability of the task set in this interval.

Formula (5) enables us to compute processor demand in time complexity linear in the number of periodic tasks in τ. This interval is bounded by a certain value which guarantees we can find a deadline miss if the task set is not schedulable. Due to the resulting large number of computations because there could be a very large number of absolute deadlines that need to be checked, the test was enhanced later to reduce exponentially the calculation times [30] [17].

A pseudo-polynomial-time test was given by Baruah et al. (PPT test) [3] for a restricted class of task sets in which processor utilization is strictly less than one.

5. Scheduling in RTEH systems

5.1. The ED-H scheduling algorithm

As under EDF, task's deadlines are used in the scheduling decisions of ED-H to select the next task to be executed. However, ED-H[25] is not work-conserving

since the processor can be let idle even if there are pending tasks. The main principle of ED-H is to authorize job executions as long as no future starvation could occur. Energy starvation on a job, say J_i, can only be caused by a job, say J_j, which executes before the arrival of J_i such that $d_j > d_i$. Let us note that energy starvation of J_i caused by J_j with $d_j \leq d_i$ could not be avoided. Clairvoyance on both jobs arrivals and energy production permits ED-H to anticipate possible energy starvation.

We define the so-called *preemption slack energy* (resp. *slack time*) for current time t_c, denoted $PSE_{\mathcal{T}}(t_c)$ (resp. $ST_{\mathcal{T}}(t_c)$), as the maximum energy that could be consumed from t_c by the currently active job while preventing energy starvation for jobs that may preempt it (resp. the maximum processing time consumed from t_c while preventing deadline missing). In other terms, ED-H comes down to EDF except that the processor is idling whenever the preemption slack energy is zero.

The ED-H scheduler consists of more than just a single rule for choosing a ready job for execution. It also includes a rule for dynamically managing the processor activity by specifying the intervals in which the processor should remain on standby, and those on which it should execute a job. Let t_c be the current time. According to the ED-H scheduling algorithm, the processor cannot be active at t_c if either the energy storage unit is deplenished or executing any job would prevent at least one future occurring job from being executed timely because of energy starvation i.e. the system has no preemption slack energy. In addition, the processor cannot be inactive if either the energy storage unit is fully replenished or if making the processor idle would prevent at least one job from beeing executed timely because of time starvation i.e. the system has no slack time at t_c. When the storage unit is neither full nor empty and the system has both slack time and preemption slack energy, the scheduler may decide on the processor state.

More formally, considering that $L_r(t_c)$ denotes the list of jobs ready at time t_c, ED-H scheduling algorithm applies the following rules:

- **Rule 1:** The priorities assigned by EDF are used to select the next ready job in $L_r(t_c)$.
- **Rule 2:** The processor is on standby during $[t_c, t_c + 1)$ if $L_r(t_c) = \emptyset$.
- **Rule 3:** The processor is on standby during $[t_c, t_c + 1)$ if $L_r(t_c) \neq \emptyset$ and one of the following conditions is true:
  1. $E(t_c) \approx 0$
  2. $PSE_{\mathcal{T}}(t_c) \approx 0$
- **Rule 4:** The processor is busy during $[t_c, t_c + 1)$ if $L_r(t_c) \neq \emptyset$ and one of the following conditions is true:
  1. $E(t_c) \approx C$
  2. $ST_{\mathcal{T}}(t_c) = 0$
- **Rule 5:** The processor can be either busy or on standby if $L_r(t_c) \neq \emptyset$, $0 < E(t_c) < C$, $ST_{\mathcal{T}}(t_c) > 0$ and $PSE_{\mathcal{T}}(t_c) > 0$.

For simplicity, $E(t) \approx C$ stands for $C \leq E(t) < C + e_m$. In contrast, the energy storage is considered as fully discharged at time t if $0 \leq E(t) < e_m$ denoted by $E(t) \approx 0$. Rule 3 states that no job may be executed if the storage unit is empty, or if execution would lead to an unavoidable energy shortage due to insufficient preemption slack energy. Rule 4 indicates that the processor cannot remain idle if the storage unit is at full capacity, or if idling would cause a missed deadline as a result of zero slack time. If the storage unit is neither empty nor full, and if the system has non-zero slack time and non-zero preemption slack energy. Rule 5 states that the processor can be either on standby or busy without affecting the validity of the subsequent schedule. Note that with ED-H energy is only wasted if the storage unit is full and no jobs are ready.

The following theorem, proved in [25], states that ED-H is the best candidate for scheduling uniprocessor RTEH systems:

Theorem 3. *[25] The ED-H scheduling algorithm is optimal.*

Theorem 3 means that, if a given set of tasks cannot be feasibly scheduled with ED-H, the task set will be unschedulable by any other scheduler on the same platform. This property is general and still valuable whatever is the energy source profile.

5.2. Energy demand and slack energy

For the remainder of this section let us consider the RTEH model as defined in Section 3.2. Now, every job J_i in ω (possibly generated by the request of a periodic task in τ) is characterized by a fourth parameter, its energy requirement E_i. We need to introduce novel concepts - *the energy demand* and *the slack energy* - particularly helpful when analyzing the feasibility of a jobset ω with both energy and deadline constraints.

Definition 4. *The energy demand of jobset ω on the time interval $[t_1, t_2)$ is*

$$g_\omega(t_1, t_2) = \sum_{t_1 \leq a_k, d_k \leq t_2} E_k \tag{6}$$

The energy demand $g_\omega(t_1, t_2)$ of ω over the interval $[t_1, t_2)$ represents the cumulative energy required by jobs in ω that have arrival times at or after t_1 and deadlines at or before t_2.

Definition 5. *The static slack energy of jobset ω on the time interval $[t_1, t_2)$ is*

$$SSE_\omega(t_1, t_2) = C + E_p(t_1, t_2) - g_\omega(t_1, t_2) \tag{7}$$

$SSE_\omega(t_1, t_2)$ gives the largest energy that could be made available within $[t_1, t_2)$ after executing jobs of ω with arrival times at or after t_1 and deadlines at or before t_2.

Definition 6. *The static slack energy of jobset ω is*

$$SSE_\omega = \min_{0 \leq t_1 < t_2 \leq d_{Max}} SSE_\omega(t_1, t_2) \tag{8}$$

The static slack energy of ω represents the maximum energy that could be consumed or wasted from any time instant while still satisfying all the energy and timing constraints of ω.

5.3. ED-H schedulability test

In [25] we devise an exact schedulability test for ED-H, i.e. feasibility test by virtue of its optimality.

We prove that a jobset is feasible if and only if the processor demand on any interval does not exceed the length of the interval and the energy demand on any interval does not exceed the maximum energy available in the interval. The complexity of this test is $O(n^2)$ in the number of jobs.

Theorem 4. *[25] Jobset ω is feasible if and only if for any t_1, t_2 satisfying $t_1 < t_2$,*

$$h_\omega(t_1, t_2) \leq t_2 - t_1 \tag{9}$$

$$g_\omega(t_1, t_2) \leq C + E_p(t_1, t_2) \tag{10}$$

The feasibility test can also be expressed as follows:

Theorem 5. *[25] Jobset ω is feasible if and only if $SST_\omega \geq 0$ and $SSE_\omega \geq 0$.*

Theorem 5 permits to reduce the feasibility determination question to first one concerning the processor demand and separate second one concerning the energy demand.

5.4. Robustness considerations

Any feasibility analysis is carried out with respect to the performance parameters of a given platform. Let us recall that we focus on a real-time system in which the main concern is to guarantee the timing requirements of the jobs. When the source power is variable, the only method to certify feasibility is to apply a deterministic test, even if pessimistic. Consequently, the fundamental issue is to determine the most approaching bound so as to limit the negative

impact on the dimensioning of the energy harvester and that of the energy storage.

Let us recall that in the RTEH context, the platform is characterized by the speed of the processor, the capacity of the energy storage unit and the profile of the energy source i.e. the source power $P_p(t)$ for all time instants t from the initial time until the finishing time of the application. Moreover, this analysis assumes certain processing requirements and energy requirements coming from the jobs.

Here, we define *robustness* as the property of a scheduling algorithm/feasibility test that continues to meet all the deadlines if the source power $P_p(t)$ proves to be greater than assumed in the schedulability analysis, for every time t. In other words, a scheduling policy and/or a schedulability test for a scheduling policy is robust if any task set deemed schedulable by the test remains schedulable when the source power is increased.

If ED-H is proved to be robust, we can perform a feasibility analysis with a lower bound upon the instantaneous power drawn from the harvester at any instant. This is a fundamental issue for the following reasons:

- Many environmental energy sources including vibrations and radio frequency are unstable by nature. As they exhibit unpredictable fluctuations, no exact test can be performed off-line to guarantee schedulability at run time.

- It may be impossible for instrumentation to measure the source power with preciseness even if stable.

Theorem 6. *The feasibility test is robust with respect to source power.*

Proof:

Let us recall that $E_p(t_1, t_2) = \int_{t_1}^{t_2} P_p(t)dt$ is the amount of electrical energy collected by the HM between t_1 and t_2 with collected power given by function $P_p(t)$. The total energy harvested with power profile $P_p(t)$ such that $P_p(t)$ is then greater than $P'_p(t)$ for every time t is obviously greater than the total energy

harvested with power profile given by $P'_p(t)$. If inequality (14) in Corollary 1 given hereafter is satisfied with the value $P'_p(t)$, then it will be satisfied with any value $P_p(t)$ greater than $P'_p(t)$ for every time t. And this does not affect the time-feasibilty test which is decoupled from the energy feasibility test. Thus, the set of jobs will still be feasible. □

Theorem 6 says that any task set, periodic or not, which is schedulable by ED-H with a given energy harvester, will still be schedulable with any more powerful harvester. Consequently, the idea is to perform the feasibility test based on a worst-case scenario for which the energy harvester delivers a lower but unvariable power. As shown in the next section, the feasibility test then reduces to examine schedulability only in the first hyperperiod, for any periodic task set, synchronous or not. Clearly, the performance of the sufficient test will strongly depend on the accuracy of the lower bound on the power. The better the approximation, the better the schedulability test will perform in terms of exactness.

However, the power harvested from the environment may exhibit stochastic fluctuations at runtime and no lower bound on the harvested power can be computed off-line. Only an on-line admission test may be implemented in order to recalculate schedulability based on the prediction of a constant harvested power. In that case, reducing the online schedulability checking to one hyperperiod permits to limit both run-time overhead and memory utilization.

5.5. ED-H illustrative example

The ED-H schedule is illustrated in Figure 5 by considering the same jobs shown in Table 1, with energy requirements $E_1 = 5$, $E_2 = 1$ and $E_3 = 8$. We assume that the energy storage unit has a nominal capacity of 5 and is fully charged at time $t = 0$, i.e. $E(0) = C = 5$. We also suppose a non-uniform source power $P_p(t)$ such that the amount of energy produced flutuactes over time.

The energy demand and the static slack energy for jobset ω are reported in Table 3. Again, they are only computed in those intervals starting with an

arrival time and finishing with a deadline.

Table 3: Static slack energy computations for ω

$[t_1, t_2)$	$[0, 2)$	$[0, 6)$	$[0, 9)$	$[4, 6)$	$[4, 9)$
$g_\omega(t_1, t_2)$	1	6	14	5	5
$SSE_\omega(t_1, t_2)$	8	7	5	2	8

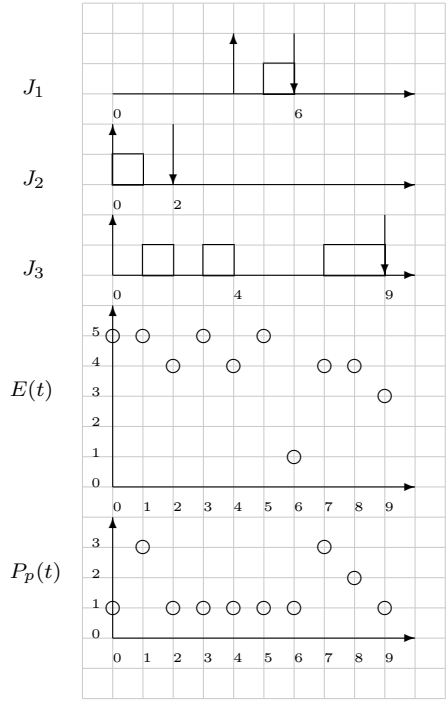

Figure 5: ED-H Schedule

Theorem 5 guarantees that jobset ω is feasible (i.e. schedulable by ED-H optimal scheduler) since we have both $SST_\omega \geq 0$ (see Table 2) and $SSE_\omega \geq 0$ (see Table 3).

This example underlines the capability of ED-H to cope with any energy production scenario. In fact, ED-H has been proved optimal for any application in which the profile of energy consumption may be different from the profile of energy production. This is particularly the case when we consider periodic tasks which consist of infinite sets of deadline constrained jobs with different arrival times. As the system is constrained both by time and energy, it requires an adaptive scheduler which takes into account jointly the two constraints. The role of ED-H is to define precisely the busy and idle time intervals of the processor so as to never waste or lack energy while executing all the tasks before deadlines.

Furthermore, this example illustrates the robustness property of the feasibility test (see Theorem 6). Let us consider a constant power profile with $\forall t$, $P_p(t) = 1$. It can be easily verified that with this more constrained energy supply, the test still guarantees the feasibility of the application. This implies that under *any* non-uniform power profile with an instantaneous power always greater than or equal to 1, the application will remain feasible.

6. The feasibility analysis problem

We shall now consider the feasibility problem for periodic task sets in the RTEH model. Since an optimal scheduler such as ED-H feasibly schedules any feasible task set (periodic or not), it is sufficient that one such algorithm be used for providing a feasibility test. Any periodic task set generates exactly one collection of jobs, and the feasibility problem is concerned with determining whether this collection of jobs can be scheduled to meet all deadlines under ED-H. We shall show that the feasibility checking can be done off-line when the energy profile is precisely characterized for all the application lifetime. Otherwise, the checking should be realized at runtime in dependance with the horizon of the prediction technique.

Definition 7. *For a periodic task set τ and any two real numbers t_1 and t_2 with $t_1 < t_2$, the energy demand $g_\tau(t_1, t_2)$ of τ over the interval $[t_1, t_2)$ is defined to be*

cumulative energy requirement by jobs generated by tasks in τ that have arrival times at or after t_1 and deadlines at or before t_2.

Let $N(\tau_i, t_1, t_2) = \max\left(0, \lfloor \frac{t_2 - r_i - D_i}{T_i} \rfloor - \lceil \frac{t_1 - r_i}{T_i} \rceil + 1\right)$. That is, $N(\tau_i, t_1, t_2)$ is the number of jobs of task τ_i which occur in the interval $[t_1, t_2)$ with a deadline less than or equal to t_2. Any feasible scheduling algorithm must give at least $N(\tau_i, t_1, t_2)C_i$ processing time units and $N(\tau_i, t_1, t_2)E_i$ energy units to τ_i in this interval.

Lemma 2. *For any t_1, t_2 satisfying $t_1 < t_2$,*

$$g_\tau(t_1, t_2) = \sum_{i=1}^{n} N(\tau_i, t_1, t_2)E_i \tag{11}$$

Proof:

The number of jobs generated by task τ_i that lie within the interval $[t_1, t_2)$ corresponds to the number of non-negative integers k that satisfy the inequalities $t_1 \leq r_i + kT_i$ and $t_2 \geq r_i + kT_i + D_i$. There are exactly $\max\left(0, \lfloor \frac{t_2 - r_i - D_i}{T_i} \rfloor - \lceil \frac{t_1 - r_i}{T_i} \rceil + 1\right)$ such k's. Each job of τ_i requires at most E_i energy units. The lemma follows since the cumulative energy required by all the tasks in τ is given by $\sum_{i=1}^{n} E_i.max\left(0, \lfloor \frac{t_2 - r_i - D_i}{T_i} \rfloor - \lceil \frac{t_1 - r_i}{T_i} \rceil + 1\right)$. $\square$

Lemma 2 gives a closed-form expression for computing the energy demand in time linear in the number of tasks.

Lemma 3. *For any t_1, t_2 satisfying $\max_{1 \leq i \leq n} r_i \leq t_1 < t_2$,*

$$g_\tau(t_1 + H_\tau, t_2 + H_\tau) = g_\tau(t_1, t_2) \tag{12}$$

Proof:

$g_\tau(t_1, t_2) = \sum_{i=1}^{n} N(\tau_i, t_1, t_2)E_i$ with $N(\tau_i, t_1, t_2) = \max\left(0, \lfloor \frac{t_2 - r_i - D_i}{T_i} \rfloor - \lceil \frac{t_1 - r_i}{T_i} \rceil + 1\right)$. Consequently, $g_\tau(t_1 + H_\tau, t_2 + H_\tau) = \sum_{i=1}^{n} N(\tau_i, t_1 + H_\tau, t_2 + H_\tau).E_i$ with $N(\tau_i, t_1 + H_\tau, t_2 + H_\tau) = \max\left(0, \lfloor \frac{t_2 + H_\tau - r_i - D_i}{T_i} \rfloor - \lceil \frac{t_1 + H_\tau - r_i}{T_i} \rceil + 1\right)$. Clearly, $N(\tau_i, t_1 + H_\tau, t_2 + H_\tau) = \max\left(0, \lfloor \frac{t_2 - r_i - D_i + H_\tau}{T_i} \rfloor - \lceil \frac{t_1 - r_i + H_\tau}{T_i} \rceil + 1\right)$. As H_τ is multiple of T_i for any $1 \leq i \leq n$, it follows that $N(\tau_i, t_1, t_2) + H_\tau) =$

$\max\left(0, \lfloor \frac{t_2 - r_i - D_i}{T_i} \rfloor + \frac{H_\tau}{T_i} - \lceil \frac{t_1 - r_i}{T_i} \rceil + 1\right) - \frac{H_\tau}{T_i}$. As $N(\tau_i, t_1 + H_\tau, t_2 + H_\tau) = N(\tau_i, t_1, t_2)$, we deduce that $g_\tau(t_1 + H_\tau, t_2 + H_\tau) = g_\tau(t_1, t_2)$. □

The energy demand $g_\tau(t_1, t_2)$ is a measure of the minimum energy required by the jobs of the periodic task set to comply with deadlines in the time interval $[t_1, t_2]$. Lemma 3 says that in all subsequent intervals $[t_1 + H_\tau, t_2 + H_\tau]$, $[t_1 + 2 \times H_\tau, t_2 + 2 \times H_\tau]$,..., etc., the energy demand will be the same.

Corollary 1. ***(Feasibility test)*** *An asynchronous periodic task set τ is feasible up to time t with the ED-H scheduler if and only if*

$$h_\tau(t_1, t_2) \leq t_2 - t_1 \tag{13}$$

$$g_\tau(t_1, t_2) \leq C + E_p(t_1, t_2) \tag{14}$$

for any t_1, t_2, t satisfying $0 < t_1 < t_2 < t$,

Proof:

This directly follows from Theorem 5. □

Inequality (13) (respectively (14)) in Corollary 1 is called time-feasibility condition respectively energy-feasibility condition.

Theorem 7. *Let τ be a periodic task set. If τ is schedulable in the synchronous case using the ED-H scheduler, then τ is schedulable in all asynchronous cases.*

Proof:

We show the theorem by contradiction and we assume that the task set τ is unschedulable using the ED-H scheduler with the offset assignment $r_1, ..., r_n$ with $\min_{0 \leq i \leq n} r_i = 0$ and $r_{Max} = \max_{0 \leq i \leq n} r_i$ and schedulable in the synchronous case. Since the asynchronous task set is not schedulable, it follows that a job misses its deadline (say at time t^*). Hence a deadline corresponds with time t^*. A deadline missing can be due to either time starvation (case 1) or energy

starvation (case 2).

Case 1: At time $t^* - 1$, the processor is busy executing a job with deadline less than or equal to t^*. Let us define time t_1 as follow: t_1 is the smallest instant less than t^* such that in the interval $[t_1, t^*)$ the processor is busy executing jobs with deadline less than or equal to t^*. Let us consider in this schedule only jobs which occur after or at time t_1: we have an asynchronous task set where all tasks are started after or at time t_1 with the same schedule from t_1 to t^* as before (see Figure 6). From this asynchronous task set, we construct a synchronous task set where all jobs are started at time t_1 with a deadline missing (see Figure 7). So, we consider an asynchronous system where each task τ_i starts its execution at time $r_i = t_1 + \delta_i$. $r_1 = t_1 + \delta_1$ is then the instant of the first request of task τ_1. Since in the interval $[t_1, t^*)$ the processor remains busy for requests whose deadlines occur before or at time t^*, this is also the case if we start τ_1 at time t_1 and the numbers of such requests is greater than or equal to the one in the previous situation. More precisely, $\sum_{i=1}^{n} k_i.C_i > t^* - t_1$ (where k_i represents the number of requests of τ_i occurring in the interval $[t_1, t^*)$ with a deadline less than or equal to t^*) if $r_1 = \delta_1$ as well if $r_1 = 0$ and for this reason an overflow occurs before or at time t^* in both situations. From a schedulability point of view the situation is worst and in this new asynchronous system an overflow

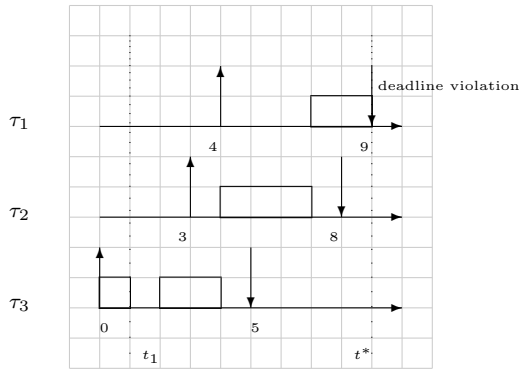

Figure 6: ED-H Schedule with a processing time starvation

occurs before or at time t^*. Repeating the same argument for all tasks, we construct a schedule for the ED-H scheduler where all tasks are started at time t_1, without processor idle time, with an overflow. This leads to a contradiction, since the new schedule from 0 is a ED-H schedule, for which there is no overflow.

Case 2: At time $t^* - \epsilon$ with $\epsilon \geq 0$, either the storage unit is deplenished, or the system has no more preemption slack energy. Let us define time t_1 as follow: t_1 is the smallest instant less than $t^* - \epsilon$ such that in the interval $[t_1, t^* - \epsilon)$ the storage unit is not empty and the processor is busy executing jobs with energy consumption less than or equal to that available in the storage unit. Let us consider in this schedule only jobs which occur after or at time t_1: we have an asynchronous task set where all tasks are started after or at time t_1 with the same schedule from t_1 to t^* as before (see Figure 8). From this asynchronous task set, we construct a synchronous task set where all jobs are started at time t_1 with an energy starvation. In this asynchronous system $r_1 = t_1 + \delta_1$ is the instant of the first request of task τ_1; since in the interval $[t_1, t^* - \epsilon)$ the processor remains busy for requests whose deadlines occur before or at time t^*, this is also the case if we start τ_1 at time t_1 and the total energy consumed by the requests is greater than or equal to the one in the previous situation. More precisely, $\sum_{i=1}^{n} k_i E_i > C + E_p(t_1, t^*)$ (where k_i represents the number of requests of τ_i occurring in the interval $[t_1, t^*)$ with a deadline less than or equal to t^*) if $r_1 = \delta_1$ as well if $r_1 = 0$ and for this reason an energy starvation occurs before or at time t^* in both situations. From a schedulability point of view the situation is worst and in this new asynchronous system an energy starvation occurs before or at time t^*. Repeating the same argument for all tasks, we construct a schedule for the ED-H scheduler where all tasks are started at time t_1, without processor idle time, with an energy starvation. This leads to a contradiction, since the new schedule from 0 is a ED-H schedule, for which there is no energy starvation. □

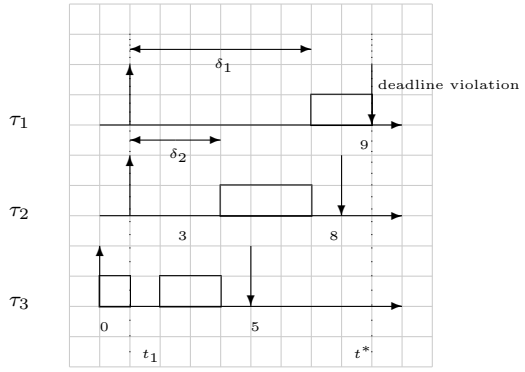

Figure 7: Reducing to a synchronous task set

In many applications, the initial release times of the periodic tasks are not known in advance because they depend on unpredictable events. Then, the feasibility decision can be performed based on the synchronous case. Theorem 7 expresses the fact that for the ED-H scheduler and from a schedulability point of view the synchronous case is the worst-case scenario. Consequently, in the case of asynchronous (or non-concrete) tasks for which the first release times of the tasks can be arbitrary, feasibility can be checked considering the worst-case synchronous scenario in which all the tasks are first released at the same time. However, for non-concrete asynchronous periodic task sets, the feasibility conditions are no longer necessary, but only sufficient.

In this section, we have established a feasibility test for periodic task sets that must be implemented at run-time and whose complexity depends on the variability characterization of the power profile. In most cases, we cannot predict the energy profile on a long-term interval. Consequently, we cannot implement the feasibility test in a pre-operational phase. One solution to that problem could be to approximate the energy supply by a worst-case value on the instantaneous incoming power. That's why the next section will focus on the feasibility analysis under the assumption of a constant source power.

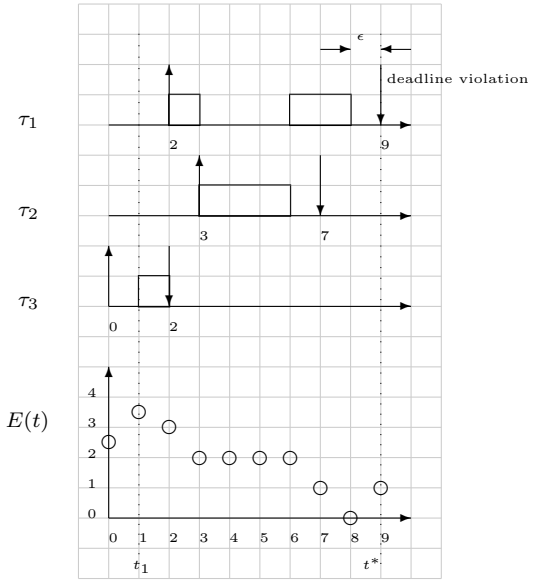

Figure 8: ED-H Schedule with an energy starvation

7. Feasibility test with constant source power

Solar energy can be harvested efficiently through photovoltaic conversion. The solar radiation fluctuates with time since it follows a diurnal cycle. Nevertheless, we may consider that the harvested power is constant on long time intervals, regarding periods of the tasks. In other words, a prediction technique can be applied to determine the constant power harvested during the next time interval in the order of multiple task hyperperiods. Energy can also be harvested from human body with stable power generation characteristics using thermoelectric generators. Those wearable devices allow to exploit temperature differences that exist between the environment and the surface of the human body while providing a constant and continuous power supply.

Let P_p denote the invariable power drawn from the environmental source assumed to be precisely known for a long term interval. We assume that the periodic task set τ satisfies $U_\tau^p \leq 1$ and $U_\tau^e \leq P_p$. In other terms, we suppose that the harvested power is sufficient to support the long term energy demand

of the periodic task set; Otherwise, deadline misses are unavoidable.

We now consider the following question: *how to decide if a synchronous periodic task set τ (such that $r_i = r_j$ for all integer $1 \leq i, j \leq n$) is feasible or not?* $H_\tau = lcm\{T_1, T_2, ..., T_n\}$ is an obvious time-bound within which to construct the schedule by an optimal scheduler such as EDF when energy is not limited. Nevertheless, it is not obvious that this time-bound still exists with the ED-H scheduler with energy harvesting considerations.

Example 2. *To understand better how ED-H behaves, let us consider a task set whose attributes are shown in Table 4. The energy storage unit has a capacity $C = 4$ and we consider a full reservoir at time zero i.e. $E(0) = C$. The energy production power is constant with $P_p = 1$. The resulting ED-H schedule on the first hyperperiod is depicted in Figure 9.*

Table 4: Taskset $\mathcal{T}$

Task	r_i	C_i	E_i	D_i	T_i
τ_1	0	3	6	7	20
τ_2	0	2	2	4	5
τ_3	0	1	2	8	10

Let us consider a feasible task set. We now prove that, if the energy storage is full at system initialization time, it will still be full at the end of the first hyperperiod, and that of all the next hyperperiods.

Lemma 4. *Let τ be a periodic task set with $U_\tau^p \leq 1$ and $U_\tau^e \leq P_p$. If the ED-H schedule produced within $[O, H_\tau)$ is valid, then $E(H_\tau) = C$.*

Proof:

We prove the lemma by contradiction. We assume all deadlines in the interval $[O, H_\tau)$ are met and $E(H_\tau) < C$. The total energy consumed by the task set

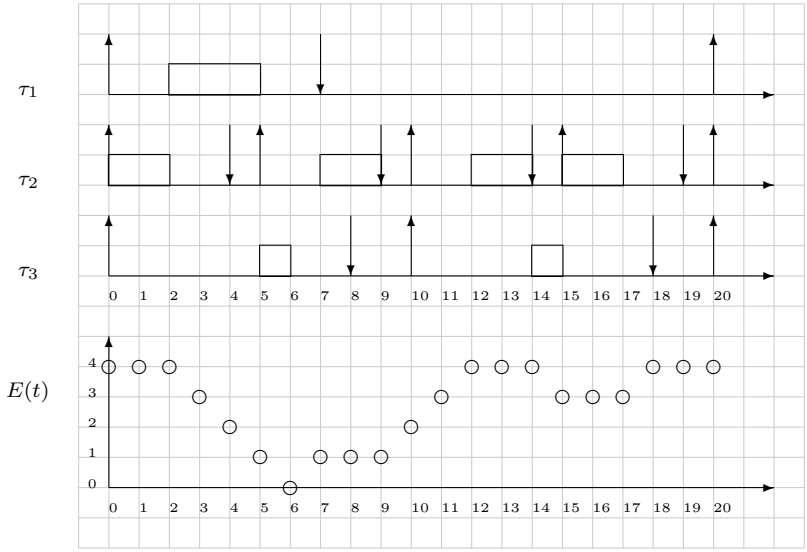

Figure 9: ED-H schedule on the first hyperperiod

within $[O, H_\tau)$ is $U_\tau^e H_\tau$. The task set τ satisfies $U_\tau^e \leq P_p$. This implies that at least one energy unit has been wasted. According to ED-H scheduling, energy can be wasted when there is no job waiting for execution and the energy storage unit is fully replenished. Let t_1 be the last time instant before time H_τ that satifies this condition. The total number of jobs in execution between t_1 and H_τ is given by $N(\tau_i, t_1, H_\tau)$ equal to $max\left(0, \lfloor \frac{H_\tau - t_1 - D_i}{T_i} \rfloor - \lceil \frac{H_\tau - t_1}{T_i} \rceil + 1\right)$. This quantity is less than or equal to $\frac{H_\tau - t_1}{T_i}$. Consequently, each task τ_i consumes at most $\frac{(H_\tau - t_1)E_i}{T_i}$ energy units in the interval $[t_1, H_\tau)$. The total energy consumed is bounded by $(H_\tau - t_1)\sum_{i=1}^{n} \frac{E_i}{T_i}$ which is less than or equal to $(H_\tau - t_1)P_p$ since $U_\tau^e \leq P_p$. The total energy harvested from the source in the interval $[t_1, H_\tau)$ is equal to $(H_\tau - t_1)P_p$. As no energy is wasted after time t_1, there is at least $E(t_1) = C$ energy units in the storage at time H_τ which contradicts that $E(H_\tau) < C$. □

Theorem 8. *The synchronous task set τ is feasible if and only if all deadlines in the interval $[O, H_\tau)$ are met under the ED-H scheduler.*

Proof:

(only if part). Obvious since the ED-H scheduler is optimal.

(if part). Let assume that the ED-H schedule produced within $[O, H_\tau)$ is valid. From Lemma 4, $E(H_\tau) = C$. For each positive integer j, let us define t_j as the time instant jH_τ. At time t_1, the system is exactly in the same situation that it is at time zero i.e. $E(t_1) = C$ and all the tasks have completed their execution since the schedule is valid. As the power harvested from the source is invariable, the schedule that will be produced by ED-H scheduling will be the same in the second hyperperiod $[t_1, t_1 + H_\tau)$ and we will have $E(t_2) = C$. So the schedule repeats indefinitely with period equal to H_τ. □

The immediate consequence of Theorem 8 is that the energy feasibility test reduces to check absence of energy shortage in one hyperperiod only as for the time feasibility test. Such a test is therefore of high practical relevance having a low-computational complexity implementation.

8. Summary and Future Work

Research into monoprocessor real-time scheduling and schedulability analysis of periodic tasks has markedly advanced since the seminal paper of Liu and Layland in 1973 [13]. Nonetheless, there are still significant and fundamental research challenges that remain into scheduling with energy harvesting considerations. Energy harvesting is an emerging technology that overcomes the limited lifetime of battery-operated devices. It allows continuous recharging of the energy storage from various environmental sources during use. Such autonomous devices are very tightly constrained in terms of size and weight and must additionnaly satisfy stringent real-time constraints. Feasibility testing is then a critical aspect of designing an energy harvesting system because it must be determined whether the available power output of the harvester is sufficient to satisfy all the constraints (i.e. timing and energy requirements) during the whole lifetime of the system.

In this paper, we have supposed that the available average power from the environmental energy source is sufficient to support the long-term power demand of the periodic task set. Otherwise, energy shortage with resulting deadline violations are unavoidable. We have formulated an efficient schedulability test for the optimal scheduler ED-H in case of periodic task sets. In addition, given an energy source which can be approximated by constant lower bound on the harvested power, we may determine whether all deadlines can be met on checking schedulability on one hyperperiod only. We also proved that under a constant harvested power, the synchronous scenario is the worst-case one as with no energy consideration. This enables us to apply an efficient sufficient schedulability test when initial task arrivals are unknown beforehand.

As a conclusion, we mention that we plan to investigate the feasibility of RTEH systems with: (i) DVFS (Dynamic Voltage and Frequency Scaling) capabilities and (ii) multiple cores. The problem of energy minimization has been investigated for these two models [8]. However, this is a new challenge to incorporate the issue of energy self-sufficiency which characterizes RTEH systems.

References

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

[2] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M.-B. Srivastava, Design Considerations for Solar Energy Harvesting Wireless Embedded Systems, *Proc. of the International Symposium on Information Processing in Sensor Networks*, Apr. 2005.

[3] X. Jiang, J. Polastre, and D.-E. Culler, Perpetual Environmentally Powered Sensor Networks, *Proc. of the International symposium on Information Processing in Sensor Networks*, Apr. 2005.

[4] Y. Shin, K. Choi, and T. Sakurai, Power optimization of real-time embedded

systems on variable speed processors, *Proc. of International Conference on Computer Aided Design* , pp. 365368, 2000.

[5] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, Power-aware scheduling for periodic real-time tasks, *Proc. of IEEE Transactions on Computers*, 2004.

[6] E. Bini, G. Buttazzo, and G. Lipari. Speed Modulation in Energy-Aware Real-Time Systems. *Proc. of the 17th Euromicro Conference on Real-Time Systems*. pp. 3-10, 2005.

[7] M. Bambagini, M. Bertogna, and G. Buttazzo, On the Effectiveness of Energy-Aware Real-Time Scheduling Algorithms on Single-Core Platforms. *Proc. of the 20th International Conference on Emerging Technology and Factory Automation*, pp. 1-8, 2014.

[8] K. Li, Energy-Efficient Task Scheduling on Multiple Heterogeneous Computers: Algorithms, Analysis, and Performance Evaluation, *IEEE Transactions on Sustainable Computing*, 1(1):. 7-19, 2016.

[9] Google Inc. 2017, *Project Loon*. https://www.google.com/loon/, 2017.

[10] K. Ryokai, P. Su, E. Kim, and B. Rollins, EnergyBugs: Energy Harvesting Wearables for Children. *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*. pp. 1039-1048. 2014.

[11] S. Baruah, S. Funk, and J. Goossens, Robustness Results Concerning EDF Scheduling upon Uniform Multiprocessors, *IEEE Transactions on Computers*, 52(9): 1185-1195, September 2003.

[12] M.-L. Dertouzos, Control Robotics: The Procedural Control of Physical Processes, *Proc. of International Federation for Information Processing Congress*, 1974.

[13] C.-L. Liu, J.-W. Layland, Scheduling algorithms for multiprogramming in a hard real-time environment, *J. of the Association for Computing Machinery*, 20(1): 46-61, 1973.

[14] S.-K. Baruah, L.-E. Rosier, R.R. Howell, Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic Real-Time Tasks on one Processor. *Real-Time Systems*, 2(4): 301-324, 1990.

[15] S.-K. Baruah, R.-R. Howell, and L.E. Rosier, Feasibility Problems for Recurring Tasks on One Processor. *Theoretical Computer Science*, 118(1): 3-20, 1993.

[16] M. Spuri, Analysis of Deadline Scheduled Real-Time Systems, *Technical Report RR-2772, INRIA*, Le Chesnay France, 1996.

[17] F. Zhang, A. Burns, Schedulability Analysis for Real-Time Systems with EDF Scheduling, *IEEE Trans. on Computers* 58(9): 1250-1258, 2009.

[18] B.-A. Allavena, A. Allavena, and D. Mossé, Scheduling of Frame-Based Embedded Systems with Rechargeable Batteries, *Proc. of Workshop on Power Management for Real-time and Embedded Systems*, May 2001.

[19] C. Moser, D. Brunelli, L. Thiele, L. Benini, Real-time scheduling for energy harvesting sensor nodes, *Real-Time Systems*, 37(3): 233-260, 2007.

[20] S. Liu, Q. Qiu, and Q. Wu, Energy Aware Dynamic Voltage and Frequency Selection for Real-Time Systems with Energy Harvesting, *Proc. of Design, Automation, and Test in Europe*, Mar. 2008.

[21] S. Liu, Q. Qiu, and Q. Wu, An Adaptive Scheduling and Voltage/Frequency Selection Algorithm for Real-time Energy Harvesting Systems, *Proc. of Design Automation Conference*, Jul. 2009.

[22] J. Hsu, S. Zahedi, A. Kansal, M. Srivastava, V. Raghunathan, Adaptive duty cycling for energy harvesting systems, *Proc. of the ISLPED Conference*, Oct. 2006.

[23] J. Recas, C. Bergonzini, D. Atienza, and T.-S. Rosing, Prediction and management in energy harvested wireless sensor nodes, *Proc. of the 1st International Conference on Wireless Communication, Vehicular Technology,*

Information Theory and Aerospace & Electronic Systems Technology, pp. 6-10, May 2009.

[24] J. Lu, S. Liu, Q. Wu and Q. Qiu, Accurate modeling and prediction of energy availability in energy harvesting real-time embedded systems, *Proc. of the International Conference on Green Computing*, pp. 469-476, August 2010.

[25] M. Chetto, Optimal Scheduling for Real-Time Jobs in Energy Harvesting Computing Systems, *IEEE Transactions on Emerging Topics in Computing*, 2(2): 122-133, June 2014.

[26] 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.

[27] M. Chetto, A. Queudet, Clairvoyance and Online Scheduling in Real-Time Energy Harvesting Systems, *Real-Time Systems Journal*, March 2014, 50(2): 179-184, March 2014.

[28] U.-C. Devi, An Improved Schedulability Test for Uniprocessor Periodic Task Systems, *Proceedings of the 15th Euromicro Conference on Real-Time Systems*, 2003.

[29] J.-Y.-T. Leung and M.-L. Merill, A Note on Preemptive Scheduling of Periodic, Real-Time Tasks, *Information Processing Letters*, 11(3): 115-118, 1980.

[30] I. Ripoll, A. Crespo, A.-K. Mok, Improvement in Feasibility Testing for Real-Time Tasks, *Journal of Real-Time Systems*, 11(1): 19-39, 1996.