1

Scheduling in Energy Autonomous Objects

In an autonomous system, in other words a system supplied during its entire lifetime by ambient energy, the issue of scheduling must be addressed in jointly taking into account the two physical constraints: time and energy. The fundamental scheduling questions can be raised as follows: is a scheduler as efficient, simple and high-performance as earliest deadline first (EDF) is appropriate? Is there, in this new context of perpetual energy autonomy, a scheduler which is optimal with acceptable implementation costs? How do we dimension the energy storage unit in such a way that no energy starvation, and therefore no deadline violation can occur at any time?

This chapter proposes to answer these questions according to the following plan:

– description of the real-time energy harvesting (RTEH) system model;

– study of the behavior of EDF for the RTEH model;

Chapter written by Maryline CHETTO.

– specification of the earliest deadline-harvesting (ED-H) scheduler, optimal for the RTEH model;

– description of a necessary and sufficient schedulability test.

1.1. Introduction

Electrical energy supply is a crucial issue, in particular in the design of portable systems that by nature have to be autonomous from an energy point of view. Today, this issue is mainly handled by dynamic voltage scaling (DVS) or dynamic power management (DPM) methods that aim to reduce the energy consumption of electronic circuits. Thus, the proposed solutions allow us to extend the durations separating two successive recharges of a battery without overcoming them.

However, the new generations of embedded systems, in particular those functioning in hostile or inaccessible environments, limit human intervention. They function with the help of batteries (or any other kind of energy storage unit), which are continuously recharged over time from a renewable energy source. There is no doubt that the DVS and DPM techniques prove to be very useful in autonomous systems: they lead to using lower capacity batteries, smaller solar panels, etc. But these techniques do not allow, by themselves, to ensure infinite operation, called *energy-neutral*. Energy-neutrality is defined here by the property of the embedded system to operate in such a way as to respect all of its timing constraints and this, by only using the energy available in the storage unit without ever lacking any.

An autonomous system is built around three components (see Figure 1.1):

– The *energy harvester* whose choice depends on the nature of the environmental energy, the amount of energy required, etc.

– The *energy storage unit,* such as a battery or a supercapacitor, whose choice depends on the dynamics of the system, the design constraints and/or cost constraints, etc.

– The *energy consumer* that here represents the execution support of the real-time tasks. In this chapter, we assume that the energy consumed by the operational part of the embedded system (actuator, LED, etc.) is separately powered, as is the transmitter/receiver module. Therefore, the energy consumer denotes the electronic card built around a microcontroller or a microprocessor.

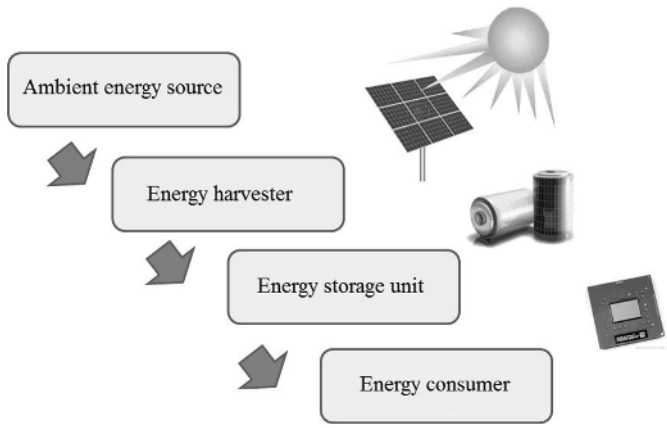

Figure 1.1. *Diagram of an ambient energy harvesting system*

Designing such a system requires the resolution of a certain number of issues related to the harvesting, storage and the use of ambient energy [PRI 09]. It has to be provided with a durable autonomy (from one to tens of years) while maintaining an acceptable real-time performance level. In this chapter, we focus on the consumer of energy, a machine whose energy needs are variable in time. These needs are required by the real-time tasks whose processing has to be done in predefined time intervals. Therefore, the energy needs are not identical and continuous over time. They depend on the timing profile of the tasks, very generally

characterized by a period and/or a deadline. An embedded system and mainly an autonomous intelligent sensor has to function during several years or even several tens of years without any possibility of intervention. This is why guaranteeing offline that it will respect its constraints is of importance. The implementation will be made difficult or even impossible by the uncertainty attached to the quantity of harvested energy. We can, therefore, see that the design of an autonomous system leads to several fundamental questions. Assuming that the energy supply is perfectly characterized (energy source profile, size of the storage battery, etc.), how do we verify and guarantee before the system becomes operational that it will have a continuous autonomy with an always acceptable performance level? This, therefore, means, first of all, to define this performance, often called Quality-of-Service, characterized by application constraints. In this chapter, we consider a firm real-time system whose performance level is mostly related to the percentage of jobs satisfying their deadlines.

From a software point of view, a real-time system is composed of application tasks and the real-time operating system (commonly referred to as RTOS) that ensures their scheduling. In Chapter 1, Volume 1 [CHE 14d], we have recalled the real-time schedulers typically implemented in current RTOSs. These schedulers have, for the most part, the particularity of being online, non-idling, priority-driven and preemptive. Their implementation does not lead to any major difficulty: one or more data structures organized in lists have to be managed. The role of the scheduler is to order these lists and update them, either using a fixed-priority policy such as rate monotonic or a dynamic-priority policy such as EDF [LIU 73]. However, these optimal schedulers offer their performance under the assumption that there is no energy limitation. Indeed, their optimality assumes that the processor has, at any time, the energy required for the execution of any job. Thus, we can see that the only constraint to be handled by the scheduler is a timing one.

Schedulability conditions associated with these schedulers are, therefore, centered around the utilization factor of the processor or the processor demand by time interval.

In an energy-autonomous system, the issue of scheduling is related to jointly taking into account the two physical constraints: time, which is measured in seconds and energy, which is measured in joules. The following fundamental questions are, therefore, raised: can an efficient and capable scheduler, such as EDF, be suitable for systems subject to, besides timing constraints, energy constraints? Are there, in this new context proper to renewable energy harvesting, schedulers which are at the same time optimal and easily implementable? The initial studies related to these questions date back to the 2000s [ALL 01].

1.2. Modeling and terminology

1.2.1. *System model*

Hereafter, we describe the RTEH model that comprises a computing element, a set of jobs, an energy storage unit, an energy harvesting unit and the environmental energy source (see Figure 1.2).

1.2.1.1. *Job model*

We consider a set of real-time jobs that is executed on a single processing unit. A single operating frequency is supported. We assume the energy consumed in the idle state to be negligible. The energy consumption comes integrally from dynamic switching. The jobs are executed by exclusively using the energy generated by the environmental source. We denote by $\tau = \{\tau_i, i = 1, \ldots, n\}$ the set of n preemptible jobs. The jobs are independent from one another. We associate the four-tuple (r_i, C_i, E_i, d_i) with the job τ_i. This job arrives at time r_i called release time, and requires a worst-case execution time of C_i time units and consumes E_i energy units

in the worst case. The quantity E_i is not necessarily proportional to C_i [JAY 06]. In other words, the effective energy consumption of a job does not vary linearly with its effective execution time. During each time unit, we know an upper bound on the energy consumption of every job equal to e_{Max} energy units. The exact amount of energy effectively drained in every time unit is, however, not known beforehand. The deadline of τ_i denoted by d_i represents the date at which τ_i has to have terminated its execution. We assume that $\min_{0 \leq i \leq n} r_i = 0$. Let $d_{Max} = \max_{0 \leq i \leq n} d_i$ and $D = \max_{0 \leq i \leq n} (d_i - r_i)$ be, respectively, the latest absolute deadline and the greatest relative deadline among those of the jobs of τ. $E_c(t_1, t_2)$ denotes the energy consumed by the jobs on the time interval $[t_1, t_2)$. If the energy consumed by a job in each time unit is no less than the energy harvested on this same time unit, we say that the job is discharging [ALL 01]. Every job of τ is discharging. Consequently, the residual capacity of the energy storage unit never increases every time a job executes.

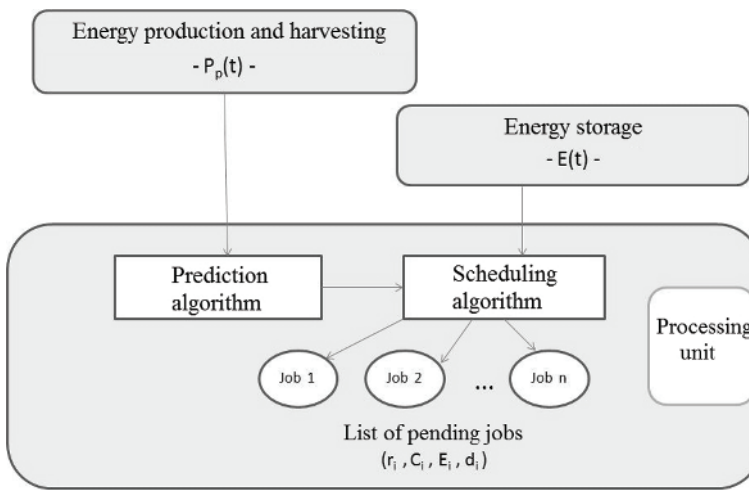

Figure 1.2. *The RTEH model*

1.2.1.2. *Energy production model*

The energy produced by the environmental source is assumed to be uncontrollable. We characterize it by a so-called *instantaneous charging rate* denoted by $P_p(t)$. This includes all the losses induced by the energy conversion and storage processes. The energy produced by the source on the time interval $[t_1, t_2)$ is denoted by $E_p(t_1, t_2)$ and is obtained by the following formula: $E_p(t_1, t_2) = \int_{t_1}^{t_2} P_p(t)dt$. We assume that the production and consumption of energy can occur simultaneously. Even though the power output at any time by the environmental source fluctuates with time, we assume to be able to predict it with precision in an immediate future and with negligible processing and energy costs.

1.2.1.3. *Energy storage model*

Our system uses an ideal energy storage unit with a nominal capacity denoted by C that is expressed in units of energy such as joule or watt per hour. The capacity may be less than the total energy consumption of a job. Let us denote by $E(t)$ the residual capacity of the storage unit at time t, which gives the current level of energy available.

We consider the energy to be wasted when the storage unit is fully charged while we continue to charge it. In contrast, the storage unit is considered fully discharged at time t if $0 \leq E(t) < e_{Max}$ denoted by $E(t) \approx 0$. The application starts with a fully charged storage unit (i.e. $E(0) = C$). The stored energy may be used at any later time and does not leak energy over time.

1.2.2. *Types of starvation*

According to the RTEH model described in the previous section, a job τ_i can miss its deadline if one of the two following situations occurs:

– *Time starvation*: when τ_i reaches its deadline at time t, its execution is incomplete because the time required to process τ_i by its deadline is not sufficient. However, there is enough energy in the storage unit when the deadline violation occurs, i.e. $E(t) > 0$.

– *Energy starvation*: when τ_i reaches its deadline at time t, its execution is incomplete because the energy required to process it by its deadline is not sufficient. The energy in the storage unit is exhausted when the deadline violation occurs, i.e. $E(t) \approx 0$.

1.2.3. *Terminology*

We now give definitions that we will be requiring throughout the remainder of the chapter.

DEFINITION 1.1.– *A schedule Γ for τ is said to be valid if the deadlines of all jobs of τ are respected in Γ starting with a full energy storage unit.*

DEFINITION 1.2.– *A system is said to be feasible if there exists at least one valid schedule for τ with a given energy storage unit and environmental energy source. Otherwise, the system is said to be infeasible.*

In infeasible systems, the limiting factors are either time, energy or both time and energy. In this chapter, we focus on feasible systems only. As in classical scheduling theory, we say that a scheduling algorithm is:

– *optimal* if it finds a valid schedule whenever one exists;

– *online* if it makes its decisions at run-time;

– *semi-online* if it is online with necessary lookahead on a certain time interval;

– *lookahead-ld* if it is semi-online with lookahead on ld time units;

– *idling* if it is allowed to keep the processor idle even when there are pending jobs. Otherwise, it is called *non-idling*;

– *clairvoyant* if it has knowledge of the future (characteristics of released jobs and energy production profile) at any time including the initial instant.

We introduce a terminology proper to the RTEH model.

DEFINITION 1.3.– *A schedule* Γ *for* τ *is said to be time-valid if the deadlines of all jobs in* τ *are met in* Γ*, considering that* $E_i = 0 \; \forall i \in \{1, \ldots, n\}$.

DEFINITION 1.4.– *A set of jobs* τ *is said to be time-feasible if there exists a time-valid schedule for* τ.

DEFINITION 1.5.– *A schedule* Γ *for* τ *is said to be energy-valid if the deadlines of all jobs in* τ *are met in* Γ*, considering that* $C_i = 0 \; \forall i \in \{1, \ldots, n\}$.

DEFINITION 1.6.– *A set of jobs* τ *is said to be energy-feasible if there exists an energy-valid schedule for* τ.

DEFINITION 1.7.– *A scheduling algorithm* A *is said to be energy-clairvoyant if it needs knowledge of the future energy production to take its run-time decisions.*

1.3. Weaknesses of classical schedulers

1.3.1. *Scheduling by EDF*

We show by a simple example that a conventional priority-driven real-time scheduler cannot build an optimal schedule for the RETH model. We consider the EDF algorithm, the most popular approach to schedule independent jobs in the absence of energy limitation and processing overload [LIU 73, DER 74]. EDF is an online scheduler that selects the ready job with the closest relative deadline. An online scheduling represents the only option in a

system whose processing overload is unpredictable, since it accommodates itself in an adaptive way to processor demand variations.

Let us give several useful definitions for the evaluation of the performance of online schedulers. The *value of a job* defines its contribution to the global performance of the system. The system obtains the value of the job if it terminates its execution before its deadline. Otherwise, the system obtains no value [BAR 91, BUT 05]. We say that an online scheduler has a *competitive factor* r ($0 < r < 1$) if it guarantees a total cumulated value of at least r times the value that the best clairvoyant scheduler may provide [BAR 92]. We say that an online scheduler is *competitive* if its competitive factor is strictly higher than 0. Otherwise, it is *non-competitive*.

The optimality of EDF remains true as long as we allow preemption between the jobs, and the jobs do not enter into competition for the access to shared resources. However, if a processing overload occurs, the optimality of EDF disappears. Let us consider the value of a job as being proportional to its execution time. Baruah *et al.* proved that no online scheduler, including EDF, can guarantee, in an overload situation, a competitive factor higher than 0.25 when jobs have uniform value densities [BAR 92]. A recent analysis shows that for the RTEH model, the competitive factor of EDF becomes zero.

THEOREM 1.1.– [CHE 14a] EDF is non-competitive for the RTEH model.

The EDF scheduler can be implemented under two distinct variants, called, respectively, as soon as possible (ASAP) and as late as possible (ALAP). It also possesses very important qualities: easy implementation, fast execution (low overhead) and reduced preemption rate. It is, therefore, natural to ask ourselves whether the EDF algorithm, even though

non-competitive, remains optimal or not when the jobs present energy needs in the context of RTEH.

1.3.2. *ASAP strategy*

The EDF variant that applies the ASAP policy is also called Earliest Deadline as Soon as possible (EDS). It consists of immediately rendering the highest priority ready job executable. We illustrate below this non-idling scheduling policy.

Example: let us consider a time-feasible set of two jobs τ_1 and τ_2 with $\tau_1 = (0, 4, 32, 9)$ and $\tau_2 = (2, 3, 24, 5)$. We harvest energy from the environment with the same power over time, $P_p = 6$. The energy storage unit has a capacity of $C = 8$ and is initially full ($E_c(0) = 8$). We notice that τ_2 misses its deadline at time 5 (see Figure 1.3). The weak point of EDS resides in its greedy way of consuming energy, which leads to emptying the energy storage unit and preventing the job τ_2 to be completely executed. This example shows that, despite a sufficient amount of energy and processing capacity, the non-idling schedulers commonly integrated into existing operating systems turn out to be incapable of efficiently dealing with energy limitations. Theorem 1.2 establishes that the EDF scheduler remains, however, the best non-idling scheduler for the RTEH model.

THEOREM 1.2.– [CHE 14a] EDF is optimal in the class of non-idling schedulers for the RTEH model.

1.3.3. *ALAP strategy*

In symmetry with the ASAP variant of EDF, let us examine the behavior of the ALAP variant, also called earliest deadline as late as possible (EDL). This consists of postponing the execution of the jobs as much as possible while respecting their deadlines. EDL is particularly adapted

to situations in which the processor must be put in an idle state for as long as possible [CHE 89]. The idea of delaying the execution of the jobs of critical periodic tasks permits us to recover the availability of the processor and to serve non-critical aperiodic tasks as soon as possible for minimizing their response time [CHE 99].

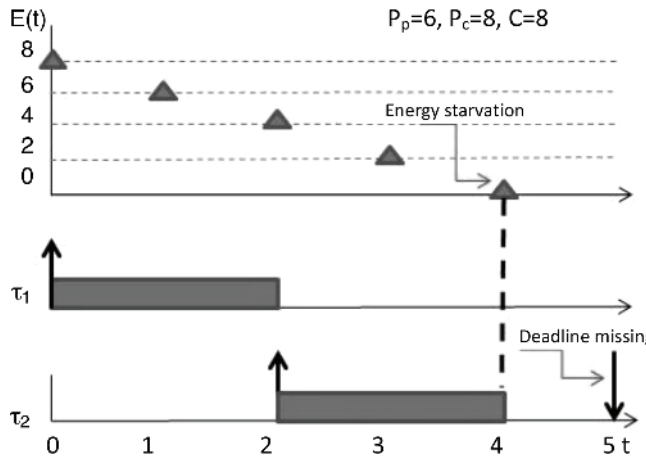

Figure 1.3. *Schedule produced by EDF with ASAP*

Example: We go back to the previous example and apply the ALAP variant of EDF (see Figure 1.4).

At the release of τ_1, its latest starting time for ALAP is computed, equal here to 5 and given by its deadline minus its execution time. The processor, in the idle state from instant 0, therefore, does not consume any energy, which allows the storage unit to fill up until time 1. The storage unit has then reached its full capacity. Thus, the energy harvested after time 1 is wasted. At time 2, the job τ_2 is released. Its latest starting time corresponds to its release time. τ_2 is executed from time 2 to time 5. Hence, this is causing a discharge of the storage unit which, at time 5, contains 2 units of energy. τ_2 starts its execution, which leads to completely emptying

the storage unit at time 6, when the application is prematurely stopped.

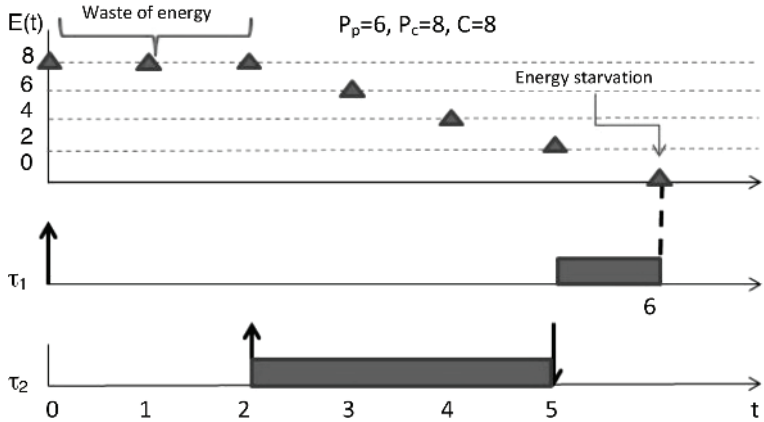

Figure 1.4. *Schedule produced by EDF with ALAP*

Contrarily to EDS, EDL does not turn out to be greedy enough, since it delays the energy consumption of the jobs. This first of all leads to a waste given the limited capacity of the storage unit that cannot store all the harvested energy. This is followed by an energy starvation depriving the job from terminating its execution before its deadline even though it has enough time to do so.

From a practical point of view and with regard to simplicity, EDF remains a first-choice scheduler for systems powered by renewable energy. Its integration does not require any particular technological device: it does not need to know the current level of energy in the storage unit, and to complete a predictive estimation of the harvested energy.

1.4. Fundamental properties

From the above, we conclude that every energy starvation (i.e. the situation in which the available energy proves to be insufficient to execute a job with respect to its deadline) has

to be anticipated sufficiently in advance in order to prevent it from exceeding a deadline. In other words, the optimality of a scheduler assumes a clairvoyance capacity.

THEOREM 1.3.– [CHE 14a] A non-clairvoyant online scheduling algorithm cannot be optimal for the RTEH model.

Thus, theorem 1.3 indicates that having a prediction on a part of the future may help in building a better schedule than that produced by a totally non-clairvoyant scheduler such as EDF. Moreover, theorem 1.4 establishes a lower bound on the omniscience of the scheduler in order for it to build an energy-valid schedule.

THEOREM 1.4.– [CHE 14a] Let D be the largest relative deadline of the application. No lookahead-ld online scheduling algorithm is optimal for the RTEH model if $ld < D$.

Thus, theorem 1.4 gives us the clairvoyance horizon required by any optimal scheduler. The value of the largest relative deadline appears as a key parameter in the application. If, for a given application we cannot predict the harvested energy profile on a time interval of length equal to at least this relative deadline, then it is illusory to benefit from an optimal online scheduler.

Estimating the amount of energy drained from the environment on a given time interval constitutes a central issue on the design of an RTEH system. The source of environmental energy can be formally modeled or even precisely determined offline in certain applications. However, when it is uncontrollable and highly unstable, only the prediction techniques applied online and cyclically on timing sliding windows allow us to determine lower bounds on the future harvested energy.

1.5. Concepts related to energy

1.5.1. *Processor demand*

The processor demand on $[t_1, t_2)$ is defined by the amount of execution time required by all jobs with release time at or after t_1 and deadline before or at t_2 (see definition 1.8). When the set of jobs τ under-utilizes the processor, the result is a residual processing availability and a timing flexibility in the execution of the jobs, hence the notion of slack time.

DEFINITION 1.8.– *The processor demand of a set of jobs τ on the time interval $[t_1, t_2)$ is given by*

$$h(t_1, t_2) = \sum_{t_1 \leq r_k, d_k \leq t_2} C_k \qquad [1.1]$$

The schedulability analysis of EDF based on the so-called processor demand approach needs to compute the processor demand for every time interval starting with a release time and finishing with a deadline. We then verify whether there is a processing overload on each tested interval. This approach comes down to compute the *static slack time* denoted by $SST_\tau(t_1, t_2)$ (see definition 1.9). For applications in which the jobs arrive in an unpredictable manner, the schedulability analysis takes the form of an online test (we then refer to an admission test) in such a way as to decide whether to accept or reject a new job [BUT 05].

DEFINITION 1.9.– *The static slack time of a set of jobs τ on the time interval $[t_1, t_2)$ is given by*

$$SST_\tau(t_1, t_2) = t_2 - t_1 - h(t_1, t_2) \qquad [1.2]$$

$SST_\tau(t_1, t_2)$ represents the longest duration of the interval included within $[t_1, t_2)$ during which the processor can remain inactive while guaranteeing the execution of the jobs of τ released at or after t_1 and with deadline at most equal to t_2. From this, we deduce the static slack time of the set τ.

DEFINITION 1.10.– *The static slack time of a set of jobs τ, SST_τ, is given by*

$$SST_\tau = \min_{0 \leq t_1 < t_2 \leq d_{Max}} SST_\tau(t_1, t_2) \quad [1.3]$$

Let t_c be the current time in the schedule produced for the set of jobs τ by a certain scheduling algorithm. Let AT_i be the remaining execution time of the uncompleted jobs at time t_c with deadlines smaller than or equal to d_i.

DEFINITION 1.11.– *The slack time of the job τ_i at the current time t_c is given by*

$$ST_{\tau_i}(t_c) = d_i - t_c - h(t_c, d_i) - AT_i \quad [1.4]$$

$ST_{\tau_i}(t_c)$ represents the total amount of processor time available in $[t_c, d_i)$ after having executed all the jobs with deadlines smaller than or equal to d_i. We can define the slack time of τ at the current time t_c as follows:

DEFINITION 1.12.– *The slack time of the set of jobs τ at the current time t_c is given by*

$$ST_\tau(t_c) = \min_{d_i > t_c} ST_{\tau_i}(t_c) \quad [1.5]$$

The slack time as computed with [1.5] represents the maximum continuous processor time that could be available from t_c during which the processor could remain inactive or execute other jobs than those of the set τ. The computation of $ST_\tau(t_c)$ uses the construction of the EDL schedule from time t_c initially described in [CHE 89].

1.5.2. *Energy demand*

We introduce here new concepts for the feasibility analysis of a set of jobs characterized by their energy needs. Let $E_p(t_1, t_2)$ be the amount of energy harvested between t_1 and t_2.

DEFINITION 1.13.– *The energy demand of a set of jobs τ on the time interval $[t_1, t_2)$ is given by*

$$g(t_1, t_2) = \sum_{t_1 \leq r_k, d_k \leq t_2} E_k \quad [1.6]$$

DEFINITION 1.14.– *The static slack energy of a set of jobs τ on the time interval $[t_1, t_2)$ is given by*

$$SSE_\tau(t_1, t_2) = C + E_p(t_1, t_2) - g(t_1, t_2) \quad [1.7]$$

$SSE_\tau(t_1, t_2)$ represents the maximum amount of energy available during the time interval $[t_1, t_2)$ and this, while guaranteeing the execution of the jobs of τ released at or after t_1 and with deadlines smaller than or equal to t_2. We can then define the static slack energy of τ as follows:

DEFINITION 1.15.– *The static slack energy of a set of jobs τ is given by*

$$SSE_\tau = \min_{0 \leq t_1 < t_2 \leq d_{Max}} SSE_\tau(t_1, t_2) \quad [1.8]$$

The static slack energy of τ represents the energy surplus that could be consumed at any time while guaranteeing the energy needs of the jobs of τ.

DEFINITION 1.16.– *The slack energy of a job τ_i at the current time t_c is given by*

$$SE_{\tau_i}(t_c) = E(t_c) + E_p(t_c, d_i) - g(t_c, d_i) \quad [1.9]$$

$SE_{\tau_i}(t_c)$ represents the maximum amount of energy consumed in $[t_c, d_i)$ while guaranteeing the energy needs of the jobs released from time t_c with deadlines smaller than or equal to d_i. If there is a job τ_i such that $SE_{\tau_i}(t_c) = 0$, then the execution between t_c and d_i of any job with a deadline higher than d_i will provoke an energy starvation for τ_i. We can now describe ED-H, the optimal scheduler for the RTEH model.

1.6. ED-H scheduling

1.6.1. *Informal description*

The intuition behind the ED-H scheduler is to run jobs according to the EDF rule in time intervals determined following energy constraints. A job is only allowed to be executed after having verified that its execution during a time unit will not lead to energy starvation neither for the job nor for a job that will arrive in the future. ED-H does not correspond either to EDS, nor to EDL. This scheduler is, therefore, based on the timing and energy characteristics of the jobs as well as on the replenishment rate of the storage unit to make decisions concerning the state of the processor. Schematically, ED-H constitutes a variant of EDF that could be qualified as energy aware, since it is capable of preventing energy starvation.

The conventional EDF scheduler is said to be greedy since it executes systematically jobs as soon as possible, and thus spends the energy stored in the storage unit disregarding needs of future occurring jobs. Let us consider a set of jobs that is time-feasible by EDF. The energy starvation for a job τ_i can only come from the execution of a job τ_j that is executed before the arrival of τ_i with $d_j > d_i$. Indeed, the energy starvation of τ_i caused by τ_j with $d_j \leq d_i$ cannot be avoided by any other scheduler. Intuitively, a minimum of clairvoyance relative to the arrival of jobs and to the production of energy will help EDF to anticipate an energy starvation, and consequently a deadline miss. The key principle of ED-H consists of allowing the execution of jobs while no energy starvation can occur. We are, therefore, led to introducing the concept of *preemption slack energy* at the current time t_c as the largest quantity of energy consumable by the active job that does not put into question the feasibility of the jobs susceptible to preempt it.

DEFINITION 1.17.– *Let d be the deadline of the active job at time t_c. The preemption slack energy of the set τ at time t_c is given by*

$$PSE_\tau(t_c) = \min_{t_c<r_i<d_i<d} SE_{\tau_i}(t_c) \qquad [1.10]$$

1.6.2. *Rules of ED-H*

Let $L_r(t_c)$ be the list of jobs ready for execution at time t_c. The ED-H scheduling algorithm respects the following rules:

– *Rule 1:* the EDF priority order is used to select the future running job in $L_r(t_c)$.

– *Rule 2:* the processor is imperatively idle in $[t_c, t_c + 1)$ if $L_r(t_c) = \emptyset$.

– *Rule 3:* the processor is imperatively idle in $[t_c, t_c + 1)$ if $L_r(t_c) \neq \emptyset$ and one of the following conditions is satisfied:

1) $E(t_c) \approx 0$;

2) $PSE_\tau(t_c) \approx 0$.

– *Rule 4:* the processor is imperatively busy in $[t_c, t_c + 1)$ if $L_r(t_c) \neq \emptyset$ and one of the following conditions is satisfied:

1) $E(t_c) \approx C$;

2) $ST_\tau(t_c) = 0$.

– *Rule 5:* the processor can equally be idle or busy if $L_r(t_c) \neq \emptyset$, $0 < E(t_c) < C$, $ST_\tau(t_c) > 0$ and $PSE_\tau(t_c) > 0$.

Rule 3 states that no job can be executed if the energy storage unit is empty or if this execution unavoidably leads to energy starvation, the preemption slack energy being insufficient. Rule 4 states that the processor cannot be inactive if either the energy storage unit is fully replenished or making the processor idle would lead to missing a deadline

due to a zero slack time. If the storage unit is neither empty nor full and if the system has a non-zero slack time and a non-zero preemption slack energy, rule 5 states that the processor can equally take the idle or busy state without compromising the validity of the resulting schedule. Note that according to ED-H, a waste of energy is only produced when the storage unit is full and no job is waiting to be executed.

This description of ED-H does not mention the particular situation in which the storage unit is full ($C \leq E(t_c) < C + e_{Max}$) with a zero preemption slack energy ($0 \leq PSE_\tau(t_c) < e_{Max}$). In order to avoid a waste of energy by putting the processor into an idle state, the execution of the highest priority job in $[t_c, t_c + 1)$ may be allowed, leading then to an energy consumption equal to at most e_{Max} units of energy. The processor then remains passive during a sufficient amount of time for the storage unit to be fully replenished. Thus, ED-H provokes a continuous switching from a busy state to an idle state in such a way that over this period the consumption of energy remains the same as the production of energy. The result is an energy waste of at most e_{Max} units.

Various implementations can be taken from ED-H depending on the choice of rule 5. ASAP and ALAP are the only special cases that can be reduced to executing jobs either systematically at the earliest as soon as the energy proves to be sufficient or at the latest without, however, provoking an overflow of the energy storage unit. The rule selected for deciding when to start and when to stop the recharging phase of the storage unit determines the variant of ED-H. Thus, we will be able to choose to execute jobs while the energy level is above a certain threshold and leaving the processor inactive in order to replenish the storage unit while its level has not reached a predefined upper value. The ED-H scheduler, therefore, presents a great flexibility in its implementation

with the only conditions that we forbid at all times the waste of energy and we prevent the system from a negative slack time and a negative preemption slack energy.

Example: let us go back to the previous example. Equation [1.9] gives us

$$SE_{\tau_2}(0) = E(0) + E_p(0, d_2) - g(0, d_2) = 8 + 30 - 24 = 12.$$

$SE_{\tau_2}(0)$ represents the maximum quantity of energy that any active job may consume from instant zero in order to preserve to feasibility of τ_2. τ_1 is allowed to consume at most 12 units of energy and is stopped at time 1, since $SE_{\tau_2}(1) \approx 0$. Rule 3 imposes that the processor be put into an idle state until the storage unit is again full or that the slack time becomes zero. Note the simultaneous fulfillment of these two conditions at time 2. $ST_{\tau_2}(1)$ is given by $d_2 - 1 - h(2, d_2) - AT_2$. Consequently, $ST_{\tau_2}(1) = 1$. $ST_{\tau_1}(1)$ is given by $d_1 - 1 - h(2, d_1) - AT_1 = 9 - 1 - (3 + 3) - 1$. $ST_{\tau_1}(1) = 3$. From formula [1.5], $ST_{\tau}(1) = 1$. The processor switches back to the active state at time 2 when $E(2) = 8$. τ_2 is executed until its completion at time 5 where $E(5) = 2$. τ_1 resumes its execution until the storage unit is empty at time 6. The processor becomes inactive in order to refill the storage unit until time 7, the time when the slack time of the system becomes zero, imposing the execution of τ_1 that is thus terminated at time 9, its deadline where $E(9) = 2$ (see Figure 1.5).

1.6.3. *Optimality analysis*

The optimality of ED-H means that if ED-H is unable to build a valid schedule for a set of jobs τ, then no other scheduler will be able to. The proof of optimality is based on lemmas 1.1 and 1.2 in which we assume that the deadline d_1 of the job τ_1 is the first missed deadline in the ED-H schedule produced for the set τ.

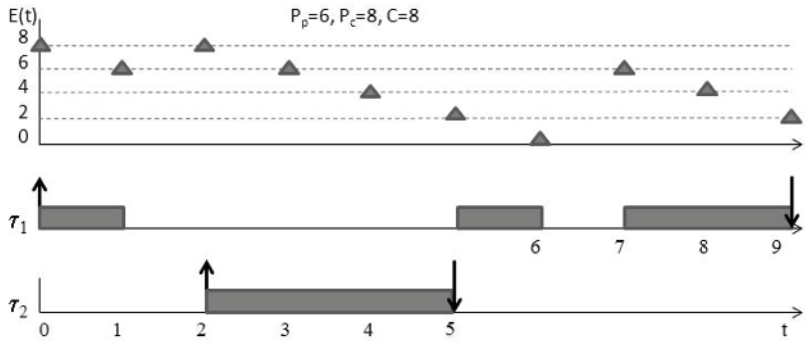

Figure 1.5. *Schedule produced by ED-H*

LEMMA 1.1.– [CHE 14c] If d_1 is missed in the ED-H schedule because of time starvation, then there exists a time instant t such that $h(t, d_1) > d_1 - t$ and no schedule exists where d_1 and all earlier deadlines are met.

LEMMA 1.2.– [CHE 14c] If d_1 is missed in the ED-H schedule because of energy starvation, then there exists a time instant t such that $g(t, d_1) > C + E_p(t, d_1)$ and no schedule exists where d_1 and all earlier deadlines are met.

The ED-H scheduler produces a valid schedule as long as there are no time intervals where the processor demand on this interval exceeds its duration and the energy demand exceeds the total available energy in this interval. Lemmas 1.1 and 1.2 then lead us to theorem 1.5.

THEOREM 1.5.– [CHE 14c] The ED-H scheduling algorithm is optimal for the RTEH model.

ED-H provides an optimal solution that is less restrictive than the lazy scheduling algorithm (LSA) algorithm described in [MOS 07]. In these works, the energy consumed by any job being executed varies linearly with its execution time.

1.6.4. *Clairvoyance analysis*

In accordance with the result outlined in theorem 1.4, we know that no online scheduling algorithm can be optimal without a clairvoyance of at least D time units. In order to make a decision at any time t_c, ED-H requires to know at the same time the arrival process of the jobs and the energy production process recovered over the following D time units, hence theorem 1.6.

THEOREM 1.6.– [CHE 14c] The ED-H scheduling algorithm is lookahead-D.

The main technological limitation associated with the implementation of ED-H is due to the estimated measure of the energy harvested over D time units. This problem is handled by targeted prediction methods that depend on the energy source.

1.6.5. *Schedulability test*

The essential question "is the set τ feasible?" refers to that of the schedulability of τ by ED-H. It has to be verified by a simple test whether there exists a valid ED-H schedule for τ, given an energy storage unit characterized by its capacity and an energy harvesting system characterized by an instantaneous production power $P_p(t)$. Theorem 1.7 shows that this feasibility test is reduced to two independent tests, one relative to the timing feasibility and the other to the energy feasibility. In other words, we show that τ is feasible if and only if τ is time-feasible and energy-feasible.

THEOREM 1.7.– [CHE 14c] A set of jobs τ compliant with the RTEH model is feasible if and only if

$$SST_\tau \geq 0 \text{ and } SSE_\tau \geq 0 \qquad [1.11]$$

This feasibility test is implemented in $O(n^2)$, since n^2 time intervals are the object of a static slack time calculation. Let us assume that we predict the ambient energy on each time interval by a finite number of values. We then show that the complexity of the energy feasibility test is in $O(n^2)$.

Example: we go back to the previous example and apply theorem 1.7 in order to test the feasibility of the two jobs τ_1 and τ_2. Since $SST_\tau(0,9) = 2$ and $SSE_\tau = 6$, we deduce that the set τ is feasible and consequently feasibly schedulable by ED-H.

1.7. Conclusion

The technology known as *energy harvesting* consists of generating electrical energy from the environment. This technology is becoming an undeniable asset for the development of autonomous communication devices, with as much regards to civil applications as military defense applications. Energy harvesting is turning out to be potentially attractive and promising. Nevertheless, its implementation assumes to resolve numerous theoretical and technological issues relative to the harvesting, conversion, storage and consumption of energy. We do not seek here to minimize the energy consumption in order to maximize the lifetime of the system (low-power technology) as in traditional portable devices. An *energy-neutral* mode of functioning has to be ensured in which the system never consumes more energy than it harvests. More precisely, the question that we have provided an answer for, is formulated as follows: how do we schedule the jobs in order to continuously respect their timing constraints by an adequate exploitation of the *processor* resource and the *ambient energy* resource?

This assumes providing the operating system with DPM functions capable of adapting the energy consumption of the

processor to the energy production profile and respecting the deadline constraints attached to the jobs.

In this chapter, we have restricted our study to a monofrequency uniprocessor platform. We have presented an optimal scheduler, ED-H, a variant of the EDF scheduler. Energy autonomous real-time systems are also the subject of studies that relate to platforms equipped with DVFS functionalities [LIU 08, LIU 09, LIU 12], to fixed-priority driven systems [ABD 13] or to multiprocessor architectures [LU 11].

1.8. Bibliography

[ABD 13] ABDEDDAIM Y., CHANDARLI Y., MASSON D., "The optimality of PFPasap algorithm for fixed-priority energy-harvesting real-time systems", *25th Euromicro Conference on Real-Time Systems*, 2013.

[ALL 01] ALLAVENA A., MOSSE D., "Frame-based embedded systems with rechargeable batteries", *Workshop on Power Management for Real-Time and Embedded Systems*, 2001.

[BAR 91] BARUAH S., KOREN G., MISHRA B., *et al.*, "Online scheduling in the presence of overload", *Symposium on Foundations of Computer Science*, 1991.

[BAR 92] BARUAH S., KOREN G., MAO D., *et al.*, "On the competitiveness of on-line real-time job scheduling", *Real-Time Systems*, vol. 4, no. 2, pp. 125–144, 1992.

[BUT 05] BUTTAZZO G., *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, Springer, 2nd. ed., 2005.

[CHE 89] CHETTO H., CHETTO M., "Some results of the earliest deadline scheduling algorithm", *IEEE Transactions on Software Engineering*, vol. 15, no. 10, pp. 1261–1270, 1989.

[CHE 99] CHETTO-SILLY M., "The EDL server for scheduling periodic and soft aperiodic tasks with resource constraints", *Real-Time Systems*, vol. 17, no. 1, pp. 87–111, 1999.

[CHE 14a] CHETTO M., QUEUDET A., "Clairvoyance and online scheduling in real-time energy harvesting systems", *Real-Time Systems*, vol. 50, no. 2, pp. 179–184, 2014.

[CHE 14b] CHETTO M., QUEUDET A., "A note on EDF scheduling for real-time energy harvesting systems", *IEEE Transactions on Computers*, vol. 63, no. 4, pp. 1037–1040, 2014.

[CHE 14c] CHETTO M., "Optimal scheduling for real-time jobs in energy harvesting computing systems", *IEEE Transactions on Emerging Topics in Computing*, 2014.

[CHE 14d] CHETTO M., (ed.), *Real-time Systems scheduling 1: Fundamentals, ISTE*, London and John Wiley & Sons, New York, 2014

[DER 74] DERTOUZOS M.-L., "Control robotics: the procedural control of physical processes", *International Federation for Information Processing Congress*, 1974.

[JAY 06] JAYASEELAN R., MITRA T., LI X., "Estimating the worst-case energy consumption of embedded software", *12th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2006.

[LIU 73] LIU C.-L., LAYLAND J.-W., "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.

[LIU 08] LIU S., QIU Q., WU Q., "Energy aware dynamic voltage and frequency selection for real-time systems with energy harvesting", *Design, Automation and Test in Europe*, 2008.

[LIU 09] LIU S., WU Q., QIU Q., "An adaptive scheduling and voltage/frequency selection algorithm for real-time energy harvesting systems", *ACM/IEEE Design Automation Conference*, 2009.

[LU 11] LU J., QIU Q., "Scheduling and mapping of periodic tasks on multi-core embedded systems with energy harvesting", *Conference on Green Computing*, 2011.

[LIU 12] LIU S., LU J., WU Q., *et al.*, "Harvesting-aware power management for real-time systems with renewable energy", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 8, pp. 1473–1486, 2012.

[MOS 07] MOSER C., BRUNELLI D., THIELE L., *et al.*, "Real-time scheduling for energy harvesting sensor nodes", *Real-Time Systems*, vol. 37, no. 3, pp. 233–260, 2007.

[PRI 09] PRIYA S., INMAN D.-J., *Energy Harvesting Technologies*, Springer-Verlag, 2009.