Energy-aware schedulers for Real-Time Energy Harvesting systems with Quality of Service requirements

Maissa Abdallah, Maryline Chetto, Audrey Queudet

Energy-aware schedulers for Real-Time Energy Harvesting systems with Quality of Service requirements

Maissa Abdallah, Maryline Chetto and Audrey Queudet
IRCCyN
LUNAM University
Nantes, France
e-mail: Firstname.Lastname@irccyn.ec-nantes.fr

***Abstract*—Our study concerns energy harvesting embedded systems that have real-time constraints. We present two energy aware scheduling algorithms, namely Green-RTO and Green-BWP which aim to optimize the quality of service of the system measured in terms of deadline success ratio. Such algorithms permit to gracefully cope with processing overload and energy starvation. A simulation study permits to show their performance in comparison with the scheduling algorithm EDeg.**

***Index Terms*—firm real-time embedded systems; energy harvesting scheduling; Quality of Service.**

I. INTRODUCTION

Embedded system technologies can be found in different areas of human activities (e.g. telecom, satellites, sensor nodes, smart cards, computer networking, etc...). They have augmented the quality of human lives by making them more comfortable and safe. Power-aware design has become a central issue in high performance embedded systems that require the optimum use of available power sources. Consequently, these systems must achieve continual functioning without the necessary periodical maintenance due to replacing or recharging batteries. However, non-renewable energy sources are limited and bound to expire eventually. In contrast, renewable energy sources are available in an unlimited quantity. Therefore, environmental energy harvesting (i.e. the process of extracting energy from the surrounding environment) is considered as a promising approach to provide power for long-term applications.

Most of embedded systems have real-time performance requirements that must be met for reasons of usability and safety. More precisely, a deadline is attached to a task execution. In this paper, we assume that the computing system receives energy in limited and variable quantity from the environment. The energy received is stored in an energy reservoir formed by a battery or a super-capacitor. Such energy reservoir is required because the embedded system needs to continue operation even when no energy can be drawn from the environment. We deal with the problem of scheduling periodic skippable tasks under both timing and energy constraints. We propose two energy-aware scheduling algorithms called Green-RTO and Green-BWP. The objective is to enhance the Quality of Service (QoS) i.e. to maximize the success deadline ratio of the system.

The remainder of the paper is organized as follows. Section II gives background materials. Related works are described in Section III. Section IV describes the two schedulers Green-RTO and Green-BWP which are illustrated in Section V. Simulation results are presented in Section VI. Finally, section VII concludes the paper.

II. BACKGROUND MATERIAL

A. Real-time computing

Real-Time Systems (RTS) are defined as those systems in which time constraints, mainly deadlines, have to be considered. They are usually classified as being hard, soft, or firm, depending on the degree of missed deadlines that the system can tolerate. In hard RTS, any deadline violation may lead to a system failure and a catastrophic consequence. In firm RTS, some deadline violations are allowed but they may lead to a system performance degradation without inducing serious consequences.

B. Quality of service in firm RTS

1) Skip-Over model: The *Skip-Over model* [4] deals with the problem of scheduling firm periodic task set (i.e. it allows occasional deadline violations) on a uniprocessor system. A task τ_i is characterized by a worst-case computation time C_i, a period T_i, a relative deadline equal to its period $D_i = T_i$, and a skip parameter s_i. s_i gives the tolerance of this task to miss deadlines in the sense that the distance between two consecutive skips must be at least s_i periods. According to this model, a task is characterized by two kinds of jobs, red ones and blue ones. A red job must complete before its deadline whereas a blue job can be aborted at any time. Consequently, for a task τ_i, $(s_i - 1)$ consecutives jobs are red and the next job is blue. If s_i equals to infinity, no skips are allowed and τ_i is a hard periodic task. In [6], the authors define the equivalent utilization processor U_p^* given by

$$U_p^* = max_{L \geq 0} \left\{ \frac{\sum_i D(i, [0, L])}{L} \right\} \quad (1)$$

Where,

$$D(i,[0,L]) = (\left\lfloor \frac{L}{T_i} \right\rfloor - \left\lfloor \frac{L}{T_i s_i} \right\rfloor) C_i. \quad (2)$$

L represents periods end points.

They proved that a set τ of skippable periodic tasks, which are deeply-red (i.e synchronous activation at time t=0), is schedulable under EDF if and only if $U_p^* \leq 1$.

2) RTO and BWP Scheduling algorithms : Koren and Shasha introduced two scheduling algorithms in [4]. The first algorithm called *Red Tasks Only (RTO)* is the simplest one. It consists in scheduling only red jobs according to either the *Earliest Deadline First (EDF)* or the *Rate-Monotonic (RM)* algorithms [1]. Blue ones are systematically rejected. The second algorithm introduced is the *Blue When Possible (BWP)* algorithm which is an enhancement of the first one. Indeed, BWP schedules blue jobs whenever possible (i.e. when no red job is ready for execution). In other words, blue jobs are served in background relatively to red jobs.

C. Problem definition

Environment harvesting energy is being considered as a promising approach to replace the current power supplies for energy constrained embedded systems. However the power source can be unstable and lead to a possible processing overload. Therefore, we propose to extend the scheduling algorithms of the Skip-over approach in order to process tasks in a best effort manner. The objective is to provide the highest QoS (i.e. the highest deadline success ratio) for the system.

III. RELATED WORKS

The EDeg scheduler:
El Ghor et al. proposed a real-time scheduling algorithm called *Earliest Deadline with energy guarantee (EDeg)* in [3]. It is based on the work presented in [2]. It consists in scheduling a hard periodic task set (i.e. all jobs must meet their deadline) on a uniprocessor system which uses the energy stored in a rechargeable battery and the energy harvested from the environment. Every task is characterized by the energy consumption E_i in addition to the traditional timing parameters (i.e. C_i, D_i, T_i).

All jobs are executed according to the EDF algorithm, as soon as possible and as long as the energy level in the battery is sufficient to provide energy for current and future occurring tasks without involving deadline violations. Hence, the system starts executing a job only if the so-called *slack energy* (i.e. the energy surplus that can be consumed by a job while still satisfying energy constraints of highest priority jobs) is positive and the battery is not empty.

Slack energy enables us to quantify the energy consumed by future jobs and prevents from violating deadlines in case of energy shortage. If energy is not sufficient to execute current or future occuring jobs, the system stops its activity as long as the slack time is positive and the battery is not fully replenished.

IV. CONTRIBUTIONS:

A. Model and terminology

We consider a uniprocessor system that consists of n firm periodic tasks. This system is powered by a rechargeable battery. We assume that firm periodic tasks comply with the skip-over model so each task exhibits computation, energy and QoS requirements. Let $\mathcal{T}$ be a firm periodic task set defined as follows: $\mathcal{T} = \{\tau_i(C_i, D_i, T_i, s_i, E_i), i = 1...n\}$. E_i is the Worst Case Energy Consumption (WCEC). Tasks are independent, preemptable and deeply-red (i.e. synchronous activation at time t=0).

Let us define:

$$g_i^*(0,L) = (\left\lfloor \frac{L}{T_i} \right\rfloor - \left\lfloor \frac{L}{T_i.s_i} \right\rfloor) E_i \quad (3)$$

as the energy consumed by red jobs of task τ_i in the interval $[0, L]$. We define the equivalent energy factor U_e^*, as follows:

$$U_e^* = max_{L \geq 0} \left\{ \frac{\sum_{i=1}^{n} g_i^*(0,L)}{E(0) + E_r(0,L)} \right\}. \quad (4)$$

E(0) represents the initial level of energy in the battery, $E_r(0, L)$ represents the energy received by the battery during the interval [0,L]. In this paper, we assume that the power received by the environment is constant during an hyperperiod P (i.e. P = LCM($T_1 s_1, .., T_i s_i, .., T_n s_n$)). Then as $P_r(t) = P_r$ $\forall t$, $E_r(0, L) = P_r.L$ where L represents the red job end points during the interval [0,P[.

B. Feasibility test

Theorem IV.1. *A skippable periodic task set $\mathcal{T}$ is schedulable only if*
$\forall L | \sum_{i=1}^{n} g_i^*(0,L) \leq E(0) + E_r(0,L)$.

Proof:
We suppose that there is a value t for which $\sum_{i=1}^{n} g_i^*(0,t) > E(0) + E_r(0,t)$. $\sum_{i=1}^{n} g_i^*(0,t)$ represents the total energy required by tasks to feasibly execute in [0,t]. $E(0) + E_r(0,t)$ represents the maximum energy which is available between 0 and t. $\sum_{i=1}^{n} g_i^*(0,t) > E(0) + E_r(0,t)$ clearly implies that at least one task will not be able able to complete execution due to energy starvation. Then $\mathcal{T}$ is not feasible. ■

Theorem IV.2. *A set of skippable periodic tasks $\mathcal{T}$ is schedulable only if $U_p^* \leq 1$ and $U_e^* \leq 1$*

Proof:
Suppose that $U_p^* > 1$ or $U_e^* > 1$. Assume that $U_p^* > 1$. Considering only timing constraints, Caccamo and Buttazo proved in [6] that a set of skippable periodic tasks is schedulable only if $U_p^* \leq 1$. Consequently, $U_p^* > 1$ implies that $\mathcal{T}$ is no feasible.
Assume that $U_e^* > 1$: The previous theorem proves that $\mathcal{T}$ is no feasible.

■

C. Green-RTO Scheduler

We propose the *Green-RTO scheduler* which is based on RTO and EDeg algorithms. Under EDeg all jobs must complete before their deadlines while under Green-RTO only red jobs have to be executed before their deadlines.
Green-RTO runs as follows:

- When the battery is not empty, the processor is active if there is a pending red job and the system has enough slack energy. Ready red jobs are processed according to the EDF algorithm.

- The processor is inactive if the slack time is not null or there is no red jobs ready to be executed.

The slack time of the system at time *t* represents the maximum time available from *t* to postpone red jobs while still satisfying all timing constraints. Hence, it permits to put the processor to an idle state and recharge the battery. Its computation uses the *Earliest Deadline as Late as possible (EDL)* algorithm [2].

The slack energy of the system at time *t* will be given by the lowest slack energy of red jobs with a priority greater than that of the highest priority job ready at time *t*.

The slack energy of a red job J_i of the task τ_i at time *t* represents the maximum amount of energy that can be used from *t* by higher priority jobs (i.e. jobs with a deadline less than or equal to J_i's deadline, d_i):

$$SlackEnergy(t, J_i) = E(t) + \int_t^{d_i} P_r(x)dx - A_i \qquad (5)$$

where *E(t)* is the residual capacity at time *t*, $P_r(x)$ is the power of the fluctuating energy source at time *x* and A_i is the total energy required by red jobs ready to be executed after *t* with a deadline less than or equal to d_i.

D. Green-BWP Scheduler

The *Green-BWP scheduler* is a mixture of BWP and Green-RTO algorithms. According to BWP algorithm, red jobs are executed in priority according to the EDF rule and blue jobs are executed whenever possible (i.e. when there is no ready red job) by taking into consideration both timing and energy constraints of red jobs.

The Green-BWP algorithm has the same framework as Green-RTO and uses the same dynamic data. Accordingly, the main differences between Green-RTO and Green-BWP can be summarized as follows:

- Under Green-RTO, the *slack time* is computed only on basis of the current and future occurring red jobs.
  Under Green-BWP, the sequence is also constructed upon the execution of current and future occurring red jobs. As a completed blue job is always followed by a blue one, a shift is introduced (or offset) in the activation pattern of red jobs. Therefore the computation of slack time requires the determination of the red jobs' releases according to the current status of blue jobs (i.e. either they are aborted or totally executed). Blue jobs are always aborted with Green-RTO.

- Under Green-RTO, the *slack energy* represents the maximum amount of energy that can be consumed by a red job while still guaranteeing all timing constraints of red jobs **only**.
  Under Green-BWP, the slack energy at time *t* is defined as the maximum amount of energy that can be consumed by either a red or a blue job while still guaranteeing all red job constraints. If the current job in execution is red at time *t*, the computation of the slack energy is the same as Green-RTO. Otherwise, if the current job in execution at time *t* is blue with deadline d_i, the slack energy of the system represents the minimum between the slack energy of this job and the slack energy of all red jobs with deadline less than or equal to d_i.

V. ILLUSTRATIVE EXAMPLE

We consider a task set $\mathcal{T} = \{\tau_i(C_i, D_i, T_i, s_i, E_i)\}$ with $\tau_1(3, 6, 6, 2, 10)$, $\tau_2(4, 10, 10, 2, 13)$ and $\tau_3(5, 15, 15, 2, 16)$.

We give *E(0)* $= E_{max} = 5$ and $P_r = 3$.
$U_p^* = 0.833 < 1$. As $U_p^* < 1$, red tasks are schedulable, abstracting from energy constraints.
$U_e^* = 0,831 < 1$. As $U_e^* < 1$, red tasks are schedulable, abstracting from timing constraints.

Figure 1 depicts both Green-RTO and Green-BWP schedules. As Green-BWP does best effort for blue jobs execution, it will give a better QoS than Green-RTO: 70% of jobs are completely executed with Green-BWP whereas 50% of jobs are executed with Green-RTO.

Regarding Figure 1(b):
The battery is full at time $t = 0$. Red jobs are scheduled according to EDF until time $t = 15$. As there is no ready red job, τ_2's blue job is executed since it has the earliest deadline among blue jobs.
At time $t = 19$, the residual capacity becomes 0, no job can be processed thus the battery must be recharged. The slack time is computed using the EDL method. Then, the processor is put in the idle state as long as the system has slack.
At time $t = 21$, the battery is fulfilled. Nevertheless, no red job is ready then τ_1's blue job, which has the earliest deadline among all ready blue jobs, is executed.
As it is completely processed , τ_1's next job is blue. Since no red job is ready and τ_3's blue job has the earliest deadline among all ready blue jobs, it begins its execution as the slack energy of the system is positive and the battery is not empty.
At time $t = 29$, τ_2's blue job has the highest priority however as the slack energy of the system is negative then the processor remains idle during the time interval given by the slack time computation.

VI. EXPERIMENTAL EVALUATION

A. Set up

In order to experimentally evaluate the effectiveness of the proposed algorithms on performance improvement, we

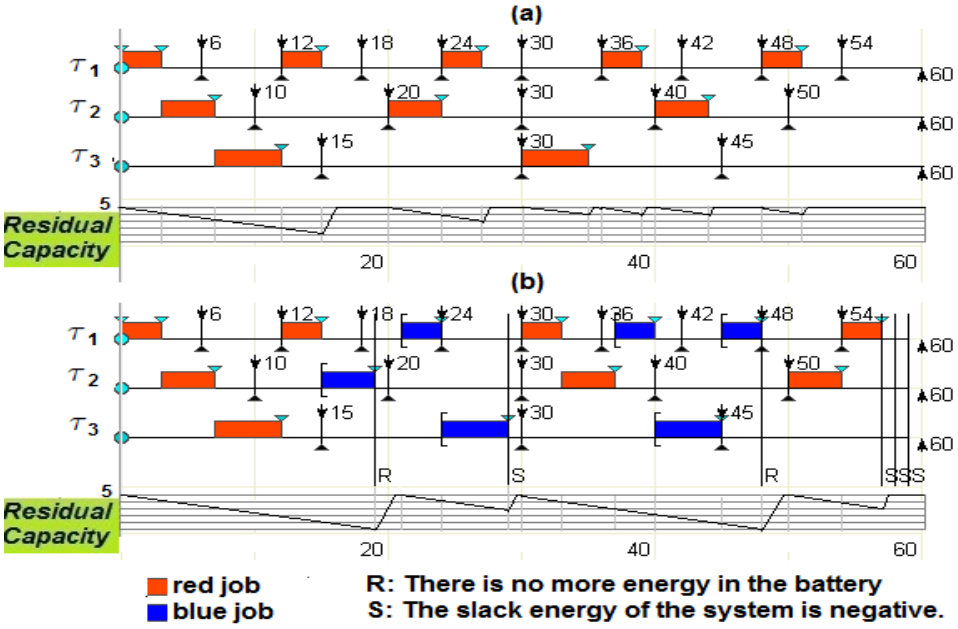

Fig. 1. (a) Green-RTO and (b) Green-BWP scheduling

develop a discrete-event simulator in C language. In the simulator, we implement Green-RTO and Green-BWP. For the sake of comparison, we also implement EDeg proposed in [3]. Task sets used for evaluations are randomly generated. Each task set is characterized by the following parameters:

- The relative deadline D_i is equal to the period T_i.
- The period T_i of each task is randomly chosen in an interval [Tmin, Tmax], so that the least common multiple of the task set (*LCM*) be less than a given *LCMmax*.
- The skip parameter s_i is the same for each task.
- The energy consumption E_i is randomly based on the average power consumption $\overline{P_e}$ given by the following expression:
  $\overline{P_e} = \sum_{i=1}^{n} \frac{E_i}{T_i}$
  . We define also $R_e = \frac{\overline{P_e}}{P_r}$ as the energy criticity ratio.
- The simulator generates 100 task sets for a given processor utilization U_p and an energy utilization factor $\overline{P_e}$. Each set generated is composed of 10 tasks.
- The battery is initially fully charged and the power received P_r is constant.

B. Experimental result

The goal of the simulation is to comparatively measure the QoS performance (i.e. the average of success ratio) according to different values of U_p, $\overline{P_e}$ and P_r. Hereafter, we fix a value of R_e and we vary U_p. As depicted in Figure 2, Figure 3 , Figure 4, Figure 5 and Figure 6, the X-axis shows the processor utilization U_p and the Y-axis shows the QoS percentage.

Case 1 : $R_e = 1$ *(i.e. the average power consumption is equal to the power received)*

In Figure 2 and Figure 3, R_e is equal to 1 and we vary U_p. We assume that *E(0)=300*. Figure 2 and Figure 3 show that *EDeg* gives the best QoS when U_p is less than 1 (i.e. there is no processing overload). However when U_p exceeds 1, *EDeg* behaves badly and *Green-BWP* gives a better QoS than *Green-RTO*. As shown in Figure 2, for low skip values (i.e. when many instances are skipped) the performance of *Green-BWP* is much better than *Green-RTO* because it tries to execute blue jobs. In addition to that, when U_p is less than 1, the QoS under *Green-BWP* tends to be similar to the one under *EDeg*. However, because of red jobs priority, *Green-BWP* always executes red jobs first then blue jobs may not complete their deadlines even if $U_p < 1$ this is why the QoS is not equal to 100%.
As shown in Figure 3, higher is the skip parameter, higher is the effectiveness of *Green-RTO*. Moreover under *Green-BWP*, if the skip value equals 10, the QoS performance tends to 100%.

Case 2 : $R_e = 1.2$ *(i.e the average power consumption is greater than the power received) and* $s_i = 2$ In Figure 4, Figure 5 and Figure 6, R_e is equal to 1.2 and $s_i = 2$. In each Figure, we fix *E(0)* and we vary U_p. As the average power consumption is greater than the power received, we compare the QoS performance between Green-RTO, Green-BWP and EDeg algorithms according to the battery capacity.

As depicted in Figure 4, the battery energy storage is large, $E(0) = P * P_r$, then *EDeg* gives the best QoS while U_p is less or equal to 1.
Green-BWP gives a QoS performance that tends to 100% when U_p is less or equal to 1. Moreover, it still gives an acceptable QoS (between 60% and 80%) when U_p is greater than 1. Green-RTO gives a constant QoS (50%) for all values

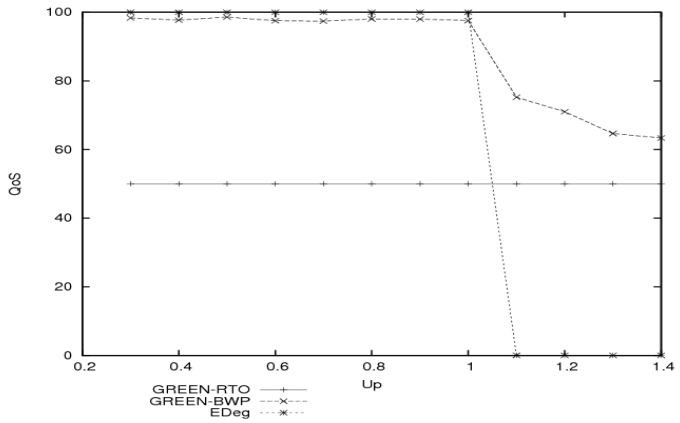

Fig. 2. QoS Performance comparison ($R_e = 1$ and $s_i = 2$)

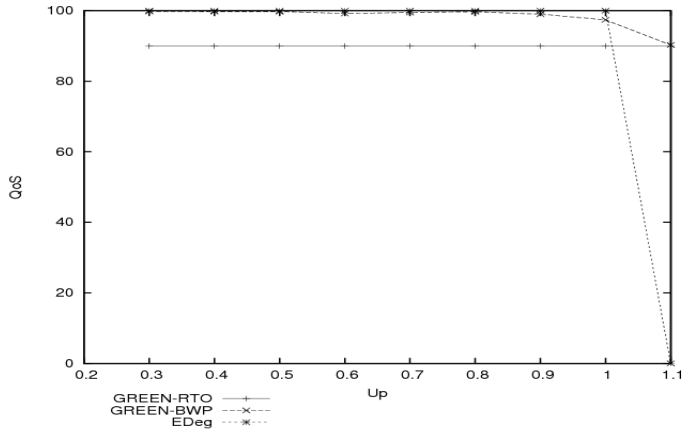

Fig. 3. QoS Performance comparison ($R_e = 1$ and $s_i = 10$)

of U_p.

As depicted in Figure 5 and Figure 6, when the battery energy storage is not very large, Green-BWP and Green-RTO still give an acceptable QoS in contrast to EDeg. Green-BWP gives the best QoS while Green-RTO gives a constant QoS (50%) which depends on s_i.
Higher is R_e, closer are the performances of Green-BWP and Green-RTO.

VII. CONCLUSION AND FUTURE WORKS

Energy harvesting is one of the most promising solutions for increasing the lifespan of autonomous systems. Consequently, energy-aware schedulers should consider the variation of energy drawn from the environment and make tradeoffs between performance and energy consumption. In this paper, we have proposed two scheduling strategies namely Green-RTO and Green-BWP. The proposed algorithms are based on the skip-over model and permit to gracefully cope with both processing overload and shortage of energy. The simulation study has shown that given a task set $\mathcal{T}$, Green-RTO gives the least acceptable QoS for all values of the energy criticity ratio (R_e) and the processor utilization (U_p). When R_e is less

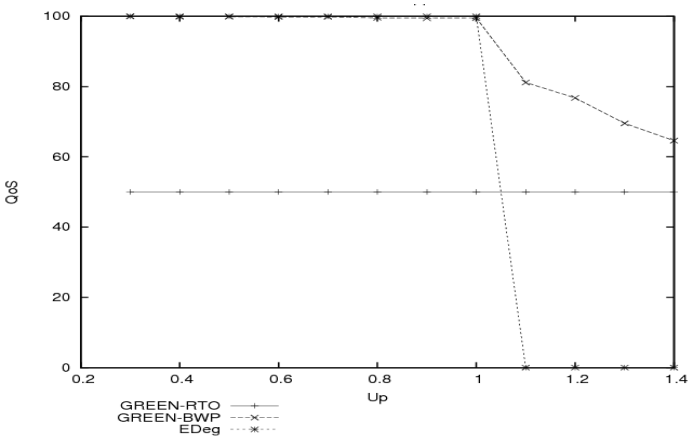

Fig. 4. QoS Performance comparison ($E(0) = P * P_r$)

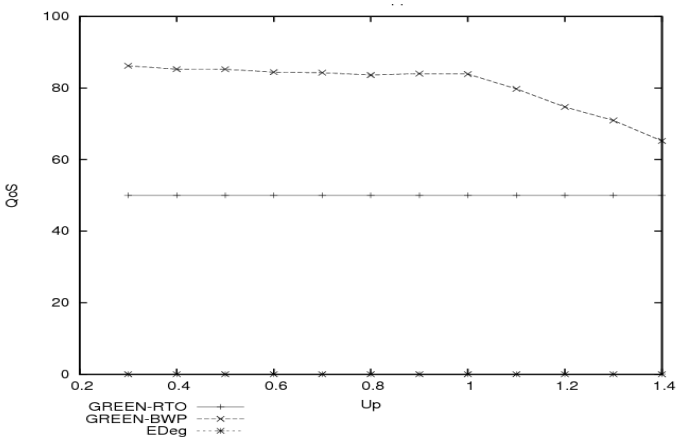

Fig. 5. QoS Performance comparison ($E(0) = \frac{P*P_r}{10}$)

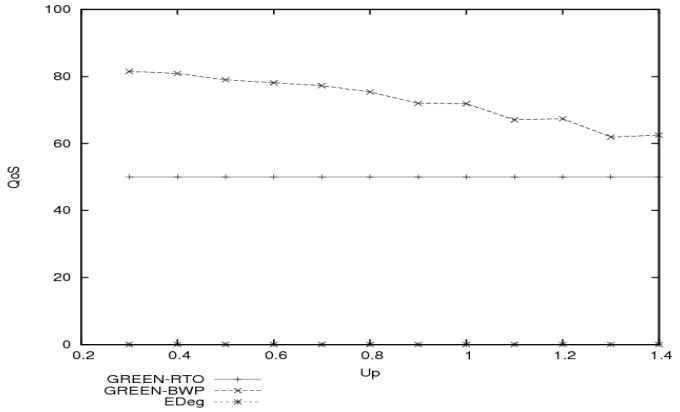

Fig. 6. QoS Performance comparison ($E(0) = \frac{P*P_r}{100}$)

or equal to 1 (i.e. the average power consumption is less or equal to the power received), EDeg gives the best QoS only if there is no processing overload. Whereas Green-BWP gives a QoS that tends to 100% when there is no processing overload and still gives an acceptable QoS when there is a processing

overload. However, when the energy criticity ratio is greater than 1, EDeg works only if the battery energy storage is quite large in constrast to Green-RTO and Green-BWP which give acceptable QoS with smaller battery energy storages. For the future, we would like to consider aperiodic tasks or extend our model to dependent tasks.

ACKNOWLEDGMENT

The work presented in this paper was partially realized in the framework of the GreenEmbedded project (2011-2012), supported by the CEDRE Programme Hubert Curien.

REFERENCES

[1] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. In *Journal of the ACM, 1973.*

[2] H. Chetto and M. Chetto. Some results of the earliest deadline scheduling algorithm. In *IEEE Transactions on Software Engineering, 1989.*

[3] H. El Ghor, M. Chetto and R. Hajj Chehade, A Real-Time Scheduling Framework for Embedded Systems with Environmental energy Harvesting. In *Computers & Electrical Engineering, 2011.*

[4] G. Koren and D. Shasha. Skip-over algorithms and complexity for overloaded systems that allow skips. In *Proceedings of the 16th IEEE Real-Time Systems Symposium (RTSS'95), 1995.*

[5] J.W.S. Liu. Real-Time Systems, Prentice-Hall, 2000.

[6] M. Caccamo and G. Buttazo, Exploiting Skips In Periodic Tasks For Enhancing Aperiodic Rsponsiveness. In *Proceedings of the 18th IEEE Real-Time Systems Symposium, 1997.*