Stability and Robustness Issues in Real-Time sustainable wireless sensors

Maissa Abdallah, Maryline Chetto, Audrey Queudet, Rafic Hage Chehade

Stability and Robustness issues in Real-time Sustainable Wireless Sensors

Maissa Abdallah*, Maryline Chetto*, Audrey Queudet* and Rafic Hage Chehade†
*University of Nantes
IRCCyN UMR CNRS 6597, 44321 Nantes, France
Email: firstname.lastname@irccyn.ec-nantes.fr
†IUT Saida, Lebanese University, Saida, Lebanon
Email: rhagechehade@ul.edu.lb

***Abstract*—In this paper, we address the problem of periodic task scheduling in a sensor node powered with energy harvester. The scheduler can be occasionally forced to skip jobs because of energy shortage or processing overload. Every task executes jobs in conformance with the so-called Skip-Over model where blue jobs may be aborted at any time in contrast to red ones that should complete before deadline. The work presented here aims to consider stability and robustness issues for the Skip-Over model in a uniprocessor energy harvesting system. We present two scheduling strategies, called Green-BWP-LF and Green-BWP-MS specifically adapted to that context. A simulation study shows that these policies outperform the conventional Green-BWP algorithm based on the classical Earliest Deadline rule.**

***Index Terms*—energy harvesting; real-time scheduling; fairness stability; robustness; deadlines.**

I. INTRODUCTION

Energy harvesting defined as a process in which ambient energy is converted into electrical energy, has emerged to power wireless sensor nodes. Energy harvesting has the potential to address the conflicting design goals of lifetime and performance. It exploits renewable energy and resolves the issues of battery life and replacement. Energy harvesting targets consumer applications as well as industrial and healthcare ones. Today, technology for harvesting includes piezoelectric, radio frequency, thermoelectric, inductive coupling, wind, and solar power.

The micropower sources used in energy harvesting applications raise specific challenges for energy management. The ambient energy is often intermittently available. Consequently, this leads to store excess power in order to match supply and demand.

Exploiting an energy harvesting source is fundamentally different from simply using a battery. Rather than a limit on the maximum energy available, it has a limit on the maximum rate at which the energy can be used. Further, the harvested energy availability typically varies over time in a nondeterministic manner. While a deterministic metric (i.e. residual energy level) suffices to characterize the energy availability in the case of batteries, a more sophisticated characterization may be required for a harvesting source. The time-varying characteristics of renewable energy sources creates a shift in research focus from energy-efficient to energy-neutral approaches. Wireless sensor networks (WSN) are deployed in infrastructures such as buildings or bridges and enable various data collection applications (e.g. structural monitoring). Sensors collect information about their surrounding environment, update a base station and respond to frequent or sporadic monitoring requests.

A wireless sensor has a real-time behaviour since the overall correctness of the system depends on both the functional and the timing correctness. A firm real-time system must meet its deadlines with a degree of flexibility in contrast to hard real-time systems where all deadlines have to be met. A missed deadline will just degrade the system's Quality of Service (QoS).

The energy cost of sensing applications relates heavily to the frequency of data requests and updates between sensors and the base station. The frequency in turn affects accuracy of the collected data. In systems with energy harvesting capabilities, we envision that sensors only communicate when there is sufficient harvested energy. There is therefore a tight coupling between the ability of the system to harvest energy and data accuracy: intuitively better harvesting leads to better data quality, poor harvesting conditions imply loss of accuracy.

Our contribution includes: (1) exploiting application tolerance to quality degradation to adapt the sensor data collection process under unstable energy harvesting conditions, (2) designing an energy harvesting management framework with 2 stages (online and offline) that utilizes energy harvesting prediction and knowledge of application tolerance energy cost to maintain system sustainability and optimize data quality and (3) evaluating the performance of the management framework compared with other strategies. Our simulator is also a valuable tool for designers to tune system parameters, to check feasibility of application constraints under various energy harvesting conditions and to study system performance.

The remainder of this paper is organized as follows. Section II presents some related work. Section III states the problem. The energy harvesting system model and two existing schedulers are presented in Section IV. Two novel scheduling policies compliant with the definitions of stability and robustness are described in Section V. Their performance is evaluated in Section VI. Finally Section VII concludes the paper.

II. Related Work

A. Scheduling and processor overload

Earliest Deadline First (EDF) [8] is today one of the most attractive real-time scheduler. However, should the processor experience a transient overload, Earliest Deadline scheduling can not directly ensure that almost the most important tasks of the application are guaranteed. The Skip-Over model [7] aims to consider situations in which periodic tasks may occasionally have deadline violations because of transient processor overloads. 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 and a skip parameter s_i. The distance between two consecutive skips must be at least s_i periods. When s_i equals to infinity, no skips are allowed and τ_i is a hard periodic task. Every job of a task is either red or blue [7]. A red job must complete before its deadline whereas a blue job can be aborted at any time.

Two Skip-over scheduling algorithms were introduced about fifteen years ago by Koren and Shasha in [7]. The first one proposed is the Red Tasks Only (RTO) algorithm. Red jobs are scheduled as soon as possible according to EDF algorithm [8], while blue ones are always rejected. The second one, called Blue When Possible (BWP) algorithm, is an improvement of RTO. BWP schedules blue jobs whenever their execution does not prevent the red ones from completing within their deadlines. In other words, blue jobs are served in background relatively to red jobs.

B. Scheduling and energy harvesting

Liu et al. [9] and Moser et al.[11] propose scheduling techniques for energy harvesting systems at operating system layer. In [14], Han et al. propose an adaptive data collection protocol which aims to minimize energy consumption and prolong battery life-time. This approach is designed for battery powered sensor systems. Our work, on the other hand, exploits error tolerance in both offline and online stages to adapt the system to fluctuations of renewable energy.

Based on the work in [3], the authors in [5] proposed a real-time scheduling algorithm called Earliest Deadline with energy guarantee (EDeg). According to EDeg, the processor executes tasks as soon as possible according to the EDF rule. However, the system starts executing a task only if the so-called slack energy is positive and the reservoir is non empty. Slack energy enables us to quantify the energy consumed by future jobs and prevent them to violate their deadlines because of energy shortage. The system may be inactive as long as the slack time is positive and the reservoir has not fully replenished. The key issues in this algorithm are properly predicting the energy production and measuring the current energy level of the reservoir.

III. Problem statement

Our system consists of a wireless sensor node. Every sensor periodically collects information about its surrounding environment by reading values from its embedded sensor and periodically sends an update to the base station(s). This value can be a property of the environment such as temperature, humidity or sound, that the application needs to monitor. We assume that the sensor node is equipped with an harvesting circuitry and an energy buffer that supplies power for the operation of the sensor.

The first challenge is to utilize the prediction information about future harvested energy to sustain the system and maximize the overall Quality of Service (i.e the success deadline ratio). If high data accuracy is assigned to an interval with predicted low energy, the energy supply will not meet the energy demand and the system might run out of battery and shut down, suspending monitoring activities. If low data accuracy is assigned to an interval with predicted high energy, the harvested energy is not utilized and might be wasted.

IV. Modeling and Harvesting scheduling

A. Definitions

We extend the Skip-over model to real-time energy harvesting applications. We assume that tasks may miss their deadline due to either transient processor overload (i.e. time limitation) or energy overload (i.e. energy shortage). We consider a uniprocessor system that executes a set of firm periodic tasks as described previously. In addition, each task τ_i consumes a certain amount of energy, E_i, called Worst Case Energy Consumption (WCEC). It follows that a task set τ is characterized as : $\tau = \tau_i(C_i, D_i, T_i, s_i, E_i), i = 1...n$.
Let us define:

$$g_i^*(0, L) = (\left\lfloor \frac{L}{T_i} \right\rfloor - \left\lfloor \frac{L}{T_i.s_i} \right\rfloor)E_i \quad (1)$$

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

$$U_e^* = max_{L \geq 0} \left\{ \frac{\sum_{i=1}^{n} g_i^*(0, L)}{E(0) + E_r(0, L)} \right\}. \quad (2)$$

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 $\boldsymbol{H}$ with $\boldsymbol{H}$ = LCM($T_1 s_1, .., T_i s_i, .., T_n s_n$). As $P_r(t) = P_r \ \forall t$, $E_r(0, L) = P_r.L$, where *L* represents the red jobs' end points during the interval [0,H(.

$U_e^* \leq 1$ and $U_p^* \leq 1$ are necessary feasibility conditions where U_p^* is the equivalent utilization processor defined in [2].

Our approach consists in using the spare time saved by the skipped jobs to recharge the battery whenever necessary as described hereafter. We next recall two scheduling strategies initially presented by the authors in [10].

The *Slack Time* is the maximum allowable time to postpone red jobs considering all timing constraints. It is computed using EDL (Earliest Deadline as Late as possible) algorithm [3] and will be used to recharge the battery.
The *Slack Energy* at time *t* [4] is the maximum amount of energy that can be consumed from *t* until d_i, the deadline of

the highest priority red job ready at *t* when still guaranteeing all timing constraints of red jobs. It represents the minimum slack energy of red jobs that have to be executed between *t* and d_i.

The *Slack energy of red job* $\Re_i$ at time *t* is the amount of energy that can be consumed from *t* until d_i while still satisfying timing and energy constraints [1]:

$$SlackEnergy(t, \Re_i) = E(t) + \int_t^{d_i} P_r(x)dx - \sum_{j=1}^{n} E_j \quad (3)$$

E(t) is the residual capacity at time *t*, $P_r(x)$ is the power of the fluctuating energy source at time *x* and E_j is the energy required by red jobs ready to be executed between *t* and d_i.

B. Green-RTO Scheduler

Green-RTO [1] results from RTO and EDeg algorithms. EDeg considers hard real-time periodic tasks in the sense that all jobs must be executed before deadlines. Only red jobs have to be executed before their deadlines under Green-RTO.

Green-RTO runs as follows: The processor is active if the system has positive slack energy and the battery is not empty. Then it will execute ready red jobs according to EDF algorithm. The processor is inactive if the slack time is not equal to zero or if there are no ready red jobs to be executed.

C. Green-BWP Scheduler

Green-BWP is based on BWP and Green-RTO algorithms. It incorporates modifications to enhance the QoS in the sense that according to BWP algorithm, blue jobs are executed whenever possible (i.e as soon as there is no ready red jobs) considering both timing and energy constraints of red jobs. Red jobs are ordered according to the EDF rule.

Green-BWP uses a similar framework to Green-RTO and the same dynamic data. However, the main differences between Green-RTO and Green-BWP can be summarized as follows:

- under Green-RTO, *slack time* is computed only from the current and future occuring red jobs. Under Green-BWP, it is computed taking into account both red and blue jobs.
- under Green-RTO, *slack energy* is the maximum amount of energy that can be consumed by a red job while still satisfying all timing constraints of red jobs only. Under Green-BWP, *slack energy*, at time *t*, is the maximum amount of energy that can be consumed by either a red or a blue job while still guaranteeing all timing constraints of red jobs. If the job in execution at time *t* is red, *slack energy* is computed like under Green-RTO. If the job in execution at time *t* is blue, *slack energy* represents the minimum between *slack energy* of the blue job and *slack energy* of red jobs which have to be executed between *t* and d_i (i.e the deadline of the ocurring blue job).

Let us denote the blue job in execution, β_i. The slack energy of β_i is computed as follows [1]:

$$SlackEnergy(t, \beta_i) = E(t) + \int_t^{d_i} P_r(t)dt - E_i - \sum_{j=1}^{n} E_j \quad (4)$$

E_i is the energy required by β_i and $\sum_{j=1}^{n} E_j$ is the amount of energy required by red jobs ready to be executed between *t* and d_i.

V. STABILITY AND ROBUSTNESS OF ENERGY HARVESTING SYSTEMS

A. Definitions

Robustness and stability have multiple definitions. Thus, we will use the following one: Robustness of a real-time scheduling strategy refers to the global success ratio (i.e. the total number of job completions over the total number of jobs launched) for a given task set. Let us consider the following definition of robustness for a computer system:.

***Definition** 1:* [12] A scheduling algorithm X is more robust than a scheduling algorithm Y if the global success ratio with X is greater than the global success ratio with Y.

Loosely speaking, a scheduling solution for a task set is said to be stable if small perturbations to the task set (e.g. variations in processor workload) result in a new scheduling solution that stays close to the original solution. More precisely, we consider here the definition in terms of success balancing similar to that of fairness:

***Definition** 2:* [12] A scheduling algorithm X is more stable than a scheduling algorithm Y if the greatest difference in success ratio of any tasks with X is less than the greatest difference in success ratio of any tasks with Y.

Note that stability does not refer to the ability of the system to maintain a certain level of performance.

The performance evaluation of a firm scheduling strategy should be performed by measuring its robustness (i.e. the global success ratio) and its stability (i.e. the individual performance of each task).

The analysis reported in [10] shows that the classical EDF is not a stable scheduler. Scheduling blue jobs according to the EDF rule tends to priviledge some tasks relatively to other ones. EDF is clearly a robust scheduler but not a stable one.

B. Two novel scheduling policies

We define Green-BWP-LF (Blue When Possible - Last Failure) which schedules at each time instant, the ready blue job whose number of successive successes from the last failure is the lowest one. The earliest deadline rule is used to break ties between blue jobs of equal priorities.

Green-BWP-MS (Blue When Possible - Minimum Success) schedules at each time instant, the ready blue job whose individual success ratio, computed from the initialization time, is the least. As for Green-BWP-LF case, ties are broken in favor of the task with the earliest deadline. These two variants of the Green-BWP scheduling framework guarantee that any task gets the highest priority at the end of a finite time interval.

C. Illustrative example

We consider a task set $\tau = \{\tau_i(C_i, D_i, T_i, s_i, E_i)\}$ with $T_1(5, 10, 10, 2, 16)$, $T_2(4, 15, 15, 2, 14)$ and $T_3(2, 6, 6, 2, 7)$.
We give *E(0)*=$E_{max} = \frac{H*P_r}{20} = 9$ with $P_r = 3$.
$U_p = \sum_{i=1}^{n} \frac{C_i}{T_i} = 1.1$ and $R_e = \frac{P_e}{P_r} = 1.23$ where U_p is the processor utilization, $P_e = \sum_{i=1}^{n} \frac{E_i}{T_i}$ is the average power consumption and R_e is the energy criticity ratio.
As $U_p > 1$ and $R_e > 1$, the system is overloaded in terms of both energy and time. $U_p^* = 0,733 < 1$. As $U_p^* < 1$, red tasks are schedulable, abstracting from energy considerations.
$U_e^* = 0.698$. As $U_e^* < 1$, red tasks are schedulable, considering only the energy constraints.

Figures 1, 2 and 3 represent the resulting schedules under Green-BWP, Green-BWP-LF and Green-BWP-MS respectively during an hyperperiod *H*. They show that blue jobs are scheduled differently according to the 3 strategies.
In Figure 1, the individual success ratios are respectively equal to 66.66% for T_1, 100% for T_2 and 70% for T_3. The maximal difference between the individual success ratios is 33.33%.
Figure 2 shows that individual success ratios are equal to 83.33% for T_1, 75% for T_2 and 70% for T_3. Then the maximal difference between the individual success ratios is 13.33%.
In Figure 3, the blue job with the minimum success ratio is executed first among blue ready jobs then the individual success ratios are more similar: 66.66% for T_1, 50% for T_2 and 70% for T_3 then the maximal difference between the individual success ratios is only 20%. This example shows the impact of blue job executions on the stability of the system during an hyperperiod. We notice that Green-BWP-LF and Green-BWP-MS are more stable that Green-BWP. As the blue job are executed according to the EDF algorithm, there is a big dispersion of the individual success ratios.

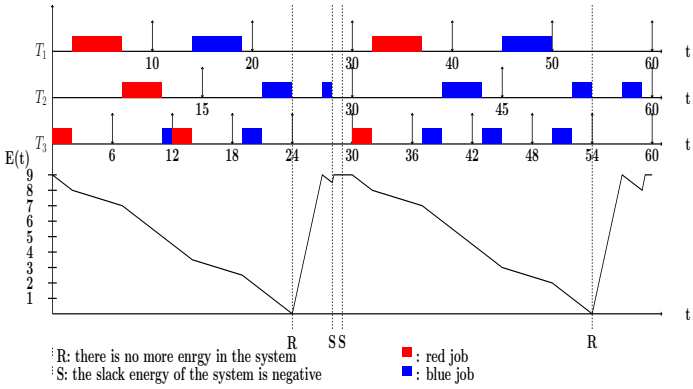

Fig. 1. Green-BWP scheduling

VI. EVALUATION

We now briefly describe simulation results that illustrate the tradeoff between robustness and stability. For our experiments, we use a home-grown simulator written in C. We first describe the experimental setup to evaluate the effectiveness of our proposed scheduling variants and then we compare the results with the classical Green-BWP scheduler in terms of robustness and stability.

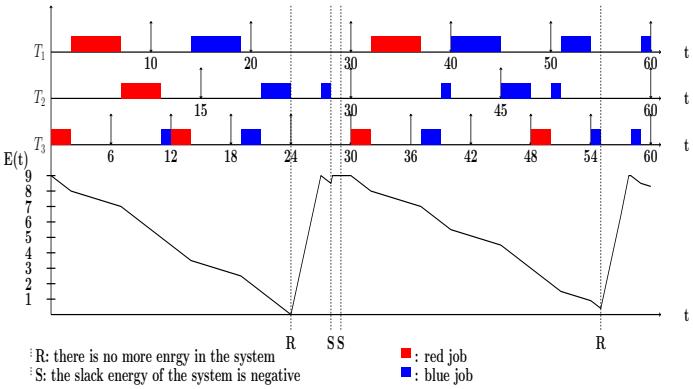

Fig. 2. Green-BWP-LF scheduling

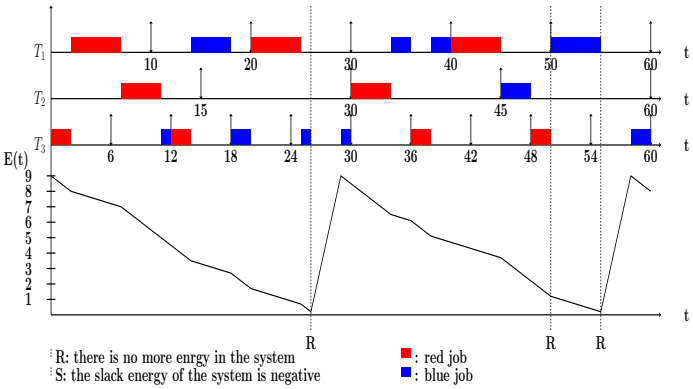

Fig. 3. Green-BWP-MS scheduling

A. Experimental Setup

The simulator generates 100 periodic task sets for a given processor utilization U_p and an average power consumption P_e. Each task set is composed of 10 tasks with a least common multiple equal to 3600. Deadlines are assumed to be equal to periods. The worst case execution time (WCET) of a task is randomly choosen and depends on the processor utilization. The energy consumption E_i is randomly based on the average power consumption P_e. Simulations have been processed over 10 hyperperiods. We assume than the battery is initially fully charged and the power received P_r is constant. We consider a system which is overloaded in terms of energy. Hereafter, we will present an illustration of the system behaviour by applying Green-BWP, Green-BWP-LF and Green-BWP-MS to the generated task sets with a constant energy criticity ratio $R_e = 1.2$ and making varying the processor utilisation U_p. We fix $s_i = 2$ and $E(0) = \frac{P*P_r}{10}$ where *H* is the hyperperiod (i.e *H* = LCM($T_1 s_1, .., T_i s_i, .., T_n s_n$)).

B. Experimental results

We report here some simulation results in order to evaluate stability and robustness of Green-BWP variants.

Green-BWP

Green-BWP algorithm executes ready blue jobs if firstly there is no ready red jobs and secondly if the slack energy of the system is positive. Then, the job which has the earliest deadline will always be executed.

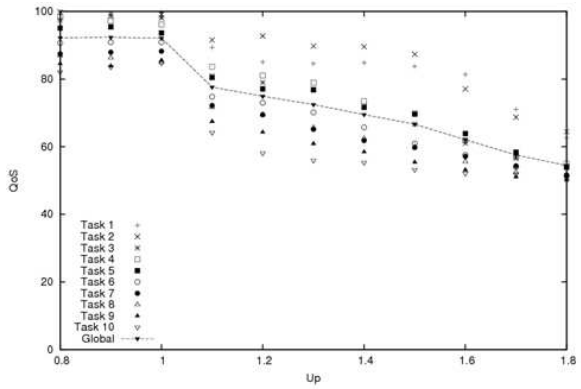

Fig. 4. Individual Success Ratio under Green-BWP

Figure 4 shows the variation of the individual and global success ratios according to processor utilization values. Note that tasks are ordered such that $i < j$ implies that task periods $T_i > T_j$. We notice that τ_1 with the largest period, gets the best success ratio. Indeed, largest be the period, lowest be the number of successful jobs for the given task.

Moreover, we observe that individual success ratios are very dispersed around the plain curve which represents the global success ratio. For $U_p = 1.2$, τ_1 has 85% of successfully executed jobs while τ_{10} has only 58%. We conclude that Green-BWP has a poor behaviour in terms of stability.

Green-BWP-LF

Figure 5 depicts the individual success ratios and the global success ratio under Green-BWP-LF. We notice that

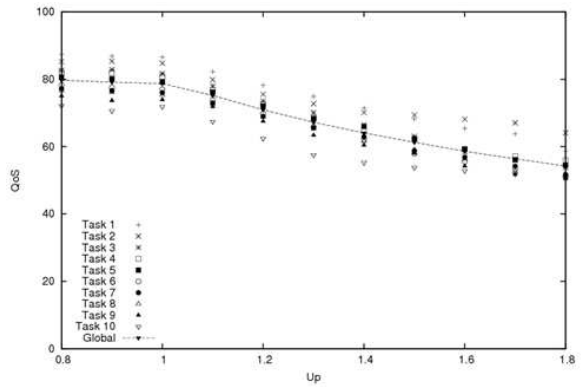

Fig. 5. Individual Success Ratio under Green-BWP-LF

Green-BWP-LF is more stable than Green-BWP because the individual success ratio curves are less dispersed around the global success ratio. For $U_p = 1.2$, τ_1 has 78% of successfully executed jobs whereas τ_{10} has 62%. Then the task with the shortest period gets a better QoS under Green-BWP-LF compared to Green-BWP.

Green-BWP-MS

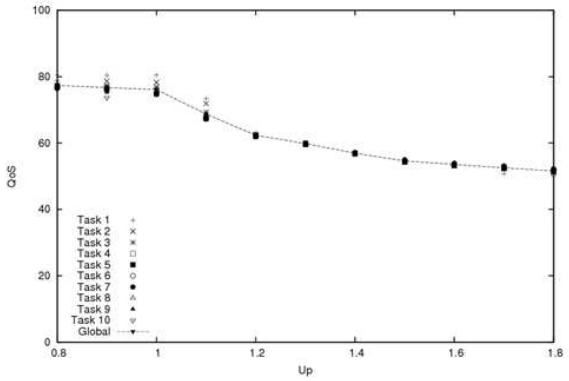

Fig. 6. Individual Success Ratio under Green-BWP-MS

As depicted in Figure 6, stability is very high with Green-BWP-MS. For $U_p > 1.1$, all curves have the same shape.

C. Performance Comparison

Table I summarizes the main criteria in order to compare the stability performance of Green-BWP, Green-BWP-LF and Green-BWP-MS. d_{mean} is the mean difference, d_{max} is the maximal difference and σ is the standard deviation between individual success ratios of tasks. We notice that under Green-BWP, the maximal difference equals 30.56% while it equals to 17.43% under Green-BWP-LF. It is reduced to 6.74% under Green-BWP-MS. Moreover, the mean distance between two individual success ratios is respectively equal to 14.76% under Green-BWP, 8.33% under Green-BWP-LF and 1.96% under Green-BWP-MS. Finally, as the standard deviation with Green-BWP-MS is the lowest one (i.e. 0.73%), we conclude that in terms of stability, Green-BWP-MS is the best algorithm while Green-BWP is the worst one.

Algorithms	d_{max}	d_{mean}	σ
Green-BWP	30.56	14.76	8.08
Green-BWP-LF	17.43	8.33	4.10
Green-BWP-MS	6.74	1.96	0.73

TABLE I
RELEVANT STABILITY CRITERIA

Figure 7 represents the global success ratio under Green-BWP, Green-BWP-LF and Green-BWP-MS. For any strategy, the global success ratio decreases when the processor utilization increases. Green-BWP gives the best global success ratio. Note that the global success ratio observed under Green-BWP-LF is sligthly higher than the one offered by Green-BWP-MS. We conclude that when $R_e > 1$, Green-BWP is highly robust for all values of U_p.

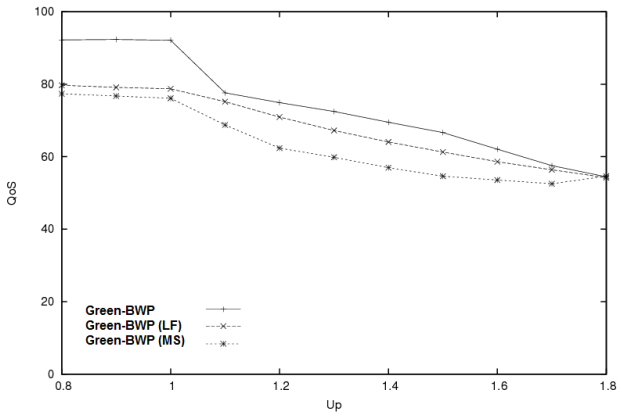

Fig. 7. Global Success Ratio under Green-BWP,Green-BWP-LF and Green-BWP-MS

VII. CONCLUSION AND FUTURE WORKS

In this paper we studied energy-aware scheduling algorithms with the objective of achieving robustness and stability in a real-time energy harvesting system that may experience energy shortage and processor overload. We propose two scheduling policies, namely Green-BWP-LF and Green-BWP-MS. Simulations show that Green-BWP-MS is a very stable algorithm with a maximal difference of individual success ratios equal to 6.76%. Green-BWP-LF is more stable than Green-BWP with a maximal difference of individual success ratios equal to 17.43% against 30.56% under Green-BWP. Some applications (e.g. wireless sensor network) have to deal with both stability and robustness in order to react quickly and provide stable performance. Hence, according to the characteristics of the application (overloaded system in terms of energy or time), designers have to choose a strategy among the 3 strategies studied in this paper. For example, in real-time surveillance applications, multi-cameras are used to provide robustness and accuracy of the monitoring scene. Even if the system is overloaded in terms of energy, images should be processed at the same frequency. Therefore, Green-BWP-LF or Green-BWP-MS should be used.

For future work, we plan to extend that study to less restrictive task models that include synchronization constraints and aperiodic 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 (bilateral project with France and Lebanon).

REFERENCES

[1] M. Abdallah, M. Chetto and A. Queudet, Energy-aware schedulers for Real-Time Energy Harvesting systems with Quality of Service requirements. In *2nd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA), beirut, Lebanon,2012.*

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

[3] H. Chetto, and M. Chetto. Some results of the earliest deadline scheduling algorithm. In *IEEE Transactions on Software Engineering, 15(10): 1261-1269, 1989.*

[4] M. Chetto and H. El Ghor. Real-time Scheduling of periodic tasks in a monoprocessor system with a rechargeable battery. In *17th IEEE International Symposium on Real Time Systems (RTSS 2009), wip session, Washington, 2009.*

[5] 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.*

[6] Q. Han, S. Mehrotra, N. Venkatasubramanian. Energy Efficient Data Collection in Distributed Sensor Environments. In *23rd International Conference on Distributed Computing Systems (ICDCS), USA, 2003.*

[7] 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), Pisa, Italy, 1995.*

[8] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. In *Journal of the ACM, 20(1):46-61, 1973.*

[9] S. Liu, Q. Wu, Q. Qiu . An adaptive scheduling and voltage/frequency selection algorithm for real-time energy harvesting systems. In *DAC '09 Proceedings of the 46th Annual Design Automation Conference, USA, 2009.*

[10] A. Marchand and M. Chetto. Stability and Robustness Issues in Scheduling Periodic Tasks with Firm Real-Time Requirements. In *18th Euromicro Conference on Real-Time Systems, WIP session, Dresden, Germany, 2006*

[11] C. Moser, D. Brunelli, L. Thiele, and Luca Benini, Lazy Scheduling for Energy-Harvesting Sensor Nodes. In *Proceedings of Fifth Working Conference on Distributed and Parallel Embedded Systems, 2006.*

[12] M. Silly-Chetto, On the stability of scheduling algorithms for real-time control, IMACS/IEEE-SMC Computational Engineering. In *Systems Applications Multiconference, Lille, France, 1996.*