



Performance Evaluation of Real-Time Scheduling Heuristics for Energy Harvesting Systems

Maryline CHETTO and Hui ZHANG

IRCCyN – University of Nantes
1 Rue de la Noé, F-44321 Nantes FRANCE

Maryline.chetto@univ-nantes.fr

Hui.zhang@univ-nantes.fr

Abstract

Energy constrained systems can increase their usable lifetimes by extracting energy from their environment. This is known as energy harvesting. This paper investigates scheduling issues in uni-processor real time embedded systems using regenerative energy. Task scheduling should account for the properties of the regenerative energy source which fluctuates, capacity of the energy storage as well as deadlines of the time critical tasks that characterize most of real time embedded systems. In this context, designing efficient scheduling strategies is significantly more complex compared to conventional real-time scheduling. In this paper we compare several scheduling heuristics with the optimal algorithm known as LSA (Lazy Scheduling Algorithm). We report results of an experiment study in terms of percentage of deadlines satisfied.

1. Introduction

Real-time embedded systems play a considerable role in our society, and they cover a spectrum from the very simple to the very complex. Examples of current systems include the control of domestic appliances like washing machines and televisions, the control of automobile engines, telecommunication systems, military command and control systems, industrial process control, flight control systems, and space shuttle and aircraft avionics... For example, a system that monitors temperature in a nuclear power plant would require that the readings be reported to a base

station within enough time for a proper response to be made to a rapid increase in the temperature.

Harvesting energy in surrounding environment to power embedded systems for the lifespan appears nowadays to be the alternative to conventional batteries. Harvesting systems constructed to date extract power efficiently from the source. However, they do not use it adequately under real-time running conditions. As a result, they need a much larger harvester (e.g. solar panel) than necessary to yield the same level of power as a more efficient one, or they rely on a larger, more expensive, higher capacity battery than needed in order to sustain extended operation.

In this new context, the main problem to solve comes from the instantaneous power level that tends to vary over a wide. The autonomous nature of operation makes it imperative that the system learns its own energy environment and adapts its power consumption accordingly. Goal of this adaptation is to maximize the utility of the application in a long-term perspective. The resulting mode of operation is sometimes called *energy neutral* operation.

Then, the crucial issue is to find scheduling mechanisms that can adapt the performance to the available energy profile. Up to now, when designing a real-time embedded system, the first concern has been usually time, leaving energy efficiency as a hopeful consequence of empiric decisions. Now, the primary concern is that power from solar panels or other free sources that cannot be stored (or stored with limited capacity) should be fully consumed greedily, or else this energy will be wasted.

In a real-time environment where tasks have to meet deadlines and execute periodically, energy harvesting and task scheduling are strongly dependent since they have to jointly handle timing constraints and variability of available energy.

In that paper, we propose five scheduling heuristics, easy to implement with limited overhead. Our objective is to compare their performance to an optimal scheduler, namely LSA with high overhead. The main question is to appreciate precisely the relative performance of all these schedulers thanks to simulation. All of them are based on the famous Earliest Deadline First rule. We report results of an experiment study in terms of percentage of satisfied deadlines and wasted energy. The quantitative analysis is achieved without considering computation overheads.

The remainder of this paper is organized as follows. In section 2, we describe the main issues in energy harvesting and briefly describe projects that involve energy harvesting. Section 3 gives background materials around real-time scheduling and precisely describes the optimal uni-processor scheduler, LSA, under energy harvesting assumptions. In section 4, we present five scheduling heuristics and we propose to compare them to LSA. Section 5 reports results of a simulation study and enables us to bring to light that some heuristics may have similar performance to LSA without incurring high overhead. We conclude in section 6 with a brief discussion on ongoing research.

2. Energy harvesting

2.1 Motivations

With the multitude of mobile devices that are used in all areas of social and commercial life it is increasingly important to design systems in an energy-efficient manner. These autonomous systems (including sensor-actuator networks) are being envisioned to carry out complex task sets under real-time requirements without human intervention. However, they require power in order to operate, and if power outages occur, critical data may be lost. The true autonomy of such systems depends on their reliable and guaranteed operation for extended times without maintenance.

Most prior wireless monitoring systems in last decades have relied on continuous power supplied by batteries such as lithium-ion cells. Their disadvantage is that they become depleted, must be periodically replaced or recharged and consequently place hard restrictions on products' usability, lifetime, and cost of ownership. Moreover, while processing power roughly doubles ever two years, battery technology advances at a much more sluggish pace (battery capacity has

doubled every 10 years). In addition to the very slow growth in their energy capacity, traditional batteries have a limit to the total practical energy density they can provide.

Even if it is possible to increase their energy density by tenfold within a few years, we must still consider practical safety concerns. First, given improper use, batteries with extremely high energy densities can become dangerous, explosive devices. Second, in many embedded applications, battery replacement is impractical or has high labour costs associated with maintenance.

Besides, batteries suffer from self-discharge, temperature, and other environmental effects that work to bound their usable lifetime, even in the case of zero use. Consequently for long-term, economical deployment, embedded systems must gather energy from the environment around it, a technique known as *energy harvesting* or *energy scavenging*.

It concerns as well the high technology sectors as the general public products in which wireless embedded systems are used in a variety of applications, such as environmental applications (forest fire and flood detection, monitoring of drinking water and level air pollution), military applications (battlefield surveillance, reconnaissance of enemy forces), health applications (tele-monitoring of human physiological data, tracking and monitoring of doctors and patients), home applications (intrusion detection), or commercial applications (monitoring of product quality, climate control in large buildings). While some of these applications are marginal today, they will become commonplace one day. Devices with maintenance-free life of hundreds of years can now be envisaged if we provide them with efficient strategies for harvesting, storing and managing environmental energy. The current perspectives of this market are thus very big and promising.

2.2 History

A number of projects have used energy harvesting technologies to deliver sustainable power for autonomous sensors. Photovoltaic energy harvesting is by far the most prevalent form of technology used in current projects in part due to the plentiful supply of light in many deployment settings, and the low cost of photovoltaic modules. Nodes conventionally store electrical energy in super-capacitors or batteries to achieve operation. *Heliomote* has a solar panel and two AA type Ni MH batteries [1]. The solar panel is directly connected to its battery through a diode. Even though ample power may be available on the solar panel, a wireless sensor node can still draw current from the battery. *Prometheus* has a super-capacitor as

a primary buffer, a Li-Polymer battery, and a solar panel [2]. The solar panel first charges the super-capacitor, from which the system draws current when enough power is available on the solar panel. The system draws current from the battery only when the charge level of the primary buffer is less than a certain threshold, and it seldom draws power from the battery.

Heliomote and Prometheus have permitted to show that systems may operate perpetually through scavenging solar energy. However, the common drawback of these first prototypes is that they do not target at real-time and quality of service requirements that characterize most of embedded applications.

Several prototype systems incorporating vibration energy harvesting have been developed too. For example, the S5NAP uses a commercially-available electromagnetic vibration energy harvester to power an accelerometer based condition monitoring system. In this system, energy harvested from vibrations is buffered in super-capacitors to permit nodes to draw large bursts of power during radio transmissions and sensing operations [3].

Another project named ShiMmer uses piezoelectric transducers to evaluate a portion of a structure (i.e. a bridge) to determine if damage exists. It relies on a wireless platform that combines active sensing and localized processing with energy harvesting to provide long-lived structural health monitoring. One of the objectives of ShiMmer project is to create a robust and flexible software controller that can manage both the energy and the task execution [4].

3. Scheduling with energy constraints

3.1 Background materials

Most of embedded applications require periodic activities that have to be cyclically executed at fixed rates and within special deadlines. Typically, each periodic instance is assigned a relative deadline equal to the task period and is treated as a hard job. Thus, a periodic task is executed only if all its instances are guaranteed to complete within their deadlines. Schedulability analysis of periodic task sets can easily be performed both under fixed and dynamic priority assignments. In particular, a lot of work has been done for the Rate Monotonic (RM) and the Earliest Deadline First (EDF) algorithms [5]. Schedulability analysis has also been extended for the case in which tasks use shared resources or run in the presence of aperiodic activities, under fixed priority scheduling and in dynamic priority systems as well [6] [7].

While EDF (dynamic priority depending on urgency) and RM (fixed priority depending on period) can

support sophisticated task set characteristics such as deadlines, precedence constraints, shared resources, jitter, etc., they are all open loop scheduling algorithms. Open loop refers to the fact that once schedules are created they are not "adjusted" based on continuous feedback. Systems with open-loop schedulers are usually designed based on worst-case parameters. Such an approach can result in a highly underutilized system based on extremely pessimistic estimation of workload (or energy). While open-loop scheduling algorithms can perform well when the workload and the processing performance are accurately modelled, they perform poorly in unpredictable dynamic systems including regenerative energy dependent ones.

Only in the past decade, researchers started to address power and scheduling issues with the objective of either minimizing power usage under timing constraints or maximizing the system performance under the energy constraints. Nevertheless, they did not consider the rechargeability of the batteries. For example, EDF and RM scheduling have been extended to variable-voltage processors. The idea is to save power by slowing down the processor just enough to meet the deadlines [8]. But solely applying these techniques has limitations in energy harvesting systems because they minimize CPU power, rather than they dynamically manage power according to the profiles of both available energy and processor workload.

The performance of a practical energy harvesting real-time system is measured by the deadline miss rate and heavily depends upon the stored energy and the energy harvested from the environment. Unfortunately, the scavenging power is time-varying and thus very unstable. Therefore, the accurate modelling for energy source plays a key role in designing a good policy to schedule the tasks and reduce the deadline miss rate.

3.2 An optimal scheduling algorithm

The first work that really makes adaptive power management for energy harvesting systems with real time constraints has been published in [9]. There, C. Moser et al. propose a real-time scheduling algorithm, called Lazy Scheduling Algorithm (LSA) that uses task postponement. Algorithm LSA is energy-clairvoyant, i.e., the generated energy in the future is known. Taking into account available time as well as processable energy, an optimal task ordering can be determined based on the prediction of the available energy in the future.

This work deals with a mono-processor architecture that draws the energy from storage and uses it to

process tasks (periodic or non periodic) with arrival time, deadline, and worst case execution time. The worst case execution time represents the maximum energy demand of the task. The arrival time of the task is not known beforehand. The deadline as well as the worst case execution time of the task is unknown before it is released. However, as long as the task is released, all these parameters are determined. They assume that tasks are preemptable and execute according to the earliest deadline first policy.

At any time, the energy source module harvests the energy from its ambient environment and then converts it into electrical energy. The electrical energy can be stored in the energy storage (battery), whose capacity is precisely known. The stored energy is assumed to be known at the system level at any time and is no more than the storage capacity. It is assumed that the energy storage is ideal and the battery can be recharged up to its capacity. Likewise, it can also be completely discharged to as less as zero. If the stored energy reaches the capacity, the incoming harvested energy overflows the storage and is discarded.

According to LSA, the processor executes all tasks at full power when the battery is full time, and the system starts executing a task if the task is ready and has the earliest deadline among all ready tasks and the system is able to keep on running at the maximum power until the deadline of the task.

Contrary to greedy scheduling algorithms, LSA hesitates to power tasks until it is necessary to respect timing constraints. In that sense, tasks are executed neither as soon as possible nor as late as possible. In this paper, the authors also discuss an admittance test that decides, whether a set of real-time tasks can be scheduled without violating deadlines. Another crucial question which has been solved is how to dimension the capacity of the battery that ensures continuous operation. The simulation study demonstrates that achievable capacity savings between 20% and 45% are obtained comparing the classical Earliest Deadline First algorithm. However, all these measurements ignore on line computational costs.

While optimal in the case of a single speed processor, LSA algorithm has the following drawbacks:

- The consumption power of the task is assumed to be characterized by some value. This implies that for every task, its total energy consumption is directly connected to its execution time through the constant power of the processing device. However, in practice, the total energy which can be consumed by a task has no correlation with the worst case execution time.

- Renewable energy sources must be accurately modelled, otherwise the performance of LSA will be degraded.

- Scheduler LSA requires a lot of mathematical computations to be performed on-line. So, in practice, we have to consider its computational overhead, i.e. the cost of its operation both in terms of time and energy consumption.

4. Description of scheduling heuristics

To evaluate the effectiveness of the LSA algorithm on energy saving and performance improvement, we developed a discrete-event simulation and compared LSA to several scheduling heuristics, all using the simple and easy to implement earliest deadline rule:

- Heuristic 1: EDt. Before starting the execution of the highest priority task which is ready, a test is performed to compare the energy level of the battery to the total energy required by the task for its execution. If the energy available in battery is sufficient, the task is authorized to execute. Otherwise, the processor is put into sleep mode until the battery contains enough energy to run it. According to this scheduler, we never start execution of a task if there is no sufficient energy to execute it totally.

- Heuristic 2: EDi. All the tasks are processed as soon as possible according to EDF until the battery is empty. Then, the processor is put into sleep mode until the next release date i.e. the next instant corresponding to the arrival of another ready task. During that period, the battery will replenish and necessarily, the processor will be active at that instant for executing the highest priority task.

- Heuristic 3: EDd. As previously, all tasks execute as soon as possible according to EDF until there is no more energy available in the battery. Then, all ready tasks are discarded and the processor is put into sleep mode until the next release time.

- Heuristic 4: EDu. Compared to EDi, EDu is similar but lets the processor in sleep mode just during one time unit whenever the battery is empty.

- Heuristic 5: EDc. Compared to EDd, EDc is similar but just deletes the current active task instead of discarding all the tasks waiting for execution. The processor is put into sleep mode until the arrival of a new task even if the list of ready tasks may not be empty.

Edi, EDd, EDu and EDc execute tasks as soon as possible i.e. as long as the battery contains at least one unit of energy. These are typical greedy scheduling strategies which mainly differ each other in the way of managing energy lack situations. In contrast, EDt will test dynamically energy availability before running tasks and does not greedily consumes energy. Let us note that for every heuristic, as soon as a deadline is missed, the corresponding task is aborted for

immediately stopping the overload situation and limiting wasted energy.

5. Experiments

5.1 Description of the simulator

In order to evaluate and compare the performance and the effectiveness of the scheduling heuristics to the optimal algorithm, we developed a discrete-event simulation in C/C++. The simulator has been designed specifically for any periodic task set under energy constraints. By using it, we can report details of the schedule produced for any task set with given energy storage capacity and energy source profile.

In our study, we consider the scheduling problem of periodic tasks. We assume that deadlines are less than or equal to periods and greater than or equal to computation times. So we use a task generator to produce 30 periodic task sets, each consisting of 6 tasks with a least common multiple of the periods (often called hyper-period) equal to 300 time units.

The rechargeable power is variable with time in practice. But in this paper, we report results when the source power is constant along time, equal to 6. In the first experiment, the battery capacity is 10. In other terms, 10/6 units of time are required to replenish the energy storage from the environmental power source.

Simulations have been processed over 5 hyperperiods with a mono-frequency processor which is characterized by its maximum consumption power, here equal to 8 (watts). In any case, we assume that the energy storage is fully charged at the beginning of the simulation.

5.2 Experiment 1: Varying the processing load

Figure 1 presents the ratios of satisfied deadlines for the heuristics and the optimal algorithm. It clearly shows that the performances of all algorithms degrade when the processor utilization factor increases. Figure 1 naturally shows that optimal LSA outperforms all other policies. And LSA has the softest degradation, and EDd has the strongest one. Velocities of degradations for the other heuristics are intermediate.

For 60% processor utilization, LSA permits to satisfy 90% task deadlines while EDd, the worst one satisfies about 40% task deadlines. And for 100% processor utilization, LSA succeeds in satisfying about 60% task deadlines while the performance of EDd drops to less than 10%. From a general view, higher is the processor utilization, higher is the gain of LSA compared to heuristics, given a power source profile and a battery capacity. For every processor load, EDd provide the worst performance. Nevertheless, performance

improvement of LSA over EDi and EDu is always less than 20% and less than 10% for very highly loaded systems.

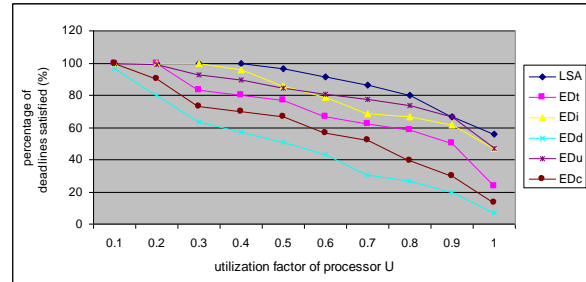


Figure 1: Percentage of satisfied deadlines, making vary the processing load for a given battery capacity

5.3 Experiment 2: Varying the battery capacity

We choose three different values for U, 0.2, 0.5 and 0.8 respectively representing low, medium and high system load. And for each load and a given source power, we make vary the capacity of the battery. The objective of such study amounts to determine optimal dimensioning of the battery for each scheduler and a given tolerance expressed in terms of missed deadlines.

Figure 2 (a) is relative to low processing load ($U=0.2$) and consequently to low energy requirement. When the battery capacity is less than 12, difference of performance between LSA and EDi is very large and as the battery capacity increases, the gaps are getting smaller. For a battery capacity equal to 3, 100% deadlines are satisfied under LSA, EDi and EDu while less than 10% respectively 30% under EDd respectively EDt and EDc. When the battery capacity is larger than 18, all schedulers perform quasi identically. When the battery capacity is greater than 27, all schedulers achieve exactly 100% satisfied deadlines. This simulation result clearly indicates that for low processing loads and low battery capacities, LSA, EDu and EDi outperform very significantly the other heuristics in terms of deadline meeting.

Figure 2 (b) shows that the performances of all schedulers are degraded when the processing load is 0.5. When the size of the battery is less than 30, no strategy can achieve 100% satisfied deadlines including LSA and EDi. To guarantee that LSA achieves 100% satisfied deadlines, the battery capacity should be larger than 40. Consequently, we see that if we make the load 2.5 times, we have to make the battery capacity more than 10 times to guarantee the same level of performance. As in previous curve, it clearly appears that behaviour of EDt, EDc and EDd

are very sensitive to the size of the energy storage. To ensure 40% satisfied deadlines, they require a battery capacity at least equal to 5 while they require a battery capacity equal to 30 to guarantee 90% satisfied deadlines. In the same capacity range, LSA will respectively guarantee between 90% and 100% deadlines.

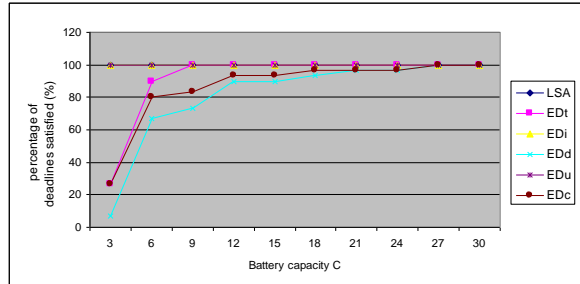
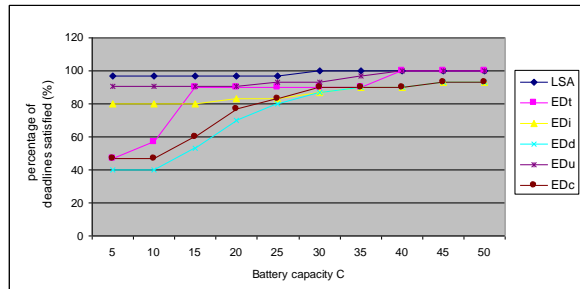
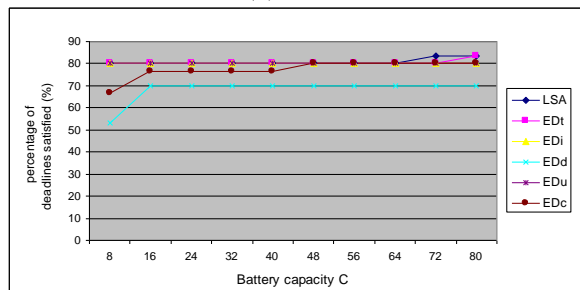
(a) $U=0.2$ (b) $U=0.5$ (c) $U=0.8$

Figure 2: Percentage of satisfied deadlines, making vary the battery capacity for a given processing load

Figure 2 (c) reports the results for high processing loads ($U=0.8$). Comparing to the previous curves, a higher battery capacity is required to provide the same performance level. No scheduler is able to provide 100% satisfied deadlines even if the battery capacity is more than 80. The system is highly time constraint and there is no flexibility for constructing the schedule which may be unfeasible even for high battery capacities. Except for very low battery capacities, all policies yield similar performance with a difference limited to 10%. There is consequently no significant

motivation to increase the battery size for improving performance.

6. Conclusion

Careful energy management is the key to providing the best possible performance in real-time harvesting systems. In this paper we have presented scheduling heuristics in order to compare them to the optimal algorithm known as LSA. We have implemented the policies and reported results showing that the optimal policy outperforms the heuristics that we examined. Results were in terms of percentage of satisfied deadlines which is commonly used to measure the performance of real time systems. However, the experiment reveals that, under all processing loads, the gain is not significant compared to heuristic EDi that executes tasks greedily according to Earliest Deadline First until the energy storage unit be empty and then lets the processor idle until the next release time.

Moreover, practical implementation of LSA requires exact prediction of environmental energy in order to compute dynamically the exact start time of every task. Approximation on the above quantities will make LSA a sub-optimal scheduler, actually providing worst performance relative to the proposed heuristics while leading to higher computational overheads. This interesting issue needs more attention that will be in our immediate research plan. We are measuring the impact of approximating energy availability on the effective performance of LSA and the actual gain of LSA if still existing.

References

- [1] V. Raghunathan, A. Kansal, et al, "Design considerations for solar energy harvesting wireless embedded systems", In Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks, pp 457-462, UCLA, Los Angeles, USA, April 2005.
- [2] X. Jiang, J. Polastre, and D. E. Culler, "Perpetual environmentally powered sensor networks", In Proceedings of the Fourth International symposium on Information Processing in Sensor Networks, pp 463-468, Los Angeles, USA, April 2005.
- [3] S.E. George, M Bocko and G.W. Nickerson, "Evaluation of a vibration-powered, wireless temperature sensor for health monitoring", In IEEE Aerospace Conference, pp 3775-3781, 2005.

[4] D. Dondi, A. Di Pompeo, C. Tenti, and T. S. Rosing, "SHiMmer: a Wireless Harvesting Embedded System for Active Ultrasonic Structural Health Monitoring", presented at SENSORS, 2010 IEEE, 1-4 November. 2010.

[5] C.L. Liu and J.W. Layland, "Scheduling Algorithms for Multiprogramming in Hard Real-time Environment", *Journal of ACM* 20(1), pp. 46-61, 1973.

[6] H. Chetto and M. Chetto, "Some Results of the Earliest Deadline Scheduling Algorithm", *IEEE Transactions on Software Engineering*, Vol. 15, No. 10, pp 1261-1269, 1989.

[7] J.W.S. Liu, "Real-Time Systems", Prentice Hall, 610 pages, march 2000.

[8] G. Quan and X. S. Hu, "Designing Embedded Processors, A Low Power Perspective", Springer Netherlands, Chapter 10, 2007.

[9] C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Real-time scheduling with regenerative energy", in *Proc. of the 18th Euromicro Conference on Real-time Systems (ECRTS06)*, pp 261-270, Dresden, Germany, October, 2006.