EDeg-fs

A Heuristic for Scheduling and Dynamic Power Management in Embedded Energy Harvesting Systems with DVFS Facilities

Hussein El Ghor[1] and Maryline Chetto[2]
[1]*Lebanese University, IUT of Saida, Saida, Lebanon*
[2]*IRCCyN Laboratory, University of Nantes, 1 Rue de la Noe, F-44321 Nantes, France*
hussein.ghor@ul.edu.lb, maryline.chetto@univ-nantes.fr

Keywords: Real-time, Energy Harvesting, Power Management, Dynamic Voltage and Frequency Selection (DVFS), Task Scheduling.

Abstract: This work investigates the problem of dynamic power management and scheduling for a uniprocessor system with dynamic voltage and frequency scaling (DVFS) facilities. This one is qualified as real-time since jobs have to be executed before deadlines. In addition, it is an energy harvesting system since its supply energy is drawn from the environment. We assume that the preemptable jobs receive dynamic priorities according to the Earliest Deadline First (EDF) rule. We propose to extend the dynamic power management scheme called EDeg that must be adapted to a processor with DVFS capabilities. We show how to adjust dynamically the frequency in order to gain quality of service measured in terms of deadline miss rate.

1 INTRODUCTION

Environmental energy harvesting is deemed promising for the supply of embedded and wireless sensor systems. Many sensing environments provide energy that can be harvested and converted into electricity so as to power these systems on an infinite time. Consequently, energy harvesting can make them self-sufficient often for decades (Roundy et al., 2004), having a behavior which is called energy neutral. For example, wireless distributed sensor networks (WDSN) (Lewis, 2004) are mainly used in critical conditions like natural catastrophe and artificial disruptions. The main role of WDSN is to help in monitoring physical and environmental conditions like temperature and pressure. However, WDSN have some drawbacks that are limited battery life and less processing efficiency (Nallusamy and Duraiswamy, 2011). Therefore, energy harvesting seems to be an appropriate approach to increase the life time of WDSNs in environmental applications. Several technologies to extract energy from the environment have been demonstrated including solar, motion-based, biochemical, and vibrational energies. Many other ones are being developed (Kotz and Carlen, 2000). And many prototypes for energy harvesting have been described in the scientific literature. The first ones are certainly Heliomote (Raghunathan et al., 2005) and Prometheus (Jiang et al., 2005).

The work presented in that paper provides the following contributions to research: We present an online algorithm that permits to answer the three following questions dynamically: how and when to decide whether to put the processor in idle or active mode? How to select the active task? How to compute the frequency of the processor for executing the selected task? Our scheduler is model-free with respect to the energy source i.e. it can be implemented without prior information about the source which may be uncontrollable and time-varying. The power management policy is based on trading the slack time and the slack energy for energy efficiency. And the DVFS technology enables us to achieve the lowest energy dissipation. The Quality of Service in terms of deadline success ratio is improved.

The rest of this paper is organized as follows. The energy harvesting system model and some assumptions are described in Section 2. Section 3 gives necessary background materials. Section 4 introduces the proposed scheduling and power management framework. Section 5 presents the related research works. We summarize and describe our current work in Section 6.

2 SYSTEM MODEL AND TERMINOLOGY

The real-time energy harvesting system considered in this work consists of three major units: energy harvesting unit (EHU), energy storage unit (ESU) and energy dissipation unit (EDU). The energy harvesting unit harvests the energy from external sources like sun, wind, etc. The harvested energy is stored in the energy storage unit for future use at any time since it does not leak any energy over time. This helps continuous execution of tasks even at times of deficiency. Apart from the applications running in the energy dissipation unit, there is an additional software running in the uniprocessor system, namely the scheduler. Earliest deadline First (EDF) is the first dynamic priority scheduler used in our algorithm. The other used scheduler is the DVFS which slows down task execution under deadline constraints depending on the energy harvested and energy in the storage unit.

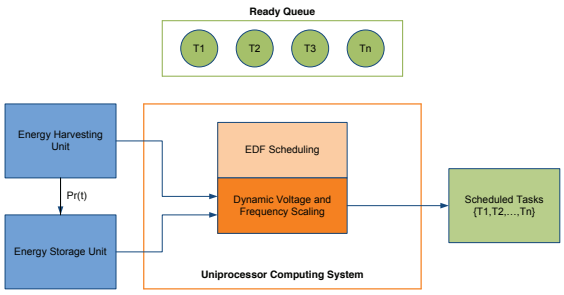

Figure 1: A Real-Time Energy Harvesting System model.

2.1 Energy Harvesting Unit (EHU)

We assume that ambient energy is harvested and converted into electrical power. We cannot control the energy source but we can predict the expected availability with a lower bound on the harvested source power output, namely $P_s(t)$. Clearly, we make no assumption about the nature and dynamics of the energy source, making our approach more easily implemented in real systems where data about the energy source may not be available beforehand.

The energy harvested in an interval of time $[t_1, t_2]$ is denoted by $E_s(t_1, t_2)$ and can be calculated using the following formula:

$$E_s(t_1, t_2) = \int_{t_1}^{t_2} P_s(t)dt \qquad (1)$$

2.2 Energy Storage Unit (ESU)

We use in our work an ideal energy storage unit (supercapacitor or battery) that can be recharged up to a nominal capacity C. Since we use an ideal energy storage unit, then the amount of energy wasted in the process of charging and discharging is neglected. The energy level has to remain between two boundaries C_{min} and C_{max} with $C = C_{max} - C_{min}$. The lower limit of the energy storage unit (C_{min}) is not zero since there must always be a reserved energy in the energy storage unit for worst case scenarios.

2.3 Energy Dissipation Unit (EDU)

We consider a real-time system equipped with a DVFS-enabled processor. The variable speed processor is assumed to be working with N discrete frequencies ranging from $f_{min} = f_1 \leq f_2 \leq \cdots \leq f_N = f_{max}$. The total energy consumption of a running job in the processor depends on the processor's frequency. The power consumption and voltage level relative to clock frequency f_k are denoted by P_k and V_k respectively.

We consider the slowdown factor S_k as the frequency f_k which is normalized with respect to the maximum frequency f_{max}. S_k ranges from S_{min} to 1:

$$S_k = \frac{f_k}{f_{max}} \qquad (2)$$

We consider that each task is characterized by different power dissipation values according to the selected processor frequency. Any job that results from task execution has maximum energy consumption if executed entirely at maximum frequency; And higher is the processor frequency higher is the energy consumption of the job.

Consequently, besides its timing parameters, every task τ_i is characterized by N values that respectively correspond to their worst case energy consumption for each of the N different processor frequencies.

The application software is composed of independent preemptable periodic tasks: $\Gamma = \{\tau_i | 1 \leq i \leq n\}$. A 3-tuple (C_i, D_i, T_i) characterizes task τ_i, where C_i, D_i and T_i indicate the worst case execution time (WCET) at maximum frequency, the relative deadline and the period respectively. τ_i generates an infinite set of jobs that release at times kT_i, $k = 0, 1, 2, \cdots$. When we stretch any job of task τ_i by a slowdown factor S_k, then its actual execution time ($C_i(a)$) at frequency f_k will be C_i/S_k. When the processor is running at it maximum frequency, then $C_i(a) = C_i$. We assume that $0 \leq C_i \leq D_i \leq T_i$ for each $1 \leq i \leq n$.

3 BACKGROUND MATERIAL

3.1 EDF Scheduling

The problem of scheduling deadline constrained tasks on one processor with no energy consideration has been an active area of research for more than thirty years (Liu, 2000). One of the most popular approaches is the dynamic priority driven scheduling algorithm, known as Earliest Deadline First (EDF). EDF schedules at each instant of time t, the ready job (i.e. the job that may be processed and is not yet completed issued from the task set) whose deadline is closest to t. EDF has been proved to be optimal in that sense that if a task set is schedulable by any algorithm, then it can be feasibly scheduled by EDF. Moreover, EDF fully exploits the processor, reaching up to 100% of the available processing time. In general, implementation of EDF consists in ordering jobs according to their absolute deadline either as soon as possible (EDS) or as late as possible (EDL) (Chetto and Chetto, 1989), (Silly-Chetto, 1999).

In a system with limitations and fluctuations in energy availability, simply executing jobs according to the EDF rule, either as soon as possible (EDS) or as late as possible (EDL) may lead to violate some deadlines because of energy starvations. This is why, in energy constrained systems, dynamic power management plays a crucial role due to its impact on the resulting performance. The dynamic power management rule will permit to decide when to put the processor in the active mode and for how long time. The objective of such a policy is to prevent from energy starvation while still preserving the system from deadline violation.

3.2 EDeg Scheduling

In a recent work, we provided a dynamic power management policy called EDeg (Earliest Deadline with energy guarantee) . EDeg orders the ready jobs according to the EDF rule but performs a test before dispatching the highest priority job so as to prevent from energy starvation. More precisely, if the decision test receives a "yes" answer, the processor is autorized to be in the active mode since two conditions can be satisfied. Firstly, the energy level in the battery is sufficient enough to execute the active job. Secondly, executing the active job will not provoke any energy starvation including for a future occurring job.

We introduced the concept of *slack energy of a job* defined as the maximum energy that could be consumed by a lower priority job for avoiding its energy starvation. And the so-called *system slack energy* represents the maximum energy that could be consumed at the current time while guaranteeing absence of energy starvation in the system.
The decision test may lead to a "no" answer thus signifying that the processor has to sleep so that the energy storage unit recharges sufficiently. Deciding for how long time recharging should be performed is very flexible. Nevertheless, the best solution is to recharge the battery at its entire capacity whenever possible, as long as all the deadlines can still be met despite execution postponement. As a consequence, such approach requires to compute at run-time the so-called *slack time of the system* whenever necessary in order to avoid any deadline missing.

3.2.1 Computation of Slack Time

The slack time of a hard deadline set of jobs (which generally are issued from a periodic task set) at current time t is the length of the longest interval starting at t during which the processor may be idle continuously while still satisfying all the timing constraints. The determination of slack time at run-time for jobs issued from a periodic task set can be performed efficiently through a method initially described in (Silly-Chetto, 1999). Some parts of the computation are performed once for all and permit to limit the online overhead.

3.2.2 Computation of Slack Energy

Let $P_s(k)$ be the source power that varies with time k and $C(t)$ be the energy storage capacity at time t. Let A_j be the total energy demand of jobs ready to be processed between t and d_j with a higher priority than job J_j. Consequently, $A_j = \sum_{d_k \leq d_j} E_k$. The slack energy of any job J_j, $Slack.Energy(J_j,t)$, is given by $C(t) + \int_t^{d_j} Ps(k)dk - A_j$.

The slack energy at current time t, Slack.Energy(t), is defined as the maximum amount of energy that can be consumed from t by the currently highest priority job. Consequently, Slack.Energy(t) is given by the minimal slack energy from the all the jobs with a priority higher than one of the active job.

4 THE EDeg-fs ALGORITHM

We are now ready to describe a new scheduling and power management scheme called EDeg-fs (EDeg with frequency scaling facilities). We will show how to optimize the energy consumption of jobs with the dynamic voltage and frequency scaling facilities of

the processor. In addition to the functionalities of EDeg, EDeg-fs selects the execution speed of a running job from the stored energy as well as the available harvested energy and deadline of that job.

4.1 Description of EDeg-fs

The idea behind the EDeg-fs algorithm is to still order the jobs according to the EDF rule. The EDeg-fs schedule differs from the EDF schedule in that the jobs are not executed as soon as possible with the classical work-conserving manner. In addition, EDeg-fs differs from EDeg in that it decides when to execute jobs at full processor speed and when to decrease the processing: speed.

Before authorizing a job to execute, it must be guaranteed that the energy level in the storage be sufficient to provide energy for all future occurring jobs, be given their energy requirements and the replenishment rate of the storage unit. In other words, sufficient energy must permit to execute this job completely. This is realized by computing the slack energy of the active job as described in the EDeg algorithm. If the slack energy is negative, this signifies that energy feasibility can be guaranteed only by decreasing the energy consumption of some jobs through adaptation of the processor frequency.

Let assume that jobs are ordered according to release time.If the job J_i starts execution at current time t, its finishing time could be at least $ft_i = t + Slack.Time(t))$ while guaranteeing the timing feasibility of the system. That means that in the worst case, the maximal energy that could be consumed between t and ft_i is $C(t) + E_s(t, ft_i)$. Consequently, we look for the slowdown factor, namely S_1 that leads to an energy consumption for job J_i that is less than $C(t) + E_s(t, ft_i)$. Necessarily, the selected slowdown value should satisfy $S \geq S_1$. In the other hand, we compute the processor speed so that $C_i(a) = Slack.Time(t)$. The slowdown factor must satisfy $S \leq S_2$ with $S_2 = \frac{C_i}{Slack.Time(t)}$. Finally, the selected slowdown factor is the lowest value that satisfies both $S \geq S_1$ and $S \leq S_2$. Such a computation of the slowdown factor permits to guarantee both the timing and energy requirements of the system. If no such a value exists then, the job should be discarded.

The major components of the EDeg-fs algorithm are $C(t)$, $Slack.Energy(t)$ and $Slack.Time(t)$ where t is the current time, $C(t)$ the amount of energy that is currently stored at time t i.e. the remaining amount of energy in the energy storage at time t. $Slack.Energy(t)$ and $Slack.Time(t)$ are respectively the slack energy of the system and the slack time of the system at time t. The function $execute()$ puts the processor in the active mode for executing the ready job with the earliest deadline.

The EDeg-fs algorithm works as follows: First, EDeg-fs checks if there are jobs in the ready queue. If not, the processor is made idle until the next release time. Otherwise, EDeg-fs selects the highest priority job ready for execution.

Before authorizing the execution of that job, EDeg-fs computes first the slack energy and second the energy level in the storage. If the system slack energy is positive and there is sufficient energy for execution, then the job will be executed with the highest speed. Otherwise, the processor speed is computed from the slack time of the system which leads to stretch the execution of the job without violating deadlines. As decreasing the instantaneous processing speed leads to decrease the total energy consumed by the running job, the level of energy in the storage will be decreasing more slowly than with the highest processor speed.

The main contributions of EDeg-fs can be summarized as follows: The energy optimization process is based on both energy limitations and timing constraints. It fully explores the possibility of trading both the slack time and slack energy for energy saving especially when considering multiple jobs in the queue at the same time. It allows DVFS techniques to achieve lowest energy dissipation while respecting deadlines. And it avoids wasting the over low energy. Energy is wasted only when there are no pending jobs and the storage unit is full.

4.2 Illustrative Example

Let us consider the three following periodic tasks. $\tau_1 = (1,3,5)$, $\tau_2 = (2,7,10)$ and $\tau_3 = (3,12,20)$. We assume that the energy storage capacity is $C = 200$ energy units at $t = 0$. For simplicity, we assume that the rechargeable power P_s is constant along the hyperperiod (least multiple of the periods) equal to 10. The processor has eight discrete slowdown factors: 1, 0.8, 0.7, 0.5, 0.4, 0.3 ,0.2 and 0.1. The total energy consumption of every periodic task τ_i is shown in table 1.

Let us execute the jobs generated by the task set Γ according to EDeg-fs within the first hyperperiod from 0 to 20. The EDeg-fs schedule for Γ is illustrated by figure 2.

We notice that Γ is schedulable since all tasks are executed without violating deadlines and without getting out of energy. At time $t = 0$, all tasks are ready. τ_1 is the highest priority task and is executed until $t = 1$ where $C(1) = 180$ energy units. At time $t = 1$, τ_2 is executed until $t = 3$ where $C(3) = 120$ energy units.

Table 1: Energy dissipation of tasks τ_i.

Energy Dissipation	$S=1$	$S=0.8$	$S=0.7$	$S=0.5$	$S=0.4$	$S=0.3$	$S=0.2$	$S=0.1$
Task τ_1	30	25	20	15	11	8	6	3
Task τ_2	80	65	55	40	32	24	16	10
Task τ_3	180	140	125	90	70	55	35	20

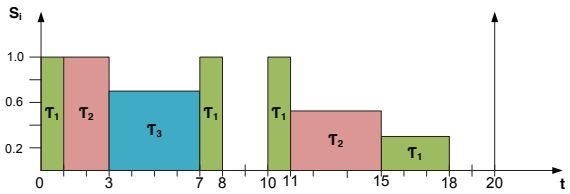

Figure 2: Schedule produced by EDeg-fs.

At $t=3$, τ_3 is the highest priority task ready to be processed but it cannot run at maximum speed because of insufficient energy in the battery. So, we have to slowdown the processor in such a way that the deadline is not violated. We have $Slack.Time(3)=4$. Thus, the actual execution time for τ_3 could be 4 and the slowdown factor is $3/4$. Consequently, the energy dissipation for τ_3 is $E_3=140$. (see table 1). In the other side, as $C(3)+E_s(3,7)=160$, we notice that it is possible to execute τ_3 as regards it new energy consumption equal to 140 energy units.

Now, τ_3 is executed from $t=3$ to $t=7$ with a slowdown factor 0.75 where $C(7)=20$ energy units. At time $t=7$, τ_1 is executed until $t=8$ where the battery is fully discharged. The processor is then idle from time $t=8$ until $t=10$ where $C(10)=20$ energy units. At time $t=10$, τ_1 is executed until $t=11$ where the battery is fully discharged. τ_2 is now ready to be processed, but it cannot run at maximum speed since the battery is fully discharged. Slack time is equal to 2, the actual execution time for τ_2 is equal to 4 and the slowdown factor is 0.5. According to table 1, $E_2=40$ energy units. Now, τ_2 is executed from $t=11$ to $t=15$ with a slowdown factor equal to 0.5 where $C(15)=0$ energy units. This procedure continues till the end of the hyperperiod where $C(20)=39$ energy units.

Let us now consider the EDeg schedule where all the jobs are processed at the constant highest frequency (see figure 3). The total energy consumed under EDeg-fs is about 22 percent less than under EDeg

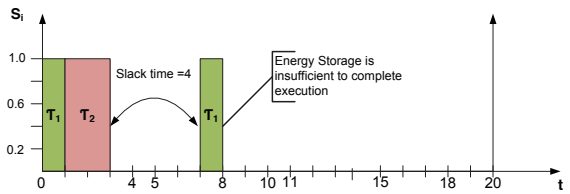

Figure 3: Schedule produced by EDeg.

while still guaranteeing the timing requirements of all the jobs. We may consider that EDeg-fs has significant improvement over EDeg for that illustrative example.

5 RELATED WORK

Energy harvesting systems design has been extensively studied in the past years. In practice, the total energy which can be consumed by a task is not necessarily proportional to its execution time and slack time is not used for energy savings. Clearly, as an embedded system uses a unique energy storage considered as the critical resource of the system, a successful power-aware scheme must consider these non-computation activities and coordinate their power usage as a whole system. Under these assumptions, we presented a scheduling algorithm, EDeg (Earliest Deadline with energy guarantee) (Ghor et al., 2011), that takes into consideration the limits of both time and energy. EDeg relies on two basic concepts: slack time and slack energy. The main idea behind EDeg is to run tasks according to the earliest deadline first rule. However, before authorizing a task to execute, we must ensure that the energy storage is sufficient to execute all future occurring tasks. When this condition is not verified, the processor has to stay idle so that the storage unit recharges as much as possible and as long as all the deadlines can still be met despite execution postponement.

Later in (Chetto et al, 2011), we proved by performance evaluations the efficiency of this scheduler. However, EDeg is a clairvoyant algorithm since it needs the characteristics of the future tasks and the energy source profile to build an optimal schedule. To achieve better system performance and energy efficiency, classical priority driven scheduling has been extended to variable-voltage processors. The idea is to save power by slowing down the processor just enough to meet the deadlines. In (Allavena and Mosse, 2001), A. Allavena et al. describe an off-line scheduler that uses voltage and frequency selection (DVFS) for a frame based system. While they permit to reduce power consumption by slowing down task execution under deadline constraints, their approach relies on the unrealistic assumption that both

the instantaneous consumption power and production power are constant.

S. Liu et al. (Liu et al., 2008) propose an energy aware dynamic voltage and frequency selection algorithm, called EA-DVFS, for periodic tasks. The purpose of EA-DVFS is to efficiently use the slack to reduce the deadline miss rate. Processors must select between running with maximum power or reduced power based on the available energy. If the system has sufficient energy, the task is executed at its full speed; otherwise, it is stretched and executed at a lower speed. In case of low workload, EA-DVFS algorithm reduces deadline miss rate by 50% compared to LSA and decreases the minimum storage size by 25% when the deadline miss rate is zero. The advantage of EA-DVFS is that it reduces the deadline miss rate and storage capacity in case of low overload.

Later in (Liu et al., 2012), Liu et al. presented a harvesting-aware DVFS (HA-DVFS) algorithm to improve the system performance by fully exploiting the task slack under timing and energy constraints. HA-DVFS utilizes adaptive scheduling techniques combined with dynamic voltage and frequency selection to reduce the deadline miss rate when compared to LSA and EH-DVFS.

6 CONCLUDING REMARKS

In this paper we have described an integrated framework for deadline constrained job scheduling, dynamic power management and voltage/frequency selection in real-time energy harvesting systems. The scheduler is a dynamic priority driven one that uses the Earliest Deadline First rule. The dynamic power management policy consists in checking the energy feasibility through computation of the so-called slack energy. This is to verify that deadlines will be met while guaranteeing no energy starvation; And the voltage/frequency selection policy slowdowns task execution whenever the system has slack time.

We are now conducting an experimental study in order to compare the EDeg-fs and EDeg algorithms. We want to evaluate (in comparison to EDeg) how much EDeg-fs permits to increase the resulting Quality of Service i.e. the deadline success ratio. Moreover, it could be interesting to show the improvement of EDeg-fs over EDeg regarding the minimum energy storage capacity requirement to achieve zero deadline miss ratio.

REFERENCES

S. Roundy, D. Steingart, L. Frechette, P. K. Wright, and J.M. Rabaey. *Power sources for wireless sensor networks*. In Proc. Euro.Workshop Wireless Sensor Networks, pp. 117, 2004.

R. Kotz and M. Carlen. *Principles and applications of electrochemical capacitors*. In Electrochimica Acta 45, pages 2483-2498. Elsevier Science Ltd., 2000.

V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. B. Srivastava. *Design considerations for solar energy harvesting wireless embedded systems*. In Proc. Int. Symp. Inf. Process. Sensor Netw., pp. 457462, 2005.

X. Jiang, J. Polastre, and D. E. Culler. *Perpetual environmentally powered sensor networks*. In Proc. Int. Symp. Inf. Process. Sensor Netw., pp. 463468, 2005.

F. L. Lewis. Smart Environments: technologies, protocols, and Applications. John Wiley, New York, 2004.

R. Nallusamy and K. Duraiswamy. *Solar Powered Wireless Sensor Networks for Environmental Applications with Energy Efficient Routing Concepts: A Review*. Information Technology Journal, 10: 1-10, 2011.

A. Kansal, J. Hsu. *Harvesting aware power management for sensor networks*. In IEEE Proceedings of design automation conference, 2006.

J-W-S. Liu. Real-time systems. Prentice-Hall, 2000.

H. Chetto, M. Chetto. *Some results of the earliest deadline scheduling algorithm*. IEEE Transactions on Software Engineering 1989;15(10):12619, 1989.

M. Silly-Chetto. *The EDL server for scheduling periodic and soft aperiodic tasks with resource constraints*. Real-Time Systems; 17(1):125, 1999.

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.

M. Chetto, H. EL Ghor and R. Hage Chehade. *Real-Time Scheduling for Energy Harvesting Sensors*. The 6th International Conference for Internet Technology and Secured Transactions, Abu Dhabi, UAE, December 11-14, pp. 396 - 402, 2011.

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

S. Liu, Q. Qiu, Q. Wu. *Energy aware dynamic voltage and frequency selection for real-time systems with energy harvesting*. In: Proceedings of the conference on design, automation and test in Europe, p. 236241, 2008.

S. Liu, J. Lu, Q. Wu and Q. Qiu. *Harvesting-Aware Power Management for Real-Time Systems With Renewable Energy*. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 20, No. 8, August 2012.