LAZY SCHEDULING FOR ENERGY HARVESTING SENSOR NODES

C. Moser[1], D. Brunelli[2], L. Thiele[1], L. Benini[2]

[1]*Computer Engineering and Networks Laboratory*
Swiss Federal Institute of Technology (ETH) Zurich, Switzerland

[2]*Department of Electronics, Computer Science and Systems*
University of Bologna, Italy

Abstract The paper studies the case of a sensor node which is operating with the power generated by an environmental source. We present our model of an energy driven scheduling scenario that is characterized by the capacity of the node's energy storage, the deadlines and the power dissipation of the tasks to be performed. Since the execution of these tasks requires a certain amount of energy as well as time, we show that the complexity of finding useful scheduling strategies is significantly increased compared to conventional real-time scheduling. We state online scheduling algorithms that jointly account for constraints arising from both the energy and time domain. In order to demonstrate the benefits of our algorithms, we compare them by means of simulation with the classical Earliest Deadline First Algorithm.

1. Introduction

Wireless sensor networks have been the subject of intensive research over the past several years. As for many other battery-operated embedded systems, a sensor's operating time is a crucial design parameter. As electronic systems continue to shrink, however, less energy is storable on-board. Research continues to develop higher energy-density batteries and supercapacitors, but the amount of energy available still severely limits the system's lifespan. Recently, energy harvesting has emerged as viable option to power sensor nodes: If nodes are equipped with energy transducers like e.g. solar cells, the generated energy may increase the autonomy of the nodes significantly.

In [6], technologies have been discussed how a sensor node may extract energy from its physical environment. Moreover, several prototypes (e.g. [2, 3]) have been presented which demonstrate both feasibility and usefulness of sensors nodes which are powered by solar or vibrational energy.

The authors of [4] propose algorithms for tuning a node's duty cycle dependent on the parameters of the energy source. Nodes switch between active and sleep mode and try to achieve perpetual operation. Other approaches addressed offline scheduling with regenerative energy by means of Dynamic Voltage Scaling (DVS) [1, 7]. In contrast to this work, we present online algorithms to dynamically schedule arriving tasks and thereby, we are not restricted to a certain technique like Dynamic Voltage Scaling.

In [5], Lazy Scheduling Algorithms (LSA) have been presented for the first time. The latter work primarily focuses on proving the optimality of LSA and derives schedulability conditions from that proof. This paper, on the other hand, presents a detailed description of the algorithms as well as extensive simulative studies revealing the benefits of this new scheduling discipline.

The Earliest Deadline First (EDF) algorithm has been proven to be optimum with respect to the schedulability of a given taskset in traditional time-driven scheduling. The following example shows why greedy scheduling algorithms (like EDF) are not necessary optimal in the context of this paper.

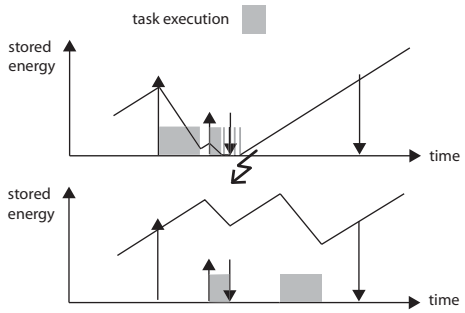

Figure 1. Greedy vs. Lazy Scheduling

Imagine a sensor node with an energy harvesting unit that replenishes a battery with constant power. Now, this node has to perform an arriving task that has to be finished until a given deadline. In Figure 1, the arrival time and deadline of this task are depicted by "long" -up and down- arrows respectively. Meanwhile, a less energy-intensive task has to be executed within a short time interval that is again given by an arrival time and a deadline (indicated by the "short" arrows). As depicted in the top diagram, the EDF scheduler violates the deadline of the second task since it uses greedily the stored energy to drive the first, energy-intensive task. When the energy is required to execute the second task, the battery level is not sufficient to meet the deadline. In this example, however, a scheduling strategy that hesitates to spend energy meets both deadlines. The bottom plot illustrates how the *Lazy Scheduling* paradigm described in this paper outperforms a naive, greedy approach like EDF in the described situation.

2. Problem Statement

Let us consider a sensor node as depicted in Fig. 2. In the following, the single components of this node will be explained in detail.

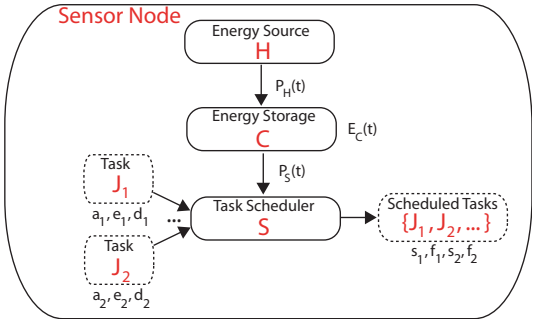

Figure 2. Scheduling Scenario

Energy Source

We denote $P_H(t)$ the charging power that is actually fed into the energy storage and hence incorporates all losses due to power conversion. Next, the corresponding energy E_H scavenged in the time interval $[t_1, t_2]$ is given by the integral $E_H(t_1, t_2) = \int_{t_1}^{t_2} P_H(t)\mathrm{d}t$.

Energy Storage

We assume an ideal energy storage (e.g. a battery) that may be charged up to its capacity C , i.e., $E_C(t) \leq C$. According to the scheduling policy of the sensor node, power $P_S(t)$ and the respective energy $E_S(t_1, t_2)$ is drained from the storage to execute tasks. In particular, if the node decides to assign power $P_i(t)$ to the execution of task J_i during the interval $[t_1, t_2]$, we denote the corresponding energy $E_i(t_1, t_2)$. If no tasks are executed and the storage is consecutively replenished by the energy source, an energy overflow occurs.

Task Scheduling

As illustrated in Fig. 2, we use the notion of a task scheduler that assigns energy E_C from the storage to arriving tasks. Only one task is executed at the same time and preemptions are allowed. For the sake of simplicity, we bound the power consumption of all tasks to the maximum value p_d. In other words, we introduce the abstraction of a single processing device that determines how much power $P_S(t)$ it uses at any moment in time, i.e.

$$0 < P_S(t) < p_d\,.$$

A task is characterized by its arrival time a_i, its energy demand e_i and its deadline d_i. The effective starting time s_i and finishing time f_i of a task are dependent on the scheduling strategy used: A task started at time s_i will finish as soon as the required amount of energy e_i has been consumed by it. We can write

$$f_i = \min \{t : E_i(s_i, t) = e_i\} .$$

Tasks are considered to be preemptive i.e. the currently active task may be interrupted at any time and continued at a later time. If the full processing power p_d is continuously assigned to a single task J_i, the task is finished after a minimum execution time $w_{i,min} = \frac{e_i}{p_d}$.

3. Lazy Scheduling with $p_d = \infty$

We start with a node that executes tasks with infinite power $p_d = +\infty$. This theoretical model of a node which runs a task in zero time can be a good approximation for many practical scenarios. If processing times w_i are negligible compared to the time to recharge the battery (i.e. $p_d \gg P_H(t)$), the assumed model can be regarded as reasonable.

Moreover, we assume the processing device on the sensor node to select between three power modes. The node may process tasks with the maximal power $P_S(t) = p_d$ or not at all ($P_S(t) = 0$). In between, the node may choose to spend only the currently incoming power $P_H(t)$ from the harvesting unit on tasks. Altogether, we consider a node that decides between $P_S(t) = P_H(t)$, $P_S = 0$ and $P_S = +\infty$ to advance arriving tasks.

As already indicated in the introduction, the naive approach of scheduling tasks with the EDF algorithm may result in unnecessary deadline violations. Given a node with $p_d = \infty$, LSA avoids spending energy on tasks too early by executing all tasks at their deadline. At time d_j, task J's remaining amount of unprocessed energy $(e_j - E_j(a_j, d_j))$ is drained from the energy storage with $P_S = \infty$. Only if we hit the capacity limit ($E_C(t) = C$) at some time t, we execute the task with the currently earliest deadline using power $P_S(t) = P_H(t)$. The above two rules formulated as pseudo-code are shown in Alg. 1.

In the next section we will see that Alg. 1 is an optimal algorithm for respecting the deadlines of an arbitrary taskset. Note that it degenerates to an *earliest deadline first* (EDF) policy, if $C = 0$. On the other hand, we find an *as late as possible* (ALAP) policy for the case of $C = +\infty$.

4. Lazy Scheduling with finite p_d

Using a device with finite power consumption p_d, one has to take into account finite execution times w_i, too. Obviously, starting at a task's deadline d_j is not appropriate anymore and also determining straightforward starting times $s_j = d_j - \frac{e_j}{p_d}$ does not help: Already a second task J_n arriving shortly

Algorithm 1 (Lazy Scheduling for $p_d = \infty$)

Require: maintain a set of indices $i \in Q$ of all ready but not finished tasks J_i
  $P_S(t) \Leftarrow 0$;
  while (true)
    $d_j \Leftarrow \min\{d_i : i \in Q\}$;
    process task J_j with power $P_S(t)$;
    $t \Leftarrow$ current time;
    if $t = a_k$ **then** add index k to Q;
    if $t = f_j$ **then** remove index j from Q;
    if $t = d_j$ **then** $E_C(t) \Leftarrow E_C(t) - e_j + E_j(a_j, t)$;
      remove index j from Q;
      $P_S(t) \Leftarrow 0$;
    if $E_C(t) = C$ **then** $P_S(t) \Leftarrow P_H(t)$;

after s_j and having an earlier deadline $d_n < d_j$ inevitably causes a deadline violation. Clearly, this kind of timing conflict can be solved by starting tasks earlier. In doing so, however, we risk to run into energy conflicts as pointed out in the introduction. In the following, we focus on finding optimal starting times s_j for a task J_j that balance the *time* and *energy* constraints for our scheduling scenario.

In order to find an optimal starting time s_j for task J_j, LSA requires the knowledge of the incoming power flow $P_H(t)$ for all future times $t \leq d_j$. In addition we make the realistic assumption that $P_H(t) < p_d$, that is, the incoming power $P_H(t)$ from the harvesting unit never exceeds the power consumption p_d of a busy node. Finding useful predictions for the power $P_H(t)$ can be done for example by analyzing traces of the harvested power over a finite duration. If these measurements of the past are also representative for the future, the prediction will be close to the real value of $P_H(t)$.

At first, we consider task J_j illustrated in Fig. 3. We calculate the starting time s_j^* as

$$s_j^* = d_j - \frac{E_C(a_j) + E_H(a_j, d_j)}{p_d}.$$

Once again, we assume that a second task J_n arrives after J_j but has to finish before J_j ($d_n < d_j$). Since at the time of its arrival task J_n has an earlier deadline, it is reasonable to adhere to the well-known EDF policy and interrupt execution of task J_i. At this point, we demonstrate that starting task J_j before or after s_j^* may lead to unnecessary deadline violations.

On the one hand, starting before s_j^* ensures the completion of task J_i whereas task J_n may "starve" because of missing energy and thus finishes after its deadline. Fig. 3 illustrates that a too early starting time may cause this conflict since the nested task J_n has to process with the harvested power flow $P_H(t)$ instead of p_d. Note that in the diagrams also the energy/power assigned to the respective tasks is displayed. On the other hand, starting after s_j^* can result in deadline violations due to lack of time while a sufficient amount of energy is

stored on the node. As depicted in Fig. 3, task J_j violates its deadline if the complete energy $E_C(a_j) + E_H(a_j, d_j)$ is needed to process tasks J_j and J_n.

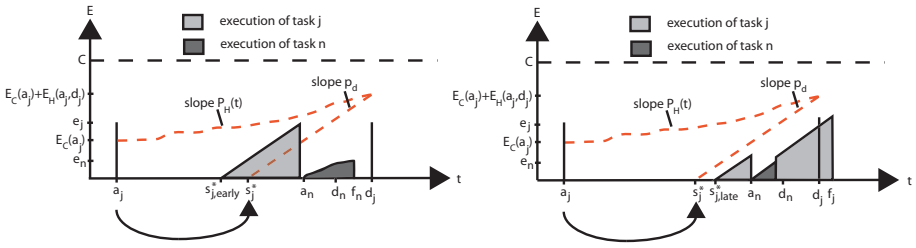

Figure 3. Starting execution of task J_i before or after s_j^*

If the complete energy $E_C(a_j) + E_H(a_j, d_j)$ is available during the interval of full utilization, the starting time s_j^* is certainly optimal considering both energy and time constraints. But what happens if the stored energy E_C reaches its maximum value C before the starting time s_j^*? – Of course, the overflowing part of the energy $E_H(a_j, s_j^*)$ is used in an EDF-manner to execute task J_j. As a consequence, only energy $C + E_H(s_j^*, d_j)$ can be processed continuously with p_d and it is not possible to maintain full utilisation of the device until the deadline d_j. A better, lazier starting time s_j' can be found by numerically solving the following equation (see Fig. 4):

$$E_H(a_j, s_j') - C = E_H(a_j, d_j) + (s_j' - d_j)p_d$$

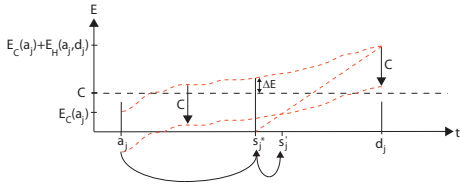

Figure 4. Recalculation of the starting time due to storage limitation

If we now choose the maximum starting time $s_j = \max\left(s_j', s_j^*\right)$ we have finally found the optimal starting time. The calculation of s_j must be performed once the scheduler selects the task with the earliest deadline. Then, LSA either executes task J_j with power p_d if the current time $t \geq s_j$, or it adheres to the EDF strategy with $P_S(t) = P_H(t)$ if the storage is full. Alg. 2 shows the pseudo-code for LSA with constant power consumption p_d.

If the scheduler is not energy-constraint, i.e. if the available energy is more than the device can consume with power p_d within $[a_j, d_j]$, the starting time s_j

Algorithm 2 (Lazy Scheduling with $p_d = const.$)

Require: maintain a set of indices $i \in Q$ of all ready but not finished tasks J_i
  $P_S(t) \Leftarrow 0$;
  while (true)
    $d_j \Leftarrow \min\{d_i : i \in Q\}$;
    calculate s_j;
    process task J_j with power $P_S(t)$;
    $t \Leftarrow$ current time;
    if $t = a_k$ **then** add index k to Q;
    if $t = f_j$ **then** remove index j from Q;
    if $E_C(t) = C$ **then** $P_S(t) \Leftarrow P_H(t)$;
    if $t \geq s_j$ **then** $P_S(t) \Leftarrow p_d$;

will always be before the current time t. Then, the resulting scheduling policy is EDF, which is reasonable, because only time-constraints have to be satisfied. On the other hand, whenever the sum of stored energy E_C and generated energy E_H is small, the scheduling policy changes towards an ALAP policy. In doing so, LSA avoids spending scarce energy on the "wrong" tasks too early. In summary, LSA can be regarded as an adaptive, energy-clairvoyant algorithm that schedules tasks in an Earliest Deadline First fashion.

THEOREM 1 (OPTIMALITY OF LAZY SCHEDULING, $P_S = const.$) *If the Lazy Scheduling Algorithm cannot schedule a given taskset, then no other scheduling algorithm can. This holds even if the other algorithm knows the complete taskset in advance.*

The proof of Theorem 1 is omitted due to space constraints, but can be found in [5]. As a direct consequence of its optimality, LSA successfully schedules a taskset with the minimum possible capacity C.

5. Simulation Results

We implemented the Lazy Scheduling Algorithm LSA as well as the Earliest Deadline First EDF algorithm in a simulation framework. Since LSA and EDF may exhibit identical behaviour for a taskset in dependency of p_d, we run all simulations in this section with power $p_d = \infty$. Figure 5 shows the harvested power $P_H(t)$ generated by a random number generator for 1000 time units.

Since the performance of a given algorithm will be severely affected by the properties of the arriving tasks, we performed all simulations in this section for two different tasksets: Taskset T_1 consists of 30 periodic tasks with a common period of 300 time units. Initial phases, energy demands and relative deadlines of the tasks are randomly assigned by a random number generator. Taskset T_2 consists of 8 periodic tasks, also with a common period of 300 time units. In contrast to taskset T_1, phases, energies and deadlines of taskset T_2 are manually assigned. Figure 5 displays the respective values of taskset T_2 within one period.

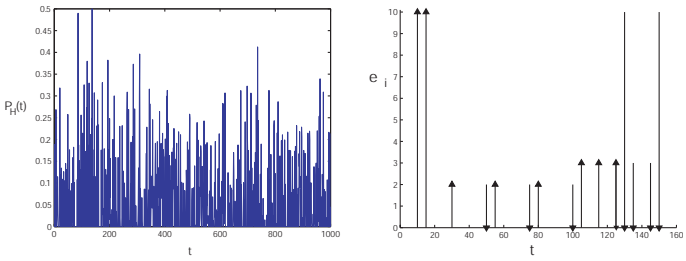

Figure 5. Randomly generated power curve $P_H(t)$ and taskset T_2

Time until First Deadline Violation

We are now interested in the first deadline that cannot be hold with a certain scheduling strategy. With the help of some offline analysis, we tuned the amplitude of the power source $P_H(t)$ to enforce early deadline violations of tasksets T_1 and T_2 respectively. We assume $E_C(0) = C$, i.e. at the beginning of the simulation the battery is fully charged. After a deadline violation is detected, the simulation terminates.

A first simulation result is depicted in Fig. 6 for taskset T_1. Obviously, both curves are monotonically increasing since a longer time is needed to deplete the battery if the initial energy $E_C(0) = C$ is higher. Due to the optimality of LSA, the time of the first deadline violation with EDF is always earlier than with LSA. Though, for some values of the capacity C, the difference between LSA and EDF is very low.

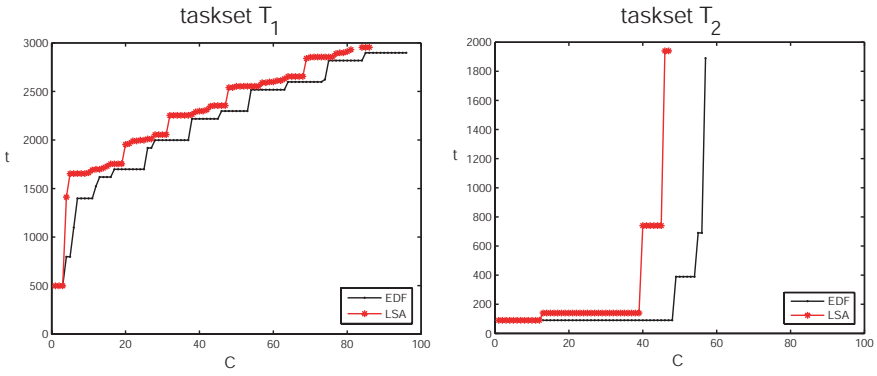

Figure 6. Time t until the first deadline of taskset T_1/T_2 is violated

The same simulations have been repeated for taskset T_2 (see Fig. 6). For values of the capacity over 60, both algorithms schedule taskset T_2 without deadline violations in the simulated time ($t < 2000$). For capacities between

40 and 60, however, we find significant differences between LSA and EDF. Clearly, taskset T_2 is tailored to the weak points of EDF. But the principal arrangement of taskset T_2 is not unreasonable if tasks for radio communication, sensor activity or data processing have to be executed on the same device. Large differences of the tasks' energy demands and overlapping arrival times and deadlines are the ideas underlying taskset T_2.

Number of violated deadlines

Another approach is to disregard missed deadlines and to count only the number of violations. We decided to use an extended version of the LSA algorithm that continues task execution even after a deadline violation. To this end, the scheduler drains $P_H(t)$ from the storage until the task is finally finished.

Fig. 7 presents the number of recorded deadline violations that occurred during a period of 3000 time units for taskset T_1. Since Lazy Scheduling makes the best use of the scavenged energy, it outperforms EDF for all capacities. The difference between LSA and EDF is varying between 0 and 9 deadline violations in the considered interval. For high values of C, no deadline violations could be detected for both algorithms because of the high energy availability.

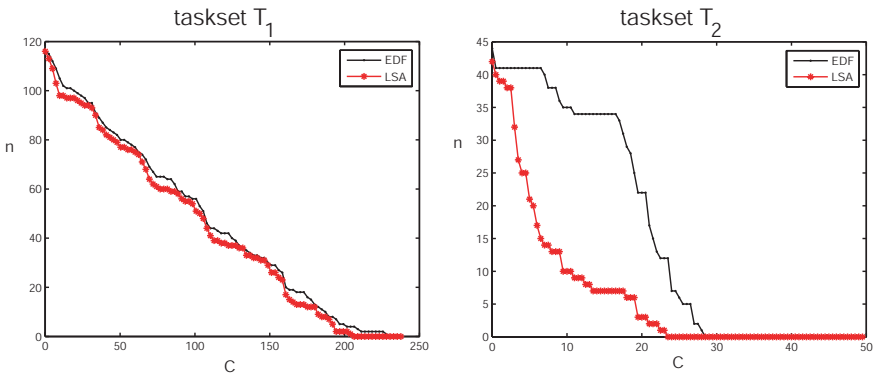

Figure 7. Number n of violated deadlines of taskset T_1/T_2

For taskset T_2, LSA performs significantly better in terms of violated deadlines than EDF. Especially for values of C between 10 and 20, LSA's energy management pays off. On the other hand, for a given number n of violated deadlines, a much higher capacity C is necessary under EDF scheduling. For example, to obtain $n = 0$ deadline violations, LSA requires a capacity $C = 24$ while EDF needs a 25% higher capacity ($C = 30$) to respect all deadlines.

6. Conclusion

In this paper, we studied the case of an energy harvesting sensor node that has to schedule a set of real-time tasks. These tasks require a certain amount of energy as well as time to complete. We have discussed Lazy Scheduling Algorithms for online scheduling. However, LSA algorithms are energy-clairvoyant, i.e., the profile of the energy generated in the future has to be known to the scheduler in advance. Finally, simulation results demonstrate how LSA outperforms the Earliest Deadline First Algorithm and that significant reductions of the battery size are possible when running LSA.

Acknowledgments

The work presented in this paper was partially supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322. In addition, this research has been founded by the European Network of Excellence ARTIST2.

References

[1] A. Allavena and D. Mosse. Scheduling of frame-based embedded systems with rechargeable batteries. In *Workshop on Power Management for Real-Time and Embedded Systems (in conjunction with RTAS 2001)*, 2001.

[2] Y. Ammar, A. Buhrig, M. Marzencki, B. Charlot, S. Basrour, K. Matou, and M. Renaudin. Wireless sensor network node with asynchronous architecture and vibration harvesting micro power generator. In *sOc-EUSAI '05: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence*, pages 287–292, New York, NY, USA, 2005. ACM Press.

[3] 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, IPSN 2005*, pages 463–468, UCLA, Los Angeles, California, USA, April 25-27 2005.

[4] A. Kansal, D. Potter, and M. B. Srivastava. Performance aware tasking for environmentally powered sensor networks. In *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2004*, pages 223–234, New York, NY, USA, June 10-14 2004. ACM Press.

[5] C. Moser, D. Brunelli, L. Thiele, and L. Benini. Real-time scheduling with regenerative energy. In *18th Euromicro Conference on Real-Time Systems, ECRTS 2006*, Dresden, Germany, July 5-7 2006.

[6] S. Roundy, D. Steingart, L. Frechette, P. K. Wright, and J. M. Rabaey. Power sources for wireless sensor networks. In *Wireless Sensor Networks, First European Workshop, EWSN 2004, Proceedings*, Lecture Notes in Computer Science, pages 1–17, Berlin, Germany, January 19-21 2004. Springer.

[7] C. Rusu, R. G. Melhem, and D. Mosse. Multi-version scheduling in rechargeable energy-aware real-time systems. In *15th Euromicro Conference on Real-Time Systems, ECRTS 2003*, pages 95–104, Porto, Portugal, July 2-4 2003.