REAL-TIME 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 Energy harvesting has recently emerged as a feasible option to increase the operating time of sensor networks. If each node of the network, however, is powered by a fluctuating energy source, common power management solutions have to be reconceived. This holds in particular if real-time responsiveness of a given application has to be guaranteed. Task scheduling at the single nodes should account for the properties of the energy source, capacity of the energy storage as well as deadlines of the single tasks. We show that conventional scheduling algorithms (like e.g. EDF) are not suitable for this scenario. Based on this motivation, we have constructed optimal scheduling algorithms that jointly handle constraints from both energy and time domain. Further we present an admittance test that decides for arbitrary task sets, whether they can be scheduled without deadline violations. To this end, we introduce the concept of energy variability characterization curves (EVCC) which nicely captures the dynamics of various energy sources. Simulation results show that our algorithms allow significant reductions of the battery size compared to Earliest Deadline First scheduling.

1. Introduction

Wireless sensor networks – consisting of numerous tiny sensors that are unobtrusively embedded in their environment – have been the subject of intensive research. 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. As a result, size as well as weight of most existing sensor nodes are largely dominated by their batteries.

On the other hand, one of the main advantages of wireless sensor networks is their independence of pre-established infrastructure. That is, in most common scenarios, recharging or replacing nodes' batteries is not practical due to (a) inaccessibility and/or (b) sheer number of the sensor nodes. In order for sensor networks to become a ubiquitous part of our environment, alternative power sources should be employed. Therefore, environmental energy harvesting is deemed a promising approach: If nodes are equipped with energy transducers like e.g. solar cells, the generated energy may increase the autonomy of the nodes significantly.

In [14], several technologies have been discussed how, e.g., solar, thermal, kinetic or vibrational energy may be extracted from a node's physical environment. Moreover, several prototypes (e.g. [2, 5]) have been presented which demonstrate both feasibility and usefulness of sensors nodes which are powered by solar or vibrational energy.

From a networking perspective, classical protocols cannot harness the full potential provided by the harvesting technology. Here, several publications exist which make routing or clustering decisions within the network *harvesting aware* [8, 17]. Based on the knowledge of the currently generated power at the single nodes, the network lifetime can be optimized by shifting the communication and computation load. In the example of a solar powered network, nodes which are directly exposed to sunlight have to disburden nodes who are receiving less radiation due to shadowing effects.

In contrast, we focus on the *temporal* variations of the energy source experienced by a single node instead of *spatial* variations between several nodes. The obtained results can e.g. be applied to networks, whose nodes are independently from each other transmitting data to a base station. But even if we are dealing with a multi-hop scenario, a single sensor node may need to perform its activities in a timely manner using a limited and uncertain energy source. For example, a node may need to communicate with others by means of a energy-saving wireless protocol, e.g. by switching on transmitters only at well-defined time instances. In addition, there are application scenarios for which it is indispensable to fulfill *real-time* requirements like it is the case in, e.g., fire or intruder detection systems. In general, one can classify real-time application scenarios for wireless sensor networks into safety critical systems, smart spaces as well as entertainment [16]. For all these scenarios, our research reveals fundamental problems and tradeoffs when real-time behaviour has to be guaranteed although a sensor's driving energy source is highly unstable.

The example in Figure 1 illustrates why greedy scheduling algorithms (like Earliest Deadline First EDF) are not suitable in the context of regenerative energy. Let us consider a node with an energy harvesting unit that replenishes a battery. For the sake of simplicity, assume that the harvesting unit provides a constant power output. Now, this node has to perform an arriving task "1" that

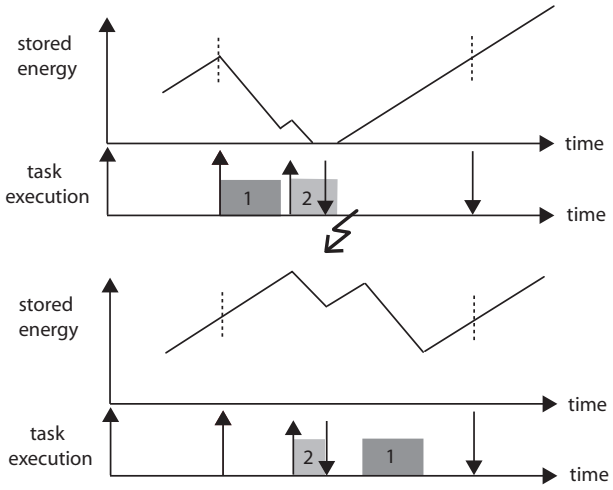


Figure 1. Greedy vs. lazy scheduling

has to be finished until a certain deadline. Meanwhile, a second task "2" arrives that has to respect a deadline which is earlier than the one of task "1". In Figure 1, the arrival times and deadlines of both tasks are indicated by up and down arrows respectively. As depicted in the top diagrams, a greedy scheduling strategy violates the deadline of task "2" since it dispenses overhasty the stored energy by driving task "1". 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 on task "1" meets both deadlines. The bottom plots illustrate how a *Lazy Scheduling Algorithm* described in this paper outperforms a naive, greedy approach like EDF in this situation. Lazy scheduling algorithms can be categorized as non-work conserving scheduling disciplines. Unlike greedy algorithms, a lazy scheduler may be idle although waiting tasks are ready to be processed.

The research presented in this paper is directed towards sensor nodes. But in general, our results apply for all kind of energy harvesting systems which must schedule processes under deadline constraints. For these systems, new scheduling disciplines must be tailored to the energy-driven nature of the problem. This insight originates from the fact, that energy – contrary to the computation resource "time" – is storable. As a consequence, every time we withdraw energy from the battery to execute a task, we change the state of our scheduling system. That is, after having scheduled a first task the next task will encounter a lower energy level in the system which in turn will affect its own execution. This is not the case in conventional real-time scheduling where time just elapses either used or unused.

The rest of the paper is organized as follows: In the next section, we highlight the contributions of our work. Subsequently, Section 3 gives definitions that are essential for the understanding of the paper. In Section 4, we present *Lazy Scheduling Algorithms* for optimal online scheduling and proof their optimality. Admittance tests for arbitrary task sets are the topic of Section 5. Simulation results are presented in Section 6 and Section 7 deals with practical issues. At the end, Section 8 summarizes related work and Section 9 concludes the paper.

2. Contributions

The following paper builds on [11] and [12], where Lazy Scheduling Algorithms have been presented for the first time. We combine both view points, extend the theoretical results by a formal comparison of the schedulability regions of EDF and LSA, include corresponding simulation results as well as a discussion of implementation aspects. Thereby, we outline how our algorithms could be implemented on real sensor nodes which illuminates the relevance of the proposed theory. The contributions described are as follows:

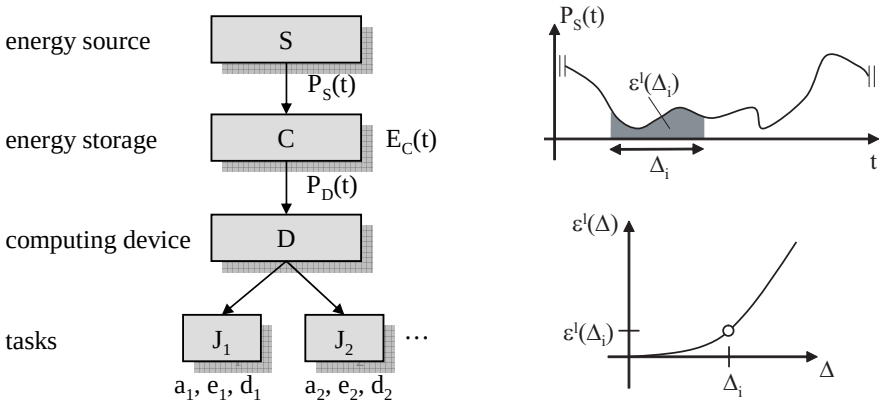
- We present an energy-driven scheduling scenario for a system whose energy storage is recharged by an environmental source.
- For this scenario, we state and prove optimal online algorithms that dynamically assign power to arriving tasks. These algorithms are energy-clairvoyant, i.e., scheduling decisions are driven by the knowledge of the future incoming energy.
- We present an admittance test that decides, whether a set of tasks can be scheduled with the energy produced by the harvesting unit, taking into account both energy and time constraints. For this purpose, we introduce the concept of energy variability characterization curves (EVCC). In addition, a formal comparison to EDF scheduling is provided.
- By means of simulation, we demonstrate significant capacity savings of our algorithms compared to the classical EDF algorithm. Finally, we provide approximations which make our theoretical results applicable to practical energy harvesting systems.

3. System Model

The paper deals with a scheduling scenario depicted in Fig. 2(a). At some time t, an energy source harvests ambient energy and converts it into electrical power $P_S(t)$. This power can be stored in a device with capacity C. The stored energy is denoted as $E_C < C$. On the other hand, a computing device drains power $P_D(t)$ from the storage and uses it to process tasks with arrival time a_i,

energy demand e_i and deadline d_i. We assume that only one task is executed at time t and preemptions are allowed.

The problem statement presented in this section comprises two major constraints which have to be satisfied: First, tasks can be processed exclusively with energy E_S generated by the energy source. And second, timing constraints in terms of tasks' deadlines d_i must be respected. For this purpose, two degrees of freedom can be exploited. The scheduler may decide which task J_i of all ready tasks to execute and what amount of power P_D to assign to this task. The following subsections define the relations between these quantities in more detail.



(a) Energy-driven scheduling scenario

(b) Power trace and energy variability characterization curve EVCC

Figure 2. Representation of the system model

3.1 Energy source

Many environmental power sources are highly variable with time. Hence, in many cases some charging circuitry is necessary to optimize the charging process and increase the lifetime of the storage device. In our model, the power P_S incorporates all losses caused by power conversion as well as charging process. In other words, we denote $P_S(t)$ the charging power that is actually fed into the energy storage. The respective energy E_S in the time interval $[t_1, t_2]$ is given as

$$E_S(t_1, t_2) = \int_{t_1}^{t_2} P_S(t)\mathrm{d}t \quad .$$

In order to characterize the properties of an energy source, we define now energy variability characterization curves (EVCC) that bound the energy har-

vested in a certain interval Δ: The EVCCs $\epsilon^l(\Delta)$ and $\epsilon^u(\Delta)$with $\Delta \geq 0$ bound the range of possible energy values E_S as follows:

$$\epsilon^l(t_2 - t_1) \leq E_S(t_1, t_2) \leq \epsilon^u(t_2 - t_1) \quad \forall t_2 > t_1$$

Given an energy source, e.g., a solar cell mounted in a building or outside, the EVCCs provide guarantees on the produced energy. For example, the lower curve denotes that for any time interval of length Δ, the produced energy is at least $\epsilon^l(\Delta)$ (see Fig. 2(b)). Three possible ways of deriving an EVCC for a given scenario are given below:

- A sliding window of length Δ is used to find the minimum/maximum energy produced by the energy source in any time interval $[t_1, t_2)$ with $t_2 - t_1 = \Delta$. To this end, one may use a long power trace or a set of traces that have been measured. Since the resulting EVCC bounds only the underlying traces, these traces must be selected carefully and have to be representative for the assumed scenario.

- The energy source and its environment is formally modeled and the resulting EVCC is computed.

- Approximations to EVCCs can be determined on-line by using appropriate measurement and estimation methods, see Section 7.1.

In Section 5, the lower EVCC ϵ^l will be used in an admittance test which decides, whether a task set is schedulable given a certain energy source. Furthermore, both EVCCs will serve as energy predictors for the algorithms simulated in Section 6.

3.2 Energy storage

We assume an ideal energy storage that may be charged up to its capacity C. According to the scheduling policy used, power $P_D(t)$ and the respective energy $E_D(t_1, t_2)$ is drained from the storage to execute tasks. If no tasks are executed and the storage is consecutively replenished by the energy source, an energy overflow occurs. Consequently, we can derive the following constraints

$$\begin{aligned} 0 \leq E_C(t) &\leq C && \forall t \\ E_C(t_2) &\leq E_C(t_1) + E_S(t_1, t_2) - E_D(t_1, t_2) && \forall t_2 > t_1 \end{aligned}$$

and therefore

$$E_D(t_1, t_2) \leq E_C(t_1) + E_S(t_1, t_2) \qquad \forall t_2 > t_1 \,.$$

3.3 Task scheduling

As illustrated in Fig. 2(a), we utilize the notion of a computing device that assigns energy E_C from the storage to dynamically arriving tasks. We assume that the power consumption $P_D(t)$ is limited by some maximum value P_{max}. In other words, the processing device determines at any point in time how much power it uses, that is

$$0 < P_D(t) < P_{max} \, .$$

We assume tasks to be independent from each other and preemptive. More precisely, the currently active task may be preempted at any time and have its execution resumed later, at no additional cost. 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

$$E_i(t_1, t_2) = \int_{t_1}^{t_2} P_i(t) \mathrm{d}t \quad .$$

The effective starting time s_i and finishing time f_i of task i are dependent on the scheduling strategy used: A task starting at time s_i will finish as soon as the required amount of energy e_i has been consumed by it. We can write

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

The actual running time $(f_i - s_i)$ of a task i directly depends on the amount of power $P_i(t)$ which is driving the task during $s_i \leq t \leq f_i$. At this, the energy demand e_i of a task is independent from the power P_i used for its execution. Note that we are *not* using energy-saving techniques like Dynamic Voltage Scaling (DVS), where $e_i = \mathrm{f}(P_i)$. In our model, power P_i and execution time w_i behave inversely proportional: The higher the power P_i, the shorter the execution time w_i. In the best case, a task may finish after the execution time $w_i = \frac{e_i}{P_{max}}$ if it is processed without interrupts and with the maximum power P_{max}.

Current hardware technology does not support variable power consumption as described above. So clearly, the continuous task processing model presented in this section is idealized. However, a microprocessor for example may step-wise advance a task by switching power on ($P_i = P_{max}$) and off ($P_i = 0$). By tuning the so-called duty cycle accordingly, devices can approximate the average power $0 \leq \overline{P}_i \leq P_{max}$. For a more detailed discussion about practical task processing and the system model in general, see Section 7.

4. Lazy Scheduling Algorithms LSA

After having described our modeling assumptions, we will now state and prove optimal scheduling algorithms. In subsection 4.1, we will start with the

analysis of a simplified scheduling scenario where tasks need only energy as computation resource but may execute in zero time. By disregarding the computation resource time, we focus on the energy-driven nature of the scheduling scenario presented in this paper. In Section 4.2, we will consider finite execution times as well and construct a more general algorithm which manages to optimally trade off energy and time constraints. Theorems which prove optimality of both algorithms will follow in subsection 4.3.

4.1 Simplified Lazy Scheduling

We start with a node with infinite power $P_{max} = +\infty$. As a result, a task's execution time w_i collapses to 0 if the available energy E_C in the storage is equal to or greater than the task's energy demand e_i. This fact clearly simplifies the search for an adequate scheduling algorithm but at the same time contributes to the understanding of the problem.

As already indicated in the introduction, the naive approach of simply scheduling tasks with the EDF algorithm may result in unnecessary deadline violations, see Fig. 1. It may happen, that after the execution of task "1" another task "2" with an earlier deadline arrives. If now the required energy is not available before the deadline of task "2", EDF scheduling produces a deadline violation. If task "1" would hesitate instead of executing directly, this deadline violation might be avoidable. These considerations directly lead us to the principle of *Lazy Scheduling*: Gather environmental energy and process tasks only if it is necessary.

The *Lazy Scheduling Algorithm* LSA-I for $P_{max} = \infty$ shown below attempts to schedule a set of tasks J_i, $i \in Q$ such that deadlines are respected. Therefore, the processing device has to decide between three power modes. The node may process tasks with the maximal power $P_D(t) = P_{max}$ or not at all ($P_D(t) = 0$). In between, the node may choose to spend only the currently incoming power $P_S(t)$ from the harvesting unit on tasks. The algorithm is based on the three following rules:

- **Rule 1**: If the current time t equals the deadline d_j of some arrived but not yet finished task J_j, then finish its execution by draining energy $(e_j - E_j(a_j, t))$ from the energy storage instantaneously.
- **Rule 2**: We must not waste energy if we could spend it on task execution. Therefore, if we hit the capacity limit ($E_C(t) = C$) at some times t, we execute the task with the earliest deadline using $P_D(t) = P_S(t)$.
- **Rule 3**: Rule 1 overrules Rule 2.

Note that LSA-I 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 $C = +\infty$.

Lazy Scheduling Algorithm LSA 1 ($P_{max} = \infty$)

Require: maintain a set of indices $i \in Q$ of all ready but not finished tasks J_i
  $P_D(t) \Leftarrow 0$;
  while (true) **do**
    $d_j \Leftarrow \min\{d_i : i \in Q\}$;
    process task J_j with power $P_D(t)$;
    $t \Leftarrow$ 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;
      $P_D(t) \Leftarrow 0$;
    if $E_C(t) = C$ **then** $P_D(t) \Leftarrow P_S(t)$;

4.2 General Lazy Scheduling

The LSA-I algorithm instantaneously executes a task at its deadline. However, with limited power consumption P_D and finite, minimal computation times $w_i = \frac{e_i}{P_{max}}$ a general algorithm has to determine earlier starting times $s_i \leq d_i$ in order to respect deadlines. In the following, we sketch how to determine optimal starting times s_i that balance the *time* and *energy* constraints for each single task J_i. For a more detailed derivation of the starting times s_i the reader is referred to [11].

At first sight, starting a task as late as possible (ALAP) seems to be a promising approach. The upper plots in Fig. 3 display a straightforward ALAP-translation of the starting time for task "1": To fulfill its time condition, task "1" begins to execute at starting time $s_1 = d_1 - \frac{e_1}{P_{max}}$. As illustrated, it may happen that shortly after s_1 an unexpected task "2" arrives. Assume that this unexpected task "2" is nested in task "1", i.e., it also has an earlier deadline than "1". This scenario inevitably leads to a deadline violation, although plenty of energy is available. This kind of timing conflict can be solved by shifting s_1 to earlier times and thereby reserving time for the unpredictable task "2" (see lower plots Fig. 3). But starting earlier, we risk to "steal" energy that might be needed at later times (compare Fig. 1). According to the "lazy" principle we have to take care that we don't start *too* early.

From the above example, we learned that it may be disadvantageous to prearrange a starting time in such a way, that the stored energy E_C cannot be used before the deadline of a task. If the processing device starts running at time s_i with P_{max} and cannot consume all the available energy before the deadline d_i, time conflicts may occur. On the other hand, if we remember the introductory example in Fig. 1, energy conflicts are possible if the stored energy $E_C(t)$ is 0 at some time $t < d_i$. Hence we can conclude the following: The optimal starting time s_i must guarantee, that the processor *could* continuously use P_{max} in the interval $[s_i, d_i]$ and empty the energy storage $E_C(d_i) = 0$ exactly

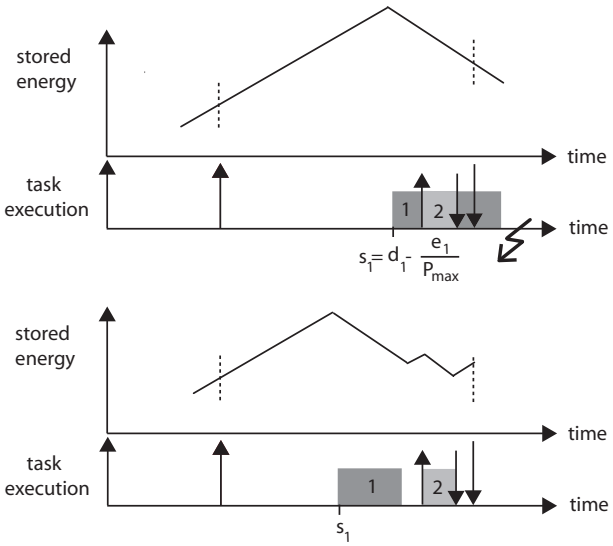


Figure 3. ALAP vs. lazy scheduling

at time d_i. Before the optimal starting time s_i, the scheduler has to conserve energy and keep the storage level E_C as high as possible.

A necessary prerequisite for the calculation of the optimal starting time s_i is the knowledge of the incoming power flow $P_S(t)$ for all future times $t \leq d_i$. Finding useful predictions for the power $P_S(t)$ can be done, e.g., by analyzing traces of the harvested power, as we will see in Section 6. In addition, we assume that $P_S(t) < P_{max}$, that is, the incoming power $P_S(t)$ from the harvesting unit never exceeds the power consumption P_{max} of a busy node. Besides from being a realistic model in many cases, this assumption ensures that no energy is wasted if the energy storage is full and the system is running with P_{max}.

To calculate the optimal starting time s_i, we determine the maximum amount of energy $E_C(a_i) + E_S(a_i, d_i)$ that may be processed by the node before d_i. Next, we compute the minimum time required to process this energy without interruption and shift this time interval of continuous processing just before the deadline d_i. In other words, we calculate the starting time s_i^* as

$$s_i^* = d_i - \frac{E_C(a_i) + E_S(a_i, d_i)}{P_{max}}.$$

If now the energy storage is filled before s_i^*, it is reasonable to advance task J_i with power P_S in order to avoid an energy overflow (compare Rule 2 of LSA-I). However, this also means that not all energy $E_C(a_i) + E_S(a_i, d_i)$ can be processed continuously in $[s_i^*, d_i]$ and we find $E_C(t) = 0$ at some

time $t < d_i$. Thus a better starting time s'_i allows for the reduced amount of energy $C + E_S(s'_i, d_i)$ which is processible in this situation:

$$s'_i = d_i - \frac{C + E_S(s'_i, d_i)}{P_{max}}$$

By choosing the maximum of s^*_i and s'_i we find the optimal starting time

$$s_i = \max\left(s'_i, s^*_i\right),$$

which precisely balances energy and time constraints for task J_i.

The pseudo-code of a general Lazy Scheduling Algorithm LSA-II is shown below. It is based on the following rules:

- **Rule 1**: EDF scheduling is used at time t for assigning the processor to all waiting tasks with $s_i \leq t$. The currently running task is powered with $P_D(t) = P_{max}$.
- **Rule 2**: If there is no waiting task i with $s_i \leq t$ and if $E_C(t) = C$, then all incoming power P_S is used to process the task with the smallest deadline ($P_D(t) = P_S(t)$).

Lazy Scheduling Algorithm LSA 2 ($P_{max} = \text{const.}$)

Require: maintain a set of indices $i \in Q$ of all ready but not finished tasks J_i
  $P_D(t) \Leftarrow 0$;
  while (true) **do**
    $d_j \Leftarrow \min\{d_i : i \in Q\}$;
    calculate s_j;
    process task J_j with power $P_D(t)$;
    $t \Leftarrow$ 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** $P_D(t) \Leftarrow P_S(t)$;
    if $t \geq s_j$ **then** $P_D(t) \Leftarrow P_{max}$;

Although it is based on the knowledge of the future incoming energy E_S, LSA-II remains an online algorithm. The calculation of s_i must be performed once the scheduler selects the task with the earliest deadline. If the scheduler is not energy-constraint, i.e., if the available energy is more than the device can consume with power P_{max} within $[a_i, d_i]$, the starting time s_i will be before the current time t. Then, the resulting scheduling policy is EDF, which is reasonable, because only time constraints have to be satisfied. If, however, the sum of stored energy E_C plus generated energy E_S 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-II can be classified as an energy-clairvoyant adaptation of the Earliest Deadline First Algorithm. It changes its behaviour according to the amount of available energy, the capacity C as well as the maximum power consumption P_{max} of the device. For example, the lower the power P_{max} gets, the greedier LSA-II gets. On the other hand, high values of P_{max} force LSA-II to hesitate and postpone the starting time s_i. For $P_{max} = \infty$, all starting times collapse to the respective deadlines, and we identify LSA-I as a special case of LSA-II. In the remainder of the paper, we will solely consider the general LSA-II algorithm derived in this section. From now on, we will denote this algorithm LSA.

4.3 Optimality of Lazy Scheduling

In this section, we will show that Lazy Scheduling algorithms optimally assign power P_D to a set of tasks. For this purpose, we formulate Theorem 1 and Theorem 2 which show that LSA makes the best use of the available time and energy, respectively. With the help of Theorem 1 and 2, we proof optimality of LSA in Theorem 3.

The scheduling scenario presented in this paper is inherently energy-driven. Hence, a scheduling algorithm yields a deadline violation if it fails to assign energy e_i to a task before its deadline d_i. We distinguish between two types of deadline violations:

- A deadline cannot be respected since the time is not sufficient to execute available energy with power P_{max}. At the deadline, unprocessed energy remains in the storage and we have $E_C(d) > 0$. We call this the **time limited** case.
- A deadline violation occurs because the required energy is simply not available at the deadline. At the deadline, the battery is exhausted (i.e., $E_C(d) = 0$). We denote the latter case **energy limited**.

For the following theorems to hold we suppose that at initialization of the system, we have a full capacity, i.e., $E_C(t_i) = C$. Furthermore, we call the computing device *idle* if no task i is running with $s_i \leq t$.

Let us first look at the time limited case.

THEOREM 1 *Let us suppose that the LSA algorithm schedules a set of tasks. At time d the deadline of a task J with arrival time a is missed and $E_C(d) > 0$. Then there exists a time t_1 such that the sum of execution times $\sum_{(i)} w_i = \sum_{(i)} \frac{e_i}{P_{max}}$ of tasks with arrival and deadline within time interval $[t_1, d]$ exceeds $d - t_1$.*

PROOF 1 *Let us suppose that t_0 is the maximal time $t_0 \leq d$ where the processor was idle. Clearly, such a time exists.*

We now show, that at t_0 there is no task i with deadline $d_i \leq d$ waiting. At first, note that the processor is constantly operating on tasks in time interval $(t_0, d]$. Suppose now that there are such tasks waiting and task i is actually the one with the earliest deadline d_i among those. Then, as $E_C(d) > 0$ and because of the construction of s_i, we would have $s_i < t_0$. Therefore, the processor would actually process task i at time t_0 which is a contradiction to the idleness.

Because of the same argument, all tasks i arriving after t_0 with $d_i \leq d$ will have $s_i \leq a_i$. Therefore, LSA will attempt to directly execute them using an EDF strategy.

Now let us determine time $t_1 \geq t_0$ which is the largest time $t_1 \leq d$ such that the processor continuously operates on tasks i with $d_i \leq d$. As we have $s_i \leq a_i$ for all of these tasks and as the processor operates on tasks with smaller deadlines first (EDF), it operates in $[t_1, d]$ only on tasks with $a_i \geq t_1$ and $d_i \leq d$. As there is a deadline violation at time d, we can conclude that $\sum_{(i)} w_i > d - t_1$ where the sum is taken over all tasks with arrival and deadline within time interval $[t_1, d]$.

It can be shown that a related result holds for the energy limited case, too.

THEOREM 2 *Let us suppose that the LSA algorithm schedules a set of tasks. At time d the deadline of a task J with arrival time a is missed and $E_C(d) = 0$. Assume further, that deadline d is the first deadline of the task set that is violated by LSA. Then there exists a time t_1 such that the sum of task energies $\sum_{(i)} e_i$ of tasks with arrival and deadline within time interval $[t_1, d]$ exceeds $C + E_S(t_1, d)$.*

PROOF 2 *Let time $t_1 \leq d$ be the largest time such that (a) $E_C(t_1) = C$ and (b) there is no task i waiting with $d_i \leq d$. Such a time exists as one could at least use the initialization time t_i with $E_C(t_i) = C$. As t_1 is the last time instance with the above properties, we can conclude that everywhere in time interval $[t_1, d]$ we either have (a) $E_C(t) = C$ and there is some task i waiting with $d_i \leq d$ or we have (b) and $E_C(t) < C$.*

It will now be shown that in both cases a) and b), energy is not used to advance any task j with $d_j > d$ in time interval $[t_1, d]$. Note also, that all arriving energy $E_S(t_1, d)$ is used to advance tasks.

In case a), all non-storable energy (i.e. all energy that arrives from the source) is used to advance a waiting task, i.e., the one with the earliest deadline $d_i \leq d$. In case b), the processor would operate on task J with $d_j > d$ if there is some time $t_2 \in [t_1, d]$ where there is no other task i with $d_i \leq d$ waiting and $s_j \leq t_2$. But s_j is calculated such that the processor could continuously work until d_j. As $d_j > d$ and $E_C(d) = 0$ this can not happen and $s_j > t_2$. Therefore, also in case b) energy is not used to advance any task j with $d_j > d$.

As there is a deadline violation at time d, we can conclude that $\sum_{(i)} e_i > C + E_C(t_1, d)$ where the sum is taken over all tasks with arrival and deadline within time interval $[t_1, d]$.

From the above two theorems we draw the following conclusions: First, in the **time limited** case, there exists a time interval before the violated deadline with a larger accumulated computing time request than available time. And second, in the **energy limited** case, there exists a time interval before the violated deadline with a larger accumulated energy request than what can be provided at best. These considerations lead us to one of the major results of the paper:

THEOREM 3 (OPTIMALITY OF LAZY SCHEDULING) *Let us consider a system characterized by a capacity C and power P_{max} driven by the energy source E_S. If LSA cannot schedule a given task set, then no other scheduling algorithm is able to schedule it. This holds even if the other algorithm knows the complete task set in advance.*

PROOF 3 *The proof follows immediately from Theorems 1 and 2. Assume a set of tasks is scheduled with LSA and at time d the deadline of task J is missed. Assume further, that deadline d is the first deadline of the task set that is violated by LSA. In the following, we distinguish between the case where the energy at the deadline is $E_C(d) > 0$ and $E_C(d) = 0$, respectively.*

In the first case, according to Theorem 1, there exists a time t_1 such that the sum of execution times $\sum_{(i)} w_i = \sum_{(i)} \frac{e_i}{P_{max}}$ of tasks with arrival and deadline within time interval $[t_1, d]$ exceeds $d - t_1$. Here, knowing arrival times, energy demands and deadlines in advance does not help, since every scheduling algorithm will at least violate one deadline in $[t_1, d]$.

In the energy limited case with $E_C(d) = 0$, according to Theorem 2, there exists a time t_1 such that the sum of task energies $\sum_{(i)} e_i$ of tasks with arrival and deadline within time interval $[t_1, d]$ exceeds $C + E_S(t_1, d)$. Hence, no algorithm can hold deadline d without violating an earlier deadline in $[t_1, d]$. This holds also for omniscient scheduling algorithms, since (a) at the beginning of the critical interval $[t_1, d]$, the energy level may be $E_C(t_1) = C$ at most and (b) the execution of the critical tasks can start at time t_1 at the earliest.

So every time LSA violates deadline d, we have either the time limited case ($E_C(d) > 0$) or the energy limited case ($E_C(d) = 0$). Since in both cases it is impossible for another algorithm to respect deadline d and all earlier deadlines simultaneously, we conclude that LSA is optimal.

If we can guarantee that there is no time interval with a larger accumulated computing time request than available time and no time interval with a larger accumulated energy request than what can be provided at best, then the task set is schedulable. This property will be used to determine the admittance test described in the next section.

On the other hand, given a task set and a certain energy source $E_S(t)$ we can also make a statement about the necessary hardware requirements of the sensor node: Due to its optimality, LSA requires the minimum processing power P_{max} and the minimum capacity C necessary to avoid deadline violations:

THEOREM 4 (OPTIMAL TUPLE $(C\,; P_{max})$) *Let us assume a given task set has to be scheduled using an energy source E_S. Among all algorithms, LSA requires the minimum capacity C and the minimum power P_{max} that are necessary to successfully schedule the task set.*

PROOF 4 *We proceed by contradiction. Let us denote C and P_{max} the minimum capacity and the minimum processing power which are needed to schedule a given task set with LSA. Assume that an adversary algorithm succeeds to schedule the task set with some $C' < C$ or $P'_{max} < P_{max}$ given the same energy source. This means the adversary algorithm can schedule the respective task set with (C', P'_{max}) and LSA cannot. This however contradicts the optimality of LSA according to Theorem 3.*

The admittance test in the next section will allow us to explicitly determine the minimum values of C and P_{max} for LSA scheduling.

5. Admittance Test

5.1 Lazy Scheduling Algorithm

In this section, we will determine an offline schedulability test in case of periodic, sporadic or even bursty sets of tasks. In particular, given an energy source with lower EVCC $\epsilon^l(\Delta)$, the device parameters ($C\,; P_{max}$) and a set of periodic tasks J_i, $i \in I$ with period p_i, *relative* deadline d_i and energy demand e_i, we would like to determine whether all deadlines can be respected.

To this end, let us first define for each task its arrival curve $\alpha(\Delta)$ which denotes the maximal number of task arrivals in any time interval of length Δ. The concept of arrival curves to describe the arrival patterns of sets of tasks is well known (request bound functions) and has been used explicitly or implicitly in, e.g., [4] or [18]. To simplify the discussion, we limit ourselves to periodic tasks, but the whole formulation allows to deal with much more general classes (sporadic or bursty) as well.

In case of a periodic task set, we have for periodic task J_i, see also Fig. 4:

$$\alpha_i(\Delta) = \left\lceil \frac{\Delta}{p_i} \right\rceil \quad \forall \Delta \geq 0$$

In order to determine the maximal energy demand in any time interval of length Δ, we need to maximize the accumulated energy of all tasks having their arrival and deadline within an interval of length Δ. To this end, we need

to shift the corresponding arrival curve by the relative deadline. We are doing this since the actual energy demand becomes due at the deadline of the respective task. In case of a periodic task J_i, this simply leads to:

$$\overline{\alpha}_i(\Delta) = \begin{cases} e_i \cdot \alpha_i(\Delta - d_i) & \Delta > d_i \\ 0 & 0 \leq \Delta \leq d_i \end{cases}$$

In case of several periodic tasks that arrive concurrently, the total demand curve $A(\Delta)$ (called demand-bound function in [4]) can be determined by just adding the individual contributions of each periodic task, see Fig. 4:

$$A(\Delta) = \sum_{i \in I} \overline{\alpha}_i(\Delta)$$

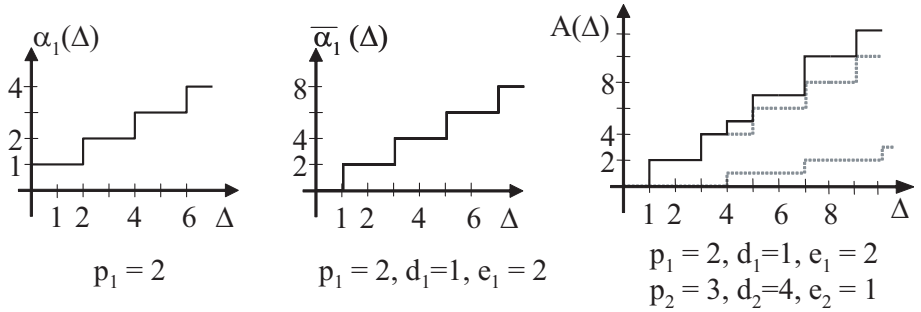


Figure 4. Examples of an arrival curve $\alpha_i(\Delta)$, a demand curve $\overline{\alpha}_i(\Delta)$ and a total demand curve $A(\Delta)$ in case of periodic tasks

Using the above defined quantities, we can formulate a schedulability test for the LSA algorithm that can be applied to a quite general class of tasks specifications.

THEOREM 5 (LSA SCHEDULABILITY TEST) *A given set of tasks J_i, $i \in I$ with arrival curves $\alpha_i(\Delta)$, energy demand e_i and relative deadline d_i is schedulable under the energy-driven model with initially stored energy C, if and only if the following condition holds*

$$A(\Delta) \leq \min\left(\epsilon^l(\Delta) + C\,,\, P_{max} \cdot \Delta\right) \quad \forall \Delta > 0$$

Here, $A(\Delta) = \sum_{i \in I} e_i \cdot \alpha_i(\Delta - d_i)$ denotes the total energy demand of the task set in any time interval of length Δ, $\epsilon^l(\Delta)$ the energy variability characterization curve of the energy source, C the capacity of the energy storage and P_{max} the maximal processing power of the system. In case of periodic tasks we have $A(\Delta) = \sum_{i \in I} e_i \cdot \left\lceil \frac{\Delta - d_i}{p_i} \right\rceil$.

PROOF 5 *The proof of the if direction is omitted, since it is a direct consequence of Theorems 1 and 2. We just prove the only-if direction.*

Remember that the total demand curve $A(\Delta)$ denotes the maximal energy demand of tasks in any interval $[t_1, t_2]$ of length Δ. It equals the maximal accumulated energy of tasks having their arrival **and** *deadline within $[t_1, t_2]$. Therefore, in order to satisfy all deadlines for these tasks, at least energy $A(t_2 - t_1)$ must be available.*

Let us suppose that the condition in Theorem 5 is violated for some Δ due to missing energy. Let us suppose also that the task arrival curve and the energy variability characterization curve are strict, i.e., there exists some time interval $[t_1, t_2]$ where the energy demand is $A(t_2 - t_1)$ and at the same time the energy $E_S(t_2 - t_1)$ is received. Then in time interval $[t_1, t_2]$ with $\Delta = t_2 - t_1$ the difference between the energy demand and the received energy is larger than the maximal stored energy C as $A(\Delta) > \epsilon^l(\Delta) + C$. As a result, the task set is not schedulable.

On the other hand, whenever the demanded computation time $\frac{A(\Delta)}{P_{max}}$ of a task set in the interval Δ is larger than the interval itself, a task set is not schedulable. Therefore it is evident, that both the energy condition $A(\Delta) \leq \epsilon^l(\Delta) + C$ and the time condition $A(\Delta) \leq P_{max} \cdot \Delta$ must be fulfilled in order to avoid deadline violations.

Theorem 5 tells us that we can decouple energy and time constraints if we have to decide whether a task set is schedulable or not. On the one hand, only if

$$P_{max} \geq \max_{0 \leq \Delta} \left(\frac{A(\Delta)}{\Delta} \right) ,$$

the system is fast enough to process the given task set. This condition is independent of the energy provided by the environmental source (i.e. ϵ^l) and the capacity of the storage device. Even increasing the capacity does not help. If a task set however satisfies the time constraint, the role of the capacity C as a design parameter for the energy harvesting system becomes important.

Suppose now that the time constraint is fulfilled. For this case, Theorem 5 states that the capacity C needed to schedule a task set with $A(\Delta)$ using a source with $\epsilon^l(\Delta)$ is constrained by

$$C \geq \max_{0 \leq \Delta} \Big(0, A(\Delta) - \epsilon^l(\Delta) \Big) .$$

Fig. 5 illustrates an example schedulability test. The left diagram displays the total demand curve $A(\Delta)$ for two periodic tasks with p_1=2, d_1=1, e_1=2 and p_2=3, d_2=4, e_2=1. Furthermore, the EVCC $\epsilon^l(\Delta)$ is given by a piecewise linear function using three pieces (0,0,0),(2,0,1),(5,3,3), where each piece i is defined by the triple of the form (initial Δ_i, initial $\epsilon^l(\Delta_i)$, slope of piece i). Now, the

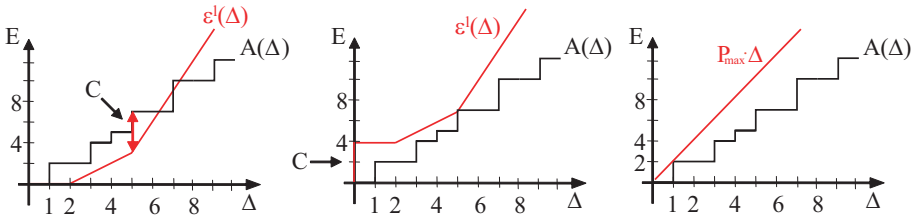


Figure 5. Determining the optimal tuple $(C; P_{max})$ according to theorem 5

maximal difference between the total demand curve A and the EVCC ϵ^l can be computed. It has value 4 and is obtained at $\Delta = 5$. Therefore, one can conclude that the set of tasks can be scheduled (with LSA) using a minimal capacity $C = 4$. The respective schedulability test with $C = 4$ is shown in the middle diagram of Fig. 5. As displayed in the right diagram, power $P_{max} = 2$ is required to fulfill the time condition in theorem 5.

As a last point to mention, let us consider the middle diagram in Fig. 5 once again. Regarding the slopes of the curves, we can guess that A and ϵ^l won't intersect after the critical time interval of length 5. Formally, this is because the minimum average power $\lim\limits_{\Delta\to\infty} \frac{\epsilon^l(\Delta)}{\Delta}$ is higher than the maximum average power demand $\lim\limits_{\Delta\to\infty} \frac{A(\Delta)}{\Delta}$ of the task set. Simply stated, the average harvested power is higher than the average power demand of the application. It becomes evident that an optimal algorithm, like LSA, can only optimize the short-term behaviour of the system by suitable power management. LSA achieves the minimum capacity C needed to temporarily buffer energy for single tasks, but on the long run the average of power P_D is dictated by the task set.

5.2 Comparison to EDF

It is useful to formally compare LSA with the well know EDF algorithm in terms of the schedulability region and the required capacity C. In order to simplify the discussion, we will investigate an energy limited scenario only, see Section 4.1 and 4.3. Then we can state the following result:

THEOREM 6 (EDF SCHEDULABILITY TEST) *A given set of tasks J_i, $i \in I$ with arrival curves $\alpha_i(\Delta)$, energy demand e_i and relative deadline d_i is schedulable with initially stored energy C, only if the following condition holds for any deadline d_k, $k \in I$:*

$$\sum_{i\in I} e_i \cdot \alpha_i(\Delta - d_k) \leq C + \epsilon^l(\Delta) \quad \forall \Delta > 0$$

In case of periodic tasks with period p_i we have $\alpha_i(\Delta) = \lceil \frac{\Delta}{p_i} \rceil$.

PROOF 6 *Remember that the left hand side of the condition denotes the maximal energy used by all the tasks in any interval* $[t_1, t_2]$ *of length* $t_2 - t_1 = \Delta - d_k$*. There is an instance of task arrivals compliant with the arrival curves* α_i *such that task* J_k *arrives at time* t_2*, i.e. by correctly adjusting the phase of all instances of task* J_k*. In this case, the deadline of the task instance arriving at* t_2 *is* $t_2 + d_k$*. In order to be able to execute this instance within its deadline, the available energy in any interval* $[t_1, t_2 + d_k]$ *must be larger than* $\sum_{i \in I} e_i \cdot \alpha_i(t_2 - t_1)$*, i.e. the energy used by tasks arriving in* $[t_1, t_2]$*. The maximal energy available in* $[t_1, t_2 + d_k]$ *is in the worst case given by* $C + \epsilon^l(t_2 + d_k - t_1)$*. Replacing* $t_2 - t_1$ *by* $\Delta - d_k$ *yields the desired result.*

The strongest bound is obtained by using the task J_k with the smallest deadline $d^{min} = \min_{i \in J}\{d_i\}$. Comparing Theorems 5 and 6 in the energy-constraint case we obtain the two constraints $\sum_{i \in I} e_i \cdot \alpha_i(\Delta - d^{min}) \leq C + \epsilon^l(\Delta)$ for EDF and $\sum_{i \in I} e_i \cdot \alpha_i(\Delta - d_i) \leq C + \epsilon^l(\Delta)$ for LSA. Clearly, EDF has a smaller schedulability region as $\sum_{i \in I} e_i \cdot \alpha_i(\Delta - d^{min}) \geq \sum_{i \in I} e_i \cdot \alpha_i(\Delta - d_i)$ for all $\Delta \geq 0$.

Finally, let us derive specialized results in the case of periodic tasks J_i with $p_i = d_i$ (period equals deadline) and a simple energy variability characterization curve $\epsilon^l(\Delta)$ shown in Fig. 6 with $\epsilon^l(\Delta) = \max\{\sigma \cdot (\Delta - \rho), 0\}$.

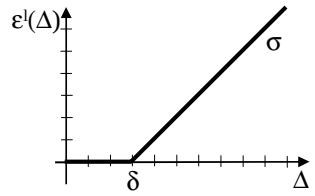


Figure 6. Simple EVCC for comparing EDF and LSA scheduling

We also suppose that the available average power σ from the energy source is sufficient to support the long term power demand $\bar{\sigma}$ of the task set

$$\sigma \geq \bar{\sigma} = \sum_{i \in I} \frac{e_i}{p_i}$$

as otherwise, deadline violations are unavoidable. In the following comparison, we suppose that the energy source has a minimal average power, i.e. $\sigma = \bar{\sigma}$, i.e. it is as weak as possible. Under these assumptions (periodic task with periods equal deadlines, energy-limited scenario) and using the results of Theorems 5 and 6, one can compute the minimal possible capacities of the energy storage for the two scheduling methods LSA and EDF as follows:

$$C^{EDF} = \sum_{i \in I} e_i + (\delta - p^{min}) \cdot \bar{\sigma} \qquad ; \qquad C^{LSA} = \delta \cdot \bar{\sigma}$$

Therefore, the relative gain in the necessary storage capacity between the two scheduling methods can be quantified and bounded by

$$0 \leq \frac{C^{EDF} - C^{LSA}}{C^{LSA}} = \frac{1}{\delta \cdot \bar{\sigma}} (\sum_{i \in I} e_i + p^{min} \cdot \bar{\sigma}) \leq \frac{p^{max} - p^{min}}{\delta}$$

For the bounds we use the fact that $p^{min} \leq (\sum e_i)(\sum (e_i/p_i)) \leq p^{max}$ where p^{min} and p^{max} denote the minimal and maximal period of tasks J_i, respectively. In other words, the maximal relative difference in storage capacity depends on the differences between the task periods. The larger the difference between the largest and smallest period is, the large the potential gain in storage efficiency for the LSA algorithm.

6. Simulation Results

In the previous section, a method to compute the minimum capacity for a certain energy source characterization ϵ^l was presented. In the following, we will call this optimal value C_{min}. The value C_{min} obtained represents a lower bound since it is obtained for energy-clairvoyant LSA scheduling. In addition, it remains unclear, which capacities C^*_{min} are required if other scheduling disciplines are applied. For this reason, we performed a simulative study to evaluate the achievable capacity savings in a more realistic scenario. The EDF algorithm – which is optimal in traditional scheduling theory– serves as a benchmark for our studies.

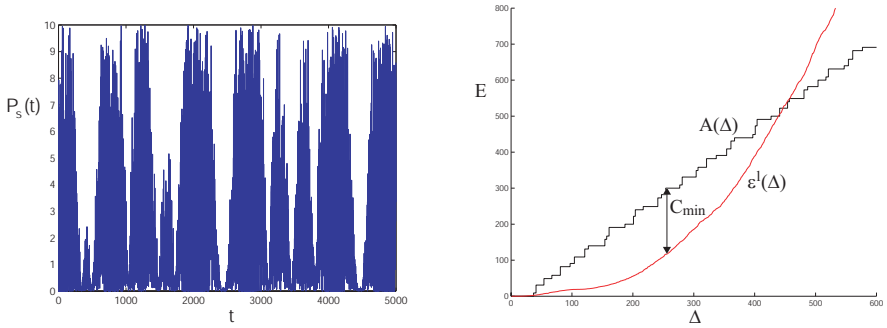
We investigated variants of LSA which utilize the measured EVCCs ϵ^l and ϵ^u to *predict* the future generated energy $E_S(t)$ for the LSA algorithm. Each time a starting time s_i has to be calculated for the task i with the earliest deadline d_i, the energy $\epsilon^l(d_i - a_i)$ (or $\epsilon^u(d_i - a_i)$) plus the stored energy $E_C(a_i)$ is assumed to be processible before the deadline.

The trace of the power source $P_S(t)$ is generated by a random number generator according to

$$P_S(t) = \left| 10 \cdot N(t) \cdot \cos(\frac{t}{70\pi}) \cdot \cos(\frac{t}{100\pi}) \right| \ ,$$

where $N(t)$ denotes a normally distributed random variable with mean 0 and variance 1. The values of P_S have been cut off at the value $P_{S,max} = 10$. As illustrated in Fig. 7(a), the obtained power trace $P_S(t)$ exhibits both stochastic and deterministic, periodic behaviour. The latter is simulating day and night periods similar to those experienced by solar cells in an outdoor environment. From this trace we compute the average power $\overline{P_S}$ as well as upper and lower EVCCs ϵ^u and ϵ^l.

A task set consists of an arbitrary number of periodic tasks. Periods p are taken from a set $\{10, 20, 30, \ldots, 100\}$, each value having an equal probabil-



(a) Power trace $P_S(t)$ (b) Determining C_{min} for a random task set

Figure 7. Calculation of C_{min} in two steps: (1) Extract $\epsilon^l(\Delta)$ from $P_S(t)$ and (2) Compute C_{min} for every task set with respective energy demand $A(\Delta)$

ity of being selected. The initial phases φ are uniformly distributed between [0,100]. For simplicity, the relative deadline d is equal to the period p of the task. The energies e of the periodic tasks are generated according to a uniform distribution in $[0, e_{max}]$, with $e_{max} = \overline{P_S} \cdot p$.

We define the utilization $U \in [0, 1]$ of a scheduler as

$$U = \sum_i \frac{\frac{e_i}{\overline{P_S}}}{p_i} .$$

One can interpret U as the percentage of processing time of the device if tasks are solely executed with the average incoming power $\overline{P_S}$. A system with, e.g., $U > 1$ is processing more energy than it scavenges on average and will deplete its energy reservoir.

In dependence of the generated power source $P_S(t)$, N task sets are generated which yield a certain processor utilization U. For that purpose, the number of periodic tasks in each task set is successively incremented until the intended utilization U is reached. Hence, the accuracy of the utilization U is varying $\pm 1\%$ with respect to its nominal value.

At the beginning of the simulation, the energy storage is full. We set $P_{max} = 10$. The simulation terminates after 10000 time units and is repeated for 5000 task sets. In order to show the average behaviour of all task sets in one plot, we normalized the capacities C with the respective C_{min} of the task set. Fig. 7(b) shows the calculation of C_{min} for a random task set.

Fig. 8 illustrates the percentage of tasks that could be scheduled without deadline violations for different utilizations U. Clearly, no deadline violations occur for energy-clairvoyant LSA scheduling and values of $\frac{C}{C_{min}} \geq 1$. For all values of U, both approximations of LSA with ϵ^l and ϵ^u outperform the EDF

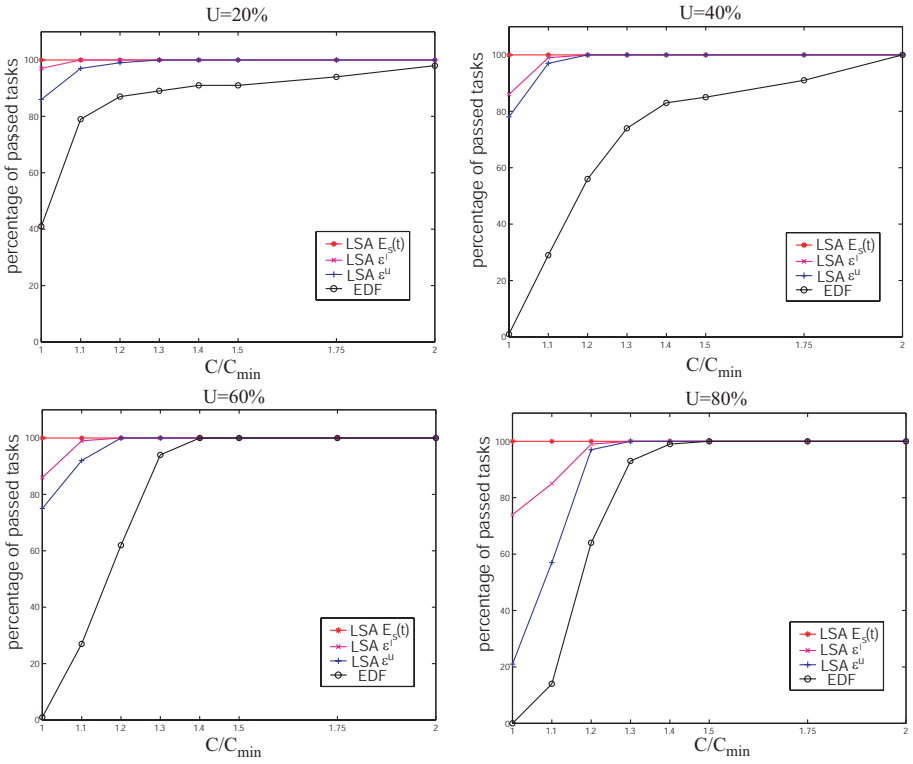


Figure 8. Comparison of pure LSA, LSA using ϵ^l, LSA using ϵ^u and EDF for different utilizations U

algorithm, whereat the lower curve ϵ^l seems to be the better approximation. At $U = 40\%$ and $C = C_{min}$, e.g., almost no task set is schedulable with EDF. Here, LSA with ϵ^u is able to schedule $\approx 78\%$ of all task sets; LSA with ϵ^l even $\approx 85\%$.

Concerning the relative capacity savings achieved with our algorithms, we are especially interested in the smallest capacities C necessary to avoid any deadline violations. The highest savings are obtained at $U = 20\%$, where EDF needs more than $2.0 \cdot C_{min}$ to respect deadlines whereas LSA using ϵ^l shows the same behaviour at $1.1 \cdot C_{min}$. This translates into capacity savings of $\approx 45\%$. At higher values of the utilization U these savings are decreasing, yet they are still significant: At utilizations of $U = 40\%, 60\%, 80\%$, the capacity savings of LSA with ϵ^l compared to EDF are still $\approx 40\%, 21\%, 20\%$, respectively.

Albeit randomized task sets are not necessary representative for all kind of applications, these simulation results demonstrate that significant capacity savings are possible. If the application involves bursty instead of periodic task processing, the benefits of Lazy Scheduling may be indeed even more strik-

ing: As showed in [11], a greedy algorithm like EDF may violate an arbitrary number of deadlines and may suffer from worst case scenarios. This holds in particular for sensor nodes, where the energy demands of different tasks are highly varying (e.g. communication, sensing and data processing tasks) and tasks have to satisfy various timing constraints (e.g. urgent and less urgent tasks which have to run in parallel).

7. Practical Considerations

The system model introduced in Section 3 and used throughout this paper implies idealized modelling abstractions, which demand further explanations. Therefore, this section is dedicated to general implementation aspects and possible application scenarios

7.1 Energy Source Predictability

Clearly, the performance of LSA is strongly dependent on the accuracy of the predicted power $P_S(t)$ of the harvesting unit. The better the approximation, the better the algorithm performs in terms of optimality. As illustrated by the simulation results of the previous section, energy variability characterization curves (EVCC) are suitable for that purpose. Especially for small utilizations U of the sensor node, EVCCs appear to converge towards the optimal, energy-clairvoyant LSA. It should be mentioned, that the prediction of $E_S(t)$ by EVCCs may even be improved if the sensor node is learning the characteristics of the energy source adaptively: By observing energy values $E_S(\Delta')$ for past intervals Δ', the prediction for future intervals Δ can be optimized online. This extension, however, increases at the same time the computational demand of the scheduler, which is one of the advantages of using simple EVCCs.

Solar energy harvesting through photovoltaic conversion is deemed a number one candidate for the power source P_S described in our model. If we assume the sensor node to be placed in an outdoor environment, the impinging radiation is variable with time, but follows a diurnal as well as annual cycle. Moreover, during short time intervals, the produced power P_S can be regarded as constant and sudden changes of the light intensity are improbable. Due to this specific nature of solar energy, a two-tiered prediction methodology is self-evident: On the one hand, long-run predictions must be made for less urgent tasks with rather late deadlines. Here, using exponential decaying factors to weight the history of recorded powers P_S is one possibility. An alternative is to combine daily and seasonal light conditions of the past with the knowledge about a sensor's environment and possible shadowing. One can think of a plurality of prediction mechanisms, which are clearly out of the scope of this paper.

For urgent tasks with close deadlines within milliseconds or seconds, intelligent prediction algorithms may not be necessary. Here, tasks like, e.g., sending a few bits over the wireless channel may be planned assuming constant power $P_S(t) = P_{S,const}$ during $s_i \leq t \leq d_i$. For stationarity of the power inflow P_S the calculation of the starting time s_i for a task i simplifies to

$$s_i = d_i - \min\left(\frac{E_C(a_i) + (d_i - a_i)P_{S,const}}{P_{max}}, \frac{C}{P_{max} - P_{S,const}}\right).$$

In the worst case, a sensor node is powered by an energy source with pure stochastic behaviour. If nothing is known about this source, the currently stored energy E_S is the only indicator for making scheduling decisions. By iteratively updating the starting time $\hat{s}_i = d_i - \frac{E_C(t)}{P_{max}}$ (and thereby increasing the computational overhead) starting task i too early can be avoided. However, once the device is running with P_{max}, the incoming energy $E_S(\hat{s}_i, d_i)$ may not be processible in the remaining interval $[\hat{s}_i, d_i]$. Consequently, optimality cannot be guaranteed for this scenario since the starting time $\hat{s}_i$ is always earlier then the optimal starting time s_i.

7.2 Task Processing

The task processing model presented in this paper exhibits two major assumptions:

1 We assume the power $P_D(t)$ driving a task to be *continuously* adjustable with respect to its value in $[0; P_{max}]$ as well as with respect to time t. That is, at any point in time a task can be advanced with an accurately defined amount of power.

2 We assume a *linear* relationship between the power P_D used for executing a task and the execution time w. We can say: the higher the power P_D, the shorter the execution time w.

The first modelling assumption is only needed in situations when the energy storage is full ($E_C(t) = C$). In practice, there is no existing hardware that supports a continuous consumption of the scavenged power $P_S(t)$ as claimed by LSA. A microcontroller, e.g., drains roughly constant power from the battery when running a piece of program. The same holds for the radio interface. When transmitting a certain amount of data, most radios won't operate properly with unstable power supply. Therefore, we assume that the respective hardware attempts to approximate the power level of the power source by continuously switching power on ($P_D = P_{max}$) and off ($P_D = 0$). In Fig. 9, the achieved average power $\overline{P_D} \approx P_S$ is sketched.

It becomes evident that in an implementation, one will have to respect a certain granularity. The scheduler needs to determine when the energy storage is

full and then, a task is executed for a given interval of time Δt which results in an energy consumption of $\Delta E = P_{max} \cdot \Delta t - E_S(\Delta t)$. For a microcontoller or a sensing unit, the time intervals Δt can be considered rather short while radio communication may require larger Δt due to packetized nature of the transmitted data. During the subsequent idle time, the stored energy is recovering again. In the worst case, this "duty cycling" results in a stored energy that is reduced by

$$\Delta E_{max} = P_{max} \cdot \max\{\Delta t\} - \epsilon^l(\max\{\Delta t\}))$$

in comparison to ideal LSA. With $\max\{\Delta t\}$ we denote the maximum period of continuous processing that can be observed for a given task. In terms of the admittance test in section 5, a task set is schedulable if it is schedulable under ideal LSA with reduced capacity $C - \Delta E_{max}$.

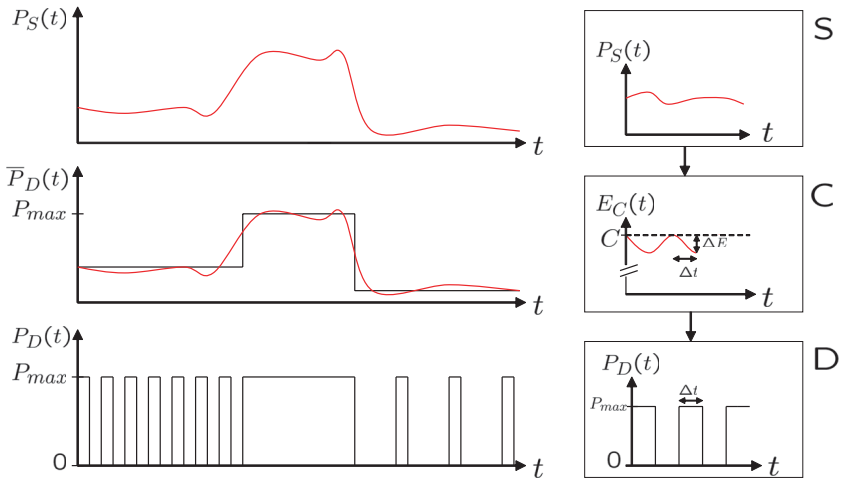


Figure 9. Approximated power consumption $\overline{P_D} \approx P_S$ by means of duty cycling and resulting non-ideal storage level $E_C(t) = C - \Delta E$

In the light of the considerations regarding assumption 1, also assumption 2 becomes plausible. Unlike common power management solution like Dynamic Voltage Scaling DVS, the energy e_i consumed by a task is the same regardless of the power P_D used during its execution. Although attractive due to its potential to *save* energy, DVS will come to its end if feature sizes of ICs get smaller. Hence, DVS or similar techniques were not considered in our work. In contrast, time and energy are assumed to be directly proportional – as indicated in Fig. 9 – with power $\overline{P_D}$ as constant of proportionality. The greater the number and size of time slots allocated to a given task, the higher the average power $\overline{P_D}$ and the faster the execution.

7.3 Energy Storage Model

An important step for the validation of the theory presented in this paper is the discussion of the energy storage model. Looking at the various devices available on the market, there are two principal methods to store energy in a small volume or mass device: using an electro-chemical process or just performing physical separation of electrical charges across a dielectric medium. The first technique is used by rechargeable batteries and it is currently the most common and for long time it was the only method to achieve high capacities in a small size. Nevertheless, research in the last years has found new materials in order to increase the specific energy of capacitors, producing devices that are called supercapacitor or ultra-capacitor [7]. As shown in the so-called 'Ragone plot' in Fig. 10, supercapacitors offer a trade-off between power- as well as energy-density, filling the gap between batteries and capacitors. Beyond their ability to support higher power flows than batteries, supercapacitors overcome many other drawbacks of batteries: They have very long lifetimes and tolerate an almost unlimited number of charge/recharge cycles without performance degradation. Unlike batteries, no heat is released during charging/discharging due to parasitic, chemical reactions. In the following, we will focus on supercapacitors as possible candidates for an energy storage device.

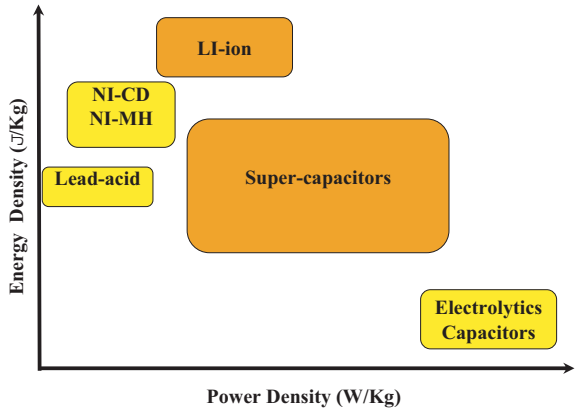


Figure 10. Power and energy characteristics of storage devices

Charge retention. Self-discharging is a natural phenomena that occurs in all kind of storage devices. It is caused by leakage currents that flow inside the device discharging it. Supercapacitors exhibit leakage currents that are typically in the order of magnitude of μA (see e.g. [9]). Moreover, it should be mentioned that the leakage is proportional to the energy level. In [5], the

leakage behaviour of different supercapacitors have been tested. From Fig.2 in the latter work it becomes evident, that there exist a potential to minimize the leakage of fully charged supercapacitors by appropriate choice (manufacturer technology) and arrangement (serial, parallel) of devices.

Apart from the intrinsic energy leakage of supercapacitors, also the idle power consumption of the sensor node has to be considered. This "external leakage" can be reduced by switching to a low-power mode if no tasks are executed. In case of a low power wireless sensor node like Moteiv's Tmote Sky [10], its ultra low power Texas Instruments MSP430 F1611 microcontroller exhibits a maximum current of $3.0\mu A$ in low power mode (LPM3). The wakeup to active mode is finished after $6\mu s$.

Altogether, it can be assumed that the energy conservation laws described in Section 3.2 hold and introducing an additional term allowing for energy leakage is dispensable.

Monitoring the stored energy. An important feature of energy harvesting systems is the capability to estimate the remaining energy in the storage device. Unlike batteries, the energy of supercapacitors can be measured in a straightforward way: The equations describing the physical behaviour of supercapacitors are nearly the same as the ones for ordinary capacitors, and the energy stored is hence $E_C \approx \frac{1}{2}CV^2$.

Storage efficiency. The efficiency η of a supercapacitor can be regarded as the quantity that relates the power flows and energies displayed in Fig. 10. Since charging a supercapacitor with P_S and discharging it with P_D can be seen as symmetrical operations, let us consider the efficiency η when the supercapacitor is charged. In this case, the increment of stored energy in time interval $\Delta = t_2 - t_1$ can be written as

$$E_C(t_2) - E_C(t_1) = \int_{t_1}^{t_2} P_S(t)\mathrm{d}t = \eta \int_{t_1}^{t_2} P_{S,raw}(t)\mathrm{d}t\,,$$

where $P_{S,raw}$ denotes the power fed into the supercapacitor. In general, supercapacitors may suffer from low efficiencies η due to their high equivalent series resistance [3]. This circumstance, however, does not jeopardize our modeling assumptions as such since, as stated in section 3.1, all losses are included in the definition of P_S. Actually, the more important property of the efficiency η is its independence of the energy level $E_C(t)$ at time t, which is approximately true for supercapacitors. Moreover, supercapacitors barely produce thermal heat which could reinforce non-linear charging/discharging behaviour.

In summary, we conclude that a linear charging/discharging behaviour with constant efficiency η is a reasonable abstraction for the example of a supercapacitor and hence our modeling assumptions hold. It should be mentioned,

that possible variations of the efficiency η have been disregarded in related work like [5] and [6], too.

8. Related Work

In [6], the authors use a similar model of the power source as we do. But instead of executing concrete tasks in a real-time fashion, they propose tuning a node's duty cycle dependent on the parameters of the power source. Nodes switch between active and sleep mode and try to achieve sustainable operation. This approach only indirectly addresses real-time responsiveness: It determines the latency resulting from the sleep duration.

The approach in [15] is restricted to a very special offline scheduling problem: Periodic tasks with certain rewards are scheduled within their deadlines according to a given energy budget. The overall goal is to maximize the sum of rewards. Therefore, energy savings are achieved using Dynamic Voltage Scaling (DVS). The energy source is assumed to be solar and comprises two simple states: day and night. Hence the authors conclude that the capacity of the battery must be at least equal to the cumulated energy of those tasks, that have to be executed at night. In contrast, our work deals with a much more detailed model of the energy source. We focus on scheduling decisions for the online case when the scheduler is *indeed* energy-constraint. In doing so, we derive valuable bounds on the necessary battery size for arbitrary energy sources and task sets.

The research presented in [1] is dedicated to offline algorithms for scheduling a set of periodic tasks with a common deadline. Within this so-called "frames", the order of task execution is *not* crucial for whether the task set is schedulable or not. The power scavenged by the energy source is assumed to be constant. Again – by using DVS – the energy consumption is minimized while still respecting deadlines. Contrary to this work, our systems (e.g. sensor nodes) are predominantly energy constrained and the energy demand of the tasks is fixed (no DVS). We propose algorithms that make best use of the available energy. Provided that the average harvested power is sufficient for continuous operation, our algorithms minimize the necessary battery capacity.

The primary commonality of [13] and our work is the term "lazy scheduling". In [13], lazy packet scheduling algorithms for transmitting packetized information in a wireless network are discussed. The approach is based on the observation that many channel coding schemes allow to reduce the energy per packet if it is transmitted slower, i.e. over a longer duration. Goal of this approach is to minimize the energy, whereas in our work the energy consumption is fixed (given by the task set). Furthermore, their scheduling algorithms are fully work conserving, which is not true for our algorithms.

9. Conclusions

We studied the case of an energy harvesting sensor node that has to schedule a set of tasks with real-time constraints. The arrival times, energy demands and deadlines of the tasks are not known to the node in advance and the problem consists of assigning the right amount of power in the right order to those tasks. For this purpose, we constructed optimal Lazy Scheduling Algorithms LSA which are energy-clairvoyant, i.e., the generated energy in the future is known. Contrary to greedy scheduling algorithms, LSA hesitates to power tasks until it is necessary to respect timing constraints. As a further result, we discuss an admittance test that decides, whether a set of energy-driven tasks can be scheduled on a sensor node without violating deadlines. This admittance test simultaneously shads light on the fundamental question of how to dimension the capacity of the energy storage: Provided that the average harvested power is sufficient for continuous operation, we are able to determine the minimum battery capacity necessary. Furthermore, achievable capacity savings between 20% and 45% are demonstrated in a simulative study, comparing the classical Earliest Deadline First algorithm with a variant of LSA, which uses energy variability characterization curves EVCC as energy predictor. Finally, practical considerations are provided that suggest practical applicability of the theoretical results.

By starting to study a single node, we believe that extensions towards multihop networks where end-to-end deadlines have to be respected are possible. If sensors jointly perform a common sensing task, distributed energy management solutions are needed. An example for the common task could be redundantly deployed sensors with overlapping coverage regions.

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] P. Barrade and A. Rufer. Current capability and power density of supercapacitors: Considerations on energy efficiency. In *European Conference on Power Electronics and Applications (EPE), 2003*, Toulouse, France, September 2-4 2003.

[4] S. K. Baruah. Dynamic- and static-priority scheduling of recurring real-time tasks. *Real-Time Systems*, 24(1):93–128, 2003.

[5] 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.

[6] 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.

[7] R. Kötz and M. Carlen. Principles and applications of electrochemical capacitors. In *Electrochimica Acta 45*, pages 2483–2498. Elsevier Science Ltd., 2000.

[8] L. Lin, N. B. Shroff, and R.Srikant. Asymptotically optimal power-aware routing for multihop wireless networks with renewable energy sources. In *Proceedings of IEEE INFOCOM 2005*, pages 1262 – 1272, Miami, USA, March 13-17 2005.

[9] Maxwell technologies, Inc. Boostcap ultracapacitor - pc series data sheet. http://www.maxwell.com/pdf/uc/datasheets/PC_Series.pdf, June, 2006.

[10] Moteiv Corporation. Tmote sky - ultra low power ieee 802.15.4 compliant wireless sensor module, datasheet. http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf, June, 2006.

[11] C. Moser, D. Brunelli, L. Thiele, and L. Benini. Lazy scheduling for energy-harvesting sensor nodes. In *Fifth Working Conference on Distributed and Parallel Embedded Systems, DIPES 2006*, pages 125–134, Braga, Portugal, October 11-13 2006.

[12] 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 (ECRTS 06)*, pages 261–270, Dresden, Germany, July 2006.

[13] B. Prabhakar, E. Uysal-Biyikoglu, and A. E. Gamal. Energy-efficient transmission over a wireless link via lazy packet scheduling. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, pages 386–394, Anchorage, AK, USA, April 2001.

[14] 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.

[15] 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.

[16] J. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou. Real-time communication and coordination in embedded sensor networks. In *Proceedings of the IEEE*, vol.91, no.7pp. 1002- 1022, July 2003.

[17] T. Voigt, H. Ritter, J. Schiller, A. Dunkels, and J. Alonso. Solar-aware clustering in wireless sensor networks. In *Proceedings of the Ninth IEEE Symposium on Computers and Communications*, June 2004.

[18] E. Wandeler, A. Maxiaguine, and L. Thiele. Quantitative characterization of event streams in analysis of hard real-time applications. *Real-Time Systems, Springer Science+Business Media B.V.*, 9(2):205–225, 2005.

Clemens Moser is a Ph.D. student at the Computer Engineering and Networks Laboratory of the Swiss Federal Institute of Technology, Zurich. His research interests include design, analysis and optimization of energy harvesting sensor networks. He studied electrical engineering and information technology at the Technical University of Munich, where he received the B.Sc. and Dipl.Ing. degree in 2003 and 2004, respectively. For his diploma thesis in 2004, he joined DoCoMo Euro-Labs to work on topology aspects of wireless multihop networks.

Davide Brunelli is currently pursuing the PhD degree at the University of Bologna, Italy. He received the electrical engineering degree from the University of Bologna, in 2002. His research interests concern design and analysis of wireless sensor networks for ambient intelligence and ubiquitous computing, with particular emphasis on energy scavenging techniques.

Lothar Thiele joined ETH Zurich, Switzerland, as a full Professor of Computer Engineering, in 1994. He is leading the Computer Engineering and Networks Laboratory of ETH Zurich. His research interests include models, methods and software tools for the design of embedded systems, embedded software and bioinspired optimization techniques. In 1986, he received the "Dissertation Award" of the Technical University of Munich, in 1987, the "Outstanding Young Author Award" of the IEEE Circuits and Systems Society, in 1988, the Browder J. Thompson Memorial Award of the IEEE, and in 2000-2001, the "IBM Faculty Partnership Award". In 2004, he joined the German Academy of Natural Scientists Leopoldina. In 2005, he was the recipient of the Honorary Blaise Pascal Chair of University Leiden, The Netherlands.

Luca Benini is an Full Professor at the University of Bologna. He also holds a visiting faculty position at the Ecole Polytechnique Federale de Lausanne (EPFL). Dr. Benini's research interests are in the design of systems for ambient intelligence, from multi-processor systems-on-chip/networks on chip to energy-efficient smart sensors and sensor networks. He has published more than 350 papers in peer-reviewed international journals and conferences, three books, several book chapters and two patents. He has been program chair and vice-chair of Design Automation and Test in Europe Conference. He is Associate Editor of the IEEE Transactions on Computer-Aided Design of Circuits and Systems and of the ACM Journal on Emerging Technologies in Computing Systems. He is a Fellow of the IEEE.