Optimal Service Level Allocation in Environmentally Powered Embedded Systems

Clemens Moser, Jian-Jia Chen, and Lothar Thiele
Computer Engineering and Networks Laboratory (TIK)
Swiss Federal Institute of Technology (ETH), Zurich, Switzerland
Email: cmoser@tik.ee.ethz.ch, jchen@tik.ee.ethz.ch, thiele@tik.ee.ethz.ch

ABSTRACT

Energy management is a critical concern in the design of embedded systems to prolong the lifetime or to maximize the performance under energy constraints. In particular, the emerging embedded systems with renewable energy sources rise new problems and trigger the revision of conventional energy management. If, e.g., the size of a solar cell limits the available power/energy of an electronic device, decisions like *when* to provide *which* service have to be made in order to satisfy the needs of the user as well as possible. In this paper, we explore how to maximize the system reward of diverse applications for an energy harvesting system. By utilizing the notion of rewards to express the different priorities of services, we answer the fundamental question of how to optimize the use of energy provided by a scarce and time-varying environmental source. For this purpose, we provide algorithms to optimally adjust service parameters dynamically. Our work is supported by simulation results which are based on long-term measurements of the power generated by real solar cells. Furthermore, we demonstrate how to dimension the embedded system, e.g., the battery capacity and elaborate on implementation details which are of practical importance.

Categories and Subject Descriptors: D.4.1 Software OPERATING SYSTEMS – Scheduling

General Terms: Algorithms, Design

Keywords: Energy harvesting systems, embedded systems, reward maximization, solar cells.

1. INTRODUCTION

Energy management has become vitally important to battery-operated embedded systems. In the past years, significant research effort has been dedicated to achieve efficient energy utilization. This holds in particular for systems adopting, e.g., dynamic voltage scaling [4, 25] or dynamic power management [6, 9]. Such methods tempt to increase the battery lifetime and maximize energy savings while still maintaining an acceptable service level for the user. To minimize energy consumption under some performance constraints, applications are associated with soft or hard real-time constraints. In the past decade, energy-efficient scheduling for real-time tasks has been an active topic in both academics and industry. Chen and Kuo [4], recently, provide a comprehensive survey for energy-efficient scheduling for real-time systems.

Many embedded systems like, for instance, wireless sensor nodes or portable multimedia devices require energy autonomy. Unfortunately, improvements in battery capacity have not kept pace with the dramatic increasing power consumption in micro-electronics technology. In order to overcome limitations of batteries, energy harvesting techniques such as conversion of, e.g., photovoltaic or vibrational energy have opened up an exciting field of research [20]. Energy harvesting is exceptionally interesting in wireless sensor networks. Here, the energy generated by small solar panels suffices to execute most common data gathering applications. Consequently, numerous researchers have started to design energy harvesting circuits to efficiently convert and store solar energy [8, 10, 18, 24].

Instead of energy consumption minimization under performance constraints, in energy harvesting devices, the energy consumption of the system should depend on the energy harvested from the environment to maximize the performance. There exist two major constraints which arise due to the burstiness of common energy sources: (1) The harvested energy is temporarily low and the service must be lowered or suspended. (2) During bursts, the harvested energy exceeds the battery capacity. Driven by solar energy, the main challenge for such a system is to optimize its performance while respecting the time-varying amount of energy. How to design and play out a given battery capacity becomes a key concern.

Therefore, energy management must consider performance maximization for different environment states. Exploiting solar energy for performance optimization has been studied recently. Kansal et al. [11] explore how to maximize the utilization of solar energy by minimizing the round-trip losses of the battery, while Moser et al. [14] develop lazy-scheduling to avoid deadline violation in energy-harvesting systems. Liu et al. [13] extend the results in [14] to reduce the rate of deadline misses by taking into account processors with the capability of dynamic voltage scaling.

Distinct from the above performance metrics, we explore systems in which the quality, in terms of *reward*, of the provided service depends on the amount of computation. In general, the reward associated with a service increases with the amount of computation required to provide the service. Prominent models used in literature are the imprecise computation model [12] and the increasing reward with increasing service (IRIS) model [7, 23]. For numerous practical applications, such as image and speech processing, time-dependent planning, and multimedia applications, the reward function is modeled as a concave function of the amount of computation [3].

Reward maximization under energy constraints has been studied for real-time systems such as [2, 5, 21, 22]. Specifically, Rusu et al. [22] explore the reward maximization for a set of real-time tasks with multiple versions for execution by applying energy harvesting devices. The execution frames are divided into two types, namely recharging frames and discharging frames. These two types of frames are then executed by applying their static schedules individually. If the scheduler observes more energy residual in the battery, three different approaches are proposed to distribute the additional energy for getting more system reward. Moreover, recent study for the maximization of the minimum rate of applications [16] can be treated as a special case of reward maximization.

This paper explores how to exploit energy harvested from the environment to maximize the system performance for applications with different concave reward functions. At this, our application model allows the specification of various dependencies between the provides services. The problem studied in this paper is similar to that in [22], but we focus on a more fundamental problem to maximize the system reward globally, in which the energy consumption in all recharging frames and discharging frames might be different to achieve the global optimization. Furthermore, the application model studied in this paper extends the one in [15] to several tasks which can run in parallel and may have various dependencies.

Our contribution in this paper is as follows:

- We develop algorithms to derive optimal solutions which maximize the overall reward.
- For a specified environment, we demonstrate how to dimension the embedded system, e.g., the battery capacity and elaborate on implementation details.
- Simulations illuminate the merits of our algorithms by applying real data recorded for photovoltaic cells as the harvested energy.

2. SYSTEM MODEL AND PROBLEM STATEMENT

This paper explores how to exploit energy harvested from the environment to maximize the system performance for applications with different reward functions. The harvested energy varies over time. For example, the energy harvested from a solar panel at a sunny noon is much more than the energy harvested at dawn. As a result, the system should adjust the quality of the associated applications to reflect the dynamic behavior for performance maximization instead of energy consumption minimization. In general, the more energy an application consumes, the more reward the system gains. However, an application might over-consume energy so that the possibility to gain more reward in the future is sacrificed.

This section will present the system model studied in this paper, including energy harvesting model for the energy source, the energy storage model, and the service and application models. The overview of the studied system is depicted in Figure 1, where the energy harvested from the energy source is stored in the energy storage, and the scheduler makes decisions for consuming the energy based on the information provided by the prediction unit and the available energy in the energy storage. At the end of this section, the problem definition will be presented.

2.1 Energy Harvesting Model

This paper explores embedded systems equipped with an energy harvesting device, which can generate energy depending on the environmental situation, such as a solar panel. It is difficult to evaluate the exact amount of harvested energy in the future as the surrounding environment might change. However, in many cases, the surroundings do not change very frequently. As a result, a predictor can indeed help and is quite precise to predict the amount of harvested energy [19]. This paper assumes that the energy harvesting device has a prediction unit to estimate the amounts of energy

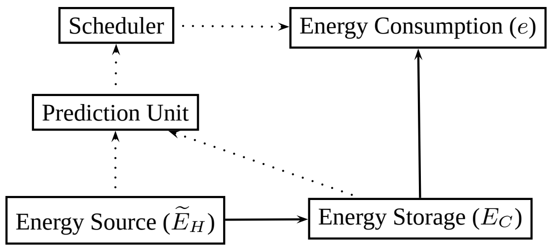


Figure 1: The system model.

harvested in each of the next K prediction intervals in the future. For brevity, a prediction interval is denoted as a *frame*, while K is referred to *number of frames of the prediction horizon*. Each frame is assumed to have the same length and is the basic time unit for scheduling.

Suppose that a prediction unit predicts the energy harvested from the environment for the next 50 minutes, in which the length of a frame is 5 minutes. The prediction horizon for this example is 50 minutes, while K is 10. We denote $\widetilde{E}_H(k)$ the accumulated energy harvested that is predicted for the k-th frame. For the rest of this paper, we assume a perfect energy predictor. In the case of solar cells, diverse estimation algorithms or even external informations from the weather forecast can be applied. Comprehensive discussions about how to choose a suitable energy prediction algorithm or how to cope with prediction mistakes can be found in [17], and, hence, the details are not included here.

2.2 Energy Storage Model

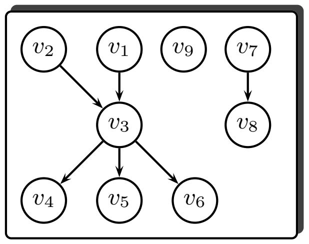
The energy storage, e.g., a supercapacitor or a battery then stores the energy harvested from the environment. As energy storage is not a perfect process, there might be some loss of energy. The *efficiency factor* defined as the actually stored energy divided by the harvested energy can be used to model the storage process. Usually, the efficiency factor, denoted by $\eta(\widetilde{E}_H)$, is a function of the harvested energy, and, by definition, is between 0 and 1. As a result, only $\eta(\widetilde{E}_H) \times \widetilde{E}_H(k)$ amount of the harvested energy will be stored to the energy storage in the k-th frame.

The amount of the harvested energy that will be stored to the energy storage in the k-th frame is denoted by $E_H(k)$, where $E_H(k)$ is $\eta(\widetilde{E}_H) \times \widetilde{E}_H(k)$. The harvested energy in the k-th frame, for the rest of this paper in sake of simplicity of presentation, is implicitly denoted by $E_H(k)$.

The energy storage is constrained by the maximum capacity $E_{\max}$ of the energy in the storage. If the energy storage is full, the additional harvested energy dissipates. Formally, suppose that $E_C(k)$ is the energy in the energy storage at the end of the k-th frame. After servicing the applications in the k-th frame with energy consumption e_k, the energy in the energy storage is $\min\{E_{\max}, E_C(k-1) + E_H(k) - e_k\}$. In other words, if $E_C(k-1) + E_H(k) - e_k$ is larger than $E_{\max}$, the system dissipates $E_C(k-1) + E_H(k) - e_k - E_{\max}$ amount of energy, which is a waste.

2.3 Application and Service Model

This paper explores how to achieve performance maximization for a variety of applications running on an embedded system, e.g., a sensor node . Two types of applications are studied in this paper: one is the *computation-based* model and the other is the *rate-based* model. For the computation-based model, the quality of



$$\rho(v_1) + \rho(v_2) = \rho(v_3)$$
$$\rho(v_4) + \rho(v_5) + \rho(v_6) = \rho(v_3)$$
$$\rho(v_7) = \rho(v_8)$$

Figure 2: An example for rate graphs.

the service provided by an application can be adjusted by assigning different computation workload, e.g., the imprecise computation model [12] and the increasing reward with increasing service (IRIS) model [7,23]. In general, for the computation-based model, the more computation a service requires, the more reward an application contributes. Moreover, more energy consumption is required to achieve higher reward of a service. For the rate-based model, the reward of a service depends on the rate to activate the application, while the computation demand and the energy consumption are fixed for one activation. For example, an application which observes an environmental phenomenon may be more precise when the sensing rate is higher, where each sensing activity takes the same time and consumes the same energy.

Moreover, for the rate-based model, several applications might be dependent in the settings of their rates. For instance, a sensor node may sense some data, and then decide whether to compress and store the data in some local memory or to directly transmit the data via the radio. Clearly, the sum of stored and transmitted data has to equal the amount of originally sensed data. In this paper, we explore rate-conserving applications that can be described by several rate-based graphs (RBGs) defined as follows: A rate-base graph is a directed tree, denoted by $G = (\mathcal{V}, \mathcal{E})$, in which a vertex in $\mathcal{V}$ is an application and a directed edge $(u \rightarrow v)$ in $\mathcal{E}$ is the relationship between the rates of two applications. Suppose that $\rho(v)$ is the rate of an application in vertex v in $\mathcal{V}$. If $(u \rightarrow v)$ in $\mathcal{E}$ is the only incoming edge to application v and is the only outgoing edge from application u, the rates of applications u and v are the same. If there are more than one incoming edge to an application v, the rate of the application v must be the sum of the rates of the applications that have directed outgoing edges to vertex v, i.e., $\rho(v) = \sum_{(u\rightarrow v)\in\mathcal{E}} \rho(u)$. Moreover, if there are more than one outgoing edge from an application v, the rate of the application v must be the sum of the rates of the applications that have directed incoming edges from vertex v, i.e., $\rho(v) = \sum_{(v\rightarrow u)\in\mathcal{E}} \rho(u)$. Figure 2 depicts an example for 9 applications and their rate relationships.[1]

For each frame, the scheduler has to determine how to provide the services of different applications. The quality of the provided service of an application is evaluated in each frame. For the rate-based model, once we assign an energy consumption to an application, the corresponding rate of the application is known. Similarly, for the computation-based model, the energy consumption for a given workload of provided service is assumed to be a convex function (when dynamic voltage scaling is adopted [5,25]) or a linear function (when the power consumption is a constant).

The reward function of an application is assumed to be a concave and increasing function of the amount of computation in the computation-based model or the rate in the rate-based model. Examples are image and speech processing, time-dependent planning, and multimedia applications. Therefore, the reward function of an application is a concave and increasing function of the energy consumption. For the rest of this paper, we will only discuss the amount of energy consumption of an application in each frame, while the required computation time or rate to complete its corresponding service in a frame with the specified energy consumption can be derived by simple calculation.

Specifically, let $r_i(\epsilon)$ denote the reward for executing the service in a frame with energy consumption ϵ for application v_i, where

- $r_i(\epsilon)$ is continuous and differentiable.
- $r_i(\epsilon)$ is monotonically increasing in ϵ.
- $r_i(\epsilon)$ is concave in ϵ, i.e.,

  $$\alpha\times r_i(\epsilon_1)+(1-\alpha)\times r_i(\epsilon_2) \leq r_i\left(\alpha \times \epsilon_1 + (1-\alpha) \times \epsilon_2\right),$$

  for any $\epsilon_1\epsilon_2 \geq 0$, $\epsilon_1 \neq \epsilon_2$, and $1 > \alpha > 0$.

2.4 Problem Definition

We are given a predictor for K frames at time 0. The energy in the energy storage at time 0 is specified as $E_C(0)$ and the energy harvested in the k-th frame is $E_H(k)$. The k-th frame starts from time $k-1$ to time k. A set $\mathcal{V}$ of N applications is going to be executed in each frame, in which each application v_i has its reward function $r_i(\epsilon_i)$ of the energy consumption. We assume that the reward functions are concave and increasing.

The objective is to determine an *inter-frame energy assignment*

$$\vec{e} = (e_1, e_2, \ldots, e_K)$$

of energy consumption for these K frames and *intra-frame energy assignments*

$$\vec{\epsilon}_k = (\epsilon_{k,1}, \epsilon_{k,2}, \ldots, \epsilon_{k,N})$$

for the N applications within the k-th frame such that the reward is maximized without violating the required energy constraints. Obviously, if one knows the intra-frame energy assignment $\vec{\epsilon}_k$ one obtains the inter-frame energy assignment

$$e_k = \sum_{n=1}^{N} \epsilon_{k,n}$$

by simple summation for every frame k. The energy consumptions of some applications in a frame might be related. For example, suppose the energy consumption of application v_i is σ_i when its rate is 1. The relationship $\rho(v_1) + \rho(v_2) = \rho(v_3)$ in Figure 2 leads to $\frac{\epsilon_1}{\sigma_1} + \frac{\epsilon_2}{\sigma_2} = \frac{\epsilon_3}{\sigma_3}$, where ϵ_i is the energy consumption of application v_i. We write these constraints from the rate-based graphs (RBG) in the form $c_m(\vec{\epsilon}_k) = 0$, where $m = 1, \ldots, M$. The constraints $c_m(\vec{\epsilon}_k)$ can be arbitrary linear combinations of the assignment $\vec{\epsilon}_k$ which conserve the rates of the application.

Let $E_C(k, \vec{e})$ be the energy in the energy storage at time k by applying the assignment of energy consumption $\vec{e}$. After completing the last frame, we would like to reserve some amount E_ℓ of energy in the energy storage for future use, and, hence, a feasible assignment $\vec{e}$ must satisfy $E_C(K, \vec{e}) \geq E_\ell$. We denote the studied problem as the *general reward maximization on energy harvesting* problem. Without loss of generality, we only explore the case that $E_C(0) - E_\ell + \sum_{i=1}^{K} E_H(i) \geq 0$ in this paper since there is no feasible solution for the other case.

We formally define the feasibility of an assignment for the general reward maximization on energy harvesting problem as follows:

[1]The approaches in this paper also work for rate-based graphs with scaling factors on the edges. For example, it could be $0.5\rho(v_1) + 0.3\rho(v_2) = \rho(v_3)$.

DEFINITION 1. *[Feasible Assignment] An energy vector* $\vec{e} = (e_1, \ldots, e_K)$ *along with* $\vec{\epsilon}_k = (\epsilon_{k,1}, \epsilon_{k,2}, \ldots, \epsilon_{k,N})$ *for all these* K *frames is feasible if*

(a) $E_C(k, \vec{e}) = \min\{E_{\max}, E_C(k-1, \vec{e}) + E_H(k) - e_k\}$, *where* $E_C(0, \vec{e})$ *is* $E_C(0)$,

(b) $E_C(k, \vec{e}) \geq 0, \forall 1 \leq k < K$,

(c) $E_C(K, \vec{e}) \geq E_\ell$,

(d) $\sum_{n=1}^{N} \epsilon_{k,n} = e_k$, *and*

(e) $c_m(\vec{\epsilon}_k) = 0, \forall 1 \leq m \leq M$.

An assignment is said *optimal* for the general reward maximization on energy harvesting problem if its reward is the maximum among all feasible assignments. We say there exists an *energy underflow* for an assignment $\vec{e}$ if there exists $E_C(k, \vec{e}) < 0$ for some $1 \leq k \leq K - 1$ or $E_C(K, \vec{e}) < E_\ell$. On the other hand, an assignment $\vec{e}$ is said with *energy overflow* if there exists some k with $E_C(k-1, \vec{e}) + E_H(k) - e_k > E_{\max}$.

3. OPTIMAL SERVICE ALLOCATION ALGORITHMS

In this section, we will show how to optimally allocate services subject to the available energy given by the environment and the energy storage. Thereby, we exploit the nature of the general reward maximization on energy harvesting problem and split it into two subproblems, namely the inter-frame problem and the intra-frame problem, which can be solved independently. Formally, the inter-frame problem focuses on inter-frame energy assignments $\vec{e}$, while the intra-frame problem copes with intra-frame energy assignment $\vec{\epsilon}_k$ for every k-th frame. In Section 3.1, for completeness, we present our recently proposed algorithm in [15] for computing inter-frame energy assignments $\vec{e}$. The energy consumption of a frame in assignment $\vec{e}$ can be treated as the energy budget in the frame. In Section 3.2, an algorithm is presented to determine the intra-frame energy assignment $\vec{\epsilon}_k$ for assigning energy budget e_k to the N applications within each k-th frame.

3.1 Inter-Frame Energy Assignment

We will first present how to derive the optimal inter-frame energy assignment $\vec{e}$. The proposed algorithm is presented in Algorithm 1, denoted by Algorithm Inter-Frame Service Allocation. The algorithm can be proved to derive optimal solutions for systems with only one application with any concave reward function. Moreover, if there is more than one application, the next subsection will show that these applications can be treated as one application with a *joint reward function*, which is concave, too. Therefore, for the sake of simplicity, we will consider only one application with a concave reward function in this subsection.

To explain how Algorithm 1 works, we will use the following simple example for demonstration: Suppose that we are given a prediction of the future harvested energy $E_H(1) = 4, E_H(2) = 5, E_H(3) = 1, E_H(4) = 0, E_H(5) = 5, E_H(6) = 5$ for a time horizon of $K = 6$, as depicted in Figure 3(a), where both the initial energy in the battery and the required energy in the battery after K frames are 2, i.e., $E_C(0) = E_\ell = 2$.

If the maximum capacity $E_{\max}$ of the energy storage is large enough (e.g., $E_{\max} = \infty$), we do not have to worry about the energy overflow issues. In such cases, to optimally exploit the concavity of the reward function and maximize the sum of rewards, one would like to consume the constant, average amount of energy

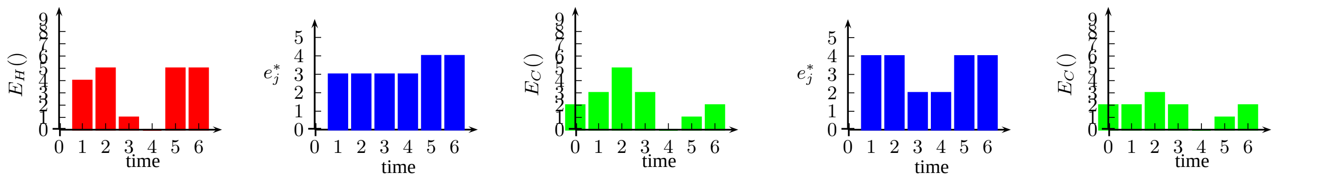
Algorithm 1 (Inter-Frame Service Allocation)

```
Input: $K$, $E_H(k)$ for $k = 1, 2, \ldots, K$, $E_C(0)$, $E_\ell$, $E_{\max}$;
Output: a feasible assignment $\vec{e}^*$ of energy consumption for the
    $K$ frames;
 1: $k \Leftarrow 0$, $n \Leftarrow 1$, $k_0 \Leftarrow 0$;
 2: while $k < K$ do
 3:     let $\tilde{e}_j$ be $\frac{E_C(k)+\sum_{i=k+1}^{j} E_H(i)}{j-k}$, $\forall j = k+1, k+2, \ldots, K-1$;
 4:     let $\tilde{e}_K$ be $\frac{E_C(k)+\sum_{i=k+1}^{K} E_H(i)-E_\ell}{K-k}$;
 5:     let $\hat{e}_j$ be $\frac{E_C(k)+\sum_{i=k+1}^{j} E_H(i)-E_{\max}}{j-k}$, $\forall j = k+1, k+2, .., K$;
 6:     $K' \Leftarrow K$;
 7:     while true do
 8:         $k^* \Leftarrow \arg\min_{k<\hat{k}\leq K'}\{\tilde{e}_{\hat{k}}\}$;
 9:         $k' \Leftarrow \arg\max_{k<\hat{k}\leq k^*}\{\hat{e}_{\hat{k}}\}$;
10:         $k^\dagger \Leftarrow \arg\min_{k<\hat{k}\leq k'}\{\tilde{e}_{\hat{k}}\}$;
11:         if $\tilde{e}_{k^*} \geq \hat{e}_{k'}$ then
12:             $k_n \Leftarrow k^*$, $e_s \Leftarrow \tilde{e}_{k^*}$;
13:             break;
14:         else if $\tilde{e}_{k^\dagger} \geq \hat{e}_{k'}$ then
15:             $k_n \Leftarrow k'$, $e_s \Leftarrow \hat{e}_{k'}$;
16:             break;
17:         else
18:             $K' \leftarrow k^\dagger$;
19:     $e_j^* \Leftarrow e_s, \forall k+1 \leq j \leq k_n$;
20:     $E_C(k_n) \Leftarrow E_C(k) + \sum_{i=k+1}^{k_n}(E_H(i) - e_i^*)$;
21:     $k \Leftarrow k_n, n \Leftarrow n+1$;
22: return $\vec{e}^*$ as the solution;
```

$\frac{4+6+1+5+5}{6} = \frac{10}{3}$ for all the six frames. However, for the example at hand, this assignment is not feasible according to Definition 1, since it results in an energy underflow $E_C(4) = -\frac{4}{3}$. The key idea to obtain an optimal assignment is to equate energies of an assignment as much as possible subject to feasibility constraints. As shown in Algorithm 1 in Step 3, the *feasible* average energies $\tilde{e}_j$ are calculated for the remaining frames $k+1, k+2, \ldots, K-1$. For the last frame K, one has to account for the final energy E_ℓ in Step 4. In our example, the average energies $\tilde{e}_j$ are $\{6, 5\frac{1}{2}, 4, 3, 3\frac{2}{5}, 3\frac{1}{3}\}$ in the first iteration of the outer while-loop. In Step 8, the index $k^* = 4$ of the minimum average energy $\tilde{e}_4 = 3$ is determined. For the case that $E_{\max}$ is large enough, the if-statement in Step 11 is always true. As a consequence, energy consumption $e_j^* = 3$ is assigned to the first four frames $j = 1, 2, 3, 4$ in Step 19. In the next iteration, the average energies $\tilde{e}_j$ are $\{5, 4\}$, and hence, we set the energy consumption $e_j^* = 4$ for the remaining frames $j = 5, 6$. The inter-frame energy assignment $\vec{e}^*$ and the stored energy $E_C()$ for $E_{\max} = \infty$ are displayed in Figures 3(b) and 3(c), respectively.

Next, we will limit the maximum amount of storage energy to $E_{\max} = 3$ to show the effects of energy overflow. Of course, energy overflows can always be avoided by increasing the energy consumption when needed. For the example in Figure 3(b) and 3(c), instead of loosing the 2 energy units at time 2, one could just increase the overall reward by increasing the energy $e_2^* = 3$ to $e_2^* = 5$. This strategy, however, would not produce an optimal assignment.

Instead, Step 5 in Algorithm 1 calculates *all* average rates $\hat{e}_j$ which are feasible with respect to an energy overflow. The average rates $\hat{e}_j$ are $\{3, 4, 3, 2\frac{1}{4}, 2\frac{4}{5}, 3\frac{1}{6}\}$. To obtain the "smoothest" increase of energy consumption during the first 4 frames, the max-



(a) $E_H()$ (b) $\vec{e}^*$ for $E_{\max} = \infty$ (c) $E_C()$ for $E_{\max} = \infty$ (d) $\vec{e}^*$ for $E_{\max} = 3$ (e) $E_C()$ for $E_{\max} = 3$

Figure 3: An example for $E_H()$ and the solutions derived by Algorithm Inter-Frame Service Allocation when K is 6 for $E_{\max} = \infty$ and $E_{\max} = 3$.

imum of the energies $\hat{e}_j$, namely $\hat{e}_2 = 4$, is chosen in Step 9. This time, the optimal energy consumption $\hat{e}_2 = 4$ necessary to avoid an energy overflow is greater then the optimal energy consumption $\tilde{e}_4 = 3$ to prevent from energy underflow. Thus, the first two energies e_j^* are set to 4 for $j = 1, 2$. After this iteration, an energy underflow is avoided at time 4 and the energies e_3^* and e_4^* are set to 2. Finally, the last iteration is the same as for $E_{\max} = \infty$ and we get $e_5^* = e_6^* = 4$. The assignment $\vec{e}^*$ and the stored energies $E_C()$ for $E_{\max} = 3$ are displayed in Figures 3(d) and 3(e), respectively.

The second if-statement in Step 14 turns out to be false for our example, since there is no energy underflow possible before the energy overflow at time 2. In general, this situation may nevertheless happen. Then the horizon K' is reduced in Step 18 and a subproblem is solved. Similarly, there may be an arbitrary sequence of energy underflows and overflows, which are handled in the order of their appearance. If an energy overflow is not avoided, the overflowing energy is just wasted and cannot be used later on.

The time complexity of the algorithm is $O(K^2)$. It is easy to verify that the derived solution $\vec{e}^*$ is feasible according to the first three conditions in Definition 1 and consumes all available energy $E_C(0) - E_\ell + \sum_{i=1}^{K} E_H(i)$, both being necessary conditions for an optimal assignment. Please note that Algorithm 1 is equivalent to Algorithm Recursive-Decomposition (RD) in [15], and is included for completeness of this paper.

THEOREM 1. *Algorithm Inter-Frame Service Allocation derives optimal assignments $\vec{e}^*$ for the general reward maximization on energy harvesting problem for an application having a concave reward function.*

PROOF. A proof for the optimality is given in [15]. □

Note that Algorithm 1 does not have to know the exact reward function. The only requirement for guaranteeing the optimality is that the (joint) reward function has to be concave.

3.2 Intra-Frame Energy Assignment

In contrast to the inter-frame energy allocation, the intra-frame energy allocation requires the precise knowledge of the reward functions $r_n()$ to distribute the energy among all running applications $n = 1, \ldots, N$. Given the energy e_k as the total energy which can be consumed in the k-th frame, we can formally write the intra-frame energy assignment problem as follows:

$$\text{maximize } \sum_{n=1}^{N} r_n(\epsilon_{k,n}) \text{ subject to:} \tag{1}$$

$$\begin{aligned} &\textstyle\sum_{n=1}^{N} \epsilon_{k,n} = e_k && \\ &c_m(\vec{\epsilon}_k) = 0 && \forall 1 \le m \le M \\ &\epsilon_{k,n} \ge 0 && \forall 1 \le n \le N \end{aligned}$$

In other words, we want to maximize the sum of rewards subject to the energy budget e_k and the constraints $c_m(\vec{\epsilon}_k)$ from possible rate-based graphs. We denote the optimal objective value of the above optimization problem in Equation (1) as function $R(e_k) = \max\{\sum_{n=1}^{N} r_n(\epsilon_{k,n})\}$. The following theorem establishes the concavity of $R(e_k)$ over the energy e_k.

THEOREM 2. *If r(x,y) is concave in (x,y) and C is a convex nonempty set, then $R(x) = \max_{y \in C} r(x, y)$ is concave in x, provided $R(x) > -\infty$ for some x.*

PROOF. A proof can be found in standard text books on convex optimization and is omitted here. □

As a result, based on the concavity of $R(e_k)$ in Theorem 2 and the optimality of Theorem 1, we can divide the overall optimization problem into two subproblems, says inter-frame and intra-frame energy assignments. Therefore, what we have to do is to determine the intra-frame energy assignment. By applying the Lagrange Multiplier method, we can solve the problem in Equation (1) optimally, as shown in Algorithm 2.

Algorithm 2 (Intra-Frame Service Allocation)

```
Input: $e_k$, $r_n()$ for $n = 1, \ldots, N$, $c_m(\vec{\epsilon}_k)$ for $m = 1, \ldots, M$;
Output: a feasible assignment $\vec{\epsilon}_k$ of energy consumption for the
    $N$ applications in frame $k$;
1: let $L(\epsilon_{k,n}, \ldots, \epsilon_{k,N}, \lambda_0, \lambda_1, \ldots, \lambda_M)$ be
   $\sum_{n=1}^{N} r_n(\epsilon_{k,n}) - \lambda_0 \cdot \left(\sum_{n=1}^{N} \epsilon_{k,n} - e_k\right) - \sum_{m=1}^{M} \lambda_m \cdot c_m(\vec{\epsilon}_k)$ ;
2: solve $\frac{\partial L()}{\partial \epsilon_{k,1}} = \cdots = \frac{\partial L()}{\partial \epsilon_{k,N}} = 0, \forall 1 \le n \le N$ under the
   energy constraints $\sum_{n=1}^{N} \epsilon_{k,n} = e_k$ and $c_m(\vec{\epsilon}_k) = 0$ ;
3: return $\vec{\epsilon}_k$ as the solution;
```

Certainly, solving these equations in Step 2 is a critical operation in Algorithm 2. In Section 5, techniques for a practical implementation of the algorithm will be discussed.

Due to the concavity of the reward functions, the solution found by using the Lagrange Multipliers $\lambda_0, \lambda_1, \ldots, \lambda_M$ is guaranteed to be the *global* optimum for the intra-frame energy assignment problem. As a result, it is not difficult to see the optimality of the derived solutions, which is a direct consequence of Theorems 1 and 2.

COROLLARY 1. *The energy assignment obtained by successive application of Algorithm 1 and 2 constitutes an optimal assignment for the general reward maximization on energy harvesting problem.*

To demonstrate how Algorithm 2 works, we use the following example for illustration. Suppose that an embedded system has to

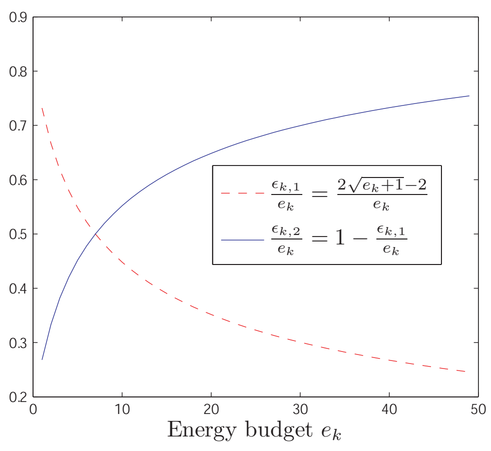


Figure 4: Intra-frame energy assignment for the example with $r_1(\epsilon_{k,1}) = \ln \epsilon_{k,1}$ and $r_2(\epsilon_{k,2}) = \sqrt{\epsilon_{k,2}}$.

execute two independent applications v_1 and v_2 with the energy budget e_k in a certain frame. The respective reward functions are given by $r_1(\epsilon_{k,1}) = \ln \epsilon_{k,1}$ and $r_2(\epsilon_{k,2}) = \sqrt{\epsilon_{k,2}}$. By setting the partial derivatives of the Lagrangian $L()$ equal to 0, we get $\frac{\partial r(\epsilon_{k,1})}{\partial \epsilon_{k,1}} = \frac{\partial r(\epsilon_{k,2})}{\partial \epsilon_{k,2}} = \lambda_0$ under the energy constraint $\epsilon_{k,1} + \epsilon_{k,2} = e_k$. Thus, we have to solve

$$\frac{1}{\epsilon_{k,1}} = \frac{1}{2\sqrt{\epsilon_{k,2}}} = \lambda_0$$

$$\epsilon_{k,1} + \epsilon_{k,2} = e_k.$$

In this case, we know that $\epsilon_{k,2} = \frac{\epsilon_{k,1}^2}{4}$. Therefore, $\frac{\epsilon_{k,1}^2}{4} + \epsilon_{k,1} - e_k = 0$, in which $\epsilon_{k,1} = -2 + 2\sqrt{1 + e_k}$. If we continue on the example from the previous subsection in Figures 3(d) and 3(e), the energy consumptions in the first frame are, e.g., $\epsilon_{1,1} = -2 + 2\sqrt{5}$ and $\epsilon_{1,2} = 6 - 2\sqrt{5}$. In Figure 4, the normalized functions $\frac{\epsilon_{k,1}}{e_k} = \frac{2\sqrt{e_k+1}-2}{e_k}$ and $\frac{\epsilon_{k,2}}{e_k} = 1 - \frac{\epsilon_{k,1}}{e_k}$ are displayed over the energy budget e_k. For values $e_k < 8$, application v_1 receives the larger percentage of e_k, whereas for $e_k > 8$, it is advantageous to assign more energy to application v_2. At $e_k = 8$, the energy budget e_k has to be shared equally among both applications.

4. DESIGN CONSIDERATIONS

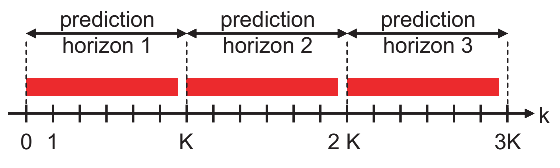
This section provides some remarks for designing embedded systems with energy harvesting devices using the techniques described in this paper. For the online usage of Algorithm 1 (Inter-Frame Service Allocation), one has two principal alternatives, as illustrated in Figure 5, *sliding horizon* or *sliding frame* operations. While the sliding horizon operation repeatedly predicts and optimizes the energy assignments for independent horizons, the sliding frame operation solves the problem for overlapping horizons for every frame. As energy prediction is assumed perfect for the study in this paper, the sliding horizon operation is adopted and will be used for presenting the simulation results in Section 5. Certainly, if there are prediction mistakes, one would preferably choose the sliding frame operation since prediction mistakes can be counteracted more efficiently. On the other hand, the sliding frame operation imposes a higher computational load for the embedded system, since Algorithm 1 is executed K times more often then in the sliding horizon operation.

To design the energy supply of an embedded system, it is important to estimate how to dimension the energy storage device. Given an initial energy $E_C(0)$, an energy source $E_H(k), 1 \leq k \leq K$ and a final energy constraint E_ℓ, we are interested in the minimum storage capacity which is needed to achieve the maximum possible reward. We denote $E_{\max,\min}$ the minimum capacity $E_{\max}$ for which the optimal reward equals the reward for an unconstrained system which $E_{\max} = +\infty$. For a single horizon, we can now determine

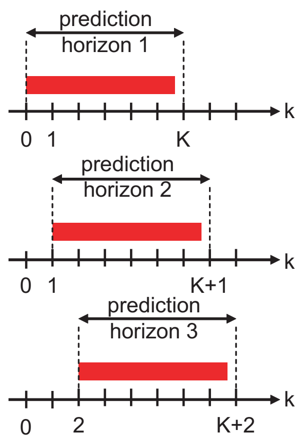
$$E_{\max,\min} = \max_{k=1,2,\ldots,K} \{E_C(0) + \sum_{j=1}^{k} (E_H(j) - e_j^*)\},$$

where $\vec{e}^*$ is an assignment computed by Algorithm 1 for specified $E_H(k)$ for $k = 1, 2, \ldots, K$, $E_C(0)$, E_ℓ and $E_{\max} = \infty$. For the sliding horizon operation, this calculation has to be repeated for representative data samples $E_H(k)$ for the energy source. This can be done by using, e.g., sufficiently large traces of solar energy which have been recorded beforehand. By choosing the maximum of all capacities $E_{\max,\min}$, it is possible to determine the minimum capacity $E_{\max}$ to optimally exploit a given environmental source.

In some systems, the services might have the requirement to consume at least some amount of energy consumption in a frame, and there might also be some constraints on the maximum energy consumption in a frame since there is no improvement in quality/reward for more energy consumption. We can revise the solution $\vec{e}^\dagger$ derived from Algorithm 1 by setting the energy consumption of the k-th frame as the maximum energy consumption if e_k^* is greater than the maximum energy consumption constraint. If there exists e_k^* which is less than the minimum energy consumption constraint, it is not difficult to see that the input does not admit an feasible assignment. Moreover, as a frame may last several hours in physical time, if the energy storage is full, there may be an energy overflow. However, this conflict can be resolved by assuming the existence of small energy buffers (capacitors or super-capacitors) or a lower capacity $E'_{\max}$.



(a) Sliding horizon operation



(b) Sliding frame operation

Figure 5: Different sliding operation for Algorithm 1 (Inter-Frame Service Allocation).

5. SIMULATION RESULTS

5.1 Simulation Environment and Setup

For the experiments, we use long-term measurements of solar light intensity recorded during 5 years from a photovoltaic plant at Mont Soleil, Switzerland [1]. The data measured in $\left[\frac{W}{m^2}\right]$ serves as harvested energy $E_H()$. Of course, to simulate a concrete system, one would have to *scale the measured power profile* with the size, number and efficiency of the actually used solar panels. The data is sampled every 5 minutes, so we have a maximum of 288 samples per day.

Since solar energy shows a similar pattern every 24 hours, multiples of a day are reasonable choices for the prediction horizon. The number of frames per day does not affect the results too much. The presented results in this section are conducted by setting 16 frames per day, which means a frame lasts for 1.5 hours. The results for different numbers of frames per day are quite similar. When investigating different values for the number of frames of the prediction horizon $K \in \{16, 32, 48, \ldots\}$, we use the shorter notation $K \in \{1d, 2d, 3d, \ldots\}$ but keep in mind that we have 16 frames per day. At the end of the prediction horizon, we want the remaining energy $E_C(K)$ to be at least equal to the initial energy $E_C(0)$, i.e., $E_\ell = E_C(0)$.

5.2 Choice of Parameters K and $E_{\max}$

A fundamental question one might ask when designing a system in a given environment is: How many days should the horizon span to obtain reasonable rewards? For this purpose, we simulate Algorithm 1 (Inter-Frame Service Allocation) for different parameters $K \in \{1d, 2d, 3d, 5d, 10d, 15d, 30d, 70d, 105d, 210d\}$. To obtain a total simulated time of 210 days for each experiment, $\{210, 105, 70, 42, 21, 14, 7, 3, 2, 1\}$ horizons are executed in the sliding horizon operation, respectively. The resulting energy assignments $\vec{e}^*$ are evaluated by applying the joint reward function $R(e_k) = \ln(0.01 + \frac{e_k}{1000})$ which assigns negative rewards (i.e., penalties) to energies $e_k < 990$, where $R(e_k)$ is the summation of reward functions $r_n(\epsilon_{k,n})$ of the N applications under energy constraint e_k. In particular, setting the services to 0 is punished with an penalty of $\ln(0.01) \approx -4.61$. For each experiment, we calculate the accumulated reward for 210 days.

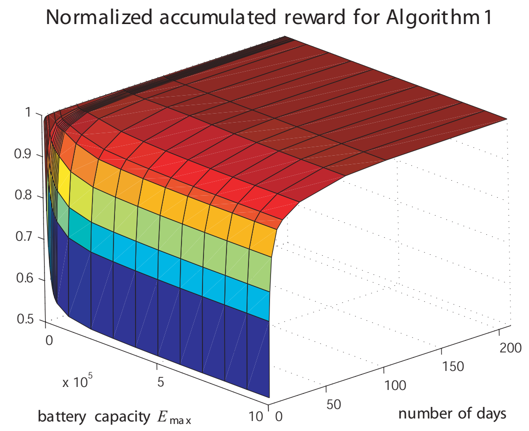


Figure 6: Convergence of the normalized accumulated reward over number of days and $E_{\max}$, $E_C(0) = E_\ell = 3000$.

For this arbitrarily chosen reward function, Figure 6 displays the accumulated rewards which are normalized by the accumulated reward obtained by the experiment with the longest horizon, namely 210 days. If one chooses a smaller $E_{\max}$, it turns out that the reward converges faster towards its longterm average with increasing d. For a horizon of $d = 15$ days, a capacity of $E_{\max} = 500000$ will result in a reward of 93,7% of the reward for the same capacity with $d = 210$ days. For smaller capacities $E_{\max} = 24000$ and 4000, the ratio increases to 97,7 % and 99.9 %, respectively. The reason for this behaviour is that Algorithm 1 is getting more and more nearsighted with smaller capacity $E_{\max}$: Due to the capacity constraint, local maxima of $\vec{e}^*$ are computed to avoid energy overflows. Hence, for small energy storage capacities $E_{\max}$, the accumulated reward can hardly be improved by increasing the prediction horizon K.

The minimum energy capacity $E_{\max}$ required to optimally exploit this energy source is $E_{\max,\min} = 759450$. Using this value for the battery, a horizon of $d = 15$ days is sufficient to achieve 93.4% of the maximum possible reward (i.e., the reward for $E_{\max} = \infty$ and $K = \infty$). For this particular reward function, however, also smaller capacities $E_{\max}$ are possible to achieve a similar reward. In summary, using our techniques, one can easily determine or trade-off suitable parameters K and $E_{\max}$ for a given environment. The solution strongly depends on the reward function $r()$.

5.3 Efficient Implementation of the Intra-Frame Energy Assignment

Concerning the intra-frame energy assignment, we already mentioned that solving the set of equations in Step 2 of Algorithm 2 may not give an explicit solution. In such situations, numerical root-finding algorithms like Newton's method can be applied to find an approximation for the energy assignment $\epsilon_{k,n} \forall 1 \le n \le N$. Newton's method is known to have a fast, quadratic convergence ratio and the precision of the result is only limited by the finite precision of the machine which is used to carry out the arithmetic. Like that, Algorithm 2 is performed offline for a large number of energy budgets e_k and approximation functions $\epsilon_{k,n} = f_n(e_k), \forall 1 \le n \le N$ are computed.

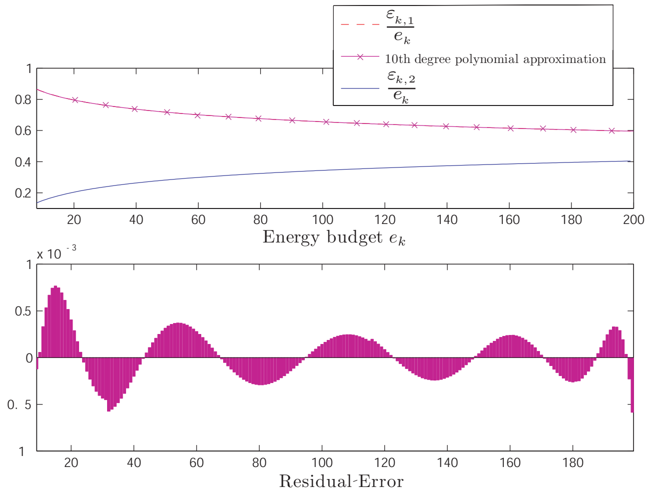


Figure 7: Approximated inter-frame energy assignment using polynomial functions.

We investigated systems consisting of several applications and discuss a representative problem next. Assume we have two independent applications v_1 and v_2 with the reward functions $r_1(\epsilon_{k,1}) = \ln \epsilon_{k,1}$ and $r_2(\epsilon_{k,2}) = \sqrt{\sqrt{\epsilon_{k,2}} + 2}$. To our best knowledge, there exists no explicit function to distribute the budget e_k in a certain frame. Using Newton's method, we approximated $\epsilon_{k,1}$ and $\epsilon_{k,2}$ for integer values $e_k \in \{1, 2, \ldots, 200\}$. Next, we determine the coefficients of, e.g., a 10th degree polynomial by least-square fitting. In Figure 7, the top plot shows the approximated values of $\frac{\epsilon_{k,1}}{e_k} = 3.7 \cdot 10^{-21} \cdot e_k^{10} - 10^{-18} \cdot e_k^{9} + 1.9 \cdot 10^{-15} \cdot e_k^{8} - 4.9 \cdot 10^{-13} \cdot e_k^{7} + 810^{-11} \cdot e_k^{6} - 8.6 \cdot 10^{-9} \cdot e_k^{5} + 6 \cdot 10^{-7} \cdot e_k^{4} - 2.8 \cdot 10^{-5} \cdot e_k^{3} + 0.00083 \cdot e_k^{2} - 0.018 \cdot e_k + 0.97$. In this plot, it is not possible to distinguish the approximated from the real values of $\frac{\epsilon_{k,1}}{e_k}$. The approximation mistake is expressed in terms of residual error and can be negligible in practice, see the bottom plot in Figure 7. The coefficients of the approximation functions have to be stored and evaluated for every task in the online case, causing neglectable overhead in terms of computation and memory requirement. Note that these approximations can be performed for miscellaneous computation-based or rate-based application models with arbitrary reward functions, as presented in Section 2.

6. CONCLUSIONS

This paper copes with embedded systems equipped with energy harvesting devices, such as solar panels. By applying prediction techniques, we explore how to globally optimize the system performance of diverse applications in terms of system rewards. The reward functions of applications are assumed to be concave and increasing with respect to the energy consumption. For a given specified prediction horizon composed of multiple frames, we develop a two-stage mechanism for energy assignments. First, we determine how much energy can be consumed in one frame without violating the energy capacity constraint. Then, an algorithm based on the Lagrange Multiplier method is applied to distribute the energy budget in one frame to the applications in the system. The effectiveness of the proposed algorithms is supported by simulations based on long-term measurements of the power generated by real solar cells. Moreover, we also demonstrate how to dimension the embedded system, e.g., the battery capacity and elaborate on implementation details for efficiency issues.

Acknowledgements

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 supported by grants from ROC National Science Council NSC-096-2917-I-564-121 and the European Network of Excellence on Embedded System Design ARTIST-Design.

References

[1] Bern university of applied sciences, engineering and information technologies, photovoltaic lab: Recordings of solar light intensity at Mont Soleil from 01/01/2002 to 31/09/2006. www.pvtest.ch, March, 2007.

[2] T. A. Alenawy and H. Aydin. On energy-constrained real-time scheduling. In *EuroMicro Conference on Real-Time Systems (ECRTS'04)*, pages 165–174, 2004.

[3] H. Aydin, R. Melhem, D. Mosse, and P. Alvarez. Optimal reward-based scheduling for periodic real-time tasks. In *Proceedings of the 20th IEEE Real-Time Systems Symposium (RTSS'99)*, pages 79–89, 1999.

[4] J.-J. Chen and C.-F. Kuo. Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms. In *RTCSA*, pages 28–38, 2007.

[5] J.-J. Chen and T.-W. Kuo. Voltage-scaling scheduling for periodic real-time tasks in reward maximization. In *the 26th IEEE Real-Time Systems Symposium (RTSS)*, pages 345–355, 2005.

[6] J.-J. Chen and T.-W. Kuo. Procrastination determination for periodic real-time tasks in leakage-aware dynamic voltage scaling systems. In *ICCAD*, pages 289–294, 2007.

[7] J. K. Dey, J. F. Kurose, and D. F. Towsley. On-line scheduling policies for a class of IRIS (increasing reward with increasing service) real-time tasks. *IEEE Transactions on Computers*, 45(7):802–813, 1996.

[8] J. Hsu, A. Kansal, J. Friedman, V. Raghunathan, and M. Srivastava. Energy harvesting support for sensor networks. In *SPOTS track at IPSN 2005*, 2005.

[9] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the Design Automation Conference*, pages 275–280, 2004.

[10] 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.

[11] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power management in energy harvesting sensor networks. *Trans. on Embedded Computing Sys.*, 6(4):32, 2007.

[12] J. W.-S. Liu, K.-J. Lin, W.-K. Shih, A. C.-S. Yu, C. Chung, J. Yao, and W. Zhao. Algorithms for scheduling imprecise computations. *IEEE Computer*, 24(5):58–68, May 1991.

[13] S. Liu, Q. Qiu, and Q. Wu. Energy Aware Dynamic Voltage and Frequency Selection for Real-Time Systems with Energy Harvesting. In *Design, Automation and Test in Europe (DATE 08)*, Munich, Germany, March 10-14 2008.

[14] C. Moser, D. Brunelli, L. Thiele, and L. Benini. Real-time scheduling for energy harvesting sensor nodes. In *Real-Time Systems*, volume 37, pages 233–260, Norwell, MA, USA, 2007. Kluwer Academic Publishers.

[15] C. Moser, J.-J. Chen, and L. Thiele. Reward maximization for embedded systems with renewable energies. *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA08)*, 0:247–256, 2008.

[16] C. Moser, L. Thiele, D. Brunelli, and L. Benini. Adaptive power management in energy harvesting systems. In *DATE '07: Proceedings of the Conference on Design, Automation and Test in Europe*, pages 773–778, NY, USA, 2007. ACM Press.

[17] C. Moser, L. Thiele, D. Brunelli, and L. Benini. Robust and Low Complexity Rate Control for Solar Powered Sensors. In *Design, Automation and Test in Europe (DATE 08)*, Munich, Germany, March 10-14 2008.

[18] C. Park and P. Chou. Ambimax: Autonomous energy harvesting platform for multi-supply wireless sensor nodes. In *Proceedings of the Sensor and Ad Hoc Communications and Networks. SECON '06.*, volume 1, 2006.

[19] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. B. Srivastava. Design considerations for solar energy harvesting wireless embedded systems. In *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks, IPSN 2005*, pages 457–462, UCLA, Los Angeles, California, USA, April 25-27 2005.

[20] 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.

[21] C. Rusu, R. Melhem, and D. Mosse. Maximizing the system value while satisfying time and energy constraints. In *IEEE 23th Real-Time System Symposium*, pages 246–255, Dec. 2002.

[22] C. Rusu, R. Melhem, and D. Mossé. Multiversion scheduling in rechargeable energy-aware real-time systems. In *EuroMicro Conference on Real-Time Systems (ECRTS'03)*, pages 95–104, 2003.

[23] W.-K. Shih, J. W.-S. Liu, and J.-Y. Chung. Algorithms for scheduling imprecise computations with timing constraints. *SIAM J. Computing*, 20(3):537–552, June 1991.

[24] F. Simjee and P. H. Chou. Everlast: long-life, supercapacitor-operated wireless sensor node. In *ISLPED '06: Proceedings of the 2006 international symposium on Low power electronics and design*, pages 197–202, New York, NY, USA, 2006. ACM Press.

[25] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 374–382, 1995.