Robust and Low Complexity Rate Control for Solar Powered Sensors

Clemens Moser, Lothar Thiele
Swiss Federal Institute of Technology Zurich
{moser,thiele}@tik.ee.ethz.ch

Davide Brunelli, Luca Benini
University of Bologna
{dbrunelli,lbenini}@deis.unibo.it

ABSTRACT

This paper is concerned with solar driven sensors deployed in an outdoor environment. We present feedback controllers which adapt parameters of the application such that a maximal utility is obtained while respecting the time-varying amount of available energy. We show that already simple applications lead to complex optimization problems, involving unacceptable running times and energy consumptions for resource constrained nodes. In addition, naive designs are highly susceptible to energy prediction errors. We address both issues by proposing a hierarchical control approach which both reduces complexity and increases robustness towards prediction uncertainty. As a key component of this hierarchical approach, we propose a new worst-case energy prediction algorithm which guarantees sustainable operation. All methods are evaluated using long-term measurements of solar energy in an outdoor setting. Furthermore, we measured the implementation overhead on a real sensor node.

1. INTRODUCTION

Energy is a primary constraint in the design of sensor networks. This fundamental energy constraint further limits everything from data sensing rates and link bandwidth, to node size and weight. For example, it has become evident that energy storage devices largely dominate the form factor of existing sensor nodes and prevent their miniaturization. Moreover, the limitation of the energy supply has constantly impeded the progress of sensor networks towards large scales and true autonomous operation. For instance, long-term monitoring of the environment is one of the sensor network visions. For this application, finite energy reservoirs like batteries render the deployment and maintenance of large scale sensor networks extremely cumbersome.

Recently, a number of solar powered prototype nodes have been presented which perform more and more efficient energy conversion [4, 10]. Latest versions accurately track the maximum power point (MPP) of a solar cell and can adapt to changing light conditions quickly. However, it has also become evident that the energy generated by small solar cells is limited. Nodes executing a given application may frequently run out of energy in times with insufficient illumination. If one strives for predictable, continuous operation of a sensor node, common power management techniques have to be reconceived. In addition to perform classical power saving techniques, the sensor node has to adapt to the stochastic nature of the solar energy. Goal of this adaptation is to maximize the utility of the application in a long-term perspective. The resulting mode of operation is sometimes also called *energy neutral* operation [5]: The performance of the application is not predetermined a priori, but adjusted in a best effort manner during runtime and ultimately dictated by the power source.

The authors of [5] are the first who propose adaptive performance scaling techniques for energy harvesting sensor nodes. Here, the problem of tuning the duty cycle of a solar-driven sensor node has been formulated as a linear program (LP). This LP has to be solved periodically. The objective in [5] is to maximize the utilization of solar energy, i.e. to maximize the average duty cycle. For this particular problem, the authors propose a heuristic algorithm which reduces the computational overhead compared to solving a LP online.

The most recent work in [9] generalizes the approach in [5] for a much larger variety of application scenarios, constraints and optimization objectives. In fact, scaling the performance by duty cycling corresponds to one single rate in this more general application model. As a main contribution, the authors of [9] suggest the use of multiparametric programming to obtain *optimal* control laws. Simple data sensing applications serve as example to demonstrate the usefulness of the approach.

However, the formulation of the control problem in [9] leads to severe performance degradations if the energy harvested in the future derives from what has been estimated. Furthermore, for applications of practical concern, the controllers proposed in [9] quickly exceed the computation and storage resources of today's sensor nodes. By proposing a novel control design which is tailored to typical sensor node applications we show how the robustness towards energy prediction mistakes can be increased. At the same time, our reformulation of the control problem reduces the on-line complexity of the controllers.

2. CONTRIBUTIONS

The following new results are described in this paper: We present a hierarchical control design which overcomes several drawbacks of previously proposed designs: First, the computation overhead and storage demand of the on-line controller is reduced significantly. In a case study, the achieved reductions amount 91% and 83%, respectively. Second, by designing the upper control layer for worst-case situations, depletion of the energy storage is avoided and robustness of the overall system is increased. Specifically, a new worst-case energy prediction algorithm accounts for the fact that the short-term energy prediction can be considerably lower than average expectations. And third, the proposed control formulation renders manual parameter tuning of finite horizon controllers unnecessary, leading to an automatical stabilization of the system. Both simulation results as well as detailed analysis of the implementation overhead on a BTnode [1] are provided.

978-3-9810801-3-1/DATE08 © 2008 EDAA

3. SYSTEM CONCEPT

The system model is depicted in Fig. 1. The whole hardware/software system is powered by an energy harvesting device that delivers in a unit time interval starting at t the energy $E_S(t)$. In the same time interval, the system uses energy $E_R(t)$. At time t, there is the stored energy $E_C(t)$ available.

It is a typical feature of recent energy harvesting circuits that solar energy E_S is either used directly by the sensor node or is stored with (low) efficiency η for later use. This possibility of bypassing the energy storage can be exploited to save energy. Therefore, we extended the general control problem formulated in [9] by the energy storage model first introduced in [5]. At the same time, however, this extension increases the complexity of the required control strategies as we will see in Section 7.

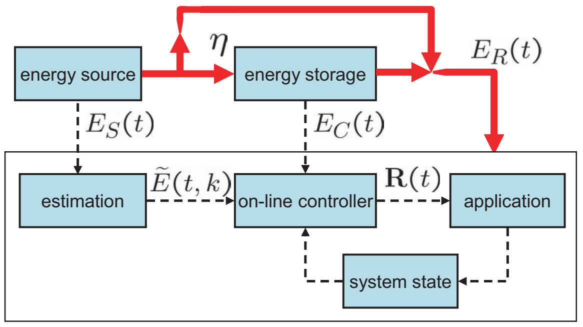

Figure 1: Illustration of the system concept.

Besides the application, there are two additional software tasks running on the target architecture. The estimator predicts future energy production of the harvesting device based on measurements of the past. The controller adapts properties of the application, e.g. task activation rates, based on the estimation of future available energy, the currently stored energy and additional information about the system state, e.g. the amount of available data memory. Parameters of the application are modified by the on-line controller. During execution, the system state (e.g. the amount of information stored in local memory and the stored energy) is changed.

4. BASIC MODELS AND METHODS

4.1 Physical power flow and energy storage model

The modeling is based on the notion of discrete time $t \in \mathbb{Z}_{\geq 0}$ where the difference in physical time between two discrete time instances is denoted as T. Energy related sensing and control may happen only at times t. In a practical setting, one may have a basic time interval T of a few minutes or even an hour.

The energy harvesting device is modeled as a power source which delivers energy $E_S(t)$ in the time interval $[t, t+1)$ of length T. Therefore, in time interval $[t_1, t_2)$ with $t_1, t_2 \in \mathbb{Z}_{\geq 0}$ it delivers energy $E_S(t_1, t_2) = \sum_{t_1 \leq u < t_2} E_S(u)$. The incoming power can be stored in an energy storage device, e.g. a rechargeable battery or a supercapacitor. We denote η the storage efficiency of a non-ideal storage device. The energy level at time t is denoted as $E_C(t)$. In dependence on the energy $E_R(t)$ drawn from the sensor node, the increment $\Delta E_C(t)$ of the stored energy is defined at each time t according to the following statement:

$$\begin{array}{lll} \textbf{if}\, E_S(t) > E_R(t) & \textbf{then} & \Delta E_C(t) = \eta \cdot (E_S(t) - E_R(t)) \\ & \textbf{else} & \Delta E_C(t) = (E_S(t) - E_R(t)) \end{array}$$

If the generated energy $E_S(t)$ is higher than energy $E_R(t)$, the sensor node is powered directly by the solar cell and excess energy is used to replenish the energy storage. As in [5], we account for the efficiency η during the charging of the energy storage. As the arrangement in Fig. 1 is fully symmetrical, one could also consider η when the storage is discharged.

The energy $E_R(t)$ is used to execute tasks on various system components. A task $\tau_i \in \mathbf{I}$ from the set of tasks $\mathbf{I}$ needs energy e_i for completing a single instance. We suppose that a task is activated with a time-variant rate $r_i(t)$, i.e. during the basic time interval T starting at t, the task is executed $r_i(t)$ times. Therefore, a task needs energy $E_i(t_1, t_2) = \sum_{t_1 \leq u < t_2} e_i \cdot r_i(u)$ in time interval $[t_1, t_2)$ for successful execution. Finally, we denote $\mathbf{R}(t)$ the vector of all task rates r_i at time t.

4.2 Receding horizon control

The estimation unit receives tuples $(t, E_S(t))$ for all times $t \geq 1$ and delivers N predictions on the energy production of the energy source. We assume that the prediction intervals are of equal size denoted as the number L (in units of the basic time interval T). We denote the total prediction horizon $H = N \cdot L$ (again in units of the basic time interval T), see also Fig. 2. At time t, the predictor produces estimations $\widetilde{E}_S(t + k \cdot L, t + (k+1) \cdot L)$ for all $0 \leq k < N$. We write $\widetilde{E}(t,k) = \widetilde{E}_S(t + k \cdot L, t + (k+1) \cdot L)$ as a shorthand notation, i.e. the estimation of the incoming energy in the $(k+1)$st prediction interval after t.

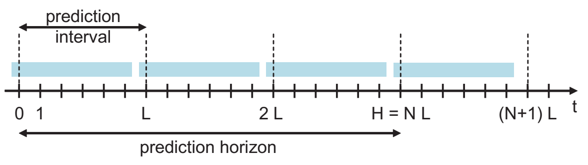

Figure 2: Illustration of the prediction horizon.

At time t, the controller is computing the control sequence $\mathbf{R}(t + k \cdot L)$ for all prediction intervals $0 \leq k < N$ based on the estimates $\widetilde{E}(t,k)$ as well as the current system state (e.g. $E_C(t)$). In other words, the rates of the different tasks r_i are planned to be constant during each prediction interval. However, *only the first* control rates $\mathbf{R}(t)$ are applied to the system during the first time step T. The rest of the control sequence is discarded. At time $t + T$, a new vector $\mathbf{R}(t)$ is computed which extends the validity of the previous vector $\mathbf{R}(t-1)$ by one time step. Again only the first control is used, yielding a receding horizon control (RHC) strategy.

4.3 Linear program specification

In this paper, we restrict ourselves to the discussion of an example application which is modeled as a linear program (LP). However, for more sophisticated application models and more general linear program specifications, the reader is referred to [9].

Let us assume a sensor node is expected to observe some phenomenon of interest in an environmental monitoring application. For this purpose, an image has to be recorded with a camera sensor and the data has to be transmitted to a base station. We can model these requirements using a data sensing task τ_1 and a data transmission task τ_2. The data sensing task τ_1 is operated with rate $r_1(t)$. At every instantiation, the sensing task τ_1 drains e_1 energy units and stores an image in some local memory. The transmission task τ_2 is transmitting images with a rate $r_2(t)$ and energy demand e_2. Thereby, task τ_2 reduces the occupied memory by one image per instantiation. The corresponding linear program is given by (1).

$$
\begin{array}{cl}
\text{maximize } J = \mu \text{ subject to:} & (1) \\
\hline
r_1(t + j \cdot L) \geq \mu & \forall 0 \leq j < N \\
r_1(t+j), r_2(t+j) \geq 0 & \forall 0 \leq j < N \\
E_C(t + k \cdot L) = E_C(t) - & \\
-\sum_{j=0}^{k-1} (e_1 \cdot r_1(t+j) + e_2 \cdot r_2(t+j)) + & \\
+\sum_{j=0}^{k-1} \widetilde{E}(t,j) - (1-\eta)\sum_{j=0}^{k-1} \lambda(j) & \forall 1 \leq k \leq N \\
\lambda(j) \geq \widetilde{E}(t,j) - e_1 \cdot r_1(t+j) - & \\
-e_2 \cdot r_2(t+j) \geq 0 & \forall 0 \leq j < N \\
M(t+k) = M(t) + & \\
+\sum_{j=0}^{k-1} (r_1(t+j) - r_2(t+j)) & \forall 0 \leq k < N \\
0 \leq M(t+j) \leq M_{max} & \forall 0 \leq j < N \\
E_C(t + N \cdot L) \geq E_C(t) + \alpha(t) & \\
\hline
\end{array}
$$

The optimization objective of the linear program (1) is to maximize the minimal rate with which the task τ_1 is operated in the finite horizon $0 \leq k < N$. In terms of intervals, the objective translates into a minimization of the maximum interval between any two consecutive measurements. This could be a reasonable objective if one attempts to minimize the unobserved time periods between two recorded images.

The auxiliary variable λ accounts for the physical switching behaviour described in Section 4.1: In intervals when the energy provided by the source is higher then the energy demand of the sensor node, the energy storage is charged with efficiency η. The other way round, the energy storage is discharged in intervals when $\widetilde{E}(t,j)$ is low and λ is forced to 0.

The last inequality in LP(1) is called a final state constraint and is used to stabilize the receding horizon controller. With the help of the energy offset $\alpha(t)$ the stored energy at the end of the prediction horizon can be influenced. In [9], α is manually tuned to some fixed value. However, it has become evident that the choice of $\alpha(t)$ severely influences the performance of the optimization: decreasing $\alpha(t)$ results in a more aggressive control behaviour, running the risk to deplete energy E_C. On the other hand, increasing $\alpha(t)$ may lead to overly conservative rates $\mathbf{R}$, poor performance and a high energy level E_C. In Section 6, a solution will be presented how α can be tuned automatically using a hierarchical control approach.

Obviously, solving at each time step t a linear program in a resource limited system is prohibitive in general. In the next Section 5, we will describe a method for solving the above linear program off-line and using the result for constructing a optimal on-line controller.

5. MULTIPARAMETRIC LINEAR PROGRAMMING

In [9], the fundamental observation made is that the considered linear programs (like the one in (1)) are parameterized and can be solved off-line. We will briefly recall the main principles of multiparametric linear programming.

A state vector $\mathbf{X}$ is defined consisting of the actual system state, the level of the energy storage as well as the estimation of the incoming energy over the finite prediction horizon (cp. Fig.1). For the example in (1), the state vector $\mathbf{X}$ can be written as

$$\mathbf{X}(t) = \Big(E_C(t)\,,\, M(t)\,,\, \widetilde{E}(t,0)\,,\, \ldots, \widetilde{E}(t,N-1)\Big)^T \qquad (2)$$

Furthermore, let us denote the vector of optimal control inputs to the system, i.e., the vector of planned rates $\mathbf{R}$ as

$$\begin{aligned}\mathbf{U}^*(\mathbf{X},t) = \big(\mathbf{R}^T(t)\,,\, \mathbf{R}^T(t+L)\,,\, \ldots, \\ \mathbf{R}^T(t+(N-1)\cdot L)\big)^T.\end{aligned} \qquad (3)$$

The state space of $\mathbf{X}$ can be subdivided into a number N_{CR} of polyhedrons. For each of these polyhedrons i (also called critical regions) the optimal solution $\mathbf{U}^*(\mathbf{X})$ of the control problem can be calculated explicitly as

$$\mathbf{U}^*(\mathbf{X}) = \mathbf{B}_j\mathbf{X} + \mathbf{C}_j \quad \text{if } \mathbf{H}_j\mathbf{X} \leq \mathbf{K}_j, j = 1, \ldots, N_{CR} \qquad (4)$$

where $\mathbf{B}_j \in \mathbb{R}^{N\times(N+2)}, \mathbf{C}_j \in \mathbb{R}^N$ and $\mathbf{H}_j\mathbf{X} \leq \mathbf{K}_j, j = 1 \ldots N_{CR}$ is a polyhedral partition of the state space of $\mathbf{X}$. For simplicity, we dropped the dependence on t of the state vector $\mathbf{X}$. The computation of the vectors and matrices of control law (4) is done off-line using, e.g., the algorithm presented in [3].

In the on-line case, the controller has to identify to which region j the current state vector $\mathbf{X}$ belongs. After this simple membership test, the optimal control rates $\mathbf{R}(t)$ for the next N prediction interval are computed by evaluating a linear function of $\mathbf{X}$. These rates $\mathbf{R}(t)$ are identical to the rates one would obtain by solving the linear program. However, the computational demand is greatly reduced compared to solving a LP on-line. After having solved the mp-LP in advance, a set of N_{CR} polyhedra with associated control laws has to be stored and evaluated at each time step t.

If the number of critical regions N_{CR} gets large, the storage demand and the computational effort grow quickly. Then many tests of the form $\mathbf{H}_j\mathbf{X} \leq \mathbf{K}_j$ must be performed. Typically, the computational effort spent for these matrix multiplications is much higher then evaluating the linear function $\mathbf{B}_j\mathbf{X} + \mathbf{C}_j$.

6. HIERARCHICAL SYSTEM MODEL

The main idea presented in this section is to decouple the control problem into two independent sub-problems, each with its dedicated energy prediction algorithm. According to Figure 3, subcontroller 1 receives estimates of the daily energy in order to ensure long-term sustainability of the energy harvesting system. Therefore, we assume a prediction interval to be 1 day and the total prediction horizon should be chosen in the order of several weeks. At the interface of both control layers, we define the energy allowance $E_D(t)$

i.e. the sum of energies which shall be used by the system at the next day.

By calculating $E_D(t)$, the upper layer determines an energy budget which serves as input to the lower control layer. Receiving hourly estimates with a prediction horizon of 1 day, subcontroller 2 decides how to set the individual rates of the application $\mathbf{R}(t)$ during the next day. At this, the efficiency of the energy utilization is optimized by e.g. exploiting the sunlight directly when available.

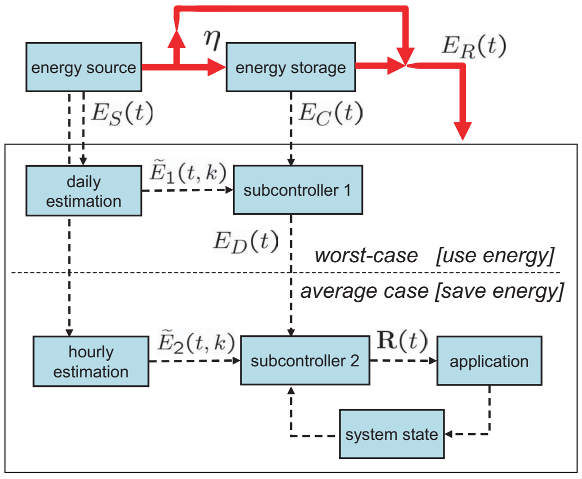

Figure 3: Illustration of the hierarchical control model.

6.1 Long-term performance optimization

The optimization objective of subcontroller 1 is to maximize the minimal available energy $E_D(t)$ per day. In other words, we would like to guarantee sustainable operation in a worst case sense. For this reason, an energy prediction for a worst case scenario is required. Unlike previously proposed energy estimation algorithms, we would like to determine the *minimal accumulated energy* for the next days, which we denote as $\widetilde{E}_1(t,k)$. For prediction intervals of size $L_1 \cdot T = 24h$, the daily energy prediction algorithm is displayed in Algorithm 1.

During the day, only a counter is running which accumulates the harvested energy of the current day. Once per day, Algorithm 1 is executed and new energy estimates for the next N days are computed. The harvested energies of the last N days are stored in the vector $E_{S,d}(d)$. In a first step **(S1)**, vector $E_{S,d}(d)$ is updated by shifting all entries by one day and storing the harvested energy $E_{S,d}(1)$ of the current day. In step **(S2)**, the daily energies $E_{S,d}(d)$ are summed up to obtain the energies $\widetilde{\epsilon}_s(\Delta)$ for time periods Δ of several days. As an internal state, the estimator stores the minimum of all vectors $\widetilde{\epsilon}_s(\Delta)$ observed so far in a variable denoted $\epsilon_s^l(\Delta)$. Next, Step **(S3)** checks whether the energies $\widetilde{\epsilon}_s(\Delta)$ harvested in the past Δ days are smaller then the minimum energies $\epsilon_s^l(\Delta)$ observed so far for the respective time interval Δ. In addition, a constant γ is added to the minimum $\epsilon_s^l(\Delta)$ to account for aging of old measurements. In doing so, the algorithm avoids pessimistic predictions based on low values of $\epsilon_s^l(\Delta)$ measured long time ago. Finally, step **(S4)** improves the estimates $\epsilon_s^l(\Delta)$ if intervals with little harvested energy $\widetilde{\epsilon}_s(\Delta)$ have occurred recently.

Algorithm 1 Daily worst-case energy prediction.

```
Input: $(t, E_S(t))$
Output: $\widetilde{E}_1(t,k)\ \forall\, 1 \le k \le N$
  if $t == 0$ then
    $\epsilon_s^l(\Delta) = \infty \quad \forall\, 1 \le \Delta \le N$
  if $t \bmod L_1 == 0$ then
    $E_{S,d}(d) = \sum_{i=(d-1)\cdot L}^{d\cdot(L-1)} E_S(t-i) \qquad \forall 1 \le d \le N$   (S1)
    $\widetilde{\epsilon}_s(\Delta) = \sum_{i=1}^{\Delta} E_{S,d}(i) \qquad \forall 0 \le \Delta \le N$   (S2)
    $\epsilon_s^l(\Delta) = \epsilon_s^l(\Delta) + \Delta \cdot \gamma \qquad \forall 1 \le \Delta \le N$   (S3)
    $\epsilon_s^l(\Delta) = \min\left[\epsilon_s^l(\Delta), \widetilde{\epsilon}_s(\Delta)\right] \qquad \forall 1 \le \Delta \le N$   (S3)
    $\widetilde{E}_1(t,k) = \max_{0 \le l \le N-k} \left\{\epsilon_s^l(k+l) - \widetilde{\epsilon}_s(l)\right\}\ \forall k \ge 1$   (S4)
```

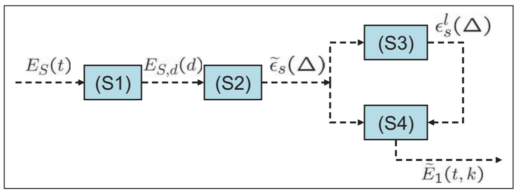

Figure 4: Visualization of the daily energy prediction in Algorithm 1.

The linear optimization problem underlying subcontroller 1 is given below. With the help of the worst case energy predictions $\widetilde{E}_1(t,k)$ it is now possible to maximize the smallest energy $E_D(t+k\cdot L)$ for all $0 \le k < N$ in the prediction horizon. However, only the first energy $E_D(t)$ is passed to subcontroller 2 and will be used during the next day.

$$\text{maximize } J = \mu \text{ subject to:} \qquad \text{(subcontroller 1)}$$

$$\begin{aligned}
E_D(t+k\cdot L) &\ge \mu && \forall 0 \le k < N \\
E_C(t+k\cdot L) = E_C(t) &+ \widetilde{E}_1(t,k) - \\
-\textstyle\sum_{j=1}^{k} &\left(E_D(t+(j-1)\cdot L)\right) && \forall 1 \le k \le N \\
0 \le E_C(t+k\cdot L) &\le E_{max} && \forall 1 \le k \le N
\end{aligned}$$

6.2 Short-term power saving

For a general control problem and a prediction $\widetilde{E}_2(t,k)$ the final state constraint of the lower control layer can be set to $\alpha(t) = E_D - \sum_{i=0}^{N_2-1} \widetilde{E}_2(t,i)$ in order to force the system to spend energy E_D during the next day. Hence, the final state constraint α is calculated automatically by the upper layer.

For the example in LP (1), we found a more elegant solution. We have to ensure that the node uses no more than energy E_D during the next day. In particular, we have to take into account a worst case situation where all energy used to drive τ_1 and τ_2 has to traverse the battery with conversion efficiency η. We can write

$$e_1 \cdot L_2 \cdot \sum_{i=0}^{N_2-1} r_1(t+i) + e_2 \cdot L_2 \cdot \sum_{i=0}^{N_2-1} r_2(t+i) = \eta \cdot E_D(t)$$

Furthermore, we want images to be transmitted within one day. This real-time requirement accounts for the fact, that an image recorded more than one day ago may be of no use to the observer at the base station. Therefore, the sum of sensing r_1 and transmission rates r_2 most coincide at the end of the day, i.e. $\sum_{i=0}^{N-1} r_1(t+i) = \sum_{i=0}^{N-1} r_2(t+i)$. Remembering that we still want to minimize the maximum unobserved time interval, we set the sensing rate $r_1 = \frac{\eta \cdot E_D(t)}{N \cdot (e_1+e_2)}$ for all N_2 intervals of the next day. Like that, the only degree of freedom to be exploited by subcontroller 2 is *when* to transmit the images with rate $r_2(t)$ in order to *save* as much energy as possible. Thus, we want to maximize the energy $E_C(N)$ at the end of the day. For this setup, we can formulate the linear program to be solved by subcontroller 2 as follows:

$$\text{maximize } J = E_C(N) \text{ subject to:} \qquad \text{(subcontroller 2)}$$

$$\begin{aligned}
& r_1 = \tfrac{\eta \cdot E_D(t)}{N \cdot (e_1+e_2)} & \\
& \textstyle\sum_{j=0}^{N_2-1} r_2(t+j) = \tfrac{\eta \cdot E_D(t)}{e_1+e_2} & \\
& r_2(t+j) \geq 0 & \forall 0 \leq j < N \\
& E_C(t+k \cdot L) = E_C(t) - & \\
& -k \cdot e_1 \cdot r_1 - e_2 \cdot \textstyle\sum_{j=0}^{k-1} r_2(t+j) + & \\
& + \textstyle\sum_{j=0}^{k-1} \widetilde{E}_2(t,j) - (1-\eta) \sum_{j=0}^{k-1} \lambda(j) & \forall 1 \leq k \leq N \\
& \lambda(j) \geq \widetilde{E}_2(t,j) - e_1 \cdot r_1 - & \\
& -e_2 \cdot r_2(t+j) \geq 0 & \forall 0 \leq j < N \\
& M(t+k) = M(t) + & \\
& + \textstyle\sum_{j=0}^{k-1} (r_1(t+j) - r_2(t+j)) & \forall 0 \leq k < N \\
& 0 \leq M(t+j) \leq M_{max} & \forall 0 \leq j < N
\end{aligned}$$

To exploit the typical profile of solar light intensity during a day, an energy predictor $\widetilde{E}_2(t,k)$ is needed which predicts the most probable light values for the next hours. In contrast to $\widetilde{E}_1(t,k)$, the hourly estimation $\widetilde{E}_2(t,k)$ should keep a history of harvested energies and use a representative, *average* value as predictor.

7. EXPERIMENTAL RESULTS

We implemented the on-line controllers for the exemplary case study using the MATLAB toolbox in [6]. We opted for a storage efficiency of $\eta = 80\%$. Measurements of solar light intensity during nearly 5 years recorded at [7] serve as energy input $E_S(t)$. The time interval between two samples is 5 minutes, so we set the basic time interval $T = 5\,\text{min}$. Using this data, we could extensively test the performance of our algorithms for time scales one usually wants to achieve with solar powered sensor networks.

7.1 Single Multiparametric Controller

For the camera sensor application introduced in Section 4.3, a sensor node with a maximum storage capacity $M_{max} = 1000$ is used. In general, radio communication is the main energy consumer in sensor networks. Hence we choose $e_1 = 0.1$ and $e_2 = 0.9$ as energy demands of the two tasks. The energy offset $\alpha(t)$ is set to 0 for all times t. As in [9] and [5], we set the prediction horizon to $H \cdot T = 24h$. Since it is of practical concern to keep the number of prediction intervals $\widetilde{E}(t,k)$ as small as possible, we subdivided the horizon in $N = 6$ intervals of lenght $L = 48$. The energy prediction algorithm we used is the same as in [9] (with parameter $\alpha = 0.5$). For this control problem, we compute the explicit solution via multiparametric programming.

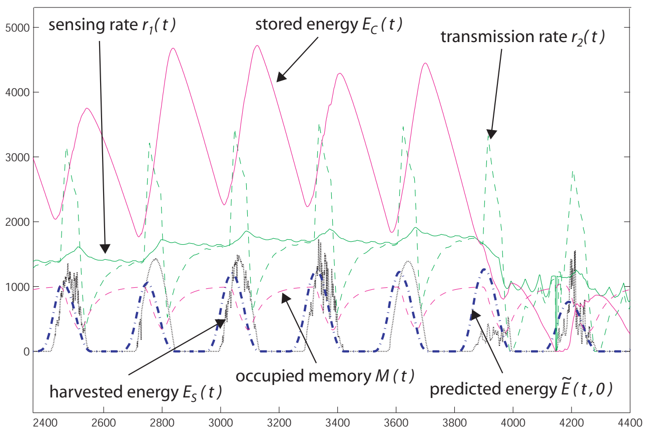

Figure 5: Single controller for (1) using average energy prediction $\widetilde{E}(t,k)$.

In Fig. 5, the generated controller is optimizing the sampling rate r_1 and the transmission rate r_2 during 7 days. During the first 5 days, the controller achieves to optimize the rate r_1 in spite of the unstable power supply $E_S(t)$. Consequently, the stored energy $E_C(t)$ is highly varying: It is increasing during day and decreasing at night. Also the transmission rate r_2 is oscillating around the sampling rate r_1. Since it is favourable to use energy when available, data is stored at night and transmitted during day. However, after two days with little sunshine, the controller is forced to suspend the sampling of data and achieves the worst objective value possible: Shortly before $t = 4200$, the sampling rate r_1 is set to 0 for 4 time intervals of length T, i.e. for 20 minutes no image is transmitted to the basestation. The main reason for this breakdown is the averaging energy predictor $\widetilde{E}$ which – per definition – is unable to foresee extreme situations. As illustrated in Fig. 5, an overestimation of the actually incoming energy E_S may deplete the stored energy quickly, resulting in an unpredicted performance degradation. From our experiments we know that the controller computed from (1) is unable to avoid these kind of breakdowns with the chosen predictor, independent of the length of the prediction horizon H and the number of intervals N.

7.2 Hierarchical Control Design

Figure 6 depicts the performance of the hierarchical control approach. The simulation result is plotted for 22 days, covering also the 7 days of simulation period of the previous experiment. This time, the controllers successfully stabilize the sampling rate at $r_1 \approx 1300$. Depending on the season, we found that rate r_1 may slightly increase (in summer) or decrease (during winter). In any case, subcontroller 1 prevents the system to run out of energy and simultaneously maximizes the amount of energy E_D per day.

Table 1 summarizes the parameters and the implementation overheads of both control designs. To avoid unnecessary control overhead, subcontroller 1 is not activated every $T = 5$ minutes, but only once per day. We set the aging of

Table 1: Complexity reduction due to hierarchical control design

control design	N	$H \cdot T$	activation frequency	N_{CR}	storage (real numbers)	ops (worst case)
single controller	6	$1day$	every $T = 5\,\text{min}$	1049	28323	52449
hierarchical, subcontroller 1	$N_1 = 30$	$H_1 \cdot T = 30days$	every $L_1 \cdot T = 1day$	30	1920	3689
subcontroller 2	$N_2 = 6$	$H_2 \cdot T = 1day$	every $T = 5\,\text{min}$	161	2898	4829

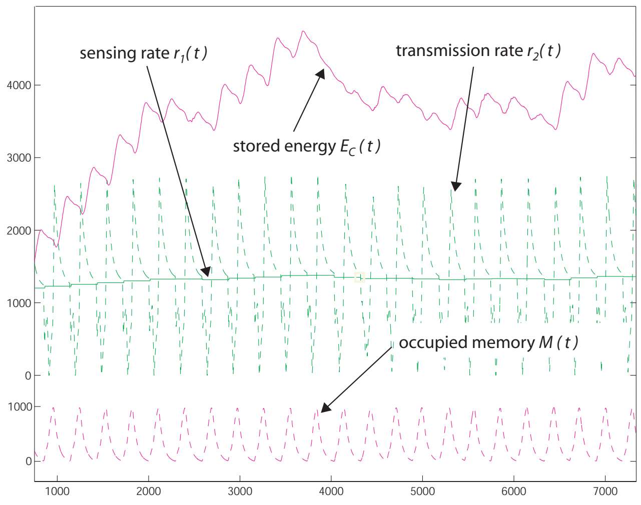

Figure 6: Hierarchical control design with worst case prediction $\widetilde{E}_1(t,k)$ and average prediction $\widetilde{E}_2(t,k)$.

old data $\gamma = 100$ for the worst case prediction algorithm $\widetilde{E}_1$. For the lower control layer, the prediction algorithm from [9] has been used, again with a factor $\alpha = 0.5$.

Concerning the storage demand, the numbers given in Table 1 are obtained for the efficient representation of the control laws proposed in [2]. The hierarchical approach yields a substantial reduction of the storage requirement of $83,0\%$. In terms of physical memory, the storage demand is reduced from 55,3 Kbyte to 9,4 Kbyte using a 16 bit integer representation per coefficient. This reduction makes the algorithms applicable to commonly used sensor nodes like the Tmote Sky [8], which exhibits 48 Kbyte Flash ROM or the BTnode [1], with 128 Kbyte.

Since the daily energy estimation and subcontroller 1 are only activated once per day, their influence on the computation demand is neglectable. The overall reduction of the worst case computation demand is $91,0\%$ compared to a single controller. Here, worst case refers to the situation where the currently active region is the last one to be checked (cp. Section 5). The dimensions of the control laws of subcontroller 1 and 2 are of similar order as those analyzed and implemented in [9]. In our implementation on a BTnode [1], we measured a worst case computation time < 190 ms for subcontroller 2, yielding a neglectable control overhead every $T = 5$ min.

8. CONCLUSION

We have successfully designed and evaluated a hierarchical control design which is tailored to the requirements of sensor nodes. Due to its worst case design, the upper control layer prevents the sensor node from running out of energy. On the lower control layer, we have shown how power saving techniques can work efficiently if stability of the operation is guaranteed by the layer above. Simultaneously, the reformulation of the control problem into two subproblems yields a significant reduction in on-line complexity. All methods are supported by extensive simulation results which are based on long-term measurements of solar energy. By implementing the found control laws on a sensor node, we demonstrate that our methods are well-suited for resource-constrained systems.

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 founded by the European Network of Excellence ARTIST2.

9. REFERENCES

[1] J. Beutel, M. Dyer, M. Hinz, L. Meier, and M. Ringwald. Next-generation prototyping of sensor networks. In *Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys 2004)*, pages 291–292. ACM Press, New York, Nov. 2004.

[2] F. Borrelli, M. Baotic, A. Bemporad, and M. Morari. Efficient On-Line Computation of Constrained Optimal Control. In *IEEE Conference on Decision and Control*, pages 1187–1192, Orlando, Florida, Dec. 2001.

[3] F. Borrelli, A. Bemporad, and M. Morari. A Geometric Algorithm for Multi-Parametric Linear Programming. *Journal of Optimization Theory and Appl.*, 118(3):515–540, Sept. 2003.

[4] 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.

[5] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power management in energy harvesting sensor networks. In *ACM Transactions on Embedded Computing Systems*, May 2006, also: NESL Tech. Report #: TR-UCLA-NESL-200605-01.

[6] M. Kvasnica, P. Grieder, and M. Baotić. Multi-Parametric Toolbox (MPT), 2004.

[7] Bern University of Applied Sciences, Engineering and Information Technologies, Photovoltaic Laboratory. Recordings of solar light intensity at Mont Soleil from 01/01/2002 to 31/09/2006. www.pvtest.ch, March, 2007.

[8] 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.

[9] 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, New York, NY, USA, 2007. ACM Press.

[10] 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.