WEA-CPS

First International Workshop on
Energy Aware Design and Analysis
of Cyber Physical Systems

In conjunction with CPS WEEK 2010

April 12th, 2010
Stockhom, Sweden
Organization committee

WEA-CPS 2010

General co-chairs

Harini Ramaprasad, Southern Illinois University Carbondale, USA
Radu Dobrin, Mälardalen University, Sweden

Program committee

Alexandru Andrei, Ericsson, Sweden
Hakan Aydin, George Mason University, USA
Enrico Bini, Scuola Superiore Sant'Anna, Italy
Bruce Childers, University of Pittsburgh, USA
Petru Eles, Linköpings University, Sweden
Tullio Facchinetti, University of Pavia, Italy
Shelby Funk, University of Georgia, USA
Tomas Lenvall, ABB Corporate Research, Sweden
Frank Mueller, North Carolina State University, USA
Sasikumar Punnekkat, Mälardalen University, Sweden
Table of Contents

Joint Voltage and Modulation Scaling for Energy Harvesting Sensor Networks
  Bo Zhang, Robert Simon, Hakan Aydin .................................................................2

Real-Time Modeling and Control of a Cyber-Physical Energy System
  Tullio Facchinetti, Marco L. Della Vedova.................................................................10

Reducing the Peak Power through Real-Time Scheduling Techniques in Cyber-Physical Energy Systems
  Tullio Facchinetti, Enrico Bini, Marko Bertogna ..................................................18

Power-Aware Scheduling in Embedded Systems: Adaptation of SDVS Scheduling using Deadline Overrun Detection
  Jonas Höglund, Stefan Olofsson, Magnus Persson, Johan Fredriksson,
  Detlef Scholle ........................................................................................................26

N-ary Sensor Model for Target Tracking in Wireless Sensor Networks
  Kenji Tei, Hikotoshi Nakazato, Yoshiaki Fukazawa, Shinichi Honiden..................34
Joint Voltage and Modulation Scaling for Energy Harvesting Sensor Networks

Bo Zhang  Robert Simon  Hakan Aydin
Department of Computer Science
George Mason University, Fairfax, VA 22030
{bzhang3, simon, aydin}@cs.gmu.edu

Abstract

Energy harvesting is rapidly becoming a critical architectural component for CPS applications that use Wireless Sensor Networks (WSNs). This paper presents an epoch-based approach for energy management in resource-constrained WSNs that utilize energy harvesting techniques. Each epoch represents a time period over which energy production can be reasonably predicted. We consider two energy harvesting hardware models, one that allows concurrent harvesting and execution, and one that does not. For both models we propose and analyze a resource management algorithm that combines energy harvesting with dynamic voltage scaling and dynamic modulation scaling. Our algorithm is optimal in the sense that it maximizes energy reserve levels at individual nodes. We have evaluated the performance of our approach with standard baseline algorithms. The results show that our algorithm outperforms the baseline algorithms under a variety of workloads and energy harvesting profiles.

1. Introduction

Many types of cyber-physical systems (CPSs), such as smart power grid applications, networks consisting of lab-on-chip nodes used for monitoring large-scale water distribution systems ([1]) and systems used for highway management, have highly distributed and computationally intensive processing requirements. Further, due to longevity, cost, ecological and managerial restrictions, these CPS applications will need to harvest environmental energy to achieve "perpetual" operation. A key technology for this class of CPS applications is a new generation of WSNs. Unlike current WSN systems, this new generation must incorporate powerful and performance sensitive processing capacity with sophisticated energy management techniques. This makes the combined use of two energy management techniques, Dynamic Voltage Scaling (DVS) ([2]) and Dynamic Modulation Scaling (DMS) ([3]), potentially quite attractive. The focus of this paper is to present a joint scaling approach for perpetually operating, computationally intensive CPS applications using WSN technology.

Due to the dynamic nature of environmental power and potential fluctuations in CPS application workload, our approach is based on an epoch-based strategy. Each epoch is a time interval during which the amount of harvested power can be reasonably predicted in advance, and remains relatively constant. Such power predictions can be conducted based on the knowledge of past harvesting records and characteristics of the environmental energy sources. In practice we expect epochs to last anywhere from several minutes to several hours. Each node then divides epoch time into a number of frames of different types based upon a periodic sensor-oriented task model.

Unlike the existing energy management approaches for battery-powered WSNs which minimize the energy consumption, we target maximizing the energy reserve of sensor nodes with energy harvesting capabilities. The benefit of this goal is that extra energy can be used to service unexpected workloads or new CPS application tasks that are introduced into the system, and to cover time periods where the harvested power is less than expected. Since such system uncertainties may cause interruption of services and consequently failures in responding to critical events, our goal is particularly important for mission-critical CPS applications. This paper achieves this goal with a joint DVS-DMS strategy. The DVS technique saves computation energy by simultaneously reducing the CPU supply voltage and frequency. The DMS technique saves communication energy by reducing the radio modulation level. In addition, our approach guarantees that all application performance requirements for computation and communication are met. The latest generations of WSN nodes, such as the iMote2, possess DVS capabilities ([4]). Although commonly used WSN radios do not typically offer DMS, we believe that as the benefits of this technique become apparent future low power radios will indeed offer that option.

Our contributions are summarized as follows: first, we propose an epoch-based control approach, exploiting DVS and DMS. As part of our approach we model three sensor tasks, sensing, computation and communication and use a periodic process model to explicitly provide support for the timely completion of these tasks. We define an energy

*Supported, in part, by US National Science Foundation grants CNS-072047 and CNS-546244 (CAREER Award)
harvesting problem that formally specifies the objective of maximizing energy reserves while meeting performance requirements. Though this specification appears to be a non-linear program, we show that it can be solved optimally. We also show how to use our algorithm in two types of harvesting models, one that allows concurrent energy harvesting and execution, and one that does not. Additionally, our joint DVS-DMS approach can be implemented on top of low-power duty cycling mechanisms which are widely used in WSN systems. Through simulation using solar energy harvesting profiles, we show that our approach can dramatically increase the level of reserved energy while still maintaining required levels of CPS system performance.

2. Related work

The joint use of DVS and DMS in wireless embedded systems has been explored in [5], [6]. In [5], Kumar et al. addressed a system-level resource allocation problem for minimizing energy consumption. They assume a system containing a mix of computation and communication tasks. [6] formulates the energy management problem as a convex optimization problem, and addresses it using genetic algorithms. Unlike our work, [5], [6] all assume battery-powered systems without energy harvesting capability. Further, the goal of their approaches is to prolong system lifetime by reducing energy consumption, without considering issues such as ensuring perpetual operation through energy harvesting or providing for energy reserve maximization.

Among the studies that explore the design of energy harvesting WSNs, in [7], Moser et al. proposed the LSA algorithm (Lazy Scheduling Algorithm) for scheduling real-time tasks in the context of energy harvesting. LSA defers task execution as late as possible in order to save energy. Liu et al. ([8]) proposed EA-DVFS (Energy-Aware Dynamic Voltage and Frequency Scaling) to improve the energy efficiency of LSA algorithm by using DVS. Both LSA and EA-DVFS manage only the CPU energy while ignoring radio energy. Other related work includes [9], which describes a probabilistic observation approach for solar energy harvesting that attempts to minimize energy allocation variance. Finally, works such as [10], [11] proposed to maximize harvested energy in order to maximize the amount of completed workload, hence the system performance.

3. System model and assumption

Our work assumes that each sensor node consists of a number of environmental sensors, a DVS-capable CPU, a DMS-capable radio, and an energy harvester. The sensor nodes are assumed to run a CPS application that is both computation and communication intensive. Each node senses the target environment, processes the sensor reading, and communicates data to other nodes. Ultimately the nodes filter data up towards a base station.

![Figure 1. A sequence of frames of different types](image-url)

3.1. Task model

Due to the simplicity in managing sensor activities and energy usage, we organize the operations of a sensor node as periodically invoked tasks. We model three basic sources of energy consumption: sense, computation and communication. A *sense* operation measures a physical quantity and generates raw reading. A *compute* operation may involve processing the raw data or aggregating data from other nodes. A *communicate* operation involves sending or receiving data packets. We will denote each of these basic operations as a subtask.

The above three subtasks are combined to form three task types. We note that sense subtasks are performed periodically and most frequently in a sensor system, followed by compute and then communicate. Based on this observation the three task types are sense-only (SO), sense-compute (SC), and sense-compute-communicate (SCC). We refer to one invocation of task as a task instance. A new task instance is invoked every $S$ time units (i.e., with period $S$) and its type is known when it arrives. The task instances are executed on frame basis ([12]). A frame refers to a time interval of length $S$ during which a task instance is invoked, executed, and completed. In order to maintain acceptable system performance, each task instance must be completed within the frame period $S$. For example, the $j^{th}$ task instance is invoked at the beginning of the $j^{th}$ frame (i.e. at time $(j-1) \cdot S$) and must complete its execution within that frame (i.e. by time $j \cdot S$). Since each frame contains only one task instance, we will use the terms frame and task instance interchangeably in the paper.

We use a tuple $\{I_{\text{sense}}^{j}, I_{\text{com}}^{j}, I_{\text{cm}}^{j}\}$ to identify the type of a frame $j$. The elements in the tuple are binary-valued with 1 indicating the existence of the specified subtasks, and 0 if not. For instance, a sense-compute frame with frame type $\{1,1,0\}$ consists of a sense subtask followed by a compute subtask, but no communicate. We assume the types of frames are determined by the application. Fig. (1) illustrates a sequence of frames of different types. SN, CP and CM represent sense, compute and communicate subtasks respectively. We denote the workload in a frame as $C$ for computation, $M$ for communication. In practice $C$ is the number of CPU cycles to be executed, $M$ is the size of data to be transmitted.

We consider a DVS-enabled CPU with $m$ discrete frequencies $f_{\min}=f_{1}<...<f_{m}=f_{\max}$, and a DMS-enabled radio with $k$ discrete modulation levels, $b_{\min}=b_{1}<...<b_{k}=b_{\max}$.
We use the terms frequency and compute speed interchangeably. In practice, the modulation level represents the number of bits encoded in one signal symbol \((I_3)\). Each modulation level \(b_i\) is associated with a communicate speed \(d_i\).

\[
d_i = R \cdot b_i
\]  

\(R\) is the symbol rate which is fixed. The execution time of the compute and communicate subtasks in frame \(j\) is equal \(C/f_j\) and \(M/d_j\) respectively. We make a common assumption that the effective transmission time dominates the overall communication time while ignoring the carrier sense time \((I_5, I_6, I_3)\). We assume the sensing time is not scalable and denote it as a constant \(t_{sen}\). The total execution time \(t_{exe}\) in frame \(j\) of type \(\{I_{sen}, I_{cp}, I_{cm}\}\) at compute speed \(f_j\) and communicate speed \(d_j\) is given as:

\[
t_{exe} = I_{sen} \cdot t_{sen} + I_{cp} \cdot e_{cm} + I_{cm} \cdot e_{cp} \]  

\(e_{sen}\) is the sensing energy, which is constant. \(e_{cp}\) and \(e_{cm}\) are the compute and communicate energy in frame \(j\):

\[
e_{cp} = \alpha f_j V_{dd,j}^2 \cdot (C/f_j) \]  

\[
e_{cm} = \beta R(2^{d_j/R} - 1) \cdot (M/d_j) \]

\(\alpha\) represents the CPU switching capacitance. \(\beta\) is a constant determined by the transmission quality and noise level \((I_3)\). The terms \(\alpha f_j V_{dd,j}^2\) and \(\beta R(2^{d_j/R} - 1)\) give the speed-dependent power of CPU and radio, that vary with \(f_j\), \(V_{dd,j}\) and \(d_j\). \(P_{ind,cp}\) and \(P_{ind,cm}\) are two constants representing the speed-independent power of CPU and radio. In DVS technique, the supply voltage \(V_{dd,j}\) can be reduced linearly alongside with \(f_j\) to obtain energy saving, making the speed-dependent CPU power essentially a cubic function of \(f_j\). Finally, we consider that energy consumed by listening radio channel activities is equivalent to transmit energy. Our model assumes a sufficient level of coordinated sleeping and transmission scheduling so that listening energy is not a significant factor. This allows us to model communication energy as the single value, i.e. \(e_{cm}\).

3.3. Energy harvesting and storage model

The sensor node is directly powered by an energy storage device (e.g., battery or super-capacitor) with capacity \(\Gamma^{max}\). The storage device receives power from the energy harvester, and delivers power to the sensor node. Generally, the harvested power is uncontrollable, but predictable based on the harvesting history \((I_{10})\). To capture the time-varying nature of the harvested power, time is divided into epochs with equal length \(L\). The harvested power is commonly assumed as an epoch-varying function denoted as \(P_i^h\) \((i\) is the epoch number). \(P_i^h\) remains constant within each epoch \(i\), but varies over epochs. Thus, the time unit used for harvesting prediction is one epoch, and we refer to the prediction horizon as \(H = N \cdot L\). Strictly speaking, \(P_i^h\) denotes the actual power received by the storage device in epoch \(i\). Hence, the power loss during the power transfer from the energy harvester to storage device is incorporated in \(P_i^h\). We assume the value of \(P_i^h\) is known (using for example, profiles \([10]\)) at the beginning of the epoch \(i\).

When a sensor node is executing tasks, it draws power from the storage device. The drawn power is controllable through DVS and DMS. The storage device stops discharging as the energy level drops to zero, and stops charging as the energy level approaches \(\Gamma^{max}\) to avoid energy overflow. \([10]\) proposed the concept of energy neutrality which basically states that the energy consumed is no larger than the energy available. This is a necessary condition for a sensor node to operate perpetually. Depending on the types of storage device, power usage and harvesting may happen concurrently or non-concurrently. Several papers assumed that concurrent usage and harvesting is commonly possible \((I_{13}, I_{10})\). However, \([14]\) pointed to the need for special hardware mechanisms to separate charge and discharge currents, which may be expensive for sensor nodes. In such systems, energy cannot be consumed (i.e. no sensing, computation, computation can take place) while harvesting.

4. Energy management with energy harvesting

In this work, we address an energy management problem in the context of energy harvesting. Our motivation is to improve the sensor nodes’ resilience to unexpected service interruption caused by depleted energy storage, while meeting the application’s timing requirements. Such energy depletion might result from many system uncertainties, e.g., workload burst or misprediction of harvested power. Motivated by this, we aim at maximizing the minimum energy level observed over time. Through this objective, we target being more resilient against unusual or emergency situations. Through the orchestrated use of DVS and DMS, along with the energy harvesting capability, we manage the consumed and harvested energy while achieving the application’s sense, computation and communication requirements. The following sections define and solve the problem formally.

In an attempt to capture all the parameters of our problem, we start with a couple of definitions. The energy level at the end of epoch \(i\) is given by:

\[
\forall i \in [1, N], \Gamma_i = \Gamma_i^{init} + \sum_{k=1}^{i} E_k^h - \sum_{k=1}^{i} E_k^e
\]
where $\Gamma^{\text{init}}$ is the initial energy level in the horizon. $E^h_k$ and $E^c_k$ are the harvested and consumed energy in epoch $k$ respectively. Starting with $\Gamma^{\text{init}}$, $\Gamma_i$ may increase or decrease depending on the consumed and harvested energy in intermediate epochs.

In our epoch-based approach, the $j^{th}$ frame of the epoch $i$ is denoted by the pair $(i, j)$. Consequently, unless otherwise stated, any frame-related variable (e.g. energy, time) $x_j$ defined in the previous section automatically becomes $x_{i,j}$ in the rest of the paper. Now, within an epoch $i$, the energy level at the end of the frame $j$, is:

$$\gamma_{i,j} = \Gamma_{i-1} + \sum_{k=1}^{j} e^h_{i,k} - \sum_{k=1}^{j} e^c_{i,k} \tag{7}$$

In other words, starting with the ending energy level $\Gamma_{i-1}$ in epoch $i - 1$ (which is also the starting energy level in epoch $i$), $\gamma_{i,j}$ is harvested by the harvested and consumed energy in frame $(i, j)$ and all its preceding frames, $e^h_{i,k}, e^c_{i,k}, k \in [1, j]$. To ensure energy neutrality, we require $\Gamma_i > 0$ and $\gamma_{i,j} > 0, \forall i, j$.

Note that there are $[L/S]$ frames in an epoch. The consumed energy in an epoch, $E^c_i$ is given as:

$$E^c_i = \sum_{j=1}^{[L/S]} e^c_{i,j} \tag{8}$$

$e^c_{i,j}$ is the energy consumption in frame $(i, j)$ (Eq. (3)). The harvested energy $E^h_i$ is given as:

$$E^h_i = \sum_{j=1}^{[L/S]} e^h_{i,j} \tag{9}$$

$$e^h_{i,j} = P^h \cdot t^h_{i,j} \tag{10}$$

$e^h_{i,j}$ is the harvested energy in frame $(i, j)$. As mentioned in the previous section, the harvested power $P^h$ is a known constant and fixed over all frames in epoch $i$. $t^h_{i,j}$ is the effective energy harvesting time in frame $(i, j)$. In concurrent harvesting model, the system can continuously harvest power throughout a frame, hence:

$$t^h_{i,j} = S \tag{11}$$

On the other hand, in non-concurrent model, task execution and harvesting cannot occur concurrently. Since $t^{exe}_{i,j}$ is the total execution time in frame $(i, j)$ (Eq. (2)), we have:

$$t^h_{i,j} = S - t^{exe}_{i,j} \tag{12}$$

At this point, we are ready to formulate our objective as an optimization problem. The objective is to maximize the minimum energy level over all frames in a horizon, $\gamma_{\text{min}} = \min\{\gamma_{i,j} \mid \forall i \in [1, N], j \in [1, [L/S]]\}$. The variables of the problem are the compute and communicate speeds $f_{i,j}$, $d_{i,j}$, used in any frame $(i, j)$ in the horizon. Recall that by managing $f_{i,j}$ and $d_{i,j}$, we can adjust the harvested and consumed energy and hence regulate the energy levels. Thus, we will need to determine the optimal speeds $f_{i,j}$, $d_{i,j}$ for each frame $(i, j)$ in the horizon that achieve our objective. We will later show that the optimal communicate speed $d_{i,j}$ is unique for a given (entire) epoch (i.e. it does not change from frame to frame). Similarly, it will turn out that for a given epoch one needs to derive only two compute speeds (one for SC and one for SCC frames, respectively).

Our optimization problem is called Energy Management with Energy Harvesting (EMEH) and given as follows:

$$\begin{align*}
\text{Max.} & \quad \gamma_{\text{min}} \\
\text{s.t.} & \quad \forall i \in [1, N], j \in [1, [L/S]] \\
& \quad 0 < \gamma_{i,j} \leq \Gamma^{\text{max}} \tag{13} \\
& \quad t^{exe}_{i,j} \leq S \tag{14} \\
& \quad f_{\text{min}} \leq f_{i,j} \leq f_{\text{max}} \tag{15} \\
& \quad d_{\text{min}} \leq d_{i,j} \leq d_{\text{max}} \tag{16}
\end{align*}$$

The constraint (14) enforces that the energy level $\gamma_{i,j}$ in any frame is confined to the range $(0, \Gamma^{\text{max}}]$. $\gamma_{i,j} > 0$ must hold in order to ensure energy neutrality. Also, we require that $\gamma_{i,j} \leq \Gamma^{\text{max}}$ to model the energy storage capacity of the sensor node. The constraint (15) ensures the timely completion of the workload in a given frame. The constraints (16) and (17) give the lower and upper bounds for compute and communicate speeds, respectively.

Notice that the problem EMEH is essentially a non-linear program, because the frame energy level $\gamma_{i,j}$(Eq. (7)) depends on the non-linear energy consumption function, $e^c_{i,j}$ (Eq. (3)). Our strategy to solve this problem will be as follows. We will first focus on designing an energy management algorithm for any single, given epoch with known initial energy level and harvested power. Then, we show that by iteratively invoking this algorithm for each epoch we can solve the horizon-based problem EMEH optimally. We start by proposing Theorem 1 as follows.

**Theorem 1.** Starting with arbitrary initial energy level in an epoch $i$, iteratively maximizing the increment of energy level of each frame $(i,j)$. $\Delta\gamma_{i,j} = \gamma_{i,j} - \gamma_{i,j-1}, j \in [1, [L/S]]$ beginning with the first frame, maximizes the energy level at the end of any frame in epoch $i$.

The proof for this theorem is given in the Appendix. Since applying Theorem 1 maximizes $\gamma_{i,j}, \forall j \in [1, [L/S]]$ in epoch $i$, the following corollary is easily justified.

**Corollary 1.** Iteratively maximizing the energy level increment in each frame $(i,j)$. $\Delta\gamma_{i,j}$, maximizes the minimum energy level observed in any frame of epoch $i$.

Theorem 1 implies the existence of an algorithm which maximizes the energy level at the end of any epoch $i$, $\Gamma_i$, by greedily accumulating energy over each frame in epoch $i$. We refer to this optimal algorithm as DVMS.
Then, we give the following observation. Starting with the initial energy level in the horizon, i.e., $\Gamma^{\text{init}}$, the ending energy level in epoch 1, $\Gamma_1$ is maximized by invoking algorithm DVMS for epoch 1, which in turn supplies the maximum possible initial energy level for epoch 2. The same reasoning would apply to the $2^{nd}$, $3^{rd}$, ..., $N^{th}$ epochs as well, as long as the new harvesting rate is fed into the algorithm DVMS at the start of each new epoch. Therefore, we conclude that by iteratively invoking algorithm DVMS for each epoch, we can achieve the objective of problem EMEH which maximizes the minimum energy level observed in any frame of the horizon, $\gamma_{\text{min}}$. The optimality of DVMS and Corollary 1 also imply the following:

**Corollary 2.** If the algorithm DVMS cannot find a feasible solution to a specific instance of the problem EMEH, then that instance does not admit any feasible solution.

Finally, we note that the violation of the constraint $\gamma_{i,j} \leq \Gamma^{\text{max}}$ will never happen in practice, simply because the energy harvester is assumed to stop charging the storage device when the energy level approaches $\Gamma^{\text{max}}$.

### 5. Epoch-based energy management

While maximization of energy increments over consecutive frames is optimal as indicated by Theorem 1, we still need to determine the optimal compute and communicate speeds to achieve that objective. Since harvesting rate changes only from epoch to epoch, a natural strategy is to solve the problem for each epoch separately. Hence, in this section, we focus on designing the epoch-based algorithm DVMS.

As mentioned in the previous section, the basic idea of DVMS is to accumulate as much energy as possible in each frame of a given epoch. This will lead to the maximum possible stored energy at the end of the epoch. We achieve this objective by iteratively solving a Single-Frame Energy Management (SFEM) problem for each frame in the epoch.

The problem SFEM has effectively two variants. In our analysis, we consider only the solution for the SCC frame because it is the most general one; the SC type is a special case of the SCC type where we have $M = 0$. Note that since its energy consumption function is not controllable through DVS and DMS, we do not include SO frames in our analysis. Although one epoch contains $\lfloor L/S \rfloor$ frames, we claim that the above problem needs to be solved only once for the first SCC frame in each epoch, that is, the optimal compute and communicate speeds derived can be fixed for all SCC frames within the epoch. This claim is supported by the observation that the harvested power and workloads are identical for all SCC frames in one epoch. One parameter that may vary is the initial energy level for different frames in the epoch. At first, it looks like different starting energy levels may result in different frame-level compute and communicate speed assignments while trying to enforce the maximum energy level constraint $\gamma_{i,j} \leq \Gamma^{\text{max}}$. However, recall that the storage device automatically stops charging when the energy level approaches $\Gamma^{\text{max}}$, hence the maximum capacity does not need to appear as a constraint in the frame level energy management problem.

This feature yields also a benefit on the implementation side: in general, a sensor node will need to notify its receiving neighbor of every change in its modulation level (communicate speed); therefore, fixing the communication speed within each epoch makes a practical implementation possible. Finally, note that the compute speed $f$ could be different for the compute subtasks in SC and SCC frames, since voltage scaling is the only energy management tool for SC frames. The problem SFEM is specified as follows:

**Max.** $\Delta \gamma_{i,j}$

**s.t.**

\[
\begin{align*}
\nu^{\text{exe}}_{i,j} & \leq S \\
\gamma_{\text{min}} & \leq f_{i,j} \leq f_{\text{max}} \\
d_{\text{min}} & \leq d_{i,j} \leq d_{\text{max}}
\end{align*}
\]

The objective is maximizing the energy level increment in a frame, while satisfying the timing and speed constraints.

Now, we provide the solution to the frame-level problem SFEM. Since we concentrate on a single frame, the epoch and frame number $(i, j)$ are removed from all the variables. Note that $\Delta \gamma$ is equal to the difference between the harvested energy and consumed energy, i.e., $\Delta \gamma = e^h - e^c$ ($e^h$, $e^c$ are given in Eq. (10) and (3)). In our solution, we consider both concurrent and non-concurrent harvesting models separately.

(1) Concurrent harvesting model

In this case, the stored energy $e^h$ is constant (Eq. (10) and (11)). Thus, maximizing $\Delta \gamma = e^h - e^c$ is equivalent to minimizing $e^c$ which is a function of $f$ and $d$ (Eq. (3)). In this case, the problem becomes:

**Min.** $e^c = e^{\text{sen}} + e^{\text{mp}}(f) + e^{\text{cm}}(d)$

**s.t.**

\[
\begin{align*}
t^{\text{sen}} + C/f + M/d & \leq S \\
\gamma_{\text{min}} & \leq f \leq f_{\text{max}} \\
d_{\text{min}} & \leq d \leq d_{\text{max}}
\end{align*}
\]

Considering the nature of compute and communicate energy functions, the objective function can be seen to be convex. This problem has two unknowns, $f$, $d$ and is denoted as SFEM-C.

(2) Non-concurrent harvesting model

In this case, the harvesting time $t^h$ is variable with $f$ and $d$, i.e., $t^h = S - t^{\text{sen}} - (C/f) - (M/d)$ (Eq. (12)). Thus, saving energy by reducing speeds will sacrifice harvesting opportunities, and may lead to smaller $\Delta \gamma$. Hence, unlike the concurrent case, minimum energy consumption does not imply maximum energy level increment. In this case, the
problem becomes:

\[ \begin{align*}
& \text{Max.} \quad e^h - e^c = t^h p^h - [e^{sen} + e^{cp}(f) + e^{cm}(d)] \\
& \text{s.t.} \quad t^{sen} + c/f + M/d + t^h = S \\
& \quad 0 \leq t^h \leq S \\
& \quad f_{min} \leq f \leq f_{max} \\
& \quad d_{min} \leq d \leq d_{max}
\end{align*} \]

This problem has a concave objective, three unknowns, \( f, d, t^h \). Since maximizing a concave objective function \( h() \) is equivalent to minimizing a convex function \( -h() \), this problem leads to a convex program as well. We denote this problem as \( SFEM-N \).

### 5.1. Solution to frame-level energy management

In order to solve \( SFEM-C \), we temporarily ignore constraint (19). By ignoring it, \( f \) and \( d \) can be scaled arbitrarily within their ranges. Thus, the overall energy \( e^c \) is minimized by minimizing the CPU energy \( e^{cp} \) and radio energy \( e^{cm} \) separately. The speeds \( f^*, d^* \) which minimize \( e^{cp} \) and \( e^{cm} \) can be found by setting the first derivatives of \( e^{cp} \) and \( e^{cm} \) to zero.

Now, we take the constraint (19) into consideration. If \( f^*, d^* \) satisfy constraint (19), we consider two special cases:

- If \( f_{min} \leq f^* \leq f_{max} \), \( d_{min} \leq d^* \leq d_{max} \), then the optimal solution is \( f^{opt} = f^*, d^{opt} = d^* \).
- If \( f^* < f_{min} \) and/or \( d^* < d_{min} \), then \( f^{opt} = f_{min} \) and/or \( d^{opt} = d_{min} \). This is because \( e^{cp} \) and \( e^{cm} \) are monotonically-increasing in \( f \in [f^*, +\infty] \) and \( d \in [d^*, +\infty] \).

In [15], Aydin et al. derived \( f^* \) under an equivalent energy model and referred it as the energy-efficient compute frequency. Also, \( d^* \) can be called the energy-efficient communicate speed. If \( f^*, d^* \) violate constraint (19), the solution is more complex. We note that problem \( SFEM-C \) can be rewritten as:

\[ \begin{align*}
& \text{Min.} \quad e^{sen} + e^{cp}(t^{cp}) + e^{cm}(t^{cm}) \\
& \text{s.t.} \quad t^{sen} + t^{cp} + t^{cm} \leq S \\
& \quad C/f_{max} \leq t^{cp} \leq C/f_{min} \\
& \quad M/d_{max} \leq t^{cm} \leq M/d_{min}
\end{align*} \]

We use the compute and communicate time, \( t^{cp} = C/f \) and \( t^{cm} = M/d \) as the new variables. This yields a separable optimization problem in the form of:

\[ \begin{align*}
& \text{Min.} \quad \sum_{k=1}^{n} F_k(x_k) \\
& \text{s.t.} \quad \sum_{k=1}^{n} x_k \leq S \\
& \forall k, x_{k,min} \leq x_k \leq x_{k,max}
\end{align*} \]

where \( n \) is the number of variables. In [15] and [16], it is shown that any problem with the above structure can be solved in time \( O(n^3) \) by manipulating the Kuhn-Tucker conditions ([17]). Since in problem \( SFEM-C \), \( n = 2 \), it can be solved in constant time. The same method can be used to solve problem \( SFEM-N \) which is also a separable problem. Due to space limits, we omit the solution to \( SFEM-N \).

In this problem, the derived \( f^{opt} \) and \( d^{opt} \) values are then used to compute the optimal speeds \( f^{opt}, d^{opt} \). Finally, \( f^{opt} \) and \( d^{opt} \) might not be available on the target hardware with discrete speed levels. However, we can use the lowest \( f \) and \( d \) which satisfy \( f \leq f^{opt} \), \( d \leq d^{opt} \), to guarantee the timely completion of the workload.

### 6. Performance evaluation

Though we have demonstrated the optimality of algorithm \( DVMS \) for energy level maximization, we ran a set of experiments to determine the actual improvement in stored energy and \( \gamma_{min} \), compared to the schemes that use either no or one of the voltage and modulation scaling techniques. The simulations are conducted using TOSSIM, the widely-used WSN simulator. In addition to the normal workload conditions where the worst-case application demand is constant, we also considered an emergency mode where there are sudden, unexpected peaks in the demand. The emergency mode is introduced to assess the schemes capacity to cope with runtime uncertainties and minimize service interruptions.

We consider two types of workloads, normal and emergency. The normal compute and communicate workloads are generated randomly according to uniform distribution within the ranges \( C = [1, 2000000] \) CPU cycles, and \( M = [1, 128] \) bytes. The emergency mode is simulated by increasing the workload of frames by \( u \) times, where \( u \) ranges in \([1, 2]\). We assume that the sensor node encountered \( v \) emergencies in the horizon, each of which lasts \( w \) consecutive epochs. \( v \) and \( w \) are both random integers in the range of \([0, 10]\). Our simulations used SCC frames.

Our basic node consists of a DVS-enabled CPU, a DMS-enabled radio, a sensor and an energy harvesting unit. We assumed that the CPU is the Intel Xscale PXA27x CPU ([18]) which is used on widely available iMote-2 sensor node. The specification of the PXA27x processor is given in Table (I). The CPU power consumption given in the

<table>
<thead>
<tr>
<th>Freq.(MHz)</th>
<th>Power(mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>104</td>
<td>116</td>
</tr>
<tr>
<td>208</td>
<td>279</td>
</tr>
<tr>
<td>312</td>
<td>790</td>
</tr>
<tr>
<td>416</td>
<td>7570</td>
</tr>
<tr>
<td>520</td>
<td>747</td>
</tr>
<tr>
<td>624</td>
<td>925</td>
</tr>
</tbody>
</table>

Table 1. Specification of Intel Xscale Pxa27x

The table is the overall power including both the frequency dependent and independent parts. We assume that the DMS-enabled radio has 4 modulation levels: \( b = \{2, 4, 6, 8\} \). The radio symbol rate is \( R = 62.5k \) symbols/sec. According to Eq. (1), the available communicate speeds are \( d = \{125, 250, 375, 500\} \) (kbps). The radio energy function is in the form of Eq. (5). Without loss of generality, based on [19] and Eq. (5), we derive the radio speed-independent power as \( p_{cm,ind} = 26.5mW \), and \( \beta = 2.74 \times 10^{-8} \). Again, without loss of generality we assumed a light sensor.
TSL2561 with sense time of 12ms for each reading and power usage of 0.72mW ([20]). We assume the harvested energy is obtained from solar radiation, and use the solar power harvesting trace over one day provided in [10] as our harvesting profile. Solar energy can be harvested in either concurrent or non-concurrent mode. We use the results in [10] to fix the harvesting cycle at $H = 24$ hours. This horizon is then divided into 96 epochs, each has a length $L = 15$ mins. We simulated the execution of each invoked frame and set the frame period at $S = 30$ ms. We assume a rechargeable battery storing at most $4000$ Joules energy. The initial energy level is $\Gamma^{\text{init}} = 2400$ Joules (60% full).

Although there are no existing schemes that are directly comparable to our approach, we have defined three new baseline schemes for comparison purposes. First, the $NPM$ (No-Power-Management) scheme fixes both frequency and modulation level at the maximum across all epochs. Second, the $DVS$ scheme scales only the frequency optimally, while fixing the modulation level at its maximum level. Third, the $DMS$ scheme scales the modulation level optimally, while fixing the frequency at the maximum level. We use the metric frame skip ratio to measure the schemes capacity to cope with uncertainties, defined as the percentage of failed frames (missed deadline) due to empty energy storage in the horizon. Notice that this ratio also captures the scope of service interruption time: the higher the frame skip ratio, the longer the service interruption time.

For each of the experiments below, we ran 96 full epochs 1000 times. We then computed the average stored energy at the end of each epoch and plotted it as a data point. In Fig. (2), we compare different schemes in stored energy of a sensor node executing normal workload, while harvesting concurrently. Among all the schemes, the energy level increases in the daytime as the sunlight intensity increases, and decreases in the evening due to the absence of sunlight. In all schemes, the $DVMS$ achieves the highest energy level. At the $\gamma_{\text{min}}$ point (that appears around 8:00am), the $DVMS$ stores 1800 joules (45.0% full) which is significantly higher than 1100 joules (27.5% full) for $NPM$, 1400 joules (35% full) for both $DVS$ and $DMS$. As opposed to $DVS$ or $DMS$ schemes, the $DVMS$ stores more energy since it has wider power scaling range, and always selects the most energy efficient speeds which yield the smallest energy consumption. All schemes have zero frame skip ratio which means no service interruption occurred.

In Fig. (3), we run the same experiment, but assuming non-concurrent harvesting. Again, the $DVMS$ stores more energy than all other schemes. The $\gamma_{\text{min}}$ point (appeared at midnight) is about 1200 joules (30.0% full) for $DVMS$ which is higher than 0 joule (empty) for $NPM$, 300 joules (7.5% full) for $DVS$, and 800 joules (20.0% full) for $DMS$. The $DVMS$ stores the most energy as it uses the speeds which optimally balancing energy consumption and harvesting. Only $NPM$ experiences a non-zero (2.1%) frame skip ratio.

In Fig. (4) and (5), we compare different schemes for a node executing emergency workloads, while harvesting concurrently and non-concurrently. The stored energy by all schemes decreases significantly compared to Fig. (2) and (3) due to the extra energy used by emergency workloads. The $DVMS$ outperforms all other schemes again in terms of stored energy and causes no service interruption. In Fig. (4), no schemes experiences service interruption, while in Fig. (5) the $DVS$ and $NPM$ have skip ratio of 4.2% and 10.4%.

In all the above figures, the $DVMS$ scheme stores significantly more energies than all other schemes, and never experiences service interruption. Under emergency mode, some schemes run out of energy in the middle of operation for up to 10% time of service which may have very serious consequences for mission-critical CPS applications. Our experiments indicate the benefits of our algorithm in terms of both stored energy and resilience to system uncertainties.

7. Conclusion

This paper presented an epoch-based approach for energy management in WSN-enabled CPS applications utilizing energy harvesting combined with DVS and DMS. We formalized this goal as the Energy Management with Energy...
Harvesting problem, and then derived an optimal algorithm to solve it. We presented a series of performance evaluation experiments. The results demonstrated that our optimal algorithm DVMSS significantly improve energy storage compared to other baseline approaches. We also show that the harvesting architecture (concurrent vs. non-concurrent) has direct impact on energy management policies and must therefore influence how designers engineer harvesting systems.

Appendix

Proof of Theorem 1

Let $\gamma_{i,j}^C$ and $\gamma_{i,j}^A$ denote the ending energy levels in frame $(i,j)$, obtained by iteratively maximizing energy level increment of each frame, and that obtained by an arbitrary scheme $A$, respectively. Similarly, we denote $\Delta \gamma_{i,j}^C$ and $\Delta \gamma_{i,j}^A$ as the energy level increments in frame $(i,j)$, obtained by our iterative scheme and scheme $A$. The initial energy level in epoch $i$ is denoted as $\Gamma_{i,0}^{init} = \Gamma_{i-1}$. We will prove the theorem by induction over the frame sequence number, $j$.

Base case: If $j = 1$, we have $\gamma_{i,j}^C = \Gamma_{i,0}^{init} + \Delta \gamma_{i,1}^C$, and $\gamma_{i,j}^A = \Gamma_{i,0}^{init} + \Delta \gamma_{i,1}^A$. Since $\Delta \gamma_{i,1}^C \geq \Delta \gamma_{i,1}^A$, by definition $\gamma_{i,1}^C \geq \gamma_{i,1}^A$ is justified.

Now, suppose the statement holds for $j = 1, 2 \ldots n - 1$ frames in epoch $i$. Based on our induction assumption, we have $\gamma_{i,n-1}^C \geq \gamma_{i,n-1}^A$. We claim that Theorem 1 also holds in frame $(i,n)$, i.e., $\gamma_{i,n}^C \geq \gamma_{i,n}^A$. We distinguish two cases:

- $\gamma_{i,n}^C < \Gamma_{i,n}^{max}$: the energy level reaches $\Gamma_{i,n}^{max}$ at the end of frame $(i,n)$. Since the energy level achieved by any scheme $A$ cannot exceed $\Gamma_{i,n}^{max}$, our claim holds.

- $\gamma_{i,n}^C \geq \Gamma_{i,n}^{max}$: Note that if $\gamma_{i,n}^C \geq \Gamma_{i,n}^{max}$, the optimal energy increment obtained by our scheme (denoted by $G$) after considering the constant harvested power and worst-case workload in frame $(i,n)$ is not constrained by the maximum capacity constraint (otherwise $\gamma_{i,n}^C$ would be equal to $\Gamma_{i,n}^{max}$). This enables us to deduce that $\Delta \gamma_{i,n}^C \geq \Delta \gamma_{i,n}^A$, since regardless of the initial energy level at frame $(i,n)$, $G$ by definition accumulates energy which is at least equal to that yielded by any other scheme $A$ during frame $(i,n)$, as long as the constraint $\gamma_{i,n}^C < \Gamma_{i,n}^{max}$ is not violated.

Also recall that by induction assumption, we have $\gamma_{i,n-1}^C \geq \gamma_{i,n-1}^A$. Therefore, we have $\gamma_{i,n}^C = \gamma_{i,n-1}^C + \Delta \gamma_{i,n}^C \geq \gamma_{i,n-1}^A + \Delta \gamma_{i,n}^C = \gamma_{i,n}^A$. Thus, our claim holds for this case as well, proving Theorem 1.

References


Real-Time Modeling and Control of a Cyber-Physical Energy System

Tullio Facchinetti  
Computer Engineering and Systems Science  
University of Pavia  
Pavia, Italy  
Email: tullio.facchinetti@unipv.it

Marco L. Della Vedova  
Computer Engineering and Systems Science  
University of Pavia  
Pavia, Italy  
Email: marco.dellavedova@unipv.it

Abstract—This paper introduces an approach for applying real-time scheduling techniques to balance electric loads in cyber-physical energy systems. The proposed methodology aims to determine, guarantee and optimize an upper bound on the peak load of electric power, which represents a desirable feature for both the electricity supplier and the user of the electrical system. For this purpose, networked electric devices are modeled using parameters derived from the real-time scheduling discipline used for computing systems. Therefore, the upper bound can be enforced by predictably and timely switching on/off the electric devices composing the electrical system.

The paper contribution include: the illustration of the relevance of electric load balancing in cyber-physical energy systems, motivating the use of real-time scheduling techniques to achieve predictability of electric loads scheduling; the presentation of a novel and powerful modeling methodology of the physical system based on a set of periodically activated loads, to enable the use of traditional real-time system models and scheduling algorithms, with adequate adaptations, to manage loads activation/deactivation. We finally derive interesting properties of real-time parameters and provide theoretical results concerning the computation of their values.

Keywords—Cyber-Physical Energy Systems; Load Balancing; Real-Time; Modeling; Peak Load

I. INTRODUCTION

The use of embedded systems to automatically monitor and control networks of electric devices is growing in interest in both home and industrial domains. In industrial applications, embedded systems have been traditionally employed to automate industrial processes. In many applications, real-time constraints arise from the physical process and must be guaranteed by computing tasks in order to meet the application requirements. Conversely, in building automation (domotics) the integration of embedded devices with domestic appliances is bringing to the possibility of implementing a large amount of new features and to advance the performance of existing ones, including multimedia, security, and energy efficiency.

In recent years, the research on embedded systems is moving towards the integration of computational resources within the physical system under monitoring and control, leading to the so-called Cyber-Physical Systems (CPSs), i.e., complex networks of interconnected embedded devices tightly integrated with the physical process under control. Key issues in CPSs are sensing and actuation, the modeling of the physical system, real-time computing, and networking. Challenging issues arise from the dynamic nature of the underlying physical process, that requires the ability of working under uncertain environmental conditions, and involve complex relationships among a high number of system components. Example applications for CPSs are in the field of manufacturing control, energy systems, automotive and avionics systems, traffic control, medical systems, cooperative robotics and smart buildings. Some detailed examples of CPSs can be found in [1].

On the other hand, home and industrial automation systems more and more often require to address the issue of energy efficiency. For this purpose, the research on CPSs has been extended to the study of Cyber-Physical Energy Systems (CPESs) [2]. In those systems, embedded computing is integrated within the electrical system to gather information about the most important electric parameters, such as voltage, current, phases, consumed energy and power. Environmental parameters, as temperature, humidity and pressure, are also relevant for the system characterization. The acquired data are then combined and processed to generate suitable control commands for the electric devices, in order to achieve the desired application goal. And such goals include, or introduce constraints, on power and energy usage.

In cyber-physical energy systems, as addressed in this paper, the physical process is composed by a set of electric devices that are monitored and suitably operated by a set of networked embedded systems. A relevant example of energy system automation is represented by the so-called smart grid [3]. Smart grids focus on the interaction between energy supply and usage, in which a two-way flow of electricity between energy providers and users is supported by a pervasive, distributed and interconnected information infrastructure. A fundamental role is played by smart meters for the intelligent monitoring of buildings, districts and town energy usage [4]. Renewable source of energy are also an important part of a smart grid, as well as distributed power generation systems by means of micro-cogeneration systems (or Combined Heat and Power, CHP, systems).

The availability of compact and flexible embedded sys-
tems allows the effective implementation of cyber-physical energy systems. Monitoring tasks and control actions can be applied on devices composing the considered physical system. Moreover, the involved embedded systems can be connected to build large distributed control networks.

This paper focuses on cyber-physical energy systems dedicated to electric power management, and particular attention is devoted to the balancing of power usage, which represents an important issue in electrical systems [5]. Balancing the use of power aims at avoiding dangerous peak load conditions, i.e., when too many user devices are simultaneously active, with the risk of overloads on the power distribution infrastructure leading to possible blackouts. The main goals of the proposed approach is to guarantee that the peak power demand remains under a given threshold, and to determine a smoother and flatter curve of power usage over time.

The proper management of peak load conditions is desirable by both consumers and energy providers [6]. The supplier can achieve a better balance over its distribution infrastructure and a tighter design of the system (e.g., optimizing the size of cables). For example, energy providers have the possibility to turn off the least efficient (i.e. the most expensive) power plants while achieving their contractual provisions; moreover, they can build the electric distribution infrastructure for tolerating well defined (and possibly, reduced) peak loads, with limited technical issues and reduced costs. On the other hand, consumers can negotiate better pricing conditions if they can guarantee an upper bound on power usage, considering that pricing strategies are often driven by policies such as peak-load pricing [7]. Additionally, energy management can be integrated with smart metering devices to increase the awareness of users about energy issues generated by their habits [8] which, in turn, can drive the behavioral change widely recognized as an important component of energy saving strategies.

The methodology proposed in this paper aims to enforce the guarantee on the peak load of power usage by adequately control the available set of electrical devices composing the physical system in a cyber-physical energy system. More precisely, the idea is to schedule the activation of devices in a timely and predictable manner, in order to limit the concurrent activation of loads, thus balancing the total consumed electric power and reducing the peak load of power usage. Moreover, it provides the possibility to determine the value of the peak load in worst case conditions; such information is important to realize a tight design of the energy distribution system (e.g., the size of cables), avoiding undesired extra-costs.

The physical system will be properly modeled to apply scheduling algorithms suitably derived from the real-time scheduling discipline which is currently developed for computing tasks executed on a microprocessor (see [9] for an introduction and a comprehensive description of hard real-time systems). One inherent benefit of this approach is to take advantage of the strong mathematical background which characterizes the results of real-time scheduling analysis. Moreover, the powerful analysis techniques developed over more than three decades of research on real-time systems will be leveraged to characterize timing and energy properties of the physical system, while facing the problem of dealing with large and complex systems, with several types of constraints, which is a typical scenario when cyber-physical energy systems are considered.

This innovative approach to electric load management opens the door to the application of sophisticated scheduling algorithms to meet power, energy and timing constraints in cyber-physical energy systems.

A. Paper organization

The paper is organized as follows: Section II illustrates the approach to model an electrical system as a set of periodically activated loads, and in Section III some relevant related works are recalled and commented. Section IV introduces the system model adopted in this work, while an example of physical process that can be represented with such a model is shown in Section VI. Section V discusses how to associate suitable real-time parameters to the physical process, and derives some interesting properties. The obtained theoretical results are applied to an example test case in Section VII. Finally, Section VIII states our conclusions and provides a sketch of the several possible enhancements to this work.

II. TOWARDS MODELING ELECTRIC LOADS USING REAL-TIME PARAMETERS

The main contribution of this paper is to propose an approach to model electric devices using real-time parameters. For this purpose, we establish an analogy between real-time computing systems and electrical systems, which represent the physical background of cyber-physical energy systems.

Real-time computing systems are used to allow the concurrent execution of processing tasks subject to timing constraints on a processor. However, in more general terms, the real-time scheduling problem can be defined as the problem of allocating resources over time to a set of time-consuming tasks, while meeting a given set of timing constraints. The key observation is that, under this definition, resources may not necessarily be processors or processing devices, as they are usually intended in computing systems.

In the last few years, the use of real-time scheduling techniques has been extrapolated from the field of computing system to be used in different application domains. In communication systems, real-time algorithms are applied to schedule and analyse the performance of messages sets over a communication channel (e.g., [10]). In this case, an analogy is made between computing and communication systems, where messages are made equal to computing tasks. The real-time scheduling is thus performed, respectively, on the communication channel and processors. In processing
systems, timing constraints need to be guaranteed on the execution times, while they must be achieved on message’s end-to-end latency in communication systems. This means that a real-time task must be guaranteed to terminate its execution before its deadline, while a message must be delivered to the receiver within the given time limit. The analogy between computing and communication systems allows to extend interesting results from one domain to the other, and vice-versa. An example of technique originally proposed for the modeling and the analysis of communication networks that has been successfully adapted to real-time computing systems is the network calculus, a theoretical framework that allows the analysis of information flow in computer networks subject to several kind of constraints [11]. Such adaptation led to the so-called real-time calculus [12]).

The above considerations foster the possibility of using real-time scheduling techniques to model, to analyse and to manage technological systems that would present a sufficient degree of affinity. Following this approach, the proposed methodology establishes an analogy between electric loads and computing tasks subject to real-time constraints.

In this paper, we borrow the well-known periodic task model [13] widely studied in real-time systems to represent electric loads as periodically triggered activities. A bound is imposed on the total amount of time that a load can stay active in each period. As for computing tasks, all the time properties of electric loads (periods, deadlines and activation time) must be selected according to their application requirements. The activation time plays the role of task’s WCET (Worst Case Execution Time) in real-time computing systems. We assume that a load consumes a given amount of electric power while active, and no power when switched off. Based on such a model, a priority-based scheduling algorithm can be applied to selectively activate/deactivate devices. The goal is to meet the timing constraints of each load, while guaranteeing an upper bound on the total instantaneous power consumed by the concurrent activation of electric devices.

III. RELATED WORKS

Power-aware scheduling techniques represent an active research topic in the real-time systems literature. The typical approach behind such techniques is to exploit some dedicated features of modern electronic components (microprocessors, motherboards, etc.) to reduce the total amount of energy consumption. For example, the Dynamic Voltage Scaling (DVS) [14] technique, i.e., the possibility of dynamically changing the power supply voltage of a microprocessor, is leveraged to reduce the energy consumption of the processing unit. Since reducing the supply voltage brings to a decrease of the clock speed, the analysis focuses on the guarantee of real-time constraints when processor’s speed is allowed to change. However, the goal of those techniques is to reduce the total use of energy, while our objective is to reduce the peak load of electric power.

Other approaches are more related with energy systems. In [15], the authors aim to find optimal schedules for microCHP (Combined Heat and Power) systems using a global optimization technique based on an Integer Linear Programming formulation of the problem. This is essentially an off-line approach, while on-line scheduling is addressed in this paper. Some recent works are concentrating on the real-time issues related with batteries charge/discharge, for example in electric vehicles [16].

In [17] the authors tackle the problem of scheduling tasks on a system powered by the energy generated from renewable sources. The goal is to produce a suitable schedule to maintain the battery energy within a predefined range. A dedicated algorithm is proposed, that uses a closed loop approach to track the change in the available energy and to adapt the schedule accordingly. Our approach, instead, uses an open loop technique to obtain the same goal, despite we do not restrict our target application to batteries. It is worth to note that closed-loop techniques are inherently more robust to noise and errors. Actually, we are planning to suitably integrate feedback techniques on top of our model as future work.

IV. SYSTEM MODEL

The considered system is composed by a set $\Lambda = \{\lambda_1, \cdots, \lambda_n\}$ of $n$ loads that need to be periodically turned on and off (or activated/deactivated), depending on their specific timing constraints. A load is said to be active when it is turned on, inactive otherwise. The load activity is controlled by a load scheduler that decides when each load is activated/deactivated. The activation of each load is independent of other loads (i.e., no precedence or other kind of constraints among loads are considered). Formally, the scheduler assigns to each load $\lambda_i$ a schedule that is modeled by the function $s_i(t)$:

$$s_i(t) = \begin{cases} 1 & \lambda_i \text{ is active at } t \\ 0 & \text{otherwise} \end{cases}$$

(1)

The schedule of all loads is then given by $s(t) = \{s_1(t), \ldots, s_n(t)\}$. The schedule will be denoted with $\mathcal{S}$ when the dependency from the time is not relevant.

The load $\lambda_i$ consumes a $p_i(t)$ amount of electric power when active at time $t$, no power otherwise. More formally, it holds

$$p_i(t) = \begin{cases} P_i & \text{if } s_i(t) = 1 \\ 0 & \text{if } s_i(t) = 0 \end{cases}$$

In this paper, we focus the goal of our approach on the control loads characterized by time-varying state variable $x_i(t)$. The notation $x(t) = (x_1(t), \cdots, x_n(t))$ will be used to denote the state vector representing all loads. The state
vector value varies over time with the law specified by Equation 2.

$$\dot{x}(t) = \rho(t)$$
$$x(0) = \overline{\pi}$$

(2)

where

$$\rho(t) = \rho^\text{in} - \rho^\text{out} s(t)$$

(3)

$$0 < \rho^\text{in} < \rho^\text{out}$$

(4)

In other words, each state variable $x_i$ linearly increases with a slope defined by $\rho^\text{in}_i$ when $\lambda_i$ is inactive (i.e., $s_i = 0$), while it linearly decreases with slope defined by $\rho^\text{in}_i - \rho^\text{out}_i$, which is lower than 0 due to inequality 4, when $\lambda_i$ is active. Notice that the choice of associating a decreasing state variable with an active load and viceversa does not affect the generality of the problem statement and its solution.

A. Real-time modeling

Considering the parameters used to model a traditional real-time computing task, we use the tuple $\{T_i, C_i, P_i\}$ to define a load $\lambda_i$, where

- $T_i$ is the time frame between two consecutive activations (as in the periodic task model for real-time computing tasks [13]);
- $C_i$ ($\leq T_i$) represents the activation time duration of $\lambda_i$ within each period $T_i$;
- $P_i$ is the nominal power associated to the activation of $\lambda_i$, as previously stated.

The utilization of $\lambda_i$ is defined as

$$U_i = C_i / T_i$$

(5)

while the total utilization of the load set is $U = \sum_{i=1}^n U_i$.

The $k$-th request for activating the load $\lambda_i$ happens at time $r_{i,k}$ (request time), where $r_{i,k} = kT_i$, $k \in \mathbb{N}$.

**Definition 1:** A schedule $\mathcal{S}$ is said to be valid if it assigns to each load $\lambda_i$ an amount of activity time equal to $C_i$ between two consecutive request times. Formally,

$$\forall \lambda_i, \forall k \int_{r_{i,k}}^{r_{i,k+1}} s_i(t) \, dt = C_i$$

(6)

A valid schedule can be generated by a real-time scheduling algorithm as Earliest Deadline First (EDF) or Rate Monotonic (RM) [13] when applied to a feasible set of loads, i.e., the specific schedulability test, applied to the given load set, is passed for the considered algorithm. Since we are considering implicit deadlines, i.e., deadlines equal to periods, utilization-based schedulability tests can be used for both algorithms.

**Definition 2:** The overall instantaneous power consumption $p(t)$ is defined as

$$p(t) = \sum_{i=1}^n p_i(t).$$

(7)

**Definition 3:** The peak load $P$ of a set of loads is defined as the maximum instantaneous power consumption over the system lifetime:

$$P = \max_{t \geq 0} p(t).$$

(8)

B. Problem statement

Given the system model and the possibility of adequately modeling the involved electric loads using real-time parameters, we are interested to determine the relationship between the physical system parameters (i.e., $\rho^\text{in}$, $\rho^\text{out}$, and $\overline{\pi}$) and real-time parameters (basically, $T_i$ and $C_i$) such that the overall peak load $P$ is minimized while meeting two constraints: i) the scheduler will produce a valid schedule (see Definition 1); ii) the instantaneous state variable value $x_i(t)$ of each load $\lambda_i$ is bounded in the range $[x_i^{\text{min}}, x_i^{\text{max}}]$. More formally,

minimize $P$

such that

$$\forall t, x_i^{\text{min}} \leq x_i(t) \leq x_i^{\text{max}}$$

(9)

In the remainder of this paper, we will also use the compact vector notation

$$x^{\text{min}} = (x_1^{\text{min}}, \ldots, x_n^{\text{min}})$$

and a similar notation will be used to indicate $x^{\text{max}}$.

V. PROPERTIES AND RESULTS OF PHYSICAL AND REAL-TIME PARAMETERS

In this section we determine some interesting properties and introduce relevant results regarding the calculation of real-time parameters of the system model presented in Section IV.

First, we establish a relationship between the load utilization $U_i$ and the dynamical properties of the related physical process. This relationship determine an useful property of the state variable itself.

**Theorem 1:** If the utilization $U_i$ of task $\lambda_i$ is set as

$$U_i = \frac{\rho^\text{in}_i}{\rho^\text{out}_i}$$

(10)

then the state variable assumes the same value $\overline{x_i}$ at every request time $r_{i,k}$, i.e.,

$$\forall k \in \mathbb{N} : k \geq 0 \rightarrow x_i(kT_i) = \overline{x_i}.$$  

(11)

**Proof:** We start by integrating Equation 2 to obtain the state variable value at the generic $k$-th request time $r_{i,k}$:

$$x_i(kT_i) = x_i(0) + \int_0^{kT_i} \rho_i(t) \, dt$$

(12)

Considering the definition of $\rho(t)$ (Equation 3), Equation 12 can be rewritten as
\[ x_i(kT_i) = \overline{x}_i + \rho_i^{\text{in}} \int_0^{kT_i} \, dt - \rho_i^{\text{out}} \int_0^{kT_i} \, s_i(t) \, dt \quad (13) \]

While the value of the first integral is trivial, the second integral can be easily evaluated by considering the definition of valid schedule (Equation 6) and the definition of task utilization \( U_i \):

\[ x_i(kT_i) = \overline{x}_i + \rho_i^{\text{in}} kT_i - \rho_i^{\text{out}} U_i kT_i = \overline{x}_i + kT_i (\rho_i^{\text{in}} - \rho_i^{\text{out}} U_i) \quad (14) \]

Finally, introducing the term \( U_i \) from Equation 10 into Equation 14, it follows

\[ x_i(kT_i) = \overline{x}_i \]

which proves the theorem.

Theorem 1 states that, to achieve the result specified by Equation 11, the load utilization \( U_i \) depends only on \( \rho_i^{\text{in}} \) and \( \rho_i^{\text{out}} \). In particular, it does not depend on the state variable range bounds. Moreover, since a state variable assumes the same value at every request time, we are allowed to derive global properties of state variable behaviour by analyzing such behaviour within the time frame delimited by one period.

Since we are interested to bound the state variable variation within a specified range, we introduce the definition of largest variation with respect to \( \overline{x}_i \):

**Definition 4:** We define the largest ascending and descending variations of \( x_i(t) \) with respect to \( \overline{x}_i \), respectively, as follows:

\[ \Delta_i^{\text{inc}} \equiv \max_t x_i(t) - \overline{x}_i \]

\[ \Delta_i^{\text{dec}} \equiv \overline{x}_i - \min_t x_i(t) \quad (16) \]

Notice that definitions (15) and (16) are calculated for every \( t \), and thus represent a global behaviour.

We now determine the properties of state variable variations within one time period.

**Lemma 1:** The largest possible ascending variation of the state variable \( x_i(t) \) with respect to \( \overline{x}_i \) on a period is

\[ \delta_i^{\text{inc}} = \rho_i^{\text{in}} (T_i - C_i) \quad (17) \]

**Proof:**
Let us define \( \hat{t} \) as the time instant after which the state variable \( x_i(t) \) can only decrease, i.e.,

\[ \forall t : \hat{t} < t \leq T_i \rightarrow x_i(t) \leq x_i(\hat{t}) \]

Therefore, the maximum value of \( x_i(t) \) must correspond to a time instant \( T^* \) such as \( 0 < T^* \leq \hat{t} \). The value of \( x_i(\hat{t}) \) can be calculated by integrating Equation 2, obtaining

\[ x_i(\hat{t}) = x_i(0) + \int_0^{\hat{t}} \rho_i(t) \, dt \quad (18) \]

which can be rewritten as

\[ x_i(\hat{t}) = \overline{x}_i + \rho_i^{\text{in}} \int_0^{\hat{t}} \, dt - \rho_i^{\text{out}} \int_0^{\hat{t}} \, s_i(t) \, dt \quad (19) \]

The first integral corresponds to the amount of time that \( x_i(t) \) increases in the range \( [0, \hat{t}] \), which is equal to \( T_i - C_i \). In fact, by definition of \( t \), the range \( [0, \hat{t}] \) contains all the amount of time that the state variable has a negative derivative and, since \( C_i \) is the amount of time that the state variable has a positive derivative in the whole range \( [0, T_i] \), than the amount of time that the state variable derivative is negative equals \( T_i - C_i \). Therefore

\[ x_i(\hat{t}) = \overline{x}_i + \rho_i^{\text{in}} (T_i - C_i) - \rho_i^{\text{out}} \int_0^{\hat{t}} \, s_i(t) \, dt \quad (20) \]

The proof concludes by noticing that Equation 20 is composed by two constants and a negative term (the value of the integral). Therefore, the maximum value of \( x_i(\hat{t}) \) holds when the negative term is equal to zero. In other words, it holds

\[ \forall t : 0 < t \leq \hat{t} \rightarrow s_i(t) = 0 \] involving \( \hat{t} = T_i - C_i \). □

Lemma 1 states that largest increasing variation of \( x_i(t) \) with respect to \( \overline{x}_i \) takes place when the state variable behaves as in Figure 1 (b). A similar result can be obtained for a decreasing variation, as stated in Lemma 2.

**Lemma 2:** The largest possible descending variation of the state variable \( x_i(t) \) with respect to \( \overline{x}_i \) is

\[ \delta_i^{\text{dec}} = (\rho_i^{\text{out}} - \rho_i^{\text{in}}) C_i \quad (21) \]

**Proof:** The proof can be carried out as for Lemma 1. □
The behaviour local to one time period is related to global largest variations by Lemma 3.

Lemma 3: If $U_i$ is assigned as in Equation 10, then:

$$\Delta_i^{inc} = \delta_i^{inc} \quad \text{and} \quad \Delta_i^{dec} = \delta_i^{dec}$$

Moreover, quantities in the Equation 22 are equal, so it can be defined:

$$\Delta_i \equiv \Delta^{inc} = \Delta^{dec}$$

Proof: Equation 22 is valid since, by Theorem 1, the initial condition is the same on each period. Therefore, results in Lemma 1 and Lemma 2, that are obtained for a generic period, have global validity over all the time $t$.

Equation 23 can be obtained by substitutions that involve Equations 5, 17, 21 and 22. •

Theorem 2 allows to calculate the upper bound on the period $T_i$ for a load $\lambda_i$ such that, if used together with the load utilization $U_i$ as in Equation 10, it guarantees that load $\lambda_i$ will maintain its state variable $x_i(t)$ within the required range $[x_i^{min}, x_i^{max}]$.

Theorem 2: If the period $T_i$ is chosen in the interval $(0, T_i^*)$, where $T_i^*$ is defined in (24),

$$T_i^* = \min \left\{ \Delta_i^{inc}, \Delta_i^{dec} \right\}$$

and $U_i$ is assigned as in Equation 10, then

$$\forall t \in \mathbb{R} : t \geq 0 \Rightarrow x_i^{min} \leq x_i(t) \leq x_i^{max}$$

Proof: Considering Lemma 3 and $\pi_i = x(0)$ as initial value, in order to keep the state variable into bounds, as stated in Equation 9, must be imposed:

$$\begin{cases} \pi + \Delta_i \leq x^{max} \\ \pi - \Delta_i \geq x^{min} \end{cases}$$

From Equations 5, 17, 23 and 25, it can be obtained by substitutions:

$$\begin{align} T_i & \leq \frac{\pi^{max} - \pi}{\rho_i^{m}(1 - U_i)} \\ T_i & \leq \frac{\pi^{min} - \pi}{\rho_i^{m}(1 - U_i)} \end{align}$$

Since both inequalities must hold, $T_i$ is upper bounded by the minor of the two quantities in Equation 26, and (24) follows. •

Since $T_i^*$ represents an upper limit on the range where the load period could be selected, a shorter period could also be preferred, if needed, that still achieves the requirements on the state variable variation. It is worth to discuss the implications of such a possible choice. Shorter periods correspond to shorter distances in time between two consecutive request times. Therefore, shorter periods determine a more frequent activation of a load within a shorter time frame. This observation holds in general when the load is considered alone, i.e., it is not affected/preempted by the activations of other loads. In fact, in presence of more than one load, preemptions may generate a similar effect, although in this case such behaviour does not emerge from the timing characteristics of a given load but arises from the interaction among load’s activations. The effect is to narrow down the state variable variation range around $\pi_i$. Although in general this behaviour may be considered as a desirable feature, a side effect needs to be taken into account, which is related with the characteristics of the physical process under control. Some types of loads, as high power electric motors, do not well tolerate sequences of activation/deactivation which are too close each other, since this may have a negative impact on the actuator’s lifetime. Therefore, a larger state variable variation range (once the state variable is guaranteed to remain within the allowed range) can achieve a longer system lifetime.

VI. EXAMPLE OF PHYSICAL SYSTEM

Given the system model and notation introduced in Section IV, in this section we provide an example of physical system having suitable characteristics to be represented using the proposed model (Figure 2). A vessel receives as input an amount of fluid with a constant flood capacity $Q^{in}$ (e.g., expressed in $[m^3/sec]$). The state variable is represented by the amount of fluid contained within the vessel. With adequate assumptions on the shape of the container, the fluid level $h(t)$ can be used as system state variable.

The fluid level needs to be maintained within a predefined range $[h^{min}, h^{max}]$ by acting on a hydro pump that, when active, pumps out a constant amount of fluid $Q^{out}$ from the vessel, being $Q^{out} > Q^{in}$. The fluid level is not altered when the pump is inactive. The hydro pump is actuated by an electric motor that consumes a $P_i$ amount of electric power when active, no power otherwise. The electric motor represents our $\lambda_i$ load. If many of such systems are deployed for a given application, the goal is to achieve the application requirements while considering physical process behaviour and, on the other hand, to adequately schedule the electric motor activations to limit the peak load of power consumption.

VII. APPLICATION OF THE PROPOSED TECHNIQUE: A SIMULATED EXAMPLE

In this section we present the application of the proposed modeling technique to a physical system composed by 3 electric devices having physical characteristics depicted in Table I. For the sake of simplicity, we limit this example to a set of loads having $U < 1$. This allows to clearly show how the absence of a proper management of load activations brings to the highest possible peak load in the worst case, while our approach improves (i.e., it decreases) the peak load. It is worth to note that, being $U \leq 1$, a real-time scheduling algorithm such as EDF is able to schedule the load set without any concurrent activation of loads. However, this fact does not limit the applicability
of the proposed results since, when $U > 1$, loads can be partitioned into groups having $U \leq 1$, as shown in [18]. Although this solution may not bring to the optimal global schedule, it allows to use the proposed method to guarantee the limitation of the state variable.

Figure 3 shows the behavior of a simple on/off control technique applied to the considered load set. Each load is independently controlled so that the load is turned on when the state variable reaches the upper bound of the working range, and it is kept active until the lower bound is reached. This control strategy easily allows to individually maintain the state variable within the required working range. However, since the activation of loads is not coordinated, it is possible that more than one load is active at the same time, which turns out in an increase of the peak load of power consumption. In fact, several times in the depicted time range, the three considered loads are activated simultaneously, thus determining a peak load $P = \sum P_i = 6$.

The same three loads have been modeled and managed with the techniques introduced in this paper. Real-time parameters are calculated using the results presented in Section V, and they are reported in Table II. Figure 4 shows the behavior of state variables and the instantaneous consumed power $p(t)$. The figure shows that state variables are confined within the desired working ranges, while the load scheduling achieves to limit the peak load to $P = \max_i P_i$. In fact, in this example the load set is scheduled using the Earliest Deadline First (EDF) scheduling algorithm [13] and, since the total utilization is $U = 0.95 < 1$, the load scheduling guarantees that only one load is active at any given time. Notice that $x_3(t)$ reaches its lower bound since the corresponding load is never preempted (it works in the worst condition specified by Lemma 2), while this is not the case for $x_1(t)$ and $x_2(t)$.

Notice that in the proposed example we expressly generate a load set having $U \leq 1$. If $U > 1$, no scheduling algorithm would be able to achieve the activation of one load only at any given time. Therefore, concurrent activations of loads should have been adequately managed by either allowing concurrent activations or rejecting the activation of some loads to achieve the schedulability condition. The former approach resembles the real-time scheduling on multiprocessors, while the latter method involves techniques for overload management, widely studied in real-time computing systems. Both approaches are subject to ongoing research.

**VIII. Conclusions and future works**

This paper presented a methodology for modeling the physical system of a cyber-physical energy system as periodic activities that can be scheduled by adapting traditional real-time scheduling algorithms. The goal of the proposed approach is to limit the peak of power consumption, which is a desirable feature for both the user and the energy provider, while achieving the requirements imposed by the application. In particular, in this paper we discussed the application to load sets where loads have linear dynamics behaviours.

The proposed approach represents a pioneering approach to the use of real-time scheduling techniques to organize the activation of electric loads in a cyber-physical energy system. In this paper, a number of simplifying assumptions have been made, such as considering periodic activations only, linear state variable’s dynamics, etc., thus future works will address the relaxation of such restrictive assumptions.

**References**


Reducing the Peak Power through Real-Time Scheduling Techniques in Cyber-Physical Energy Systems

Tullio Facchinetti  
Computer Engineering and Systems Science  
University of Pavia  
Pavia, Italy  
Email: tullio.facchinetti@unipv.it

Enrico Bini  
RETIS Laboratory  
Scuola Superiore Sant’Anna  
Pisa, Italy  
Email: e.bini@sssup.it

Marko Bertogna  
RETIS Laboratory  
Scuola Superiore Sant’Anna  
Pisa, Italy  
Email: marko@sssup.it

Abstract—This paper presents a method for applying real-time scheduling techniques to balance the power usage of electric loads in cyber-physical energy systems. The aim of the proposed approach is to achieve predictability of the activation of electric loads to guarantee an upper bound on the peak electric power consumption.

The contribution of this paper encompasses several aspects. The relevance of balancing electric loads is discussed, motivating the use of real-time scheduling techniques to achieve predictability on electric-load management. We introduce the innovation of modeling the physical system as a set of periodically activated loads, that can be effectively managed by adequately adapting traditional real-time system models and scheduling algorithms, to guarantee an upper bound on the peak power consumption. For this purpose, we present a problem formulation based on linear programming, while a low-complexity heuristic is proposed to limit the complexity of the optimization process. Simulation results are presented to assess the performance of proposed methods.

Keywords—Cyber-Physical Energy Systems; Load Balancing; Real-Time; Modeling; Peak Load

I. INTRODUCTION

Cyber-physical systems (CPSs) represent an emerging technology that aims to integrate embedded processing devices to monitor and control physical processes. Cyber-physical systems are intended to address critical applications operating in dynamic and uncertain environments, made by a high number of devices and characterized by complex relationships among components. Several factors can affect system operations, such as hardware and software failures, and partial knowledge of the system operating state. Example applications include: automotive, avionics and medical systems; critical infrastructure management, such as electric power and water resources; traffic control and safety; advanced robotics for manufacturing or telemedicine (see [1] for details on some specific systems).

While the current application design is based on the use of traditional embedded systems, which emphasizes the computational issues performed by embedded processing units, cyber-physical systems are more focused on the tight integration between physical and computational systems. This paper concentrates the attention on those systems dedicated to energy management, i.e., Cyber-Physical Energy Systems (CPESs) [2]. In such systems, the “physical” process is made by a network of electric devices that are controlled by a complex set of interconnected embedded systems.

The current technology trend is moving towards the automatic, distributed and coordinated control of electric devices. Some examples can be found in home and factory automation systems [3], large networks of electric cars [4], and automated energy supply and distribution for town and city districts organized in smart grids [5]. On the other hand, the diffusion of compact, flexible and low-cost embedded systems is making more practical and attractive the implementation of CPESs, since monitoring and control actions can be accurately applied on devices composing the considered physical system. Such embedded systems can be connected to build large distributed networks, thus being able to coordinate the actions on large systems.

The limited sources and the growing request of electric energy, together with the impact of power generation, transportation and usage on the environment and the eco-system, motivates the research on techniques to optimize the energy utilization in cyber-physical energy systems.

An “electrical system” can be defined as a set of electrical devices, or loads. Within the scope of this paper, a load is characterized by its power consumption, i.e., the maximum amount of power consumed when the load is active. Loads can be turned on/off depending on different conditions: the purpose of the device, the relationship with other devices, environmental conditions, and time.

Electrical systems may be composed of tens to thousands loads, where each device (or groups of them, which could be seen as a single logical device) must be driven in a timely fashion.

The balancing of energy utilization is fundamental for the efficient behavior of an electrical system [6], [7]. For this purpose, specific technical and economical approaches are used to control the distribution of power usage over time. One of the most widely adopted methods is the “peak-load pricing”, which assigns higher prices to larger peak-load demands [8]. The most recent analysis of this pricing policy
Reducing the Peak Power through Real-Time Scheduling Techniques in Cyber-Physical Energy Systems

Tullio Facchinetti  
Computer Engineering and Systems Science  
University of Pavia  
Pavia, Italy  
Email: tullio.facchinetti@unipv.it

Enrico Bini  
RETIS Laboratory  
Scuola Superiore Sant'Anna  
Pisa, Italy  
Email: e.bini@sssup.it

Marko Bertogna  
RETIS Laboratory  
Scuola Superiore Sant'Anna  
Pisa, Italy  
Email: marko@sssup.it

Abstract—This paper presents a method for applying real-time scheduling techniques to balance the power usage of electric loads in cyber-physical energy systems. The aim of the proposed approach is to achieve predictability of the activation of electric loads to guarantee an upper bound on the peak electric power consumption.

The contribution of this paper encompasses several aspects. The relevance of balancing electric loads is discussed, motivating the use of real-time scheduling techniques to achieve predictability on electric-load management. We introduce the innovation of modeling the physical system as a set of periodically activated loads, that can be effectively managed by adequately adapting traditional real-time system models and scheduling algorithms, to guarantee an upper bound on the peak power consumption. For this purpose, we present a problem formulation based on linear programming, while a low-complexity heuristic is proposed to limit the complexity of the optimization process. Simulation results are presented to assess the performance of proposed methods.

Keywords-Cyber-Physical Energy Systems; Load Balancing; Real-Time; Modeling; Peak Load

I. INTRODUCTION

Cyber-physical systems (CPSs) represent an emerging technology that aims to integrate embedded processing devices to monitor and control physical processes. Cyber-physical systems are intended to address critical applications operating in dynamic and uncertain environments, made by a high number of devices and characterized by complex relationships among components. Several factors can affect system operations, such as hardware and software failures, and partial knowledge of the system operating state. Example applications include: automotive, avionics and medical systems; critical infrastructure management, such as electric power and water resources; traffic control and safety; advanced robotics for manufacturing or telemedicine (see [1] for details on some specific systems).

While the current application design is based on the use of traditional embedded systems, which emphasizes the computational issues performed by embedded processing units, cyber-physical systems are more focused on the tight integration between physical and computational systems. This paper concentrates the attention on those systems dedicated to energy management, i.e., Cyber-Physical Energy Systems (CPESs) [2]. In such systems, the “physical” process is made by a network of electric devices that are controlled by a complex set of interconnected embedded systems.

The current technology trend is moving towards the automatic, distributed and coordinated control of electric devices. Some examples can be found in home and factory automation systems [3], large networks of electric cars [4], and automated energy supply and distribution for town and city districts organized in smart grids [5]. On the other hand, the diffusion of compact, flexible and low-cost embedded systems is making more practical and attractive the implementation of CPESs, since monitoring and control actions can be accurately applied on devices composing the considered physical system. Such embedded systems can be connected to build large distributed networks, thus being able to coordinate the actions on large systems.

The limited sources and the growing request of electric energy, together with the impact of power generation, transportation and usage on the environment and the eco-system, motivates the research on techniques to optimize the energy utilization in cyber-physical energy systems.

An “electrical system” can be defined as a set of electrical devices, or loads. Within the scope of this paper, a load is characterized by its power consumption, i.e., the maximum amount of power consumed when the load is active. Loads can be turned on/off depending on different conditions: the purpose of the device, the relationship with other devices, environmental conditions, and time.

Electrical systems may be composed of tens to thousands loads, where each device (or groups of them, which could be seen as a single logical device) must be driven in a timely fashion.

The balancing of energy utilization is fundamental for the efficient behavior of an electrical system [6], [7]. For this purpose, specific technical and economical approaches are used to control the distribution of power usage over time. One of the most widely adopted methods is the “peak-load pricing”, which assigns higher prices to larger peak-load demands [8]. The most recent analysis of this pricing policy
Peak-load pricing is often used by electricity suppliers and telephone utilities to enforce a limitation on the peak service time frame due to the impossibility of generating enough energy to satisfy the request of customers. Therefore, energy production may unpredictably increase in a short time frame due to the impossibility of generating enough energy to satisfy the request of customers. Therefore, energy providers — that must observe contractual obligations with their customers to supply electricity at pre-defined fixed prices — may experience a relevant financial burden. On the other hand, an adequate management of peak load conditions is desirable for energy utilities [9], so that an appropriate load management aiming at achieving predictable load conditions may lead to potential contractual benefits to the user. As a consequence, both energy providers and consumers are likely to be interested in load balancing and predictable energy consumption.

Given the aforementioned technical and economic issues, an efficient management of peak-load conditions has the following advantages:

1) the least efficient, i.e., the most expensive, power plants can be turned off if the peak power demand is guaranteed to remain under a given threshold;
2) the electric distribution infrastructure can be tailored for lower peak loads, with less technical issues and reduced costs;
3) the curve of power usage can be smoother and flatter, allowing the final users to have better pricing conditions on the free energy market, where pricing strategies are often driven by the peak-load pricing policy [8].

This work aims to apply real-time scheduling techniques to the management of loads in cyber-physical energy systems. The goal is to balance the total consumed power and the peak power consumption. The main expected advantage of this approach is to leverage the strong mathematical background of the real-time scheduling discipline for modeling and analyzing a physical system. On one hand, real-time scheduling algorithms can be used to predictably activate/deactivate electrical devices to guarantee the desired system features, in terms of timing constraints and energy consumption. On the other hand, the typical large size of cyber-physical energy systems will take advantage of efficient scheduling algorithms and analysis techniques to determine the system feasibility and properties.

It is worth noting that this paper does not deal with the architecture or the engineering of a cyber-physical energy system. The proposed approach must be intended as a viable modeling technique for the physical energy system, allowing the development of predictable and robust control strategies based on real-time scheduling methodologies. To the best of our knowledge, this is the first work addressing the application of real-time scheduling techniques to cyber-physical energy systems in order to balance the consumed power and to achieve a bound on the peak load. Therefore, the first part of the paper is dedicated to the description of the approach and of the potential contributions that real-time methodologies could bring to properly modeled CPEs.

A. Paper organization

The paper is organized as follows: a detailed explanation of the analogy between real-time computing systems and electrical systems is given in Section II, motivating the proposed approach with examples and possible scenarios. Section III introduces the system model, under which the theoretical results of Section IV are derived. Sections V and VI present, respectively, an optimal and an approximated method to reduce the peak load. The effectiveness of such methods is assessed in Section VII by means of simulations. Finally, Section VIII states our conclusions and outlines several possible directions for future enhancements of this work.

II. ELECTRIC LOADS AS REAL-TIME TASKS

This paper introduces the application of real-time analysis techniques to schedule the activation of electric devices in electrical networks. For this purpose, an analogy is established between real-time computing systems and cyber-physical energy systems.

Real-time scheduling allows managing the execution of tasks on processors under timing constraints. In more general terms, real-time scheduling can be seen as the discipline of allocating resources over time to a set of time-consuming tasks, so that given timing constraints are satisfied. However, in this more general formulation, resources may not necessarily be processors or computing devices. In fact, real-time scheduling techniques are also applied to communication systems, where real-time algorithms are used to schedule sets of messages over a communication channel [10]. In this case, an analogy holds between computing tasks and messages, as well as between processors and communication channels. The meaning of “available bandwidth” changes depending on the particular context, referring to the channel capacity in communication systems, and to
processor’s computing time in computing systems. Finally, timing constraints are enforced on the execution times in one case, and (typically) on message’s end-to-end latency in the other. In other words, a real-time task must be guaranteed to terminate its execution before its deadline, while a message must be delivered to the receiver within the given time limit. This analogy allows extending to communication networks many results that have been originally developed for real-time computing systems, and vice versa. An example is given by the “real-time calculus” [11], a real-time extension of the network calculus. The above considerations lead to the opportunity of profitably applying real-time scheduling techniques, with suitable adaptations and extensions, to systems presenting similar analogies.

Electric devices are modeled as periodically activated tasks, with a bound on the total time that a load can remain active — thus consuming power — in each period. This bound recalls the Worst Case Execution Time (WCET) of a real-time task in computing systems. As for computing tasks, all the time properties of electric loads (periods, deadlines and activation time) must be selected according to their application requirements. Section II-A provides some examples of timing constraints related to specific electrical loads. Based on the system model, a priority-based scheduling algorithm can be applied to selectively activate/deactivate each device. The goal is to meet the timing constraints of each load, while guaranteeing an upper bound on the total instantaneous power consumed by the concurrent activation of electric components.

In the real-time systems literature, there is active research on power-aware scheduling strategies to save energy while achieving timing constraints. Such scheduling policies aim at reducing the power consumption using special features of modern electronic hardware, such as Dynamic Voltage Scaling (DVS) [12], [13]. As in those works, the model proposed in this paper associates a maximum consumed power to each electric device. However, some distinctions can be identified. First of all, we do not aim at directly reducing the overall energy required by the system. The objective is, instead, to determine a bound on the peak power consumption, and to predictably enforce this bound by scheduling electric devices activations in a timely manner.

Some recent works are addressing the real-time issues related with some special cases of cyber-physical energy systems. In [14], the authors propose a technique to improve the efficiency of batteries charge/discharge, for electric vehicles. However, this work is limited to batteries, while our approach is oriented at establishing a general framework for managing energy systems in a real-time manner. In [15], the authors aim at finding optimal schedules for microCHP (Combined Heat and Power) systems. The approach is based on global optimization through an Integer Linear Programming formulation of the problem. However, the application of the proposed method is strictly limited to offline optimization, while our technique can be applied online. Moreover, we introduce the novelty of modeling electric loads using real-time parameters, to allow the use of real-time techniques for scheduling the activation of loads. Finally, in [16] the authors describe a cyber-physical energy system as a set of components modeled as dynamical systems. While the modeling of electrical devices is more advanced than the one proposed in this paper (refined modeling is subject of future research in our framework), the goal is not related with achieving peak load constraints and, again, no real-time issues are considered.

A. Load modeling

This section provides informal examples of electric devices and applications that are suitable for being integrated in a real-time management system. Their relevant characteristics are described, outlining a possible modeling of their timing properties.

**Household appliances:** Typical household devices like ovens, washing machines, dryers, dishwashers, have each a peculiar duty cycle. The tighter the timing requirements — i.e., the closer the deadline to the maximum activation time — the more constraints are imposed on the scheduling algorithm, reducing the chances of finding a lower peak load. Anyway, a certain slack is usually available in the working cycles of household appliances, and programmable devices are already used to control the activation of electric loads depending on the energy prices in the stock market. As an example, these devices are used to control washing machines in domestic environments, where postponing by a few hours the time at which the laundry is ready does not cause any problem.

**Lighting:** Consider the corridor lights of a building, that may need to be turned on in the evening, for example at 8:30pm, and turned off in the morning at 7:00am. In this case, no service interruption can be tolerated. During the active period, the total power consumption is the sum of power consumed by each lighting device, while in the rest of the time, the power consumption is negligible.

In this simple case, the load has a period of 24h, an active time of 10:30h, and a relative deadline equal to the active time. In this way, the load must be continuously scheduled at the beginning of the period, without allowing any “preemption” while the lights are switched on, as is expected from a lighting system. Clearly, this requirement has a negative impact on the level of concurrency of load activations. Electrical loads of this kind (i.e., with no activation slack) will lead to an increase in the number of concurrently active loads. Since no slack is available in the activation cycle, there is no way of reducing the impact of these loads on the resulting peak load. However, it is still possible to control the schedule of less interactive loads, so that they be activated when there is a smaller energy requirement.
III. SYSTEM MODEL

We consider a system composed of a set \( \Lambda = \{\lambda_1, \ldots, \lambda_n\} \) of \( n \) independent electric loads that request to be turned on and off (or activated/deactivated), depending on their specific timing requirements. A load is said to be active when it is turned on, inactive otherwise.

The \( j \)-th request for activating the load \( \lambda_i \) happens at time \( r_{i,j} \). The \( i \)-th load \( \lambda_i \) is modeled by the tuple \((T_i, C_i, P_i)\), where

- \( T_i \) is the minimum separation between two consecutive requests of activation \( r_{i,j}, r_{i,j+1} \) (as in the sporadic model for real-time computing tasks [17]). Hence,

\[
\forall \lambda_i, \forall j \quad r_{i,j+1} \geq r_{i,j} + T_i
\]

- \( C_i \) is the longest time the load \( \lambda_i \) can be active between two consecutive requests;
- \( P_i \) is the nominal power consumed by the load \( \lambda_i \) during its active time.

We define the utilization of \( \lambda_i \) as \( U_i = C_i / T_i \). The total utilization of \( \Lambda \) is \( U = \sum_{i=1}^{n} U_i \).

The load activity is controlled by a load scheduler that decides when each load is activated/deactivated. Formally, the scheduler assigns to each load \( \lambda_i \) a schedule that is modeled by the function \( s_i(t) \)

\[
s_i(t) = \begin{cases} 
1 & \text{\( \lambda_i \) is active at } t \\
0 & \text{otherwise}
\end{cases}.
\]

The schedule of loads is then given by \( S = \{s_1, \ldots, s_n\} \).

A schedule \( S \) is said to be valid if it assigns to each load \( \lambda_i \) an amount of activity time equal or larger than \( C_i \) between two consecutive requests. Formally,

\[
\forall \lambda_i, \forall j \quad \int_{r_{i,j}}^{r_{i,j+1}} s_i(t) \, dt \geq C_i
\]

Notice that the equality will suffice in Equation 3 if traditional scheduling algorithms (such as EDF or RM) are used to generate the schedule.

For a given schedule \( S \), the actual power consumed by the load \( \lambda_i \) at time \( t \) is

\[
p_i(t) = P_i s_i(t).
\]

The overall actual power consumption \( p(t) \) at time \( t \) is

\[
p(t) = \sum_{i=1}^{n} p_i(t).
\]

Finally, we define the peak load \( P \) of a set of loads \( \Lambda \) as the maximum instantaneous power consumption over the system lifetime

\[
P = \max_{t \geq 0} p(t).
\]

Given these hypothesis, we can formulate our problem as follows

\[
\begin{align*}
\text{minimize} \quad & P \\
\text{subject to} \quad & S \text{ being a valid schedule}
\end{align*}
\]

Unfortunately, solving the problem in this wide formulation is very hard. In the next sections, we will show how to exploit well known real-time scheduling algorithms to find a suitable solution for this problem.

IV. RT SCHEDULING ALGORITHM FOR ELECTRIC LOADS

We propose to use classic real-time scheduling algorithms, such as Rate Monotonic (RM) or Earliest Deadline First (EDF) [18], to schedule the loads in \( \Lambda \). Specifically, each load can be considered as a task with computation time \( C_i \) and period (equal to the deadline) \( T_i \). For example, when \( U \leq 1 \), the EDF scheduling algorithm can build a schedule \( S \) with the minimum possible peak power, that is \( P = \max_\lambda P_i \).

However, if \( U > 1 \) some loads must be contemporarily activated, leading to a possibly larger peak power consumption \( P \). Hence, we suggest to partition the \( \Lambda \) load set into \( m \) disjoint sets \( \Lambda_j \), \( j = 1, \ldots, m \), that we call scheduling groups. Scheduling groups are determined such that their total utilization, defined as

\[
U(\Lambda_j) = \sum_{\lambda_i \in \Lambda_j} U_i,
\]

is smaller than or equal to 1. This property enables EDF to find a valid schedule within each scheduling group. The maximum peak in this case happens when the loads with the highest powers are contemporarily activated in all the scheduling groups. Notice that Equation (5), which is evaluated over all loads \( \lambda_i, 1 \leq i \leq n \), could also be evaluated over all scheduling groups \( \Lambda_i, 1 \leq i \leq m \), since in each scheduling group only one load is active at any given time \( t \). An upper bound \( P^* \) on the peak load can be found considering the contemporary activation on all groups of the load with the largest power. Therefore,

\[
P^* = \sum_{\lambda_i \in \Lambda_j} \max_{\lambda_i} p_i.
\]

It is worth noting that \( P^* \) represents an upper bound, but it is not tight, i.e., it could be overly pessimistic.

In this section, we provide some theoretical results related to the considered system model, and propose strategies to produce a valid schedule of a given set of loads, with the goal of reducing and bounding the peak load. We will present two scheduling algorithms with different complexities. One algorithm produces a smaller peak load, although it requires a large computational effort; the second one is simpler, although it could result in a larger peak load.

Before presenting the algorithms, we first state the following theoretical result on the minimum achievable peak load.
Theorem 1: For any load set \( \Lambda \), no valid schedule can produce a peak load lower than

\[
P^{\text{min}} = \sum_{i \in \Lambda} P_i u_i. \tag{9}
\]

Proof: Assume, by contradiction, a load allocation for \( \Lambda \) grants a peak load \( P < P^{\text{min}} \). Let \( H \) be the least common multiple of all load periods \( T_1, \ldots, T_n \). The overall energy consumed by \( \Lambda \) over \( H \) when all loads are synchronously activated at time \( t = 0 \), and then periodically activated as soon as possible, is

\[
\sum_{i=1}^{n} \frac{H}{T_i} C_i P_i = H \sum_{i=1}^{n} U_i P_i.
\]

Since the peak load is assumed to be equal to \( P \), the overall energy consumed by \( \Lambda \) in \( H \) can not be greater than \( PH \). Therefore,

\[
H \sum_{i=1}^{n} U_i P_i \leq PH,
\]

and,

\[
\sum_{i=1}^{n} U_i P_i \leq P.
\]

Using Equation (9), we get

\[
P^{\text{min}} \leq P.
\]

leading to a contradiction.

V. LINEAR PROGRAMMING FORMULATION

The problem of partitioning the set of loads as introduced in Section III, can be formalized as a level packing problem [19]. In level packing, a strip must accommodate a set of rectangles such that the total height is minimized. The peculiarity of level packing is that rectangles are partitioned in horizontal levels of decreasing height from the bottom to the top. In each level, items are packed from left to right by decreasing height, similarly to the arrangement of books within a bookshelf (see Figure 1).

Since the height of a level is equal to the leftmost rectangle, such a rectangle is said to initialize the level. The advantage of level packing is that a two-dimensional problem is transformed into a pair of one-dimensional problems, namely the packing of levels, and the packing of rectangles into levels.

In this paper, the level packing problem is solved using a Binary Integer Linear Programming (BILP) technique after a proper modeling of the problem, which brings to the introduction of suitable optimization variables.

Each load is modeled as a rectangle whose height corresponds to the power consumption \( p_i \) and width is determined by its utilization \( u_i \). Without loss of generality, all loads are assumed to be sorted by decreasing power, namely \( p_i \geq p_j \iff i \leq j \). In the worst case, there are \( n \) possible levels, one for each rectangle as the starting item. A set of \( n \) variables \( y_i \in \{0,1\} \) defines level initialization. There is one such variable for each load, being \( y_i = 1 \) if item \( i \) initializes level \( i \), \( y_i = 0 \) otherwise. A level is labelled by the index of the item initializing it. The variables \( x_{i,j} \) with \( i \in \{1, \ldots, n-1\} \) and \( j > i \) define the packing of item \( j \) when it does not initialize a level. The value \( x_{i,j} = 1 \) is set if item \( j \) is packed in level \( i \), \( x_{i,j} = 0 \) otherwise.

For example, in the case depicted in Figure 1, it holds \( y_1 = y_3 = 1 \), because only items 1 and 3 initialize a level, while \( y_i = 0 \) is set for all remaining items. The allocation of other rectangles to their respective levels is encoded in

\[
x_{1,2} = x_{1,4} = x_{1,6} = 1 \text{ and } x_{3,5} = 1, \text{ with all other values being } x_{i,j} = 0.
\]

First of all, since each load can either initialize one level or it can be one of the rectangles following the initializer, the following constraint must hold:

\[
y_j + \sum_{i=1}^{j-1} x_{i,j} = 1 \quad \forall j = 1, \ldots, n \tag{10}
\]

Notice that, thanks to the ordering of the rectangles by decreasing height, item \( j \) can be allocated as one of the non-initializing items only in the levels from 1 to \( j-1 \).

A second constraint arises from the maximum width of the resource. The value \( W \) is defined to be equal to the utilization upper bound that guarantees the schedulability of a load set. For example, if Earliest Deadline First (EDF) with implicit deadlines is used, then we set \( W = 1 \). Since the horizontal dimension is interpreted as utilization, then each level can not exceed the width \( W \) of the rectangle. Therefore, it holds

\[
\sum_{j=i+1}^{n} u_j x_{i,j} \leq (W - u_i) y_i \quad \forall i = 1, \ldots, n - 1 \tag{11}
\]

To enforce the consistency of the contraint given by Equation 11, notice that when level \( i \) does not exist \( (y_i = 0) \),
then all $x_{i,j}$ are forced to 0 as well. The constraint specified by Equation 11 enforces the utilization based schedulability test. Therefore, it makes the proposed solution suitable for scheduling algorithms where feasibility can be evaluated by an utilization-based test. However, in [20], the authors propose the description of the EDF scheduling algorithm, where deadlines are less than periods, using a set of linear inequalities that could be used within the BILP framework. Therefore, the approach proposed in this paper can be easily extended to such system model.

The goal of the optimization approach based on BILP is to minimize the sum of the peak powers on each group, that is

$$\text{minimize } \sum_{i=1}^{n} p_i y_i \quad (12)$$

The evaluation of the number of variables and constraints provides an estimate the problem complexity. In the proposed scheme, the number of $y_i$ variables is $n$, because all rectangles may initialize one level. The $x_{i,j}$ variables are $\frac{n(n-1)}{2}$. Hence, the total number of variables is $\frac{n(n+1)}{2}$. Moreover, by counting the number of inequalities in Equations (10) and (11), we find that the number of constraints is $2n - 1$.

VI. LOAD BALANCING HEURISTIC

This section introduce a heuristic algorithm to address the problem of generating scheduling groups. Algorithm 1 shows the pseudo-code of the proposed method. The key point of the algorithm consists in sorting the global set of loads $\Lambda$ in a descending order with respect to powers, such as $\lambda_i < \lambda_j \iff p_i > p_j$. The algorithm is essentially a first-fit bin-packing algorithm applied to the ordered set of loads. The $\lambda_i$ load is inserted into the first scheduling group when the schedulability of the group is feasible. Otherwise, a new scheduling group is created and the current load is inserted into the newly created group.

The proposed technique recalls the RM-FFDU (Rate Monotonic First-Fit Decreasing utilization) partitioning scheme for scheduling fixed priority real-time tasks on a multi-processor system [21], where bin-packing techniques are used to allocate tasks on processors. However, the mentioned previous work does not address the optimization of the total power consumption. Moreover, the key distinction is that in our method the ordering is made with respect to the value of load’s consumed power, and utilization is not considered for this purpose.

Since no specific scheduling algorithm is assumed within each scheduling group, the feasibility test to be performed in Algorithm 1 is not specified, being dependent on the adopted scheduling policy. The complexity of the proposed method is therefore $O(\alpha \cdot n^2)$, where $\alpha$ represents the complexity of the feasibility test adopted. As an example, when using EDF with the associated utilization-based feasibility test, the complexity is $O(n^3)$.

Algorithm 1 The pseudo-code of the load balancing heuristic.

```plaintext
1: sort $\Lambda$ in decreasing order of power
2: $\Lambda_1 \ldots \Lambda_m$ are the scheduling groups
3: $m = 1$ is the initial number of scheduling groups
4: for all $\lambda_i \in \Lambda$ do
5:   for $j = 1$ to $m$ do
6:     if $\lambda_i$ is schedulable in $\Lambda_j$ then
7:       add $\lambda_i$ to $\Lambda_j$
8:     goto end-loop
9:   end if
10: end for
11: create a new scheduling group $\Lambda_{m+1}$
12: add $\lambda_i$ to $\Lambda_{m+1}$
13: $m = m + 1$
14: end-loop
```

VII. EXPERIMENTAL ASSESSMENT

This section reports some results obtained by generating random electric loads while changing some of the most relevant parameters. The goal is to investigate, under different circumstances, the reduction of the peak load achieved both by solving the optimization problem and using the heuristic approach.

The peak load achievable using the proposed schemes is compared with the worst possible case where all the loads are active at the same time:

$$P_{\text{max}} = \sum_{i=1}^{n} p_i,$$

The parameters that have been taken into account in the experiments are: the total number of loads $n$, the total utilization of the set of loads $U$, and the range for the power assigned to the loads. Given those parameters, the value of each load is randomly generated using the algorithm UUniFast presented in [22].

Figure 2 shows the efficiency of different approaches with respect to $P_{\text{max}}$, as a function of the ratio between the total utilization $U$ and the number of loads $n$. The efficiency $\eta$ is calculated as

$$\eta = \frac{P_{\text{max}} - P_{\text{meth}}}{P_{\text{max}}} \cdot 100$$

where $P_{\text{meth}}$ represents the peak load achieved by the given method: lower bound, LP and heuristic refer, respectively, to the peak load obtained from Theorem 1, the method of Section V and the approximated approach of Algorithm 1. The value of the peak load used to calculate the efficiency is an aggregated value obtained by averaging the outcome of thousands of simulation runs. The number of loads assumes values in the range $[2, 30]$, while the total utilization ranges
in the interval [2, 18]. The nominal power of each load is randomly selected in the range [20, 2000], which is a reasonable range for typical household appliances.

The results of Figure 2 show that for lower values of the $U/n$ ratio, i.e., having a high number of loads and a small total utilization, the proposed methods allow reducing the peak load up to more than 90% with respect to $P_{\text{max}}$. Therefore, the explicit control on load activations brings to a remarkable improvement in comparison to the absence of control actions. When the $U/n$ ratio tends to 1, the benefits of using a scheduling approach disappear. This is due to the fact that, when $U$ tends to $n$, the load generation algorithm presented in [22] generates an increasing number of loads having $U_i = 1$ in order to obtain the desired total utilization. In this situation, the loads cannot be efficiently aggregated into scheduling groups, so that each created scheduling group contains just a few loads (only one load in the worst case). Therefore, the number of scheduling groups tends to $n$ and the peak load achievable by all methods converges to the maximum possible peak load $P_{\text{max}}$, leading to $\eta \to 0$. Notice that when $U \geq n$, it holds $\eta = 0$.

Figure 3 shows the average peak load obtained by the different techniques as a function of the number of loads $n$ when the total utilization is constant ($U = 10$). It can be noticed that, when $U \leq 10$, the peak load achieved by the optimized methods can not be better than $P_{\text{max}}$ for the same reason above: every load $\lambda_i$ is generated with $U_i = 1$, and thus there is no opportunity to apply the scheduling of loads since each scheduling group contains exactly one load. When $U > 10$, the optimized methods guarantee an improvement that increases with $n$, accordingly with the results presented in Figure 2. Moreover, Figure 3 shows that the peak load achieved by the heuristic method is very close to the peak load guaranteed by the Linear Programming formulation which, in turn, is rather close to the lower bound $P_{\text{min}}$ imposed by Theorem 1. This characteristic behavior has been steadily detected throughout all experiments.

Finally, Figure 4 shows the average peak load obtained by the heuristic method as a function of the maximum possible power for each load; we considered $n = 100$, $U = 50$ and the minimum possible power equal to 10. Similarly to the previous results, a noticeable decrease of the peak load is achieved by the heuristic with respect to $P_{\text{max}}$. This improvement is independent from the range in which the power is selected for each load. Moreover, the solution found by the heuristic is relatively close to the lower bound $P_{\text{min}}$.

VIII. CONCLUSIONS AND FUTURE WORKS

This paper presented a methodology for modeling the physical system of a cyber-physical energy system as periodic activities that can be scheduled by adapting traditional
real-time scheduling algorithms. The goal of the proposed approach is to limit the peak of power consumption, which is a desirable feature for both the user and the energy provider.

To the best of our knowledge, this is the first attempt of using real-time scheduling techniques to organize the activation of electric loads in a cyber-physical energy system. In this paper, a number of simplifying assumptions have been made, such as considering periodic activations only, time-invariant load states, etc. Several improvements and refinements to the proposed model are thus possible: accounting for event-driven (aperiodic) load activations; an optimal scheduling strategy that would consider the interaction among loads of different groups (i.e., a global scheduling of loads); more accurate modeling of specific devices, considering different working modes, i.e. with different power consumption (full power, power saving mode, standby, etc.), time-varying load states, or accounting for the cost of “context switches”, since switching on and off an electric motor has a cost that, in the long term, may shorten its life cycle. All those topics will be subject of future research.

REFERENCES


Power-Aware Scheduling in Embedded Systems: Adaptation of SDVS Scheduling using Deadline Overrun Detection

Jonas Höglund*, Stefan Olofsson*, Magnus Persson†, Johan Fredriksson*, Detlef Scholle*

*Enea AB, Kista, Sweden
†KTH - The Royal Institute of Technology, Dept. of Mechatronics, Stockholm, Sweden
‡Mälardalen University, Mälardalen Real-Time Research Centre, Sweden

Abstract—Minimizing power consumption in real-time embedded systems is an important challenge. A common approach is to dynamically adjust the operating frequency and the supply voltage of the processor. To ensure system real-time properties, a power-aware scheduling algorithm is needed.

The contribution of this paper is the development of a new power-aware soft real-time scheduling algorithm, deadline-overrun detection DVFS (DOD-DVFS). The algorithm is based on SDVS and enhanced with run-time adaptation for one task to reduce power usage. It is easily implementable and has power savings comparable to more complex algorithms.

The algorithm has been evaluated on a Freescale development platform running the real-time operating system OSE developed by Enea AB. We show that the DOD-DVFS algorithm can produce energy savings over SDVS of 31%, while only causing a few deadline overruns in typical cases. The reduced power consumption is comparable to power savings documented for more complex DVFS algorithms.

Index Terms—real-time, embedded, DVFS, Deadline-overrun detection, power-aware scheduling

I. INTRODUCTION

Mobile phones and other hand-held devices place a very high premium on small size and weight. Since the battery is already the largest and heaviest component of a phone, increased power capacity must come from using the energy in existing batteries more efficiently, rather than increasing their size and weight. Although battery technology has progressed steadily forward, it has not kept up with increasing energy demands for mobile platforms, including multimedia mobile phones with large color displays. Although new power storage technologies, such as fuel cells, seem to have great potential to increase energy density, it is not generally expected that they will eliminate the need for power efficiency, due to the ever increasing demands of power-hungry applications [1].

The computational needs of hand-held devices can be classified either as real-time or non real-time. According to Buttazzo [2], a real-time system is made up of real-time tasks. The main difference between a real-time and non real-time task is that a real-time task is characterized by a deadline, which is the maximum time within it must complete its execution. In critical applications, a result produced after the deadline is not only late, but also wrong!

Real-time requirements is a key constraint for hand-held embedded devices such as cell phones. These systems require the ability to support both communication protocols (with hard real-time requirements in the order of a millisecond) and end-user applications, including streaming media and other soft real-time applications.

This paper begins with an overview of real-time system concepts and dynamic voltage frequency scaling (DVFS). In Section II, an overview of alternative approaches to the DVFS problem is presented. In Section III, the Deadline-Overrun-Detection DVFS algorithm (DOD-DVFS) is presented. Section IV describes the hardware and software implementation of the test bed for the proposed algorithm. In Section V, the DOD-DVFS algorithm is compared to the SDVS algorithm. The article finishes with a discussion of the results and proposed future directions.

A. Real-Time Systems

Implementation of power management on mobile devices often must consider real-time requirements. In this section, an overview of real-time system concepts will be presented as a framework for later discussions of power management with real-time constraints.

Application requirements lead to a differentiation between hard and soft real-time constraints. In a hard real-time system, time constraints are critical, and missing a deadline can have catastrophic consequences. In a soft real-time system meeting deadlines is desirable, but an occasional deadline miss is not catastrophic.

Real-time applications are divided into tasks and the order of their execution is determined by a real-time scheduler. The scheduler must ensure that the timing constraints of the application is met by executing the tasks in a certain order and the scheduling policy determines that order.

1) Basic concepts: Real-time tasks are the basic executable entities that are scheduled. We base our work on the classical periodic task model: The release time $r_i$ is the time when
a task $\tau_i$ becomes ready for execution. The deadline $d_i$ is the time when a task should be completed. The computation time $C_i$ is the worst-case execution time of a task at nominal processor frequency. The start time $s_i$ is the time a task begins its execution. The finishing time $f_i$ is the time a task completes its execution. $T_i$ is the period of a task.

The assumptions made in this paper are that all real-time tasks are periodic, independent and only exhibit relatively small variations in execution time. Task deadlines are assumed to be equal to task periods. All periods are assumed to be either harmonic or chosen in such a way that they do not yield a hyperperiod of unreasonable length.

Tasks are, in our case, statically scheduled using rate monotonic scheduling. We assume absence of release jitter. No offsets were used in the practical evaluation, although they are possible to combine with the algorithm.

We finally assume that the number of tasks is small, and although the majority of the tasks are hard real-time, at least the lowest priority task has softer real-time characteristics, and can accept a small number of deadline overruns.

### B. Dynamic voltage frequency scaling

Dynamic voltage frequency scaling (DVFS) refers to the possibility of reducing the clock frequency of the CPU during runtime, thus enabling the processor to operate at a lower voltage. Power consumption in complementary metal-oxide semiconductor (CMOS) integrated circuits (ICs) can be classified into two different categories; dynamic power consumption and static power consumption. Dynamic power consumption is the power consumption in a circuit while it is operating (e.g. switching), and static power consumption is the power consumption while it is not operating but still powered (e.g. non-switching steady state or transistor-off state).

The power consumption of a micro-electric chip, such as a CPU, is dominated by the dynamic power dissipation $P_d$ of the CMOS transistors. $P_d$ is given by $P_d = C_{eff} \cdot V_{dd}^2 \cdot f$, where $C_{eff}$ is the effective switched capacitance and $f$ is the frequency of the clock. The maximum attainable frequency is limited by the gate delay $D$ of the transistors. The gate delay is inversely related to the supply voltage, as given by the formula $D = k \cdot V_{dd}/(V_{dd} - V_t)^2$, where $k$ is a constant and $V_t$ is the threshold voltage. Hence, in CMOS circuits, the cost of switching can be lowered by reducing the supply voltage at the price of a lower maximum attainable operating frequency [1].

### II. RELATED WORK

Much work has been performed on the problem of real-time dynamic voltage scaling. Real-time task sets are usually specified with worst-case execution times, but often use much less than their worst case times to complete their execution [3]. This makes it possible to reduce energy consumption by reducing the operating frequency and still meet system deadlines.

As pointed out by Bunde et al. [4], in the context of static DVS scheduling, there is a range of applications, with on the one hand applications with very harsh power constraints (e.g. sensor networks), while, on the other hand, laptops with high performance requirements and smaller constraints on power. This trade-off appears also in the case of dynamic scheduling. There is a large range of scheduling approaches that lie somewhere on the Pareto curve between these two extremes, that all can be called power-aware.

Many articles about DVFS scheduling [5][6][7] refer to one specific article by Pillai and Shin [8]. They presented three different techniques for scheduling real-time tasks on a CPU with support for DVFS.

Static Voltage Scaling (SDVS) [8] under EDF or RM sets the processor frequency to the lowest possible value that still guarantees that all system deadlines will be met. Under SDVS, the clock frequency does not change during runtime. The operating frequency is scaled by a factor $\alpha$, $(0 < \alpha \leq 1)$. As processors usually only have a finite number of frequency settings, the closest frequency above this value is chosen. The voltage is changed to match the operating frequency and reduce energy consumption as much as possible. The SDVS algorithm for RMS is described in Algorithm 1.

---

#### Algorithm 1 Static Voltage Scaling (SDVS) under RM

1: RM-test($\alpha$)
2: if $\forall \tau_i \in \{\tau_1, \ldots, \tau_n \mid T_1 \leq \ldots \leq T_n\}$
3: $\frac{T_i}{T_1} \cdot C_1 + \ldots + \frac{T_i}{T_1} \cdot C_i \leq \alpha \cdot T_i$ then
4: return TRUE
5: else
6: return FALSE
7: end if
8: use lowest frequency $f_t \in \{f_1, \ldots, f_m \mid f_1 < \ldots < f_m\}$
9: such that RM-test($f_t/f_m$) is true.

---

In Cycle-Conserving DVS [8], the schedulability test is performed after every task completion using the actual execution times instead of the worst case times. If slack is detected, the operating frequency can be reduced. With this algorithm, the frequency changes during runtime.

Look-Ahead DVS [8] builds on the Cycle-Conserving DVS method and attempts to further reduce the operating frequency by deferring work to a later point in time. The work deferral may result in the system running at a higher speed at a later time, to be able to finish all work in time. Actual execution times of tasks are often much smaller than their worst-case values and the processor may not need to work at a higher speed to finish in time. This algorithm allows the system to operate at a low frequency while completing all tasks in time. Look-Ahead DVS better estimates how much slack is available and results in lower power consumption than both Cycle-Conserving DVS and Static DVS, according to [8][9][10].

Feedback DVS was developed by Zhu and Mueller [11]. Their scheme is a further development of Look-Ahead DVS. The scheduler divides the task into two parts based on the actual execution times of the tasks. The idea is to start with
as low frequency as possible and schedule execution at the maximum frequency in the future to ensure that the task is finished before its deadline. The difference from Look-Ahead DVS is that the partitioning of tasks into two parts is fine-tuned during runtime. The partitioning is initially based on the tasks’ worst-case execution time. The feedback mechanism attempts to perform the division based on actual execution times, so that ultimately all of the execution takes place during the low-frequency part of the task.

Zhu and Mueller performed a comparative study of the power saving potential of SDVS, Look-Ahead DVS and Feedback DVS. In their study, they implemented the DVS algorithms in a system scheduled according to Earliest Deadline First. They compared the performance of the algorithms on three different task sets. The largest power savings were recorded for a CPU load of 58%. The result of the study are shown in table I. The naive algorithm refers to running the system at the maximum frequency.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Power consumption</th>
<th>Absolute power saving</th>
<th>Power saving vs. SDVS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive</td>
<td>4.47 mW</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SDVS</td>
<td>3.20 mW</td>
<td>28.4 %</td>
<td>0 %</td>
</tr>
<tr>
<td>CC DVS</td>
<td>2.38 mW</td>
<td>46.6 %</td>
<td>25.6 %</td>
</tr>
<tr>
<td>LA DVS</td>
<td>2.21 mW</td>
<td>50.6 %</td>
<td>30.9 %</td>
</tr>
<tr>
<td>Feedback DVS</td>
<td>2.04 mW</td>
<td>54.2 %</td>
<td>36.3 %</td>
</tr>
</tbody>
</table>

TABLE I
MEASURED RESULTS DEPENDING ON ALGORITHM, AS PRESENTED BY ZHU AND MUELLER.

Aydin et al. [12][13] have since further improved upon the algorithms supplied by Zhu and Mueller, giving further optimized power savings in the cases where Zhu and Mueller only gave approximate solutions, however at the cost of a more involved algorithm and the necessity for parameter tuning: the algorithm’s aggressiveness in lowering clock frequency needs to be adjusted to be suitable for the dynamics of the tasks in the task set to be used in the actual system.

All the above described work have in common, that only hard real-time tasks are considered. Considerable work is also performed on soft real-time systems. As the notion of the word “soft” often can vary, several different approaches can be found in this area, out of which some examples are described below.

Soria-Lopez et al. [14] apply feedback control to the problem. By controlling the worst-case utilization ratio of all tasks, the energy savings ratio can be controlled to reach a certain value.

Bautista et al. [15] use heuristic mechanisms to schedule and allocate tasks on a multiprocessor, by, if necessary, increasing the clock frequency when a task arrives and decreasing when tasks finish.

Scordino and Lipari [16] apply resource reservation techniques and hierarchical scheduling in their algorithm GRUB-PA. A scheduling server is allocated to each task given the parameters deadline and virtual time. The utilization for each task is calculated, and the speed of the processor is scaled according to the total reservation of CPU capacity. Hence, real-time performance guarantees are given for each task even if they do not conform to the common periodic task model. Notable is that the paper in question has an approach that allows both hard and soft real-time tasks, which otherwise is less common. Tianzhou et al. [17] provide another reservation-based technique.

Quan et al. [18] built a power-aware scheduling algorithm based on the notion of (m,k) constraints, i.e. that out of k jobs, at most m may miss their deadline. They do this by finding a set of mandatory jobs, and execute them at a higher clock frequency. Dynamic reclamation is additionally used to use unused cycles when jobs finish early, executing jobs prematurely at a lower frequency.

Yuan and Nahrstedt [19][20] worked on characterization of typical multimedia tasks, showing that although the execution times of individual jobs vary rapidly, the distribution of the execution times typically change rather slowly. They employ this result to build a stochastic scheduling algorithm. Based on measurements giving the expected execution time distribution, they can use this information to set an appropriately chosen execution speed, such that, statistically, the expected ratio of jobs finishes before their deadline.

Harada et al. [21] considers a task model where each job consists of two parts: one mandatory and one optional. Based on the amount of time dedicated to the optional part, different Quality of Service (QoS) levels can be reached, and the paper aims to give all tasks an equivalent QoS performance. There are many such solutions at different Pareto-optimal tradeoffs between power and performance.

III. DEADLINE OVERRUN DETECTION DYNAMIC VOLTAGE FREQUENCY SCALING (DOD-DVFS)

The DOD-DVFS scheduler presented in this article is divided into two parts. Initially, the SDVS algorithm is used to calculate the lowest CPU frequency that guarantees all task deadlines using the WCET of the task set. The algorithm then decreases the CPU frequency further at run time for the lowest prioritized task, by using a heuristic dynamic deadline-overrun detection mechanism. The DOD-DVFS algorithm is triggered to run every time a job is released.

The highest prioritized tasks are run at the frequency computed from SDVS. The frequency of the lowest prioritized task is dynamically changed. The scheduler keeps track of which period each task is executing in, in relation to the hyperperiod. The scheduler uses a simple heuristic for changing the frequency by decreasing the frequency one step every new period. If the scheduler detects a deadline overrun, it will schedule the next job at the same period in relation the hyperperiod at a higher frequency. For example, task three executes three times during the hyperperiod. If it completes its execution after its deadline for the second period relative to the hyperperiod, the operating frequency is catalogued as being too low for task three, period two. The next time task three executes its second period in relation to the hyperperiod, it will run at a higher frequency.
Algorithm 2 Deadline Overrun Detection Dynamic Voltage Frequency Scaling (DOD-DVFS)

\[ f_i \] is the currently used frequency  
\[ f_m \] is the frequency determined through SDVS  
\( i \) is the index of the current frequency (1 ≤ \( i \) ≤ \( m \))  
\( j \) is a job counter within the current hyperperiod (0 ≤ \( j \) ≤ \( \frac{\text{hyperperiod}}{\text{period}} \))

1: **Initialization**()  
2: \( i \leftarrow m \)  
3: \( j \leftarrow 0 \)  
4: for all \( k : 0 \leq k \leq \frac{\text{hyperperiod}}{\text{period}} - 1 \) do  
5: \( \text{MinFreq}[k] \leftarrow 1 \)  
6: end for  
7: **NewFrequency**()  
8: \( j_{next} \leftarrow (j + 1) \mod \frac{\text{hyperperiod}}{\text{period}} \)  
9: if deadlineOverrun then  
10: \( \text{MinFreq}[j] \leftarrow \min(i + 1, m) \)  
11: \( i_{next} \leftarrow \max(\text{MinFreq}[j], \text{MinFreq}[j_{next}]) \)  
12: else  
13: \( i_{next} \leftarrow \max(i - 1, \text{MinFreq}[j_{next}]) \)  
14: end if  
15: **Switch frequency to** \( f_{i_{next}} \)  
16: \( i \leftarrow i_{next} \)  
17: \( j \leftarrow j_{next} \)

After a detected deadline overrun, the scheduler momentarily increases the frequency by one step to avoid a domino effect of deadline overruns among subsequent jobs of the task. The algorithm continues to run throughout the operation of the system. During evaluation the system typically stabilized after a few hyperperiods.

Algorithm 2 describes the operation of the (DOD-DVFS) algorithm. Each execution instance of a task \( 3 \) is referred to as a job numbered \{0-\( j \} \). The goal of the algorithm is to determine the lowest allowable frequency for each job. The allowable frequencies in the system are stored in the array \( \text{MinFreq} \) where index \( 1 \) designates the lowest frequency and \( m \) the highest. The initialization (row 1-6) is run at system startup.

Row 2: The frequency set by index \( i \) is initially the frequency determined through SDVS.
Row 3: The job counter \( j \) is initially set to zero.
Row 4: The minimum frequencies index for all jobs is set to one. Initially, all jobs are allowed to run at the lowest frequency.
Row 7: **NewFrequency**() is called at the release time of every new job.
Row 8: The job counter \( j_{next} \) is set to the following job.
Row 9: A deadline overrun is said to have occurred if the previous job has not returned before the release time of the next job.
Row 10: The minimum allowable frequency of the previous job is increased by one step in the allowable frequency array.
Row 11: The frequency for the job being released is temporarily increased for the next execution to avoid a domino effect of deadline overruns.
Rows 12-14: If no deadline overrun is detected, the frequency for the job being released is decreased by one step, unless disallowed by the allowable frequency array.
Row 15: The job- and frequency values for the job being released is set.

IV. SYSTEM DESIGN

This section describes the implementation of a real-time scheduler incorporating the proposed DOD-DVFS algorithm.
We will discuss the hardware and software platform and describe the architecture of the system.

A. Hardware platform

The DOD-DVFS algorithm was implemented on the Freescale i.MX31 ADS [22]; a multimedia development platform which consists of a baseboard, a CPU board and a power management board. The CPU board is equipped with an i.MX31 ARM11 MCU. The CPU board has a number of two-pin connections that can be used to measure the energy consumption of different parts of the system. The primary target of the i.MX31 processor is the mobile device market; thus, a lot of thought and effort has been invested to provide mechanisms for optimization of system energy performance and extended operating times for mobile devices based on the processor.

Frequencies in the range of 99-399 MHz are attainable as long as the core voltage remains above 1.20 V. However, to attain higher frequencies, in the range 400-532 MHz, the voltage must be set to 1.40 V. The power consumption at different frequencies on the i.MX31 with a CPU load of 100% was measured and is shown in Figure 1a. As can be seen, the power consumption increases linearly with increasing frequency if the supply voltage of the CPU is kept constant. Figure 1a also shows the jump in power consumption that occurs when the transition to frequencies over 399 MHz is made, as a result of the supply voltage being increased to 1.40 V. Measurements were also performed of the power consumption in the idle state and found to be marginally smaller.

There are two methods for controlling the frequency on the i.MX31; the PLL method and the PD method. The PLL method refers to controlling a Phase-locked loop (PLL) which is a hardware feedback mechanism used to generate an output frequency that has a fixed relation to the phase of a reference signal. The PD method refers to scaling the CPU input frequency with a post divider (PD). Voltage scaling on the i.MX31 platform is accomplished by communicating over a Serial Peripheral Interface (SPI) bus to the external MC13783 power management integrated circuit on the i.MX31 ADS.

The PLL method of frequency switching was rejected for dynamic frequency scaling, due to hardware-introduced delays of roughly 500 µs [23]; this was deemed unacceptably long for our test system. The decision was made to keep the PLL value fixed during real-time execution and scale the CPU frequency using the post divider method. The timing measurements from frequency scaling with the post divider mechanism is presented in Figure 1b. As can be seen, the change in frequency from 100 MHz to 200 MHz took on the order of 10 µs most of the time. However, this pattern was sporadically interrupted by high peaks in execution time. The longest measured delay time was 55 µs. The switches to and from other frequencies had a similar time profile. It is not clear what causes this behavior, but possible culprits are internal hardware mechanisms in the CPU or unknown code execution in the operating system.

B. Software platform

OSE 5.4 [24] is a preemptive, fixed-priority real-time operating system developed by ENEA AB [25]. One of the primary markets for OSE is communication applications on mobile platforms.

OSE 5.4 incorporates a software structure called Bios which offers basic services for use of the system call mechanism. The system call mechanism allows programs to call the executive or other components that have registered with Bios, without knowing the address that the executive or component is located on. It also provides a way for user mode code to switch to supervisor mode in a controlled way. Whenever an application executes a system call instruction, the execution jumps to the Bios component. OSE allocates the CPU to the idle process when no other process is running.

C. Deployment

The Frequency and voltage scaling algorithm was implemented in a process with the highest priority in the system. Frequency and voltage scaling was implemented in a fixed-priority OSE process which controls the underlying DVFS hardware through Bios calls. Tasks are implemented as fixed-priority processes with priorities set according to rate monotonic scheduling, i.e. with the highest priority assigned to the process with the shortest period.

V. Evaluation

By measuring the supply voltage and the voltage drop over a resistor in series with the CPU on the i.MX31 ADS board, power consumption can be calculated. The system is run with a task set with a hyperperiod of 24 ms described in Table II.

The tasks used were implemented as straight-forward loops, representative of tasks executing the same trace every job instance. Due to this way of implementing the used tasks, they had a relatively low execution time variance. Out of the three tasks, task 3 is the only dynamically scaled task in the system. Task 3 runs three times during the hyperperiod.

Table III describes the system settings depending on the algorithm running. The naive approach runs the CPU at the maximum frequency of 532 MHz which results in a core voltage of 1.40 V. This is the default settings of the used board. The SDVS algorithm lowers the static frequency to 399 MHz to reduce system idle time and the core voltage is adjusted to 1.20 V. The DOD-DVFS algorithm utilizes dynamic frequency scaling during runtime. Table IV shows the power savings depending on the algorithm running.

\(^1\)Note that the OSE Bios component has nothing to do with the Basic Input/Output System (BIOS) term in the desktop PC world.
Figure 2 illustrates the schedule resulting from the naive approach. As can be seen, all tasks execute at 532 MHz. System idle time constitutes 48% of available CPU capacity. Figure 3 shows the schedule resulting from the SDVS algorithm. As can be seen, all tasks execute at 399 MHz and idle time in the system is reduced to 40%. Figure 4 shows the schedule resulting from the DOD-DVFS algorithm. As can be seen, the CPU frequency is dynamically changed over the course of the hyperperiod, and as a result, system idle time is almost eliminated.

For our test case, the adaptation caused only a single deadline overrun in the initial two hyperperiods, after which the system had stabilized, illustrated in figure 5. In the general case, no hard guarantee can be given on the number of deadline overruns. The worst-case scenario is a domino situation, where a missed deadline leads to additional ones in the subsequent jobs, despite the heuristic approach of increasing clock frequency in the next job after an overrun. To give more precise guarantees, further assumptions would be needed on the execution time of the task, e.g., statistical distribution, or how it varies under frequency scaling.

### A. Discussion

Compared to SDVS, a 31% reduction in power consumption was seen. This is an impressive result when compared to the power savings achieved over SDVS utilizing more complicated
implementations as measured by Zhu and Mueller. Even though the test cases are not the same, it is an indication of the potential power saving. By making a simple comparison between the values in the rightmost column power saving vs. SDVS in Tables IV and I DOD-DVFS surpasses CC-DVS (25%), a DVFS implementation of comparable complexity, and delivers power savings comparable to much more complicated DVFS schemes such as Look-Ahead DVS (32%) and Feedback DVS (36%). An important difference between these other approaches compared to DOD-DVFS, is that the latter explicitly utilizes the fact that one task only has soft real-time requirements.

VI. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we present a deadline overrun detection DVFS (DOD-DVFS) algorithm with the goal of facilitating the implementation of DVFS in real-time system with mixed hard and soft deadlines. We show that the DOD-DVFS algorithm can produce energy savings over SDVS of 31%. This is comparable to power savings documented for more complex DVFS algorithms, and shows that even simple heuristic adaptation offers significant advantage over just SDVS.

Although our approach relies on having a soft real-time lowest priority task, we believe this is a common situation in the real world, e.g. in cell phones where hard real-time communication is combined with soft real-time multimedia. The complexity of a DVFS algorithm is important in an industrial setting. It is not even certain that well-defined WCET values are known at all in an industrial setting. The adaptation part of DOD-DVFS may be applied even though an optimal starting frequency has not been found, nor WCET values are known. The simple heuristic adaptation mechanism used in this work is easily implemented, and has low requirements on understanding of how tasks behave at run time.

A. Future Work

Work should be performed to extend the scheduling algorithm with support for dynamic process creation. If a new real-time process is created during run time, the scheduler needs to reschedule the process set dynamically.

Further, if the soft real-time task has partially hard requirements, as for example that at most $m$ out of $k$ subsequent deadlines may be missed, the current algorithm can not fulfill them. Although heuristics increase scheduling frequency if deadline overruns occur, there is no guarantee that subsequent jobs do not miss their deadline. Other alternative adaptation mechanisms may be explored to find ways to give such a guarantee. A more detailed scheduling analysis using additional information about the tasks, in each specific case, could also give more detailed guarantees.

The current version of DOD-DVFS has been developed for use with one soft task subjected to runtime adaptation only – the one with lowest priority. If no single such task exists, i.e. the least prioritized task has a very low utilization, deadline overrun detection and adaptation would have to be applied to several tasks. There are two principles of how to do the change: either each task performs adaptation independently, or they do it jointly. In the former case, there is a risk of conflict between tasks\(^2\), leading to a suboptimal result. There is however no need to change the algorithm itself, only to use several copies of persistent variables, one for each task. On the other hand, if adaptation is performed jointly, the DOD-DVFS algorithm has to be extended somehow. Further evaluation is needed to compare these two approaches, and also for the usage of the current algorithm with other load combinations and hardware platforms. Further work is also needed to extend

\(^2\)All tasks subjected to adaptation may have deadline overruns at approximately the same time, hence causing clock frequency increases. This may however be a case of over-compensation; it might have been sufficient if adaptation would have been applied to a single task only.
the algorithm to take low-power sleep states of the processor into account.

Finally, the DOD-DVFS algorithm could be adapted for usage with dynamic priority assignment, especially earliest deadline first. With dynamic instead of static priority scheduling, the CPU utilization can be increased and the system’s ability to reduce CPU idle time is improved.

REFERENCES


N-ary Sensor Model for Target Tracking in Wireless Sensor Networks

Kenji Tei†‡, Hikotoshi Nakazato*, Yoshiaki Fukazawa* and Shinichi Honiden†‡

*Waseda University, Nishiwaseda 1-6-1, Shinjuku-ku, Tokyo, Japan
†National Institute of Informatics, Hitotsubashi 2-1-2, Chiyoda-ku, Tokyo, Japan
‡the University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan

Abstract—Target tracking is a representative application in wireless sensor networks. To achieve higher resolution of trajectory, a target tracking application will consume much amount of energy, caused by data transmissions, whereas the application should save energy consumption. This motivates that the application should be able to adjust balance between the resolution and resource consumption. In this paper, we propose N-ary sensor model to adjust the balance, in order to enable developer to choose appropriate balance to satisfy requirements for the application. Moreover, we propose heterogeneous configuration algorithm which configure the sensor model for each node, to reduce unnecessary transmissions to achieve the required resolution.

Keywords—Wireless Sensor Networks; Target Tracking; Sensor Model;

I. INTRODUCTION

A Wireless Sensor Network (WSN)[1] is a network consisting of small computers, called nodes, equipping with multiple sensors and communicating each other via wireless links. Advantages of the WSN are programmability and easy deployment: a WSN can be programmed to measure various kinds of phenomena, and it can cover large filed with low cost since it does not require any wired connections. The WSN will be an infrastructure to measure physical phenomena, used by various kinds of context-aware systems, such as cyber physical systems.

Despite its usefulness, WSNs have strict resource limitation. Nodes are usually powered by limited battery. To maintain a WSN for long periods of time, energy consumption in the network has to be saved. In particular, network communication is the dominant factor of energy consumption[2]; the ratio between the energy needed for transmitting and for processing a bit of information is usually assumed to be much larger than one (more than one hundred or one thousand in most commercial platform)[3]. For example, the commonly used Crossbow Mica2 node will deplete its battery on an average of 7 days when reading its temperature sensor value and transmitting it every second[4]. Therefore, reduction of unnecessary data transmission is an important requirement for WSN applications, to prolong lifetime of the WSN.

Among many kinds of applications in WSNs, we focus on target tracking, which is a representative application in WSNs[5]. The target tracking is needed by context-aware systems in various kinds of domains, such as user navigation in smart environments, home automation, supports in battlefields, and so on. The goal of the target tracking is to trace a trajectory of a moving target in a field covered by a WSN. To track a trajectory, the target tracking application periodically estimates location of the target from datum retrieved from nodes, and estimates trajectory of the target from a history of the estimated locations.

Reducing transmissions caused by the data retrieval will improve energy efficiency of target tracking. For example, [6] proposed a sleep scheduling of nodes, and [7] proposed a sensor selection algorithm. These works tried to reduce the number of nodes being active state, in which a node consumes much energy than in sleep state. [8], [9] try to reduce the number of transmissions. [8], [9] proposed binary sensor model to reduce transmissions occurring in the sensory data retrievals in each period. Binary sensor model maps sensory data to binary value: it outputs 0 when sensory data is lower than a threshold, and 1 when sensory data is higher than one. With the binary sensor model, each sensor transmits an output, only when the current output is different from the previous one. Intuitively, target tracking based on the binary sensor model ignores slight move of the target. Therefore, the binary sensor model can reduce the transmission at the expense of reduction in the resolution of tracking, compared with conventional sensor model, which output directly sensory data.

However, the both of sensor models cannot adjust the tradeoff between the resolution and the transmission cost. Therefore, a developer of target tracking cannot adjust the resolution and the transmissions to non-functional requirements for the application.

In this paper, we propose a sensor model to enable adjustment of the tradeoff, called N-ary sensor model. The N-ary sensor model generalizes the binary sensor model. It maps sensory data to N-valued output, where the arity N is positive integer which is bigger than one. When the arity N becomes small, the resolution will get worse, but transmissions will decrease. When it becomes big, vice versa. A developer can adjust the tradeoff by setting arity N to an adequate value. Moreover, we propose a heterogeneous configuration algorithm to configure the arity N of each node. According to network density, a heterogeneous configuration, which configures suitable arity N for each node, could reduce more transmissions within the required...
resolution, compared with a homogeneous configuration, which allocating same arity to all nodes. The simulation result shows that the N-ary sensor model with the heterogeneous configuration can save transmission within required resolution in a realistic environment where nodes are placed nonuniformly.

The rest of this paper is organized as follows. Section 2 describes more detailed formulation of target tracking and existing sensor models. Section 3 describes the detail of the N-ary sensor model and configuration algorithm for the N-ary sensor model. Section 4 describes evaluation based on simulation result, and section 5 concludes this paper.

II. SENSOR MODELS FOR TARGET TRACKING

In this section, we describe the detailed of target tracking and existing sensor models.

A. Target tracking

An overview of a WSN is illustrated in Figure 1. A target moves around a two-dimensional field where there are \(q\) nodes \(\{s_1, s_2, ..., s_q\}\) consisting a WSN. Nodes know their own location \(\{L_{s_1}, L_{s_2}, ..., L_{s_q}\}\), by executing some localization method, such as a method proposed in [10], on ahead. There is no assumption as to placement of nodes, except that every node has at least one routing path to the manager node of the WSN, called basestation. Every node equips with distance sensor and can estimate distance to the target.

![Figure 1. An overview of a WSN](image)

Target tracking application periodically estimate location of target by doing procedure as follows. For every time interval, nodes measure distance to the target. A sensory data measured by \(s_i\) \((1 \leq i \leq q)\) at the time \(t\) is denoted as \(d_{s_i,t}\). After the measurement, a node calculates an output data from the sensory data according to a sensor model. Let a sensor model be a function \(f\), an output of \(s_i\) at time \(t\) is denoted as the equation (1),

\[
o_{s_i,t} = f(v_{s_i,t})
\]  

(1)

Each node transmits its output and its location to the basestation, only when the output is different from the previous one stored in its memory. More specifically, \(s_i\) transmits data tuple (\(o_{s_i,t}, L_{s_i}\)) to the basestation, only when \(o_{s_i,t} \neq o_{s_i,t-1}\). Whether a node transmits its output or not, the node stores the output in its memory. The basestation will receive a set of tuples from a part of nodes. For all outputs which the basestation has not received, previous outputs is used as current output. Finally, the basestation gets a set of the tuples of all nodes \(\{(o_{s_1,t}, L_{s_1}), (o_{s_2,t}, L_{s_2}), ..., (o_{s_q,t}, L_{s_q})\}\), and estimate location of the target at the time. Let location of the target at the time \(t\) be \(l_t\), \(l_t\) is calculated according to the target localization method.

Commonly used target localization methods are the Received Signal Strength Indicator (RSSI), the Time of Arrival (ToA), and the Time Difference of Arrival (TDoA), by using radio-frequency sensor. For example, sensory data is signal strength of received radio sent by a beacon device a target wears, in the case of the RSSI method, and difference of time between the time when the sensor receives a radio and one when the beacon send the radio, in the case of the ToA method.

From a history of estimated location of the target, the basestation achieves a trajectory of the target.

B. Range sensor model

Range sensor model outputs a sensory data directly. More specifically, a function \(f\) for the range sensor model can be defined as equation (2).

\[
f(v_{s_i,t}) = v_{s_i,t}
\]  

(2)

For each output, target tracking application converts received an output data into distance to the target. A distance from \(s_i\) to the target at the time \(t\) is denoted as \(d_{s_i,t}\). From a set of data tuple consisting of \(d_{s_i,t}\) where \(1 \leq i \leq q\), target tracking application estimates location of the target.

![Figure 2. Target tracking based on the range sensor model](image)

Consider a scenario illustrated in Figure 2. There are 4 nodes \(\{s_1, s_2, s_3, s_4\}\) in a field. At the time \(t\), the target tracking application receives data tuples \((o_{s_1,t}, L_{s_1})\), \((o_{s_2,t}, L_{s_2})\), \((o_{s_3,t}, L_{s_3})\), and \((o_{s_4,t}, L_{s_4})\). The application converts output datum into distances to the target, and achieves data tuples \((d_{s_1,t}, L_{s_1})\), \((d_{s_2,t}, L_{s_2})\), \((d_{s_3,t}, L_{s_3})\), and \((d_{s_4,t}, L_{s_4})\). The application estimates location of the target.
at the time \( t \), denoted as \( l_t \), from these tuples. Intuitively, estimated location of a target is an intersection of all circular arcs whose center is at the location of a node and whose radius is estimated distance. At the time \( t+1 \), the application repeats same procedure again, and gets \( l_{t+1} \). Finally, the application achieves a trajectory of the target, from a history of the target locations \( (l_1, l_2, ..., l_{t+1}) \).

For example, in the case of the RSSI method, each node outputs radio signal strength. Target tracking application converts the radio signal strength into distance to the target according to a graph between RSSI and distance, drawn from a theoretical signal propagation model or plotted by experimental results, such as results shown in [11].

C. Binary sensor model

Binary sensor model maps sensory data to a binary value, and outputs it. More specifically, given a \( h \), a function \( f \) for the binary sensor model can be defined as equation (3).

\[
f(v_{s_i}, t) = \begin{cases} 
0 & (v_{s_i}, t < h) \\
1 & (v_{s_i}, t \geq h)
\end{cases}
\tag{3}
\]

For each output, target tracking application converts received an output data into a region at which the target is located. The region is described as a circle whose center is a location of a node which produces the output data, and whose radius is a distance associated with the threshold of sensory data. When a node outputs 1, the target will be inside the region of the node. On the contrary, when it outputs 0, the target will be outside the region. The target tracking application estimates the region based on all outputs. The target will be inside an intersection of regions of a node which outputs 1, and be outside of all regions of node which outputs 0.

\begin{table}[h]
\centering
\caption{Outputs in a scenario}
\begin{tabular}{|c|c|c|c|c|}
\hline
Time & \( t \) & \( t+1 \) & \( t+2 \) & \( t+3 \) \\
\hline
output of \( s_1 \) & 1 & 1 & 0 & 0 \\
output of \( s_2 \) & 0 & 1 & 1 & 1 \\
output of \( s_3 \) & 1 & 1 & 0 & 0 \\
output of \( s_4 \) & 0 & 0 & 0 & 1 \\
\hline
\end{tabular}
\end{table}

Consider a scenario where there are 4 nodes in a field, and outputs of each node at the time \( t, t+1, t+2, \) and \( t+3 \) are shown in Table I. Based on these outputs, the target tracking application estimates region, at which the target will be, for each periods of time. At the time \( t \), the target will be inside the circles of \( s_1 \) and \( s_3 \), and outside the circles of \( s_2 \) and \( s_4 \). The region at the time \( t \) is the region \( l_t \) illustrated in Figure 3. In the same way, the regions at the time \( t+1, t+2, \) and \( t+3 \) are the region \( l_{t+1}, l_{t+2}, \) and \( l_{t+3} \) in Figure 3, respectively. Finally, the target tracking application achieves a trajectory of the target, from a history of the regions \( (l_1, l_2, ..., l_t, l_{t+1}, l_{t+2}, l_{t+3}) \).

D. Problems of existing sensor models

Compared with the range sensor model, the binary sensor model will reduce transmissions from nodes to a basestation. The binary sensor model maps sensory data to binary value. This means that the binary sensor model ignores slight changes in sensory data, whereas the range sensor model is sensitive to the changes.

For example, consider the scenario described in Table I. In this scenario, \( s_1 \) does not transmit its output at the time \( t+1 \) and \( t+3 \), since the output at the time is same with the previous output. Similarly, \( s_2 \) does not transmit at the time \( t+2 \) and \( t+3 \), \( s_3 \) does not transmit at the time \( t+1 \) and \( t+3 \), and \( s_4 \) does not transmit at the time \( t+1 \) and \( t+2 \). However, if the range sensor model is applied in the same situation, all nodes will transmit its output for every time it senses.

On the contrary, the binary sensor model sacrifices resolution. A trajectory achieved by the binary sensor model is represented as a history of regions, whereas one achieved by the range sensor model is represented as a history of locations. Therefore, resolution of trajectory achieved by the binary sensor model will be lower than one achieved by the range sensor model.

Appropriate sensor model depends on requirements for target tracking application. When an application has to satisfy high resolution, the range sensor model will be suitable. When it can accept low resolution, the binary sensor model will be suitable from the point of view of energy efficiency. A developer is responsible for deciding a suitable sensor model for the application.

However, existing works only provide a few alternatives: the range sensor model or the binary sensor model. There are no support to adjust a balance between the energy consumption and the resolution, in the existing sensor models. Consequently, a developer cannot sensitively adjust the balance, by selecting a sensor model.
III. N-ARY SENSOR MODEL

In this section, we describe the detail of N-ary sensor model, and an algorithm to determine suitable configurations of the N-ary sensor model for each node.

A. N-ary sensor model

N-ary sensor model is a generalization of the binary sensor model. Whereas the binary sensor model maps sensory data into binary value, the N-ary sensor model maps it into N-ary value, where the arity $N$ is an integer bigger than one. More specifically, given $N-1$ thresholds denoted as $h_1, h_2, \ldots, h_{N-1}$, where $h_i < h_j (1 \leq i < j \leq N)$, a function $f$ of the N-ary sensor model is defined as equation (4).

$$f(v_{s_i}, t) = \begin{cases} 
0 & (v_{s_i}, t < h_1) \\
1 & (h_1 \leq v_{s_i}, t < h_2) \\
\cdots \\
N-2 & (h_{N-1} \leq v_{s_i}, t < h_N) \\
N-1 & (h_N \leq v_{s_i}, t) 
\end{cases}$$

When $N = 2$, the N-ary sensor model is same with the binary sensor model, and when $N \rightarrow \infty$, it will be quite similar to the range sensor model.

One can provide some kinds of definition of the thresholds. One of the definitions is that thresholds are set to equally divide a radius of a circle where a node can sense. Intuitively, the N-ary sensor model can be represented by $N-1$ circles whose centers are a location of a node. According to the definition of thresholds, all differences between radiiuses of neighboring two circles are same. More specifically, given a function $g$ which converts a sensory data into distance, and distance $d_{\text{max}}$ which the maximum distance a sensor can sense, the thresholds $h_1, h_2, \ldots, h_{N-1}$ are defined to satisfy equation (5) and (6).

$$g(h_1) = d_{\text{max}}$$

$$|g(h_i) - g(h_{i+1})| = \frac{d_{\text{max}}}{N-1} \quad (2 \leq i \leq N-1)$$

Note that the left part of equation (6) is an absolute value of difference between radiiuses of neighboring two circles.

Figure 4 illustrates the cases that the N-ary sensor model is applied, where $N = 3$ and $N = 5$. In the left case in Figure 4, the node outputs 0, and in the right case, it outputs 5.

The arity $N$ will affect resolution and transmission of target tracking. Consider the cases illustrated in Figure 5. The both cases are same situation except that the arity $N$ is set to 3 in the left case, and 5 in the right case. In the both cases, there are two nodes $\{s_1, s_2\}$. At the time $t$, $t+1$, $t+2$, and $t+3$, the target is at the locations illustrated in Figure 5. Outputs of $s_1$ and $s_2$ at the time $t$, $t+1$, $t+2$, and $t+3$ in the both cases are described in Table II. The results show that $s_1$ in the case of $N = 3$ needs less transmissions than it in the case of $N = 5$. In the case of $N = 3$, $s_1$ transmits its output only at the time $t$, whereas in the case of $N = 5$, it transmits its output at the time $t$, $t+1$, and $t+3$. However, the results also show that the resolution of achieved trajectory in the case of $N = 3$ is higher than that in the case of $N = 5$. An achieved trajectory is represented by a history of intersection of regions of the two nodes. The intersections of regions estimated from outputs in the case of $N = 5$ are narrower than that in the case of $N = 3$.

![Figure 4](image1.png)  
**Figure 4.** Examples of N-ary sensor model ($N = 3, N = 5$)

![Figure 5](image2.png)  
**Figure 5.** Examples of N-ary sensor model ($N = 3, N = 5$)

<table>
<thead>
<tr>
<th>Time</th>
<th>$t$</th>
<th>$t+1$</th>
<th>$t+2$</th>
<th>$t+3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>output of $s_1 (N = 3)$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>output of $s_2 (N = 3)$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>output of $s_1 (N = 5)$</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>output of $s_2 (N = 5)$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

This scenario indicates that, when the arity $N$ becomes big, the target tracking application will become sensitive to changes of moves of a target. Therefore, the resolution would become high, but transmissions would increase. On the contrary, when it becomes small, the application will become insensitive. Therefore, the resolution would become
A developer can configure arity \( N \) of nodes, so as to reduce unnecessary transmissions to achieve required resolution. A configuration of tracking application based on N-ary sensor models is denoted as \( C_S = \{N_1, N_2, \ldots, N_q\} \) where \( N_i \) be an arity assigned to \( s_i \). Simple way to configure arity \( N \) of each node can be setting same arity \( N \) to all nodes:

\[
\forall N_i \in C_S : \forall N_j \in C_S : N_i = N_j \quad (7)
\]

We name the configuration as homogeneous configuration. An example of the homogeneous configuration has been shown in the scenario illustrated in Figure 5.

According to the homogeneous configuration, a developer can use the arity used for all nodes as a parameter to adjust balance between resolution and transmissions, as discussed in Section III-A.

\[\begin{align*}
\text{Sparse area} \\
\text{Dense area}
\end{align*}\]

Figure 6. Expected resolution in different density

However, in a field where nodes are placed nonuniformly, resolution achieved in a sparse part of the field will be different from one in a dense part. Consider the situation illustrated in Figure 6 where all nodes has same arity \( (N = 3) \) and they are placed uniformly. Resolutions achieved in the dense part will be higher than one in a sparse part, in average. In a sparse part where there are few intersections of regions covered by nodes, an appropriate arity \( N \) of a node should be big to satisfy required resolution. Meanwhile, in a dense part where there are many intersections, small arity \( N \) is enough to satisfy the required resolution. To achieve required resolution in entire field, an arity \( N \) should be configured to satisfy the required resolution in the most sparse part, but the arity \( N \) would be too enough for nodes in dense part. This would cause unnecessary transmissions to achieve required resolution.

This motivates us to introduce heterogeneous configuration for the N-ary sensor model. Configuring an appropriate arity \( N \) for each node would reduce unnecessary transmissions to achieve required resolution in entire field covered by nodes. Therefore we propose an algorithm to find an appropriate configuration.

Goal of the algorithm is to find a configuration to satisfy two conditions as follows.

1) all regions, divided by nodes with the arities in the configuration, satisfy the required resolution.
2) the sum of arities in the configuration is the smallest among the set of configurations satisfying the condition 1)

The first condition is related to resolution and the second one is related to transmissions. The algorithm for the heterogeneous configuration is described in Algorithm 1.

**Algorithm 1** Heterogeneous configuration for the N-ary sensor model

**Require:** \( L_S \): locations of nodes, \( \text{res}_{\text{required}} \): required resolution

**Ensure:** \( \text{result} \): the set of arities assigned to nodes

\[
\begin{align*}
1: & \quad \text{Confs}_{\text{satisfied}} \leftarrow \{\}\n2: & \quad \text{Confs}_{\text{all}} \leftarrow \text{producePossibleConfigurations}()\n3: & \quad \text{for all conf such that conf} \in \text{Confs}_{\text{all}} \text{ do}\n4: & \quad \text{regions} \leftarrow \text{getRegions}(\text{conf}, L_S)\n5: & \quad \text{isSatisfied} \leftarrow \text{true}\n6: & \quad \text{for all region such that region} \in \text{regions do}\n7: & \quad \text{if getResolution(region)} \geq \text{res}_{\text{required}} \text{ then}\n8: & \quad \text{isSatisfied} \leftarrow \text{false}\n9: & \quad \text{break}\n10: & \quad \text{end if}\n11: & \quad \text{end for}\n12: & \quad \text{if isSatisfied then}\n13: & \quad \text{Confs}_{\text{satisfied}} \leftarrow \text{Confs}_{\text{satisfied}} \cup \{\text{conf}\}\n14: & \quad \text{end if}\n15: & \quad \text{end for}\n16: & \quad \text{if Confs}_{\text{satisfied}} = \{\}\text{ then}\n17: & \quad \text{return} \quad \text{"There are no configurations to satisfy the required resolution."}\n18: & \quad \text{end if}\n19: & \quad \text{conf}_{\text{result}} \leftarrow \{\}\n20: & \quad n \leftarrow \infty\n21: & \quad \text{for all conf such that conf} \in \text{Confs}_{\text{satisfied}} \text{ do}\n22: & \quad t \leftarrow 0\n23: & \quad \text{for all arity such that arity} \in \text{conf do}\n24: & \quad t \leftarrow = \text{arity}\n25: & \quad \text{end for}\n26: & \quad \text{if } t < n \text{ then}\n27: & \quad \text{conf}_{\text{result}} \leftarrow \text{conf}\n28: & \quad n \leftarrow t\n29: & \quad \text{end if}\n30: & \quad \text{end for}\n31: & \quad \text{return conf}_{\text{result}}\n\end{align*}
\]

The algorithm mainly consists of two parts: the former...
part is the lines from 1 to 18, and the latter part is the lines from 19 to 31. The former part is responsible to satisfy the condition 1). This part tries to find a set of configurations satisfying the required resolution \( \text{res}_{\text{required}} \). \( \text{producePossibleConfiguration} \) is a function which returns all possible configurations. All possible configurations \( \text{Confs}_{\text{all}} \) is described as a set of configurations:

\[
\text{Confs}_{\text{all}} = \{ C_N | 1 \leq N_i \leq M \text{ for all } N_i \in C_N \},
\]

where \( M \) is the maximum value of an arity \( N \), determined by resolution of distance sensor device. In the lines from 3 to 15, the algorithm tries to find, from \( \text{Confs}_{\text{all}} \), a set of configurations satisfying the \( \text{res}_{\text{required}} \), named \( \text{Confs}_{\text{satisfied}} \), using \text{getRegions} function which takes a configuration and locations of nodes, and returns a set of regions divided by nodes with the configuration, and \text{getResolution} function which takes a region and returns resolution of the region. As shown in the lines from 16 to 18, if there are no configurations satisfying the required resolution, the algorithm throws an error to prompt a developer to change the required resolution to acceptable one, and terminates the execution.

The latter part is responsible to satisfy the condition 2). This part finds the configuration in which the sum of arities is the smallest among configurations in \( \text{Confs}_{\text{satisfied}} \). Finally, this algorithm outputs such a configuration \( \text{conf}_{\text{result}} \).

IV. Evaluation

In this section, we evaluate the N-ary sensor model and the heterogeneous configuration algorithm on results achieved from simulations.

A. Simulation setting

In these simulations, in a two dimensional square-shaped field whose size is \( 20m \times 20m \), a target walks along with the path illustrated in Figure 7. The target walks at the speed 3.2 km/hour which is average speed of human walking, in turn, it moves about 0.89 meters per a second. Moreover, it wears a beacon propagating radio.

In the same field, 10 nodes are placed randomly. They equip with a radio-frequency sensor, and estimate distance from the target based on radio signal propagated from the beacon, using the RSSI method. Roughly speaking, they can sense the target at most within 10 meters. They measure radio every 3 seconds, and get current output using a sensor model. When the current output is different from the previous one, the node transmits its output and location to the basestation.

Target tracking application receives outputs sent from nodes periodically, estimates location or region of the target at the time, and achieves a trajectory from a history of the estimated location or region. The simulation is performed for 300 seconds in the simulation time.

We repeated the simulation for 20 times with different node placements. For each simulation, we measured the total number of transmissions of all nodes and resolution of achieved trajectory. Note that the resolution of trajectory in these experiments is defined as dimensions of each region in the trajectory. Narrow dimensions mean that the application provides a high-resolution trajectory. The dimension is calculated by the Monte Carlo method[12].

B. Effect of N-ary sensor model

In this section, we evaluate effects of arity \( N \) to achieved resolutions and transmissions, in the homogeneous configuration. We ran 20 simulations for each arity \( N \) where \( 2 \leq N \leq 10 \). Note that the n-ary sensor model with \( N = 2 \) is same with the binary sensor model. The number of transmissions and average of resolutions are shown in Figure 8 and Figure 9, respectively.

From Figure 8 and Figure 9, it is showed that, according to the arity \( N \), the number of transmission increases, whereas the resolution decreases. For example, when \( N = 2 \) (same with the binary sensor model), the total number of
transmissions and the average of resolution are 87 and 13.1 $m^2$ respectively, whereas, when $N = 10$, they are 355 and 0.68 $m^2$ respectively. If the application requires that resolution has to be below 4 $m^2$ in average, a developer should configures the arity $N$ to be equal or bigger than 5, in this case. Moreover, configuring the arity $N$ to be small, much amount of transmissions can be saved. For example, transmissions in the case of $N = 5$ is 38% smaller than that in the case of $N = 10$. The arity $N$ can be used as a parameter to adjust the balance between them.

C. Effect of heterogeneous configuration

In this section, we evaluate effects of the heterogeneous configuration proposed in section III-B, compared with the homogeneous configuration. In section IV-B, average of resolution is used as a metric, since the homogeneous configuration cannot guarantee to satisfy the required resolution. To compare the both configuration methods, consider a scenario where resolution is required to be higher than 3.14 $m^2$. Note that the homogeneous configuration uses the arity $N$ as a parameter, but the heterogeneous configuration uses the required resolution $res_{required}$. For the homogeneous configuration, the same configurations to the simulations in section IV-B is used. For the heterogeneous configuration, a configuration is determined by the Algorithm 1 with $res_{required} = 3.14$.

The percentages of periods satisfying the required resolution are illustrated in Figure 10. The results of the heterogeneous configuration always satisfy the required resolution, whereas not every results of homogeneous configurations satisfy: only cases of $N = 9, 10$ satisfy the required resolution, the algorithm for the heterogeneous configuration is designed to always satisfy the required resolution. Figure 11 shows the total number of transmissions occurring in the same simulation. Compared with results of the homogeneous configurations with $N = 9, 10$, which satisfy the required resolution, the heterogeneous configuration reduces about 26.5 % and 29.5 % transmissions, respectively, since the heterogeneous configuration tries to minimize the transmissions within the required resolution.

The tradeoff between the total number of transmissions and average of resolution, achieved by the both configurations, are illustrated in Figure 12. In addition to the previous simulation, Figure 12 shows results of the heterogeneous configuration with $res_{required} = 4m^2, 5m^2, 6m^2, 7m^2$. Figure 12 shows that the heterogeneous configuration can reduce about 20 % transmissions providing same resolutions, in average. This indicates the heterogeneous configuration is effective to improve energy efficiency in a realistic environment where nodes are placed nonuniformly.

V. Conclusion

In this paper, we propose the N-ary sensor model and the heterogeneous configuration. The N-ary sensor model allows a developer of target tracking application to adjust balance between transmissions and resolution, whereas existing sensor model could not. Moreover, the heterogeneous configuration will reduce transmissions within the required resolutions, compared with the homogeneous configuration. Reduction of transmissions strongly contributes to improve-
ments of energy efficiency of target tracking. The N-sensor model with the heterogeneous configuration is effective in a realistic environment where nodes are placed nonuniformly.

The N-ary sensor model with the heterogeneous configuration can be applied to many kinds of distance sensor. In this paper, we applied a radio-frequency sensor with commonly used RSSI method, but we can easily apply a radio-frequency sensor with ToA, and TDoA methods. Moreover, it can be applied to with directional sensors such as camera sensor or infrared sensor, with a little changes to a function to determine intersections of region and a function to calculate dimension of a region.

In a near future, we will develop target tracking applications with our model to evaluate it in the real world where sensor noise strongly affects to the accuracy of achieved trajectories. Existing paper showed that the binary sensor model has more tolerance to the noise than the range sensor model[9]. We will evaluate the tolerance of the N-ary sensor model to the noise. Moreover, we try to propose dynamic configuration of the N-ary sensory model. Current configurations proposed in this paper are static. Therefore, nodes can not adapt to changes in environments, such as leaving nodes from a WSN, caused by battery exhaustions. To handle such a dynamic feature in a WSN, we will propose dynamic configurations for the N-ary sensor model.

ACKNOWLEDGMENT

This research was supported by the XAC Project in the GRACE Center at the National Institute of Informatics.

REFERENCES


