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) technology. This paper presents an epoch-based approach for energy management in resource-constrained WSNs that utilizing 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 (CPS), such as smart power grid, 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 Wireless Sensor Networks. Unlike current WSN systems, this new generation must incorporate powerful and performance sensitive processing power with energy harvesting. Given these, energy management techniques are of paramount importance. This makes the combined use of two energy saving 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 to develop 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 prediction can be conducted based on the knowledge of past harvesting records and characteristics of 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 highly time-varying energy replenishment. 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 the above 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. Current 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 to 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 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 solve it using an optimal algorithm. We 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 mixed set 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.

Many existing studies explored 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 improves 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 maximally utilize the harvested energy in order to maximize the amount of completed works, 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.

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, 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.

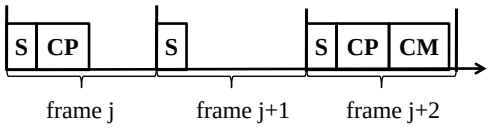

Figure 1. A sequence of frames of different types

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 rest of the paper.

We use a tuple $\{I_j^{sen}, I_j^{cp}, I_j^{cm}\}$ 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. 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 processed, 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 ([3]). Each modulation

level b_i is associated with a communicate speed d_i.

$$d_i = R \cdot b_i \tag{1}$$

R is the symbol rate which is fixed. The execution time of the compute and communicate subtasks in frame j 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 ([5], [6], [3]). We assume the sensing time is not scalable and denoted as a constant t^{sen}. The total execution time t_j^{exe} in frame j of type $\{I_j^{sen}, I_j^{cp}, I_j^{cm}\}$ at compute speed f_j and communicate speed d_j is given as:

$$t_j^{exe} = I_j^{sen} \cdot t^{sen} + I_j^{cp} \cdot (C/f_j) + I_j^{cm} \cdot (M/d_j) \tag{2}$$

3.2. Energy model

The DVS and DMS techniques enable scaling of the compute and communicate energy. The compute energy is a function of the compute speed f and supply voltage V_{dd} ([2]). The communicate energy is a function of the communicate speed d ([3]). We give the energy consumed in a frame j, e_j^c as:

$$e_j^c = I_j^{sen} \cdot e^{sen} + I_j^{cp} \cdot e_j^{cp} + I_j^{cm} \cdot e_j^{cm} \tag{3}$$

e^{sen} is the sense energy which is constant. e_j^{cp} and e_j^{cm} are the compute and communicate energy in frame j:

$$e_j^{cp} = [\alpha f_j V_{dd,j}^2 + P^{ind,cp}] \cdot (C/f_j) \tag{4}$$

$$e_j^{cm} = [\beta R(2^{d_j/R} - 1) + P^{ind,cm}] \cdot (M/d_j) \tag{5}$$

α is the constant CPU switching capacitance. β is a constant determined by the transmission quality and noise level ([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 which 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_j^{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 Γ^{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 ([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 $H = N \cdot L$. Precisely, P_i^h is the received power at the storage device incorporating the loss during the power transfer between the energy harvester and storage. We assume the prediction of P_i^h for an epoch i is known prior to the scheduling stage.

When a sensor node is executing tasks, it draws power from the storage device. The drawn power is controllable via DVS and DMS. The storage device stops discharging as the energy level drops to zero, and stops charging as the energy level approaches Γ^{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 ([13], [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*. The increased energy reserve may survives the sensor nodes when the system operates under highly uncertain state. 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 will define and solve the problem formally.

In an attempt to capture all the parameters of our problem, we start with a couple of definitions. This will allow us to formulate the problem in a precise manner. The energy level at the end of epoch i is given by:

$$\forall i \in [1, N], \Gamma_i = \Gamma^{init} + \sum_{k=1}^{i} E_k^h - \sum_{k=1}^{i} E_k^c \tag{6}$$

where Γ^{init} is the initial energy level in the horizon. E_k^h and E_k^c are the harvested and consumed energy in epoch

k respectively. Starting with Γ^{init}, Γ_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 Γ_{i-1} in epoch $i-1$ (which is also the starting energy level in epoch i), $\gamma_{i,j}$ is determined 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 $\lfloor L/S \rfloor$ frames in an epoch. The consumed energy in an epoch, E^c_i is given as:

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

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

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

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

$e^h_{i,j}$ is the harvested energy in frame (i, j). As mentioned in the previous section, the harvested power P^h_i 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, $\gamma_{min} = min\{\gamma_{i,j} \mid \forall i \in [1, N], j \in [1, \lfloor L/S \rfloor]\}$. 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:

$$\begin{aligned}
Max. \quad & \gamma_{min} && (13)\\
s.t. \quad & \forall i \in [1, N], j \in [1, \lfloor L/S \rfloor] \\
& 0 < \gamma_{i,j} \leq \Gamma^{max} && (14)\\
& t^{exe}_{i,j} \leq S && (15)\\
& f_{min} \leq f_{i,j} \leq f_{max} && (16)\\
& d_{min} \leq d_{i,j} \leq d_{max} && (17)
\end{aligned}$$

The constraint (14) enforces that the energy level $\gamma_{i,j}$ in any frame is confined to the range $(0, \Gamma^{max}]$. $\gamma_{i,j} > 0$ must hold in order to ensure energy neutrality. Also, we require that $\gamma_{i,j} \leq \Gamma^{max}$ to model the energy storage capacity of the sensor node. The constraint (15) ensures the timely completion of workloads 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 itertively 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), $\Delta\gamma_{i,j} = \gamma_{i,j} - \gamma_{i,j-1}$, $j \in [1, \lfloor L/S \rfloor]$ 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 $\gamma_{i,j}, \forall j \in [1, \lfloor L/S \rfloor]$ 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, Γ_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., Γ^{init}, the ending

energy level in epoch 1, Γ_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, γ_{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 $\gamma_{i,j} \leq \Gamma^{max}$ will never happen in practice, simply because the energy harvester is assumed to stop charging the storage device when the energy level approaches Γ^{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 single-epoch 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 $\gamma_{i,j} \leq \Gamma^{max}$. However, recall that the storage device automatically stops charging when the energy level approaches Γ^{max}, hence the maximum capacity does not need to appear as a constraint in the *frame level* energy management problem. The fixed speeds 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 b (communicate speed d); therefore, fixing d 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:

$$\begin{aligned} Max. \quad & \Delta\gamma_{i,j} \\ s.t. \quad & t^{exe}_{i,j} \leq S \\ & f_{min} \leq f_{i,j} \leq f_{max} \\ & d_{min} \leq d_{i,j} \leq d_{max} \end{aligned}$$

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 $\Delta\gamma$ also equals to the difference of 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:

$$\begin{aligned} Min. \quad & e^c = e^{sen} + e^{cp}(f) + e^{cm}(d) && (18) \\ s.t. \quad & t^{sen} + C/f + M/d \leq S && (19) \\ & f_{min} \leq f \leq f_{max} && (20) \\ & d_{min} \leq d \leq d_{max} && (21) \end{aligned}$$

The objective function can be seen to be convex due to the convex compute and communicate energy functions. This problem has two unknowns, f, d. It 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., $t^h = S - t^{sen} - (C/f) - (M/d)$ (Eq. (12)). Thus, saving energy by reducing speeds will sacrifice harvesting chance, and may lead to even smaller $\Delta\gamma$. Hence, unlike the concurrent case, minimum consumption does not imply maximum energy level increment. In this case, the problem becomes:

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

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 it 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 equalizing the first derivatives of e^{cp}, e^{cm} to zero. Now, we take 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 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:

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

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

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

In above, n is the number of variables. [15] and [16] show 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. The detail of this method can be found in [15] and [16]. The derived t^{cp}, t^{cm} 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 $f \geq f^{opt}$, $d \geq d^{opt}$, to guarantee the timely completion of the workloads.

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 γ_{min}, compared to the schemes that use either no or one of the voltage and modulation scaling techniques, under both concurrent and non-concurrent harvesting models. 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 run-time uncertainties and minimize service interruptions.

6.1. Node architecture and workload model

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

$Freq.(MHz)$	104	208	312	416	520	624
$Power(mW)$	116	279	390	570	747	925

Table 1. Specification of Intel Xscale Pxa27x

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-current mode. We use the results in [10] to fix the harvesting cycle at $H = 24hours$. This horizon is then divided into 96 epochs, each has a length

$L = 15mins$. We simulated the execution of each invoked frame and set the frame period at $S = 30ms$. We assume a rechargeable battery storing at most $4000Joules$ energy. The initial energy level is $\Gamma^{init} = 2400Joules$ (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.

6.2. Impact of joint voltage and modulation scaling

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 γ_{min} point (appeared 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 γ_{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* suffers 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* beats all other schemes again in stored energy

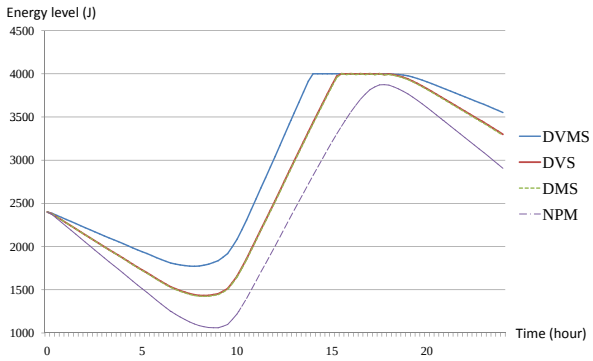

Figure 2. Normal mode, concurrent harvesting

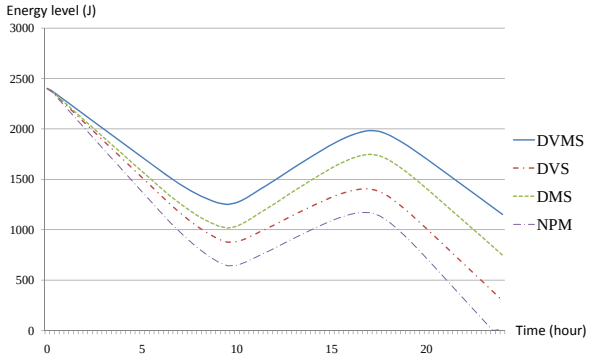

Figure 3. Normal mode, non-concurrent harvesting

and causes no service interruption. In Fig. (4), no schemes suffers 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 suffers service interruption. Under emergency mode, some schemes run out of energy in the middle of operation for up to 10% time of service which potentially causes disasters to mission-critical CPS applications. Our experiments indicate the benefits of our algorithm in term 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 algo-

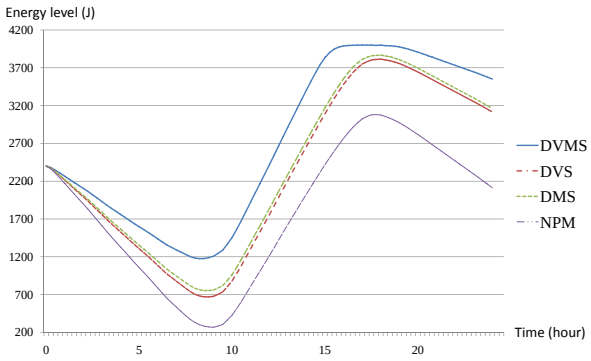

Figure 4. Emergency mode, concurrent harvesting

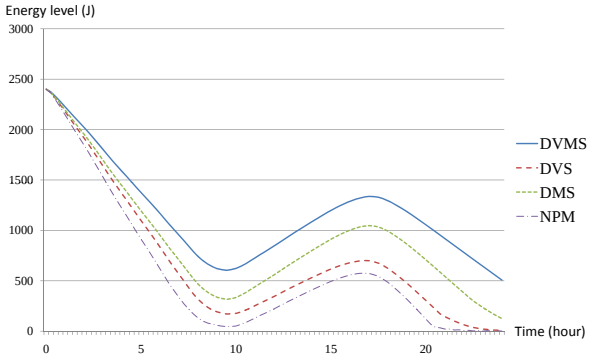

Figure 5. Emergency mode, non-concurrent harvesting

rithm *DVMS* 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 $\gamma^G_{i,j}$ and $\gamma^A_{i,j}$ denote the ending energy levels in frame (i,j), obtained by iteratively maximizng energy level increment of each frame, and that obtained by an arbitrary scheme A. Similarly, we denote $\Delta\gamma^G_{i,j}$ and $\Delta\gamma^A_{i,j}$ 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 $\Gamma^{init}_i = \Gamma_{i-1}$. We will prove the theorem by induction over the frame sequence number, j.

Base case: If j=1, we have $\gamma^G_{i,1} = \Gamma^{init}_i + \Delta\gamma^G_{i,1}$, $\gamma^A_{i,1} = \Gamma^{init}_i + \Delta\gamma^A_{i,1}$. Since $\Delta\gamma^G_{i,1} \geq \Delta\gamma^A_{i,1}$ is valid by definition, $\gamma^G_{i,1} \geq \gamma^A_{i,1}$ 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 $\gamma^G_{i,n-1} \geq \gamma^A_{i,n-1}$. We claim that Theorem 1 also holds in frame (i,n), i.e., $\gamma^G_{i,n} \geq \gamma^A_{i,n}$. We distinguish two cases:

- $\gamma^G_{i,n} = \Gamma^{max}$: the energy level reaches Γ^{max} at the end of frame (i,n). Since the energy level achieved by any scheme A cannot exceed Γ^{max}, our claim holds.
- $\gamma^G_{i,n} < \Gamma^{max}$: Note that if $\gamma^G_{i,n} < \Gamma^{max}$, the optimal energy increment obtained 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 $\gamma^G_{i,n}$ would be equal to Γ^{max}). This enables us to deduce that $\Delta\gamma^G_{i,n} \geq \Delta\gamma^A_{i,n}$, 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 $\gamma^G_{i,n} < \Gamma^{max}$ is not violated. Also recall that by induction assumption, we have $\gamma^G_{i,n-1} \geq \gamma^A_{i,n-1}$. Therefore, we have $\gamma^G_{i,n} = \gamma^G_{i,n-1} + \Delta\gamma^G_{i,n} \geq \gamma^A_{i,n-1} + \Delta\gamma^A_{i,n} = \gamma^A_{i,n}$. Thus, our claim holds as well.

Hence, we proved Theorem 1.

References

[1] N. Chaimanonart, M. Zimmerman, and D. Young, "Adaptive rf power control for wireless implantable bio-sensing network to monitor untethered laboratory animal real-time biological signals," in *Sensors, IEEE*, Oct. 2008, pp. 1241–1244.

[2] H. Aydin, R. Melhem, D. Mosse, and P. alvarez, "Power-aware scheduling for periodic real-time tasks," *IEEE Transactions on Computers*, vol. 53, no. 5, pp. 584–600, 2004.

[3] Y. Yu, B. Krishnamachari, and V. Prasanna, "Energy-latency tradeoffs for data gathering in wireless sensor networks," in *Infocom*, 2004.

[4] Crossbow, "Imote2," *www.xbow.com*, 2006.

[5] G. Kumar, G. Manimaran, and Z. Wang, "Energy-aware scheduling of real-time tasks in wireless networked embedded systems," in *RTSS*, 2007.

[6] C. Yeh, Z. Fan, and R. Gao, "Energy-aware data acquisition in wireless sensor networks," in *IMTC. IEEE*, 2007.

[7] C. Moser, L. Thiele, L. Benini, and D. Brunelli, "Real-time scheduling with regenerative energy," in *ECRTS*, 2006.

[8] S. Liu, Q. Qiu, and Q. Wu, "Energy aware dynamic voltage and frequency selection for real-time systems with energy harvesting," in *DATE*, 2008.

[9] D. K. Noh, L. Wang, Y. Yang, H. K. Le, and T. Abdelzaher, "Minimum variance energy allocation for a solar-powered sensor system," in *DCOSS*, 2009.

[10] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *Trans. on Embedded Computing Sys.*, vol. 6, no. 4, p. 32, 2007.

[11] C. Moser, L. Thiele, D. Brunelli, and L. Benini, "Robust and low complexity rate control for solar powered sensors," in *DATE*, 2008.

[12] C. Rusu, R. Melhem, and D. Mossé, "Multi-version scheduling in rechargeable energy-aware real-time systems," *J. Embedded Comput.*, vol. 1, no. 2, pp. 271–283, 2005.

[13] X. Jiang, J. Polastre, and D. Culler, "Perpetual environmentally powered sensor networks," in *IPSN*, 2005.

[14] C. Park and P. Chou, "Power utility maximization for multiple-supply systems by a load-matching switch," in *ISLPED*, 2004.

[15] H. Aydin, V. Devadas, and D. Zhu, "System-level energy management for periodic real-time tasks," in *RTSS*, 2006.

[16] H. Aydin, R. Melhem, D. Mosse, and P. Alvarez, "Optimal reward-based scheduling for periodic real-time tasks," *IEEE Trans. Comput.*, vol. 50, no. 2, pp. 111–130, 2001.

[17] C. M. S. M. Bazaraa, H. Sherali, *Nonlinear Programming: Theory and Algorithms*, 3rd ed. Wiley, 2006.

[18] Intel, "Xscale pxa27x," *www.intel.com/design/intelxscale*.

[19] Chipcon, "Cc2420," *docs.tinyos.net/index.php/CC2420*, 2006.

[20] TAOs, "Tsl2561," *www.farnell.com/datasheets/49661.pdf*.