Sustainable Computing: Informatics and Systems xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

On minimizing expected energy usage of embedded wireless systems with probabilistic workloads

Maryam Bandari, Robert Simon, Hakan Aydin*

George Mason University, Computer Science Department, 4400 University Drive, Fairfax, VA 22030, USA

ARTICLE INFO

Article history:
Received 19 May 2015
Accepted 28 February 2016
Available online xxx

Keywords:
Networked embedded systems
Energy management
Dynamic Modulation Scaling
Dynamic Voltage Scaling

ABSTRACT

A large number of embedded wireless systems must handle complex and time-varying computational and communication workloads. Further, a significant number of these systems support real-time applications. Most of the existing energy management studies for such systems have focused on relatively simple scenarios that assume deterministic workloads, and only consider a limited range of energy management techniques, such as Dynamic Voltage Scaling (DVS). Our paper addresses these deficiencies by proposing a general purpose probabilistic workload model for computation and communication. To account for the importance of radio energy consumption, we also analyse Dynamic Modulation Scaling (DMS), an often overlooked method for energy management. We define several energy control algorithms, including an optimal combined DVS–DMS approach, and evaluate these algorithms under a wide range of workload values and hardware settings. Our results illustrate the benefits of joint power control algorithms.

1. Introduction

A large class of embedded wireless systems have real-time performance requirements for both computational and communication tasks. Examples of such systems include industrial process control, highway monitoring and building surveillance [1–3]. Many of these systems are self-powered, so from both a system design and an environmental perspective efficient energy management is of paramount importance. System architects use component level tuning knobs that tradeoff power consumption with performance. For instance, a commonly used power saving technique is Dynamic Voltage Scaling (DVS) [4]. DVS controls power consumption by reducing the CPU frequency and supply voltage, thereby saving energy expenditure while requiring computations to take longer. Dynamic Modulation Scaling (DMS) is another type of tuning technique. DMS works by changing radio modulation levels and constellation sizes, reducing energy expenditures while requiring longer transmission and reception times [5]. DMS is directly supported by embedded wireless standards such as 802.15.4 [6]. The impact of DMS usage on power consumption in wireless embedded systems is relatively understudied. Moreover, for wireless embedded nodes with both substantial computational and communication workloads, both DVS and DMS techniques are relevant. Though there are a few studies that consider DVS and DMS simultaneously [7,8], those works consider exclusively deterministic workloads.

This paper addresses that gap by studying the joint use of DVS and DMS for real-time embedded wireless systems through the design of several novel energy management algorithms. We focus on systems that have deadline constraints for both computational and communication tasks. We are specifically interested in quantifying the impact of these algorithms when both computation and communication workloads are known only probabilistically. We believe this is a direction that warrants investigation, as in practical applications the most important objective is typically to minimize the *expected energy consumption* while still providing performance guarantees. To this aim, we use probabilistic workload models for both computation and communication activities. The computational model uses *cycle groups* [9–11], a concept that supports the empirical estimation of an underlying workload probability distribution. We adopt a similar approach to model the communication workload.

Our work evaluates seven different algorithms, including a joint DVS–DMS approach and a computationally simple heuristic. Using our probabilistic workload, deadline and energy models, the joint approach formulates the problem as one that can be solved through convex optimization. We present an efficient off-line solution to this problem. Our work is based on the observation that in probabilistic workload settings, the optimal solution consists of starting with low computation and communication speed levels, and then gradually increasing the speeds as the task makes progress. We

* Corresponding author. Tel.: +1 703 9933786.
E-mail addresses: mbandari@gmu.edu (M. Bandari), simon@gmu.edu (R. Simon), aydin@gmu.edu (H. Aydin).

http://dx.doi.org/10.1016/j.suscom.2016.02.004
2210-5379/

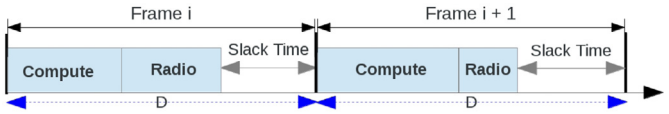


Fig. 1. Application model.

show how to compute the optimal *speed schedule* in which DVS and DMS parameters are adjusted to match the current workload conditions. We first study the optimal CPU and radio speed scheduling algorithms in the continuous speed domain. To account for the fact that in current hardware design speed levels can only change by certain step sizes, we extend the solution to cover the discrete domain.

We also present a general purpose simulation model that represents a wide variety of processor and radio types. We then describe an extensive simulation study of our various algorithms with a particular interest in evaluating the benefits of our integrated DVS–DMS approach under probabilistic workloads and as a function of the ratio of the radio power to the CPU power, by comparing to other algorithms, including those that use DVS-only or DMS-only approaches for energy management.

To our knowledge this is the first study that considers *both* DVS and DMS in a wireless embedded system using *probabilistic* computation and communication workload models. Our results precisely quantify the improvements offered by these control techniques as a function of the underlying hardware characteristics, and can be used by designers as a guideline for algorithm selection. Of particular importance of this work is the demonstration of the potential value of DMS techniques [6]. For instance, our experimental results show that an integrated DVS–DMS strategy can provide non-trivial gains on the expected energy consumption, especially when the computation and communication workloads are relatively balanced.

The rest of this paper is organized as follows. In Section 2 we present our power and application models, as well as our assumptions. By assuming an ideal system where the CPU frequency and modulation levels can be adjusted continuously, the energy minimization problem is formulated and solved in Section 3. Assuming discrete frequency and modulation levels, the same problem is formulated as a mixed binary integer programming problem in Section 4. In Section 5, a detailed performance evaluation of several algorithms, including optimal algorithms, fast heuristics and those that use only the DVS or DMS feature, is presented. Section 6 surveys related work, and we conclude in Section 7.

2. System model

This section describes the application model, presents the system level energy components and shows how to derive the expected energy.

2.1. Application model

We consider an embedded wireless node with two major activities: data processing (computation), performed by the CPU, and communication with other wireless embedded devices, performed by the radio. Specifically, as in [7], we assume that computation and communication activities form two *sub-tasks*, executed within a *frame* (Fig. 1). A frame is a time interval of length D that repeats periodically during the lifetime of the node with the rate $1/D$. Input to the radio communication sub-task depends on the output of the computation sub-task; consequently, the latter is to be executed first in each frame. Both sub-tasks must be completed within a relative deadline of D, by the end of frame. The sub-tasks may

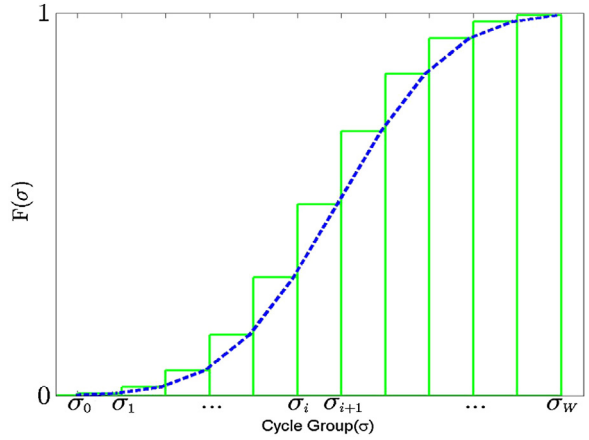


Fig. 2. Histogram-based approximation of the cumulative distribution function of the application's probabilistic workload.

have varying resource demands from frame to frame, determined according to specific probability distributions, as explained below.

2.1.1. Computation workload

In real applications, the number of CPU cycles in a given frame (the computation workload) can be known only *probabilistically* in advance. We denote the minimum and maximum computational workload demand in a single frame by C_{min} and C_{max} cycles, respectively. In general, the cumulative probability distribution function for the computation workload is:

$$F(c) = p(X \leq c)$$

where X is the random variable for the application's computation demand in a frame, and $p(X \leq c)$ represents the probability that the application will not require more than c cycles in a single frame. This function can be approximated through the histogram-based *profiling* approach [9,12,13]. Specifically, the available range of CPU cycles $[C_{\min}, C_{\max}]$ is divided into W discrete *cycle groups*, each with $\omega = (C_{\max} - C_{\min})/(W)$ cycles. We denote the upper bound on the number of cycles in the ith cycle group as σ_i, that is, $\sigma_i = C_{\min} + (i-1)\cdot\omega$.

The workload probability distribution function may be obtained by multiple means. One approach is profiling over a fixed window size for workloads with self-similarity property [14]. In general, to obtain the histogram-based profiles, the application's executions over a long time interval is monitored, and the fraction of invocations in which the number of actual cycles fall in the ith cycle group are recorded [9–11]. More precisely, the fraction of invocations where the number of executed cycles falls in the ith cycle group during the profiling phase, is assumed to correspond to the probability that the number of cycles will fall in this specific range over a long-term periodic execution. In this way, the probability that the actual number of cycles needed by the application will fall in the range $(\sigma_{i-1}, \sigma_i]$, denoted by f_i^{cp}, is derived for $i = 1, \ldots, W$. Observe that $\sum_{i=1}^{W} f_i^{cp} = 1$.

We can calculate the cumulative probability distribution function (Fig. 2) of the application's cycle demand as:

$$F_j^{cp} = \sum_{k=1}^{j} f_k^{cp}$$

F_j^{cp} denotes the probability that the application will require no more than j cycle groups (i.e., *at most* $C_{\min} + (j-1)\cdot\omega$ cycles) in one frame. Consequently, $\Gamma_j^{cp} = 1 - F_{j-1}^{cp}$ is the probability that the task will require *more than* $(j-1)$ cycle groups, or equivalently, the

Please cite this article in press as: M. Bandari, et al., On minimizing expected energy usage of embedded wireless systems with probabilistic workloads, Sustain. Comput.: Inform. Syst. (2016), http://dx.doi.org/10.1016/j.suscom.2016.02.004

probability that the jth cycle group will be executed in one frame. Note that a similar probabilistic workload formulation is used in (intra-task) DVS literature, e.g., in [9–11].

2.1.2. Communication workload

The communication workload is also expected to have a probabilistic distribution. Each node transmits and receives information via the exchange of packets. The communication workload is constrained between 1 and M packets within a frame respectively.

f_i^{cm} represents the probability distribution function of transmitting or receiving exactly i packets in one frame. It is obtained in a manner similar to histogram-based approximation of computation workload: the cumulative distribution function $F_i^{cm} = \sum_{k=1}^{i} f_k^{cm}$ is the probability of transmitting or receiving no more than i packets. The probability of the ith packet being transmitted or received in one frame is then denoted by:

$$\Gamma_i^{cm} = 1 - F_{i-1}^{cm}$$

2.1.3. Sub-task execution times and slack time

Let t^{cp} and t^{cm} denote the actual time taken by the computation and communication subtasks in a given frame, respectively. Clearly, these quantities are a function of the actual number of cycles and packets that are processed, as well as the processing frequency and modulation levels used in that specific frame. The extra time remaining in a frame which is not consumed by either of the sub-tasks is referred to as *slack time*:

$$t^{slack} = D - t^{cp} - t^{cm}$$

2.2. Power and energy models

2.2.1. CPU power

We consider DVS-enabled CPUs capable of dynamically adjusting their voltage and frequency. The total CPU power consumption p_s^{cp} is the summation of the *static* and *dynamic* power components, denoted respectively by p_s^{cp} and p_d^{cp}:

$$p^{cp} = p_s^{cp} + p_d^{cp}$$

Static power is necessary to keep the basic circuits on and the clock running; it can only be eliminated by turning the system off. Due to the excessive overhead associated with turning the system on and off for periodic real-time task application that we consider, we assume the system is constantly on and static power is not manageable as in [10,15]. Dynamic CPU power, p_d^{cp}, on the other hand, is dissipated when the CPU executes tasks and includes *frequency-independent* and *frequency-dependent* power components:

$$p_d^{cp} = p_{ind} + C_{ef} s^{\alpha}$$

The frequency-independent dynamic power component p_{ind} is due to the off-chip components such as memory and external devices and does not vary with the CPU frequency. On the other hand, the frequency-dependent active power is a convex function of the CPU frequency. On systems that are DVS-capable, the CPU supply voltage is linearly related to the variable CPU frequency and dynamic power is given by $C_{ef} \cdot s^{\alpha}$, where s is the CPU frequency (speed), C_{ef} is the effective switching capacitance, and α is a constant (typically 3 in CMOS technologies) [4]. By exploiting the convex relationship between the CPU frequency and the dynamic power, DVS enables the system to save energy at the cost of increased execution times. Since the CPU frequency s can vary with time, the total dynamic energy consumed by the CPU in the interval (t_1, t_2) is given by:

$$E_{\text{CPU}} = \int_{t_1}^{t_2} p_d^{cp}(s(t))dt$$

Finally, the time needed to execute Q cycles at a constant frequency s is Q/s, and the energy consumed while executing one *cycle group* that consists of ω cycles at frequency s is given by [9,11]:

$$e_{\omega}^{cp} = \frac{\omega}{s}\left(C_{ef} \cdot s^{\alpha} + p_{ind}\right) \tag{1}$$

2.2.2. Radio power

The radio power consists of two components: the dynamic power p_t dissipated when transmitting or receiving packets and the electronic circuitry power p_e [5].

p_t essentially corresponds to the power needed for the amplifier during data communication. in order to transmit information, bits are modulated into channel symbols. Those symbols are differentiated by waveforms. The number of different waveforms in the channel determines how many bits can be coded in one channel symbol. The number of bits per symbol in a modulation scheme, b, is called *the modulation level*. The symbol rate is denoted by R_s. In DMS, p_t can be controlled by varying the modulation level:

$$p_t(b) = C_s \cdot \phi(b) \cdot R_s \tag{2}$$

Here, $\phi(b)$ is a convex function of the modulation level and its specific form depends upon the modulation scheme. C_s is a function of the circuit implementation of the receiver's radio, current temperature, distance, and transmit media, and is independent of the modulation level. Supposing a time invariant channel, C_s can be approximated as a constant. DMS changes radio power by decreasing modulation level, at the cost of increasing transmission time.

Electronic circuitry power can be written as [5]:

$$p_e = C_e \cdot R_s \tag{3}$$

C_e is a constant that depends on the radio circuit technology. The communication time will vary with the modulation level: $t_{bit}^{cm}(b) = 1/b \cdot R_s$ is the time needed to send one bit over the communication channel. Hence, the energy needed to send one bit is $e_{bit}^{cm} = (p_t + p_e) \cdot t_{bit}^{cm}$.

The two communication parties need to agree on the exact value of the modulation level at the beginning of the transmission. One technique to do this is to use appropriate physical layer headers [6]. Note that any initialization can only occur before sending each packet, so that modulation level remains constant during the packet transmission. The energy required to send or receive one packet of ρ bits is thus:

$$e_{\rho}^{cm} = \rho \cdot (p_t + p_e) \cdot t_{bit}^{cm} = \frac{\rho \cdot (C_s \phi(b) + C_e)}{b} \tag{4}$$

Our work targets real-time embedded wireless systems and assumes a QoS-enabled MAC layer capable of minimizing energy wasted due to collisions or idle listening.

2.2.3. Overall expected energy

Given the probability distribution functions for communication and computational workloads, we can now derive the expected overall energy consumption, as the sum of expected processor and radio energy. The expected processor energy is the sum of energy dissipated to execute each of the cycle groups $(\sigma_{j-1}, \sigma_j), j = 1, \ldots, W$, multiplied by the probability that the cycle group will be actually executed in a frame, namely, Γ_j^{cp}. Similarly, the expected communication energy is the sum of energy needed for the jth packet

Please cite this article in press as: M. Bandari, et al., On minimizing expected energy usage of embedded wireless systems with probabilistic workloads, Sustain. Comput.: Inform. Syst. (2016), http://dx.doi.org/10.1016/j.suscom.2016.02.004

Table 1
List of symbols.

Symbol	Description
ρ	Packet size: number of bits per packet
ω	The size of CPU cycle group
C_s, C_e	Values of transmit and electronic circuitry power components
R_s	Symbol rate
C_{ef}	Switching activity capacitance of the task
p_{ind}	Frequency-independent power of the CPU
Γ_i^{cm}	The probability of sending the ith packet
Γ_i^{cp}	The probability of executing the ith cycle group
M	Maximum number of packets
W	Maximum number of cycle groups
b_i	The number of bits per symbol in the ith packet
s_i	CPU frequency used to execute the ith cycle group
D	Frame deadline (period)
$\phi(b)$	Modulation energy scaling function

($j = 1, \ldots, M$), multiplied by the probability that the packet will be actually transmitted/received in a frame, namely, Γ_j^{cm}.

$$e_{\text{overall}} = \sum_{j=1}^{W} \Gamma_j^{cp} e_\omega^{cp} + \sum_{i=1}^{M} \Gamma_i^{cm} e_\rho^{cm} = \sum_{j=1}^{W} \Gamma_j^{cp} \cdot \frac{\omega}{s_j} \left(C_{ef} \cdot s_j^\alpha + p_{ind}\right) + \sum_{i=1}^{M} \frac{\rho \cdot \Gamma_i^{cm}}{b_i} \cdot (C_s \cdot \phi(b_i) + C_e) \quad (5)$$

The modulation level of the ith packet and the execution frequency of the jth cycle group are shown by b_i and s_j in the formula above. Table 1 summarizes the list of the most important notations used in this section.

3. Continuous energy optimization problem

With DVS the optimal computation speed to minimize energy while meeting a timing constraint can be shown to be *constant* under the assumption that the workload is known deterministically [4,16]. This is due to the convex speed/power relationship. The same applies to the DMS technique in the deterministic communication workload case [5].

However, existing DVS research studies have identified that in the case of *probabilistic* workload, the constant speed is no longer optimal [10,12]. Rather, starting with a low speed and gradually increasing it as the task makes progress minimizes the *expected* total energy. We observe that the same considerations equally hold for the DMS case since the communication workload can vary from instance to instance and hence can be in practice known only probabilistically. Combining both DVS and DMS under probabilistic workload assumptions is a non-trivial problem.

Our solution derives a joint DVS–DMS *speed schedule* for the application. The speed schedule indicates how to adjust the computation speed (the CPU frequency) and the communication speed (modulation level) to match the current workload. More specifically, the speed schedule contains the sequence of optimal settings for each cycle group and radio packet, that we call *scheduling units*, separately. It makes the speed assignments so that the first scheduling units have low processing frequency or modulation levels while the speed is increased for the next scheduling units, in order to meet the deadline of the frame even under a worst-case scenario.

For example, consider a frame with a deadline of 95 ms, a worst-case processing time of 50 ms (under maximum CPU speed), and a worst-case communication time of 25 ms (at the maximum modulation level). Suppose the computation workload is distributed among the set of 4 cycle groups: under maximum CPU speed, each cycle group will need an execution time of 12.5 ms. The communication workload varies from one to three packets, each requiring 8.33 ms transmission time under highest modulation level. The probability distribution function of computation and communication workloads respectively are given as: $f^{cp} = \{.45, .05, .05, .45\}$, and $f^{cm} = \{.025, .85, .125\}$. Using the specification of Intel Celeron-M processor and a radio whose power consumption is twice as large as that of the CPU at the maximum modulation level, the optimal speed schedule (shown in Fig. 3) is obtained – the algorithm to obtain the optimal speed schedule will be presented in due course. This optimal CPU speed schedule suggests that in order to minimize the expected energy, one should start at the frequency 262 MHz, and if the computation sub-task is not completed within the first 15.82 ms, then the frequency should be first increased to 267 MHz, and then to 272 MHz after 31.4 and 46.7 ms, respectively. A similar communication speed schedule is suggested in the second sub-figure. Observe that as the sub-tasks experience increasing workloads, the corresponding speeds gradually increase – but under a worst-case workload, the application still meets its deadline at 95 ms.

By exploiting the probabilistic workload information and gradually increasing the computation and communication speeds, it can be shown that with the joint use of DVS and DMS and the exploitation of the probabilistic workload information, the system consumes 59%, 36% and 33% less expected energy, compared to no speed scaling, DVS-only, and DMS-only cases, respectively. This example illustrates the potential benefits of applying DVS and DMS jointly, while also taking into account the probabilistic workload information.

Having defined the energy consumption, frequency, and timing constraints of the system, we now present the main energy minimization problem. We are looking for joint communication–computation speed settings that minimize the overall **expected** energy consumption in the system, for which an analytical formula was presented at the end of Section 2. The speed schedules should meet both the timing constraints of the system and the frequency/modulation level limitations. Upper and lower CPU frequency bounds are shown by $s_{\max}$ and $s_{\min}$. In the modulation schemes we consider, the minimum number of bits per symbol is 2. The maximum modulation level is limited by the signal to noise ratio of the channel or hardware constraints. We write these two bounds as $b_{\min}$ and $b_{\max}$. Denoting the frequency for the jth cycle group as s_j and the modulation level for the ith packet as b_i, the energy optimization problem is written as:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{j=1}^{W} \Gamma_j \frac{\omega^{cp}}{s_j} \left(C_{ef} . s_j^\alpha + p_{ind}\right) + \sum_{i=1}^{M} \frac{\rho . \Gamma_i^{cm}}{b_i} \left(C_s . \phi(b_i) + C_e\right) \\
\text{subject to} \quad & \sum_{j=1}^{W} \frac{\omega}{s_j} + \sum_{i=1}^{M} \frac{\rho}{b_i R_s} \le D \\
& s_{\min} \le s_j \le s_{\max} \\
& b_{\min} \le b_i \le b_{\max}
\end{aligned}$$

Now consider the following variable substitutions, which yield a new form of the optimization problem:

$$t_j^{cp} = \frac{\omega}{s_j} \rightarrow s_j = \frac{\omega}{t_j^{cp}}$$
$$t_i^{cm} = \frac{\rho}{b_i R_s} \rightarrow b_i = \frac{\rho}{R_s t_i^{cm}}$$

Please cite this article in press as: M. Bandari, et al., On minimizing expected energy usage of embedded wireless systems with probabilistic workloads, Sustain. Comput.: Inform. Syst. (2016), http://dx.doi.org/10.1016/j.suscom.2016.02.004

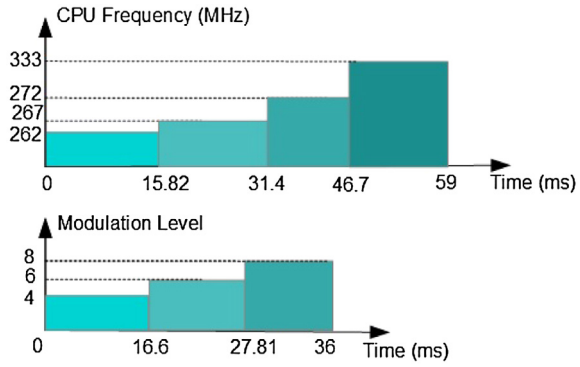


Fig. 3. Speed scheduling example.

We denote $\phi(b_i) = \phi(\rho/R_s t_i^{cm})$ as $\chi(t_i^{cm})$. $\chi()$ indicates how transmission energy changes with the transmission time to send one packet over the channel. These substitutions lead to a reformulation of the optimization problem:

$$\text{Minimize} \quad \sum_{j=1}^{W} \Gamma_j^{cp} C_{ef} \omega^{\alpha} (t_j^{cp})^{1-\alpha} + \Gamma_j^{cp}.p_{ind}.t_j^{cp} + \sum_{i=1}^{M} \Gamma_i^{cm} R_s t_i^{cm} \left[C_s \chi(t_i^{cm}) + C_e\right]$$

$$\text{subject to} \quad \sum_{j=1}^{W} t_j^{cp} + \sum_{i=1}^{M} t_i^{cm} = D$$

$$t_{\min}^{cp} = \frac{\omega}{s_{\max}} \le t_j^{cp} \le \frac{\omega}{s_{\min}} = t_{\max}^{cp}$$

$$t_{\min}^{cm} = \frac{\rho}{b_{\max}} \le t_i^{cm} \le \frac{\rho}{b_{\min}} = t_{\max}^{cm}$$

While the energy consumption is a convex function of t_i^{cm} and t_j^{cp}, a couple of observations are in order. Both *computation* and *communication* components of the expected energy have two terms, one which increases with the allocated time, and another one which decreases. This is to be expected, because the dynamic power in both cases are concave functions of the speed, while speed independent power is constant, and so off-chip and electronic circuitry energy increase by time. While reducing the speed reduces dynamic energy, it tends to increase the off-chip and electronic circuitry energy due to the need to keep the circuit in active mode for longer intervals. In other words, there is an *energy-efficient* modulation level, b_e, and frequency, s_e below which DMS and DMS are no longer effective, as observed in [5,15,17]. Their value can be determined analytically by setting the first derivative of the radio energy and cpu energy to zero (separately). Consequently, after replacing b_{min} by $b'_{\min} = \max\{b_{\min}, b_e\}$ and $s'_{\min} = \max\{s_{\min}, s_e\}$ in the above problem, we get a new optimization problem whose computation (communication) energy components are monotonically decreasing with increasing computation (communication) time allocations.

This allows us to tackle some boundary cases. In particular, if by using the maximum communication and computation time allocations (lowest speeds) for all the packets and cycle groups we can still meet the deadline, that solution is obviously optimal. Otherwise, due to the monotonic nature of the above problem, one should use the entire frame fully to maximize the energy savings, yielding a new optimization problem:

$$\text{Minimize} \quad \sum_{j=1}^{W} \Gamma_j^{cp} C_{ef} \omega^{\alpha} (t_j^{cp})^{1-\alpha} + \Gamma_j^{cp}.p_{ind}.t_j^{cp} + \sum_{i=1}^{M} \Gamma_i^{cm} R_s t_i^{cm} \left[C_s \chi(t_i^{cm}) + C_e\right]$$

$$\text{subjectto} \quad \sum_{j=1}^{W} t_j^{cp} + \sum_{i=1}^{M} t_i^{cm} = D$$

$$t_{\min}^{cp} = \frac{\omega}{s_{\max}} \le t_j^{cp} \le \frac{\omega}{s'_{\min}} = t_{\max}^{cp,}$$

$$t_{\min}^{cm} = \frac{\rho}{b_{\max}} \le t_i^{cm} \le \frac{\rho}{b'_{\min}} = t_{\max}^{cm,}$$

We developed an iterative method that solves the above optimization problem by using the Karush–Kuhn–Tucker (KKT) optimality conditions for non-linear optimization. The details of our optimal algorithm can be found in Appendix A.

4. Discrete domain energy optimization problem

The optimization framework we presented implicitly assumed an *ideal* system by considering a continuous range for the CPU frequencies and modulation levels for the radio. However, current systems offer only a limited number of CPU frequencies and radio modulation levels. For example, the Intel XScale CPU has 5 frequency levels ranging from 150 MHz to 1 GHZ. Similarly the Cortex M3 CPU has 16 frequency levels varying from 36 MHz to 96 MHz. For radio, the moduation levels in QAM are restricted to even numbers.

Consider a CPU with n discrete frequency levels $f_1, \ldots, f_n$. We use the *binary indicator variable* $\beta_{x,j}^{cp} \in \{0, 1\}$ to denote whether the xth frequency level will be used to execute the jth cycle group. Specifically, if the jth cycle group is executed at the xth frequency level, then $\beta_{x,j}^{cp} = 1$ and $\beta_{q,j}^{cp} = 0 \quad \forall q \neq x$. Similarly, on a system with m distinct modulation levels $b_1, \ldots, b_m$, we use the binary indicator variable $\beta_{y,i}^{cm} \in \{0, 1\}$ to show if the yth modulation level is used to transmit the ith packet.

Given these binary indicator variables, the problem of selecting the optimal frequency (CPU speed) and modulation levels to minimize overall expected energy can be formulated as:

$$\text{Minimize} \quad \sum_{j=1}^{W} \sum_{x=1}^{n} \beta_{x,j}^{cp} \Gamma_j^{cp} \cdot (\omega \cdot C_{ef} \cdot s_x^{\alpha-1} + p_{ind}) + \sum_{i=1}^{M} \sum_{y=1}^{m} \beta_{y,i}^{cm} \frac{\rho \cdot \Gamma_i^{cm}}{b_y} \cdot \left(C_s \cdot \phi(b_y) + C_e\right) \tag{6}$$

$$\text{subject to} \quad \beta_{x,j}^{cp} \in \{0, 1\} \quad \forall x, j \tag{7}$$

$$\beta_{y,i}^{cm} \in \{0, 1\} \quad \forall y, i \tag{8}$$

$$\sum_{x=1}^{n} \beta_{x,j}^{cp} = 1 \quad \forall j \quad 1 \le j \le W \tag{9}$$

$$\sum_{y=1}^{m} \beta_{y,i}^{cm} = 1 \quad \forall i \quad 1 \le i \le M \tag{10}$$

$$\sum_{j=1}^{W} \sum_{x=1}^{n} \beta_{x,j}^{cp} \frac{\omega}{s_x} + \sum_{i=1}^{M} \sum_{y=1}^{m} \beta_{y,i}^{cm} \frac{\rho}{b_y R_s} \le D \tag{11}$$

Please cite this article in press as: M. Bandari, et al., On minimizing expected energy usage of embedded wireless systems with probabilistic workloads, Sustain. Comput.: Inform. Syst. (2016), http://dx.doi.org/10.1016/j.suscom.2016.02.004

In the above equations, the two terms of (6) account for the *expected* CPU and radio energy. $\beta^{cp}_{x,j}$ and $\beta^{cm}_{y,i}$ represent the constraints on the binary indicator variables for choosing CPU frequencies and modulation levels, respectively (the constraint sets (7) and (8)). Note that for each data packet or computational cycle group only one of the speed choices is optimal. The optimum speed level will have the corresponding indicator variable set to 1 and other speed levels will be assigned 0. This constraint is indicated for each data packet or a cycle group, by limiting the summation of their corresponding indicator variables to 1 (the constraint sets (9) and (10)). For example, in a system with modulation levels in the set {2, 4, 6, 8} and discrete CPU frequency levels of {100, 200, 300, 400, 500, 600, 700, 800} MHz, if the optimum modulation level and CPU frequency of a certain packet p and cycle group c are $b^*_i = 4$ and $s^*_j = 700$ MHz, respectively, then we will have: $\beta^{cm}_{\{1..4\},p} = \{0, 1, 0, 0\}$ and $\beta^{cp}_{\{1..8\},c} = \{0, 0, 0, 0, 0, 0, 1, 0\}$. The deadline constraint is enforced in (11).

This formulation is an instance of the *mixed binary integer programming*. It is known that, in general, solving the mixed binary integer programming problem is an NP-Hard problem. However, for small-to-moderate size problem instances, existing optimization packages can be used to obtain the optimal solution. As the problem size grows, one needs to devise approximation algorithms or heuristic solutions that run in polynomial-time.

5. Performance evaluation

We conducted an extensive set of simulations under a wide range of computational and communication workloads and device power models, in order to accurately evaluate the performance of the joint DVS–DMS scheme, compared to the other design options. For this purpose, we developed a discrete event simulator in Matlab. The simulator is designed to accept our general purpose processor and radio energy models, as well as a range of computational and communication workloads. The simulator is fully adaptive with system's characteristics and requirements as discussed in the following.

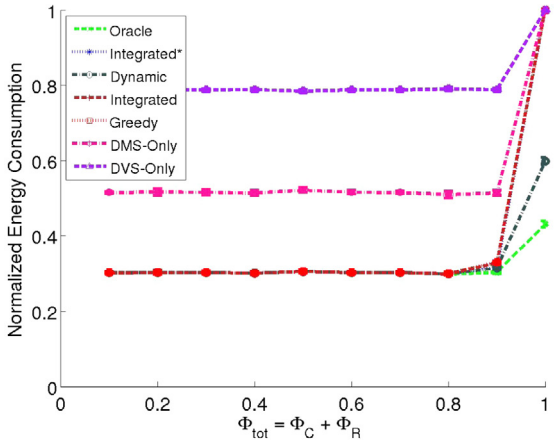
In order to cover a full dimension of CPU and radio workloads, we use the notations Φ_C and Φ_R to denote the ratio of maximum CPU processing time to the deadline, and the ratio of maximum total communication time to the deadline, in maximum speed respectively. We call these quantities the *utilization* of computation and communication subtasks. In our experiments, $\Phi_C + \Phi_R$ goes up to 1.0 (i.e., up to 100% of the available frame execution time). Our simulator is designed so that within each frame the *actual workloads* are determined using an arbitrary probability distribution. Specifically, the actual utilization of the computation sub-task is determined randomly in the range $[\Phi_C/W, \Phi_C]$ with the desirable PDF. Similarly, the actual utilization of the computation sub-component is determined randomly in the range $\left[\Phi_R/M, \Phi_R\right]$. We executed the simulations for both *uniform* and a *heavy tailed* distributions of the workload. To represent the heavy-tailed distribution, we chose a *generalized Pareto distribution*. The distribution is identified by three parameters of location (threshold) θ, scale σ, and shape ϵ. In our settings, we assigned all three parameters to 1 for computational workload and we set $\epsilon = 2$, $\sigma = 1$, $\theta = 0$ for communicational workload. It is worth noting that we tried multiple other settings for the heavy-tailed distribution, but obtained very similar results. We fixed values of both M and W to 10.

We define PR as the ratio of maximum CPU power (running at its maximum frequency) over maximum radio power (at the maximum modulation level). Changing the value of PR enables us to model a wide range of CPU and radio hardware configurations. We believe this is important; for example, the maximum power consumption can vary significantly among embedded processors – for MSP 430, Cortex M3, ATMEGA 1281, and Freescale MC1322xx, it assumes the values of 7.2, 19.8, 70, and 102.3 mW, respectively. On the other hand, CC2420 and XE1205 radios consume 60 mW and 65 mW, respectively. We ran the algorithms for PR values ranging from 10^{-2} to 10^2. For DMS modeling, we used $C_e = 15 \times 10^{-9}$, $C_s = 12 \times 10^{-9}$, $R_s = 10^6$, $b_{\min} = 2$, $b_{\max} = 8$, after [5,7]. We assumed QAM as the modulation technique. We modeled a CPU with eight equi-distant discrete frequency levels.

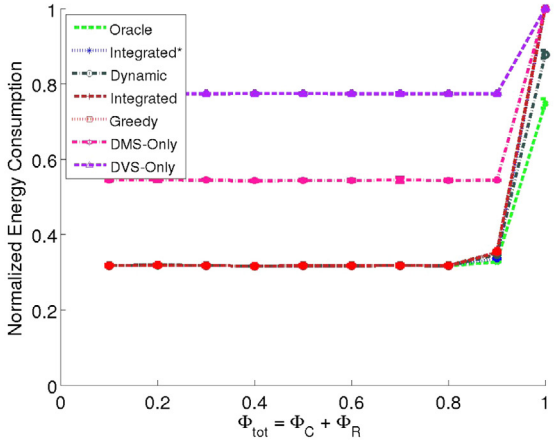
We implemented the following schemes:

- *Integrated*:* The proposed continuous domain joint DMS-DVS algorithm discussed in Section 3 that uses KKT conditions. The slack time is assigned for scaling CPU frequency and modulation level based on the value of power ratio and workload probabilities. This solution is fast but assumes a continuous CPU frequency and modulation level spectrum, as elaborated in Section 3.
- *Integrated:* This algorithm uses the mixed binary integer optimization problem formulation of Section 4, to solve the discrete domain problem optimally. Any integer programming tool could be used here, for small- to medium-size problem instances.
- *Greedy:* This is a heuristic algorithm that we developed to provide a fast but approximate solution to the discrete domain problem. The heuristic is inspired by the concept of marginal return discussed in the solution of the continuous domain. The algorithm approximates the optimal solution by finding the scheduling units (cycle groups or data packets) that provide the maximum energy gains, if slowed down by one speed level, and re-iterating as long as available slack allows. The algorithm works as follows: It first assigns the maximum speed level to all the scheduling units. The objective is to find the scheduling unit that yields the maximum potential gain by scaling by one speed level. We define a new measure called *Energy Probability Product (EPP)* that captures the expected energy gain by computing the product of the execution probability of the scheduling unit and the amount of energy saving per slack if slowed down by one level. The scheduling unit with the maximum EPP is chosen as a candidate at each iteration to scale down by one level. This step is repeated as long as the available slack is not exhausted. The algorithm's runtime complexity can be improved using a priority queue, such as max-heap, to which the scheduling units are inserted based on their EPP values. The insertion time to the priority queue is of $O(W+M)lg(W+M)$ time complexity where $(W+M)$ is the number of scheduling units. Each cycle group or data packet is added to the queue for at most n or m times, respectively, making the algorithm's overall complexity $O((W \times n + M \times m)(W+M)log(W+M)$.
- *DVS-only:* This scheme fixes modulation level to its maximum value and uses the entire slack time for scaling the CPU frequency in the manner described in [10], assuming discrete frequency levels. The extra slack time, if any, will remain unused.
- *DMS-only:* This technique assigns CPU frequency to its maximum level while applying DMS to the communication task. To do so, the time required to perform the computational workload at the maximum speed is subtracted from the deadline and the discrete optimization problem is solved by taking only the communication energy into account. DVS will not be applied in this scheme even if a part of slack time remains unused after performing modulation scaling.
- *Dynamic:* This scheme is introduced to model the systems that can, during the execution of the frames, adaptively re-compute or lookup optimal modulation levels based on the actual CPU usage. Specifically, at run-time, when the computation sub-task completes, the optimization problem is re-solved online to determine the optimal modulation level by considering the actual slack time before the deadline and the probabilistic workload profile of the communication sub-task. Since there are W possibilities for the computational workload, the designer needs to provide a

Please cite this article in press as: M. Bandari, et al., On minimizing expected energy usage of embedded wireless systems with probabilistic workloads, Sustain. Comput.: Inform. Syst. (2016), http://dx.doi.org/10.1016/j.suscom.2016.02.004

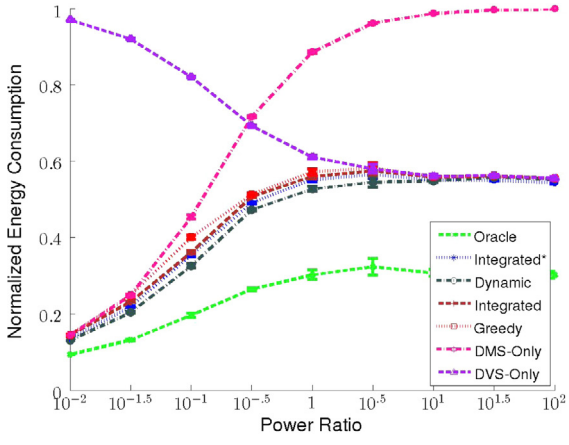


(a) Uniform Distribution

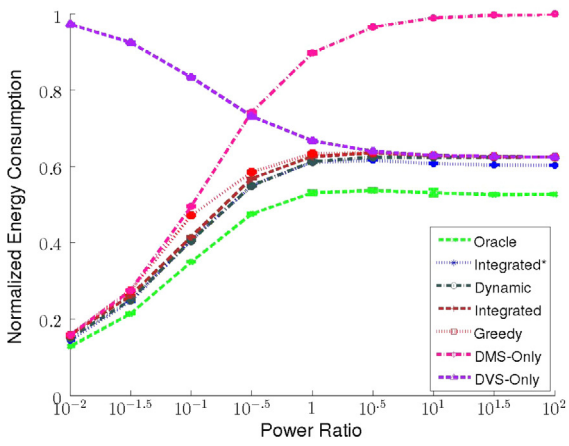


(b) Heavy Tailed Distribution

Fig. 4. Impact of total maximum workload utilization.



(a) Uniform Distribution



(b) Heavy Tailed Distribution

Fig. 5. Power ratio impact, $\Phi_C = 0.7$, $\Phi_R = 0.1$.

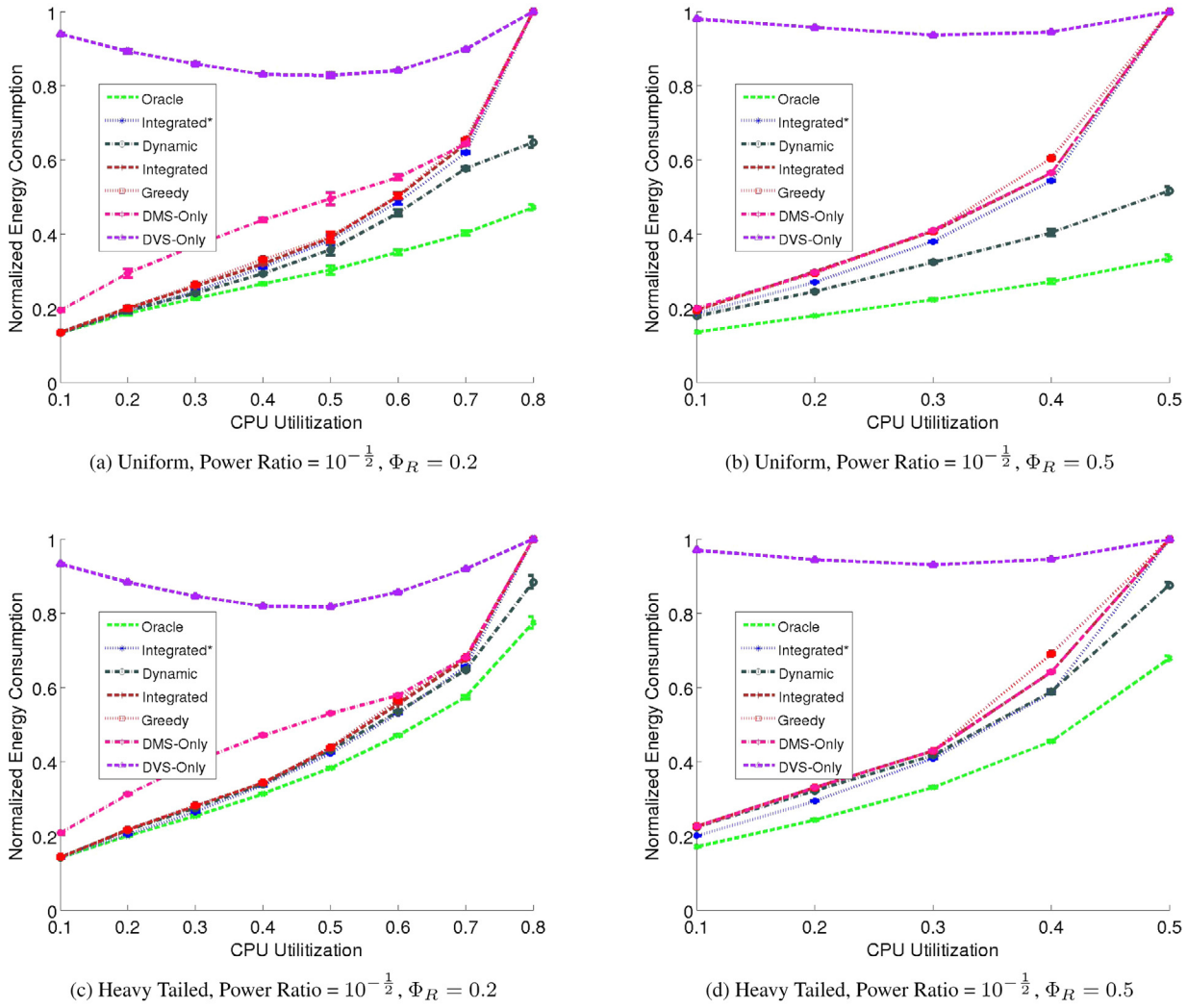
radio speed schedule for each of those possibilities and store in a look-up table.

- *Oracle:* This approach pre-supposes a clairvoyant scheduler that knows the exact value of computation and communication workloads in advance and scales both CPU frequency and modulation level to get the best use of the slack time to minimize the overall energy. The *Oracle* approach is not practical; however, it is included in the evaluation as the yardstick algorithm whose performance establishes the upper bound on the performance of *any* practical algorithm. Note that since *Oracle* is assumed to be aware of the exact workload, a speed schedule is not required and both computation and communication sub-tasks run at a constant speed during their execution to minimize the total energy.

Our results are divided into sections exploring the impact of varying total maximum workload utilization, varying *PR* (the relative maximum CPU to radio power), varying CPU utilization or radio utilization, the impact of frequency-independent CPU power, as well as the impact of the workload variability. Each data point represents the average energy consumption that results from applying the respective energy management algorithm, and is derived from averaging 1000 randomly generated tasks. For readability purposes the energy consumption is normalized with respect to the energy consumption of applying *no power management (NPM)*, so that the *lower* the reported normalized energy, the more effective is the respective energy management approach. *NPM* uses the maximum CPU frequency and highest modulation levels. The plots in the rest of this discussion show 95% confidence intervals for each scheme. As can be seen, the intervals of different schemes are small enough not to overlap, unless in regions that the schemes converge in performance. We note that due to the large number of schemes that are considered, the plots are can be inspected best in colored output (or online).

The first experiment, shown in Fig. 4, presents how the normalized energy consumption changes as a function of maximum total utilization ($\Phi_{tot} = \Phi_R + \Phi_C$). In these experiments, $PR = 10^{-1/2}$ and $\Phi_C/\Phi_R = 2.92$. As it can be seen, the results of both distributions exhibit the same trends. As expected, the *Oracle* strategy yields the highest energy savings, since it can distribute slack between DVS and DMS optimally with the pre-knowledge of the workload. *Dynamic* exhibits the next best performance. *Integrated*'s relative improvement decreases as the total maximum utilization increases, so the three off-line algorithms end up consuming the same amount of energy when $\Phi_{tot} = 1$, since they all have to run at the maximum speed to guarantee the deadline meet. This figure shows that the improvement of *DMS-only* over *DVS-only* is more pronounced in the case of the uniform distribution. The reason is that in heavy tailed distribution, the probability of having higher workloads is higher than the uniform distribution, and so the optimization algorithms perform more conservatively. Since these experiments show a wide range of workloads, the difference between other approaches is not very clear and will be more visible in the following set of experiments.

Please cite this article in press as: M. Bandari, et al., On minimizing expected energy usage of embedded wireless systems with probabilistic workloads, Sustain. Comput.: Inform. Syst. (2016), http://dx.doi.org/10.1016/j.suscom.2016.02.004



(a) Uniform, Power Ratio = $10^{-\frac{1}{2}}$, $\Phi_R = 0.2$

(b) Uniform, Power Ratio = $10^{-\frac{1}{2}}$, $\Phi_R = 0.5$

(c) Heavy Tailed, Power Ratio = $10^{-\frac{1}{2}}$, $\Phi_R = 0.2$

(d) Heavy Tailed, Power Ratio = $10^{-\frac{1}{2}}$, $\Phi_R = 0.5$

Fig. 6. Impact of varying maximum CPU utilization for uniform and heavy tailed distributions.

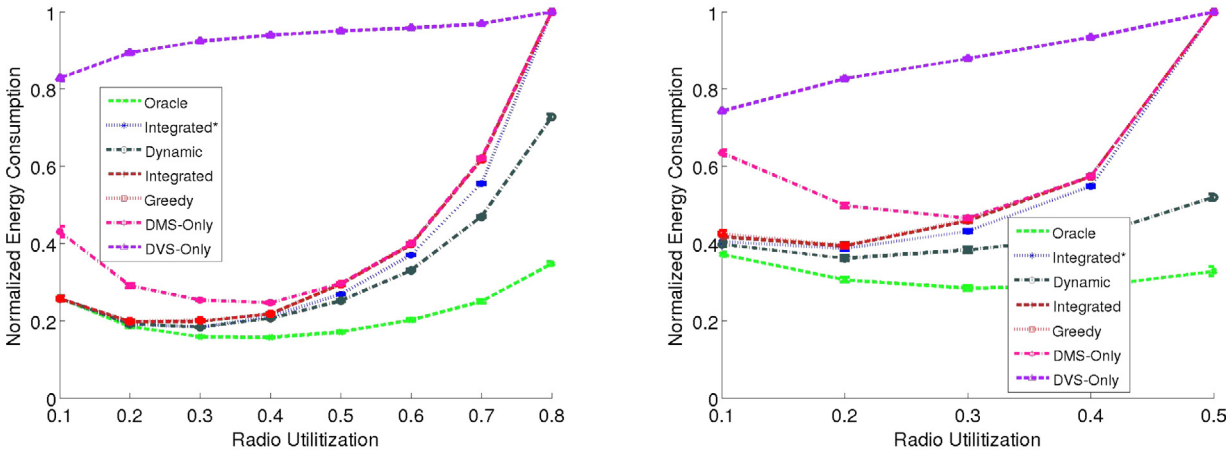
The next set of results investigate how the power ratio *PR* and the relative value of maximum CPU utilization to maximum radio utilization impact energy savings. This is important to characterize the best possible performance for a given hardware and workload settings. Fig. 5 shows the effect of power ratio on the relative performance of the schemes. Recall that the power ratio *PR* is defined as the ratio of CPU power to radio power. When the ratio is greater than 1, CPU is the higher power consumer component of the system, while radio dissipates more energy when this value is less than 1. As can be seen, *Integrated* converges towards either *DMS-only* or *DVS-only* at operating regions with extreme imbalances in the power ratio *PR*. However, *Integrated* is quite close to *Dynamic* and substantially improves energy consumption when *PR* is close to 1 (when both radio and CPU have the same share of the total power consumption). The system in this experiment has substantially higher computational demand ($\Phi_C = 0.7$ vs. $\Phi_R = 0.1$). We experimented with different combinations of CPU and radio workloads. The lower the share of the CPU in the total workload, the slower *DVS-only* converges to the optimum solution. The same observation is true about radio workload and *DMS-only*. Furthermore, *Dynamic* loses its effectiveness when CPU becomes the power hungry component of the system.

Fig. 6 shows the effect of the maximum CPU utilization. These experiments fix the ratio of maximum radio utilization to 0.2 and 0.5, and the power ratio to $10^{-1/2}$. The maximum CPU utilization varies from 0.1 to $1 - \Phi_R$. The plots at the top show the experiments with the uniform distribution and the ones at the bottom pertain to the heavy tailed distribution. The trends are the same in all four plots. With the maximum radio utilization of 0.2, the system experiences low total utilization at small values of CPU utilization, which provides significantly more slack time. As a result, *Integrated* starts off with the same performance as both *Oracle* and *Dynamic* as shown in Fig. 6a and c. When the CPU utilization increases, the algorithms cannot exploit slack time very effectively (except for *Oracle* and *Dynamic*.

The communication sub-task dominates the workload in experiments shown in Fig. 6 b and d. This experiment verifies the importance of *DMS-Only* scheme that may perform close to *Integrated* in applications with high communication workload density. Note that *DMS-Only* outperforms the *Greedy* algorithm in such cases. In both cases the gap between the different approaches is less pronounced in the heavy tailed distribution. This is due to the higher likelihood of having larger workloads and less variation in dynamic slack values. Note that even in this case, the gap between *DVS-Only* and the other optimal algorithms shows the importance of choosing the right algorithm to get the best performance out of the static slack.

The experiments shown in Fig. 7 present the relative performance when varying the maximum radio utilization for power ratio of $10^{-1/2}$. While substantially improving performance

Please cite this article in press as: M. Bandari, et al., On minimizing expected energy usage of embedded wireless systems with probabilistic workloads, Sustain. Comput.: Inform. Syst. (2016), http://dx.doi.org/10.1016/j.suscom.2016.02.004



(a) Power Ratio = $10^{-\frac{1}{2}}$, $\Phi_C = 0.2$ (b) Power Ratio = $10^{-\frac{1}{2}}$, $\Phi_C = 0.5$

Fig. 7. Impact of varying maximum radio utilization for uniform distribution.

compared to *DVS-only*, the *Integrated* scheme approaches *DMS-only* as the radio utilization increases, implying that it becomes more important to devote the slack time to slow down the radio. *Integrated* outperforms *DMS-only* at low radio utilization values. As can be seen, the schemes other than *Oracle* and *Dynamic* have the same behavior when the total utilization is 1.0. They all set CPU frequency and modulation level at the maximum value in order to prevent potential deadline violations. The gap between schemes is more distinctive when the system is more computationally intensive, that is, at high CPU utilization values (Fig. 7b). We repeated the same experiments for heavy tailed and observed the same trend but with less gap between *Oracle* and other algorithms due to less variation of the dynamic slack as explained before. Figs. 6 and 7 show that as total utilization increases and the potential slack decreases, *Integrated** performs more conservatively.

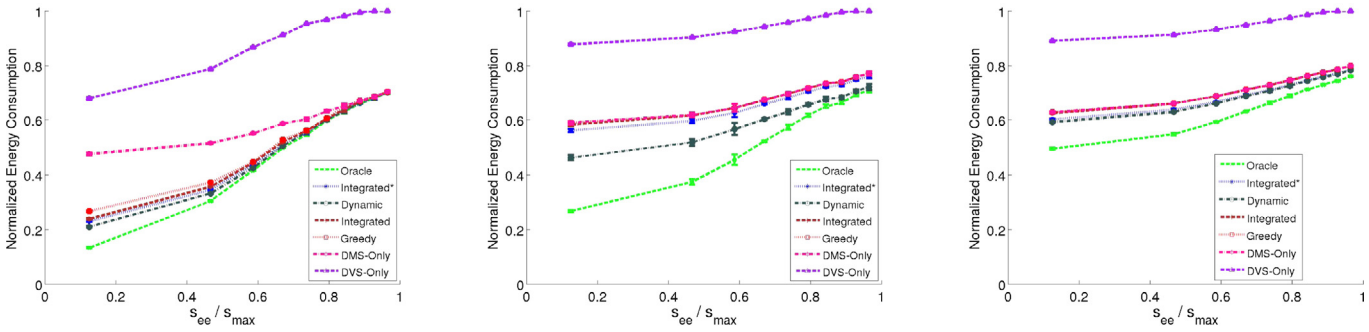
The next experiment evaluates the effect of the energy-efficient speed for DVS on the performance of the proposed algorithms. The existence of the frequency-independent power component P_{ind} implies the existence of an energy-efficient CPU frequency s_{ee} below which DVS loses its efficiency (Section 3). In general, the higher P_{ind}, the higher s_{ee}. In these experiments, we gradually increased the value of P_{ind} in such a way that s_{ee} changes in the range of $[s_{\min}, s_{\max})$. Fig. 8 shows the results of these experiments for both uniform and heavy tailed distributions when $\frac{\Phi_C}{\Phi_R} = 2.92$ and for two separate cases of $\Phi_{tot} = 0.6$, and $\Phi_{tot} = 0.9$. In the experiments, the power ratio equals $10^{-1/2}$ as before. The *DVS-Only* algorithm's performance degrades quickly as s_{ee} increases, as DVS's energy management capability is seriously constrained. However, the other algorithms have the option of allocating the slack to slow down the communication subtask. The next observation is that other algorithms' performance converges to that of *DMS-Only* when s_{ee} approaches $s_{\max}$, because there is no benefit in applying DVS in that region.

The last set of experiments explore the impact of workload variability. Specifically, we investigate how the performance of the algorithms change as a function of the best-case to worst-case workload ratio Φ^{BC}/Φ_{tot}. The smaller this ratio, the higher dynamic slack can be expected at run-time, and the better the opportunity to save power at run-time. When the ratio approaches to value 1, the algorithms can only benefit from the static slack. We ran the experiments for multiple choices of total utilization and probability distributions. In the experiments, Φ_C/Φ_R is set to 2.92. When the total utilization is rather small, the static slack would be enough for power management algorithms to get the best use of DVS and DMS and so the only difference is in the inherent capabilities of the algorithms in saving energy for the given workload mix. This can be seen in Fig. 5. However, when the utilization is high, the algorithms differ in how well they can exploit the dynamic slack. Moreover, this difference decreases as Φ^{BC}/Φ_{tot} increases. Also, the energy consumption of algorithms increases when the expected value of dynamic slack decreases. The results in the plots are normalized with respect to energy consumption of maximum speed settings at $\Phi^{BC}/\Phi_{tot} = 1$. Fig. 9b shows the effect of the workload variability when utilization is high. There is a rather large gap between *DVS-only* and other approaches when $\Phi^{BC}/\Phi_{tot} = 1$. At that point none of the algorithms can rely on dynamic slack. However, since radio is the major power consumer in this experiment ($PR = 10^{-1/2}$) and the underlying workload distribution is heavy tailed, the first packets accounts for a significant portion of the overall system energy. Consequently, the ability of scaling down the modulation level even only the first packets provides non-trivial energy savings. Note this effect is less pronounced in the case of the uniform distribution (Fig. 9c). We observe that the energy consumption figures increase more visibly with the increasing Φ^{BC}/Φ_{tot} ratio in the case of the uniform distribution due to the lower average-case workload.

6. Related work

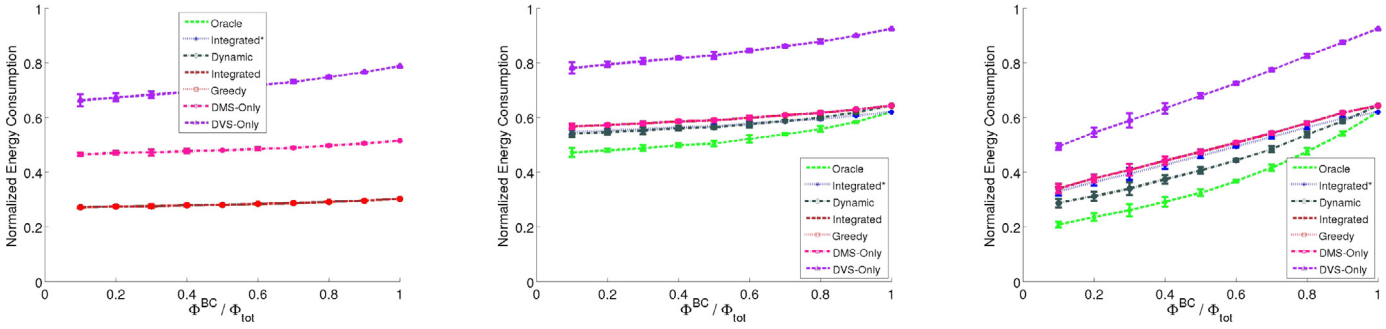
Dynamic Voltage Scaling (DVS) is a well-known energy management technique that trades off the CPU dynamic power consumption with task execution time by adjusting the processor's supply voltage and frequency. It is based on the fact that the dynamic power is a convex function of the CPU frequency. Moreover, in many applications the peak workload is significantly higher than the average-case workload. This implies that only a small fraction of jobs require the maximum performance to finish on-time. In general, unused CPU time is called the *slack time*. Slack time may be *static*, known to the designer at the design time; or *dynamic* and due to early completions of jobs. The designer can then exploit the dynamic and static slack for common workload profiles to slow down the CPU and save energy. DVS is now widely used in current microprocessors, including the Intel XScale architecture [18] and the AMD processors with the PowerNow! feature [19]. DVS methods are particularly effective for systems with time-varying workloads and real-time constraints, such as embedded control and multimedia applications [4,20]. One main research question

Please cite this article in press as: M. Bandari, et al., On minimizing expected energy usage of embedded wireless systems with probabilistic workloads, Sustain. Comput.: Inform. Syst. (2016), http://dx.doi.org/10.1016/j.suscom.2016.02.004



(a) Uniform, $\Phi_{tot} = 0.6$ (b) Uniform, $\Phi_{tot} = 0.9$ (c) Heavy Tailed, $\Phi_{tot} = 0.9$

Fig. 8. Impact of the energy-efficient speed for DVS.



(a) Heavy Tailed, $\Phi_{tot} = 0.3$ (b) Heavy Tailed, $\Phi_{tot} = 0.9$ (c) Uniform, $\Phi_{tot} = 0.9$

Fig. 9. Impact of increasing best case workload ratio.

is how to predict the slack time and how to optimally allocate the slack time to save energy.

Early works in DVS solely focused on the dynamic CPU power consumption, and discounted off-chip system components and frequency-independent power [4,16]. The consideration of the frequency-independent power introduces the concept of the *energy-efficient frequency* which is the minimum frequency below which further scaling does not improve energy savings. Off-chip power accounts for the power consumption of non-CPU components of a system such as memory and I/O devices and it does not scale with the CPU frequency. The frequency-independent power component and off-chip workloads are taken in account in [21]. The authors tackle the problem of finding the optimum frequency in a task set, given on chip and off chip task utilization and CPU's energy efficient frequency. The objective is to minimize total energy consumption considering the worst case CPU cycle for each task.

In multi-task systems if the frequency level is restricted to remain constant during task execution and can only change at task context switch times, applying DVS reduces to distribution of slack time among tasks. This problem is called the *inter-task DVS* and includes dynamic and static phases. Static phase decides the initial distribution of static slack. Dynamic slack produced as a result of early completion of tasks is reclaimed using dynamic slack reclamation algorithms. Inter-task DVS schemes are categorized as *greedy, proportional*, and *statistical* [4,10,16]. The proportional scheme distributes the slack evenly among unexecuted tasks. The greedy scheme allocates the entire slack to the next ready task, and the statistical scheme uses average execution cycles of tasks as a predictor of their future requirements. The *intra-task DVS* algorithms, on the other hand, allow frequency change during the execution of a single task. This is particularly useful to minimize the expected energy if the workload is known only probabilistically. In general, the optimal solutions start with a low speed and increase it gradually to match the actual workload, while still guaranteeing the timing constraints. PACE [12] and GRACE [9] are two common intra-task DVS techniques that differ in deadline constraint (hard vs. soft) and implementation details such as probability estimation techniques. Zhu et al. [11] investigates the trade-off between DVS and reliability guarantee and provides an algorithm to minimize the energy consumption while meeting the reliability requirements.

Device Power Management (DPM) is a system level energy saving approach that put the devices to low-power (sleep) states when they are not in use in order to save energy. DPM techniques can be stochastic, predictive, and timeout-based. Real-time systems mostly apply predictive technique which involves predicting the next usage time of the off-chip devices. The work in [13] investigates the effect of DVS and DPM for probabilistic workload. Devadas and Aydin [22] studies the interplay between DPM and DVS in presence of off-chip components. The approach is based on finding minimum energy consumption in regions distinguished by devices' *break-even times*.

The radio is a major power consumer in WSNs, making it a crucial target for network-level energy management in those systems. Dynamic Modulation Scaling (DMS) trades off power consumption with transmission time by scaling the number of bits per symbol. DMS also comes at the cost of synchronization overhead between two parties to coordinate the exact modulation level. DMS is most effective where the transmit power dominates the electronics power, which is true except for short range transmissions. It however has more complications compared to DVS due to channel variation over time which may change the coefficients in energy expressions. This could be accounted for by sampling the channel periodically and adjusting the coefficients as necessary.

Please cite this article in press as: M. Bandari, et al., On minimizing expected energy usage of embedded wireless systems with probabilistic workloads, Sustain. Comput.: Inform. Syst. (2016), http://dx.doi.org/10.1016/j.suscom.2016.02.004

Fateh and Govindarasu [23] reclaims dynamic slack produced by data redundancy using DMS in a hybrid TDMA-CDMA MAC protocol. A node may start transmission earlier if the predecessors skip their token due to data redundancy. This leaves the node with more slack time to be used by DMS. Moreover, for wireless embedded nodes with both substantial computational and communication workloads, both DVS and DMS techniques are relevant. Zhang et al. [7] suggests an algorithm to apply joint DVS–DMS at network level to maximize the minimum battery level among nodes. This is done by examining all possible radio and CPU speed settings in each node. Fateh and Manimaran [24] proposes a scheme to evaluate the maximum amount of slack in an interference aware precedence constrained task set by scheduling independent components together. They implicitly assume the same time requirement before and after scaling for both computation and communication subtasks. Those works consider exclusively deterministic workloads. Our work uses the probabilistic features of the workload to jointly schedule DVS and DMS with real-time constraints.

7. Conclusions

This paper addressed the problem of minimizing energy consumption in embedded wireless real-time systems. Our approach was to investigate the use of both Dynamic Voltage Scaling and Dynamic Modulation Scaling techniques under probabilistic workloads, and real-time constraints. We presented an integrated DVS–DMS control algorithm that minimizes overall expected energy consumption. We enhanced our solution to account for discrete levels of CPU frequency and radio modulation level. We simulated the performance of this approach against several design options, as well as a yardstick *Oracle* algorithm that knows the workload in advance. We found that under most workload mixes and relative CPU vs. radio power consumption figures our approach produces significant energy savings. For fast calculation of the speed schedule we proposed a greedy heuristic that provides close to optimum results. This work strongly suggests the desirability of using combined DVS and DMS control algorithms in embedded wireless systems.

Acknowledgements

This work is supported by NSF under grants CNS-1116122 and CNS-1205453.

Appendix A.

In this appendix, we provide the details of the solution to the optimization problem presented in Section 3. At the high-level, the algorithm proceeds as follows. We first apply the Lagrange multipliers method to solve the optimization problem by considering only the deadline constraint, temporarily ignoring the deadline constraint. This version of the problem is called the problem *DVMS-D*. Then we consider the problem where only the deadline and lower bound constraints on computation and communication times are taken into account (the problem *DVMS-L*). We solve *DVMS-L* by iteratively adjusting the solution of *DVMS-D*, if necessary. Finally, the original problem that considers also the upper bound constraints (called the problem *DVMS*), is obtained by adjusting the solution to the problem *DVMS-L*. In the following, we discuss the algorithms to solve these three optimization problems.

A.1. Problem DVMS-D: case of the deadline constraint

The problem *DVMS-D* is defined as follows:

$$\text{Minimize} \quad \sum_{j=1}^{W} \Gamma_j^{cp} C_{ef} (t_j^{cp})^{1-\alpha} + \Gamma_j^{cp}.p_{ind}.t_j^{cp} + \sum_{i=1}^{M} \Gamma_i^{cm} R_s t_i^{cm} \left(C_s.\chi(t_i^{cm}) + C_e\right)$$

$$\text{Subject to} \quad \sum_{i=1}^{M} t_i^{cm} + \sum_{j=1}^{W} t_j^{cp} = D$$

We apply Lagrangian multipliers technique to the above, which yields the following Lagrangian:

$$L(t_i^{cm}, t_j^{cp}, \lambda) = \sum_{j=1}^{W} \Gamma_j^{cp} C_{ef} (t_j^{cp})^{1-\alpha} + \Gamma_j^{cp}.p_{ind}.t_j^{cp} + \sum_{i=1}^{M} \Gamma_i^{cm} R_s t_i^{cm} \left[C_s.\chi(t_i^{cm}) + C_e\right] + \lambda \left(\sum_{i=1}^{M} t_i^{cm} + \sum_{j=1}^{W} t_j^{cp} - D \right)$$

Above λ is the dual variable. The dual function maximizes the Lagrangian function over its primal variables is given by:

$$\frac{\delta L(t_i^{cm}, t_j^{cp}, \lambda)}{\delta t_j^{cp}} = \Gamma_j^{cp} C_{ef} (1-\alpha)(t_j^{cp})^{-\alpha} + \Gamma_j^{cp}.p_{ind} + \lambda$$

$$\frac{\delta L(t_i^{cm}, t_j^{cp}, \lambda)}{\delta t_i^{cm}} = \Gamma_i^{cm} R_s \left[C_s.\chi(t_i^{cm}) + C_e + C_s \chi'(t_i^{cm}) t_i^{cm}\right] + \lambda$$

We define the *marginal energy return* function of radio packet time allocation as $w_i^{cm}(t_i^{cm})$. Similarly, the marginal energy return function of each cycle group is defined $w_j^{cp}(t_j^{cp})$. Analytically, these functions are obtained by obtaining the value of λ that sets Lagrangians to zero:

$$w_j^{cp}(t_j^{cp}) = -\Gamma_j^{cp} C_{ef} (1-\alpha)(t_j^{cp})^{-\alpha} - \Gamma_j^{cp}.p_{ind} \tag{12}$$

$$w_i^{cm}(t_i^{cm}) = -\Gamma_i^{cm} R_s \left[C_s.\chi(t_i^{cm}) + C_e + C_s \chi'(t_i^{cm}) t_i^{cm}\right] \tag{13}$$

For succinct representation, we define $\psi(t_i^{cm})$ as $\chi(t_i^{cm}) + \chi'(t_i^{cm}) t_i^{cm}$. The optimum solution to *DVMS-D* is obtained by equating all marginal returns:

$$(t_j^{cp})^* = \left(\frac{\lambda^* + \Gamma_j^{cp}.p_{ind}}{(\alpha-1)\Gamma_j^{cp} C_{ef}} \right)^{-1/\alpha}$$

$$(t_i^{cm})^* = (t_i^{cm})^+(\lambda^*) = \psi^{-1} \left(\frac{\frac{-\lambda^*}{\Gamma_i^{cm} R_s} - C_e}{C_s} \right)$$

λ^*, the common dual variable, is obtained by solving:

$$\sum_{i=1}^{M} (t_i^{cm})^+(\lambda) + \sum_{j=1}^{W} \left(\frac{\lambda + \Gamma_j^{cp}.p_{ind}}{(\alpha-1)\Gamma_j^{cp} C_{ef}} \right)^{-1/\alpha} = D$$

Please cite this article in press as: M. Bandari, et al., On minimizing expected energy usage of embedded wireless systems with probabilistic workloads, Sustain. Comput.: Inform. Syst. (2016), http://dx.doi.org/10.1016/j.suscom.2016.02.004

For example, the optimum communication time equations for QAM modulation scheme is:

$$(t_i^{cm})^* = \frac{\rho \log 2}{R_s + R_s W_f\left[\frac{C_e R_s y - C_s R_s \Gamma_i^{cm} + \lambda^*}{C_s e R_s \Gamma_i^{cm}}\right]}$$

Similarly, 2^b-PAM modulation approach yields the optimum communication times as:

$$(t_i^{cm})^* = \frac{\rho \log 4}{R_s + R_s W_f\left[\frac{3C_e R_s \Gamma_i^{cm} - C_s R_s \Gamma_i^{cm} + 3\lambda^*}{C_s e R_s \Gamma_i^{cm}}\right]}$$

W_f in the above equations is the Lambert W-function, also called the omega function, the inverse function of $W \cdot e^W$. It appears in the optimum solution of QAM and 2^b-PAM because of the term e^b/b in their corresponding energy functions.

Time complexity: There are $M+W$ unknown variables whose values need to be determined. When the closed formula for ψ^{-1} is available for the corresponding modulation, such as in above cases, the optimum values of $(t_i^{cm})^*$ and $(t_j^{cp})^*$ can be calculated each in time $O(1)$. There are $M+W$ such calculations, resulting in time complexity of $O(M+W)$.

A.2. Problem DVMS-L: case of deadline and lower bound constraints

Next, we consider adding the lower bound constraints to the problem *DVMS-D*, obtaining the problem *DVMS-L*:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{j=1}^{W} \Gamma_j^{cp} C_{ef} (t_j^{cp})^{1-\alpha} + \Gamma_j^{cp}.p_{ind}.t_j^{cp} \\
& + \sum_{i=1}^{M} \Gamma_i^{cm} R_s t_i^{cm} \left[C_s \chi(t_i^{cm}) + C_e\right] \\
\text{Subject to} \quad & \sum_{i=1}^{M} t_i^{cm} + \sum_{j=1}^{W} t_j^{cp} \le D \\
& t_j^{cp} \ge t_{\min}^{cp} \\
& t_i^{cm} \ge t_{\min}^{cm}
\end{aligned}$$

Obviously, if the solution to the problem *DVMS-D* satisfies the lower bound constraints, then it is also a solution to the problem *DVMS-L*. Otherwise, the problem becomes in essence identical to the nonlinear optimization problem discussed in [25]. As shown in [25] by manipulating the Karush–Kuhn–Tucker conditions, in that case, the optimal value of the variable which gives the minimum marginal return (according to Eqs. (13) and (12)) at the corresponding lower bound is equal to that lower bound.

This property suggests an iterative solution [25] that invokes the algorithm to solve *DVMS-L*: Call the algorithm to solve *DVMS-D*, and as long as some lower bounds are violated, in each iteration, fix the value of one variable (with smallest marginal return) to its lower bound, update the deadline D, before re-solving for the remaining variables. A straightforward implementation would be of time complexity $O(M+W)^2$, because there are at most $M+W$ iterations, and in each iteration, solving *DVMS-D* can take at most $O(M+W)$ time. Further, as shown in [25], a binary search like technique can be adopted to figure out more quickly what variables should be set to the lower bounds, yielding an overall complexity of $O((M+W)\log(M+W))$.

A.3. Problem DVMS: combining all the constraints

Finally, we add the upper bound constraints to obtain our original problem derived at the end of Section 3, that we denote as the Problem *DVMS*.

$$\begin{aligned}
\text{Minimize} \quad & \sum_{j=1}^{W} \Gamma_j^{cp} C_{ef} (t_j^{cp})^{1-\alpha} \Gamma_j^{cp}.p_{ind}.t_j^{cp} \\
& + \sum_{i=1}^{M} \Gamma_i^{cm} R_s t_i^{cm} \left[C_s \chi(t_i^{cm}) + C_e\right] \\
\text{Subject to} \quad & \sum_{i=1}^{M} t_i^{cm} + \sum_{j=1}^{W} t_j^{cp} \le D \\
& t_{\min}^{cp} \le t_j^{cp} \le t_{\max}^{cp} \\
& t_{\min}^{cm} \le t_i^{cm} \le t_{\max}^{cm'}
\end{aligned}$$

Assuming we have the solution to the problem *DVMS-L* as discussed previously, an iterative solution to the problem *DVMS* can be designed, again following the approach in [25]. Specifically, if the solution to *DVMS-L* satisfies the upper bound constraints of *DVMS*, then it is also a solution to *DVMS*. Otherwise, by using the same derivations as in [25], one can demonstrate that the variable that has the maximum marginal return value (Eqs. (13) and (12)) at the upper bound boundary should be set to that upper bound. Once again, this implies the existence of an iterative algorithm that repeatedly invokes the algorithm for *DVMS-L* as long as the upper bounds are violated, setting the value of at least one unknown to the lower bound as necessary, and updating the deadline value before invoking the algorithm for the remaining unknown variables. Since the complexity of solving *DVMS-L* is $O((M+W)\log(M+W))$, and it may be invoked at most $M+W$ times, the time complexity of solving *DVMS* is found as $O((M+W)^2 \log(M+W))$.

References

[1] L. Krishnamurthy, et al., Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea, in: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, ACM, 2005.
[2] D. Brunelli, C. Moser, L. Thiele, L. Benini, Design of a solar-harvesting circuit for batteryless embedded systems, IEEE Trans. Circ. Syst. I: Regul. Pap. 56 (11) (2009) 2519–2528.
[3] M.R. Jongerden, A. Mereacre, H.C. Bohnenkamp, B.R. Haverkort, J.-P. Katoen, Computing optimal schedules of battery usage in embedded systems, IEEE Trans. Ind. Inform. 6 (3) (2010) 276–286.
[4] H. Aydin, R. Melhem, D. Mossé, P. Mejí a-Alvarez, Power-aware scheduling for periodic real-time tasks, IEEE Trans. Comput. 53 (5) (2004) 584–600.
[5] C. Schurgers, V. Raghunathan, M.B. Srivastava, Power management for energy-aware communication systems, ACM Trans. Embed. Comput. Syst. 2 (3) (2003) 431–447.
[6] IEEEStandard for Local and metropolitan area networks, http://standards.ieee.org/findstds/standard/802.15.4g-2012.html.
[7] B. Zhang, R. Simon, H. Aydin, Harvesting-aware energy management for time-critical wireless sensor networks with joint voltage and modulation scaling, IEEE Trans. Ind. Inform. 9 (1) (2013) 514–526.
[8] T. Hamachiyo, Y. Yokota, E. Okubo, A cooperative power-saving technique using DVS and DMS based on load prediction in sensor networks, in: Fourth International Conference on Sensor Technologies and Applications (SENSORCOMM), IEEE, 2010.
[9] W. Yuan, K. Nahrstedt, Energy-efficient CPU scheduling for multimedia applications, ACM Trans. Comput. Syst. (TOCS) 24 (3) (2006) 292–331.
[10] R. Xu, D. Mossé, R. Melhem, Minimizing expected energy consumption in real-time systems through dynamic voltage scaling, ACM Trans. Comput. Syst. (TOCS) 25 (4) (2007) 9.
[11] D. Zhu, H. Aydin, J.-J. Chen, Optimistic reliability aware energy management for real-time tasks with probabilistic execution times, in: Real-Time Systems Symposium, IEEE, 2008.
[12] J.R. Lorch, A.J. Smith, Pace a new approach to dynamic voltage scaling, IEEE Trans. Comput. 53 (7) (2004) 856–869.
[13] B. Zhao, H. Aydin, Minimizing expected energy consumption through optimal integration of DVS and DPM, in: Proceedings of the ACM Conference on Computer-Aided Design, 2009.

Please cite this article in press as: M. Bandari, et al., On minimizing expected energy usage of embedded wireless systems with probabilistic workloads, Sustain. Comput.: Inform. Syst. (2016), http://dx.doi.org/10.1016/j.suscom.2016.02.004

[14] R. Marculescu, P. Bogdan, Cyberphysical systems: workload modeling and design optimization, IEEE Des. Test Comput. 28 (4) (2011) 78–87.
[15] B. Zhao, H. Aydin, D. Zhu, Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints, ACM Trans. Des. Autom. Electron. Syst. (TODAES) 18 (2) (2013) 23.
[16] P. Pillai, K.G. Shin, Real-time dynamic voltage scaling for low-power embedded operating systems, in: ACM SIGOPS Operating Systems Review, ACM, 2001.
[17] Y. Yu, B. Krishnamachari, V.K. Prasanna, Energy-latency tradeoffs for data gathering in wireless sensor networks, in: INFOCOM, Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, IEEE, 2004.
[18] 3rd Generation Intel XScale Microarchitecture, http://download.intel.com/design/intelxscale/31505801.pdf.
[19] AMD Powernow Technology, http://www.amd-k6.com/wp-content/uploads/2012/07/24404a.pdf.
[20] Z. Cao, B. Foo, L. He, M. Van Der Schaar, Optimality and improvement of dynamic voltage scaling algorithms for multimedia applications, IEEE Trans. Circ. Syst. 57 (3) (2010) 681–690.
[21] H. Aydin, V. Devadas, D. Zhu, System-level energy management for periodic real-time tasks, in: 27th IEEE International Real-Time Systems Symposium, IEEE, 2006.
[22] V. Devadas, H. Aydin, On the interplay of voltage/frequency scaling and device power management for frame-based real-time embedded applications, IEEE Trans. Comput. 61 (1) (2012) 31–44.
[23] B. Fateh, M. Govindarasu, Energy minimization by exploiting data redundancy in real-time wireless sensor networks, Ad Hoc Netw. 11 (6) (2013) 1715–1731.
[24] B. Fateh, G. Manimaran, Joint scheduling of tasks and messages for energy minimization in interference-aware real-time sensor networks, IEEE Trans. Mobile Comput. 14 (1) (2015) 86–98.
[25] H. Aydin, R. Melhem, D. Mosse, P. Mejia-Alvarez, Optimal reward-based scheduling for periodic real-time tasks, IEEE Trans. Comput. 50 (2) (2001) 111–130.

Please cite this article in press as: M. Bandari, et al., On minimizing expected energy usage of embedded wireless systems with probabilistic workloads, Sustain. Comput.: Inform. Syst. (2016), http://dx.doi.org/10.1016/j.suscom.2016.02.004