Energy management of embedded wireless systems through voltage and modulation scaling under probabilistic workloads*

Maryam Bandari, Robert Simon, and Hakan Aydin
Department of Computer Science
George Mason University, Fairfax, VA 22030
{mbandari, simon, aydin}@gmu.edu

***Abstract*—Many wireless embedded systems must deal with increasingly complex and time-varying workloads. Moreover, real-time constraints must be satisfied. Most of the existing energy management studies for such systems have focused on relatively simple models that assume deterministic workloads and consider a limited range of energy management techniques, including 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 various power control algorithms.**

I. Introduction

Energy management is critical for self-powered real-time embedded wireless systems, such as those designed for industrial process control, highway monitoring, and building surveillance. These systems have to maintain high levels of computation and communication performance while minimizing energy consumption [1]–[3]. Existing energy management techniques typically use "control knobs" that trade performance for energy savings. One commonly used power saving technique is Dynamic Voltage Scaling (DVS)[4]. DVS controls power consumption by reducing the CPU frequency and supply voltage (hence, the CPU speed). Another technique for power saving is Dynamic Modulation Scaling (DMS), which works by changing radio modulation levels and constellation sizes (hence, the communication speed) [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 under a deadline constraint. 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 five different algorithms, including a joint DVS-DMS approach. Using the 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 show how to compute the optimal *speed schedule* in which DVS and DMS parameters are adjusted to match the current workload conditions. Next, we conduct a simulation study to evaluate the benefits of our joint 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 a joint DVS-DMS strategy can provide non-trivial gains on the expected energy consumption, especially when the computation and communication workloads are relatively balanced.

*This work has been supported by the US National Science Foundation award CNS-1016974.

II. 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. One of the earliest works addressing real-time scheduling in processors with DVS capability is [12]. DVS is now widely used in current microprocessors, including the Intel XScale architecture [13] and the AMD processors with the PowerNow! feature [14]. The two major components of DVS-enabled processors, the adjustable DC-DC output voltage converter and the programmable clock generator, are studied in [15], [16] and [17]. DVS methods are particularly effective for systems with time-varying workloads and real-time constraints, such as embedded control and multimedia applications [4,18].

One focus of this paper is on probabilistic computational workloads with variable execution times. In these scenarios energy management schemes typically reclaim dynamic slack created by tasks running for less than their worst-case execution times [4,9,19]. Off-line and on-line profiling along with the intrinsic characteristics of the application may be utilized to derive a probability distribution function for the workload [20]. Research in this area includes intra-task DVS algorithms that offer optimization techniques for CPU frequency adjustment under probabilistic workloads [10,11,21].

We extend these earlier works to account for devices with radios that are Dynamic Modulation Scaling (DMS) capable. Within many embedded wireless systems the radio is a major source of power consumption, and therefore DMS control methods can lead to reduced overall energy expenditures [22]. DMS works by reducing the communication speeds with varying modulation modes or constellation sizes [5]. Despite its availability [6] relatively few works have addressed joint DVS-DMS power management approaches. One example is [23], which describes offline and online algorithms to allocate slack times based on energy gains. A similar slack distribution strategy is used in [24]. Using the concept of negotiated access periods, [25] selects job and CPU voltage levels. In a similar fashion [26,27] apply both dynamic power management (DPM) and DVS as power management approaches.

Both centralized and distributed joint DVS-DMS for energy harvesting wireless sensor networks are discussed in [7]. All combinations of frequencies and modulation levels are probed in [8] to find the settings that still meet the deadline of the predicted workload with minimum energy consumption. The authors compare the energy consumption of all members of the set to find the setting that results in minimum energy.

Our work differs from the above DVS and DMS studies by first focusing on a more realistic probabilistic workload model and then formulating an optimization problem for the minimization of the expected energy.

III. System Model

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

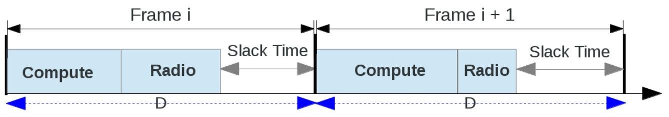


Fig. 1: Application Model

A. 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 $\frac{1}{D}$. 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 have varying resource demands from frame to frame, determined according to specific probability distributions, as explained below.

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:

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

where X is the random variable for the application's computation demand in a frame, and $p(X \le c)$ 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 [9,21,28]. Specifically, the available range of CPU cycles $[C_{min}, C_{max}]$ is divided into W discrete *cycle groups*, each with $\omega = \frac{C_{max} - C_{min}}{W}$ cycles. We denote the upper bound on the number of cycles in the i^{th} 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 [29]. 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 i^{th} cycle group are recorded [9]–[11]. More precisely, the fraction of invocations where the number of executed cycles falls in the i^{th} 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 $i = 1, \dots, W$. Observe that $\sum_{i=1}^{W} f_i^{cp} = 1$.

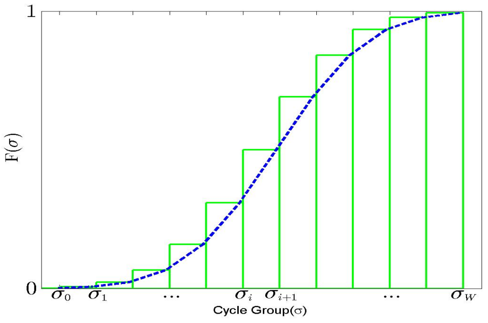


Fig. 2: Histogram-based approximation of the cumulative distribution function of the application's probabilistic workload

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 probability that the j^{th} 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]).

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 i^{th} packet being transmitted or received in one frame is then denoted by:

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

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*:

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

B. Power and Energy Models

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

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

Static power (also called *leakage* power) is necessary to keep the basic circuits on; it can be eliminated only by turning the processor off. Due to the periodic nature of our application and non-trivial overhead of turning the processor off and on at run-time, we assume that the static power is continuously consumed, and hence it is not manageable [10]. Dynamic power, on the other hand, is dissipated when the CPU executes tasks. On systems that are DVS-capable, CPU supply voltage is linearly related to the variable CPU frequency and the dynamic power is given by $p_d = C_l \cdot s^{\alpha}$, where s is the CPU frequency (speed), C_l 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 is in general a function of time (denoted by $s(t)$), the total energy consumed by the CPU in the interval (t_1, t_2) is given by:

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

As the time needed to execute Q cycles at a constant frequency s is $\frac{Q}{s}$, the dynamic energy consumed while executing one *cycle group* that consists of ω cycles at frequency s is given by [9,11]:

$$e_{\omega}^{cp} = C_l \cdot \frac{\omega}{s} \cdot s^{\alpha} = C_l \cdot \omega \cdot s^{\alpha-1} \tag{1}$$

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. 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:

$t_{bit}^{cm}(b) = \frac{1}{b \cdot R_s}$ 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 the 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.

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

$$\begin{aligned} e_{overall} &= \sum_{j=1}^{W} \Gamma_j^{cp} e_{\omega}^{cp} + \sum_{i=1}^{M} \Gamma_i^{cm} e_{\rho}^{cm} \\ &= \sum_{j=1}^{W} \Gamma_j^{cp} \cdot \omega \cdot C_l \cdot s_j^{\alpha-1} + \\ &\quad \sum_{i=1}^{M} \frac{\rho \cdot \Gamma_i^{cm}}{b_i} \cdot [C_s \cdot \phi(b_i) + C_e] \end{aligned} \tag{5}$$

The modulation level of the i^{th} packet and the execution frequency of the j^{th} cycle group are shown by b_i and s_j in the formula above. Table I summarizes the list of the most important notations used in this section.

TABLE I: 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_l	CPU switching capacitance
Γ_i^{cm}	The probability of sending the i^{th} packet
Γ_i^{cp}	The probability of executing the i^{th} cycle group
M	Maximum number of packets
W	Maximum number of cycle groups
b_i	The number of bits per symbol in the i^{th} packet
s_i	CPU frequency used to execute the i^{th} cycle group
D	Frame deadline (period)
$\phi(b)$	Modulation energy scaling function

IV. 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 [4,12]. 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 [10,21]. 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 they are increased for the next scheduling units based on the actual workload, in order to meet the deadline of the frame even under a worst-case scenario.

For example, consider a frame with a deadline of $95ms$, a worst-case processing time of $50ms$ (under maximum CPU speed), and a worst-case communication time of $25ms$ (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 $12.5ms$. The communication workload varies from one to three packets, each requiring $8.33ms$ 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

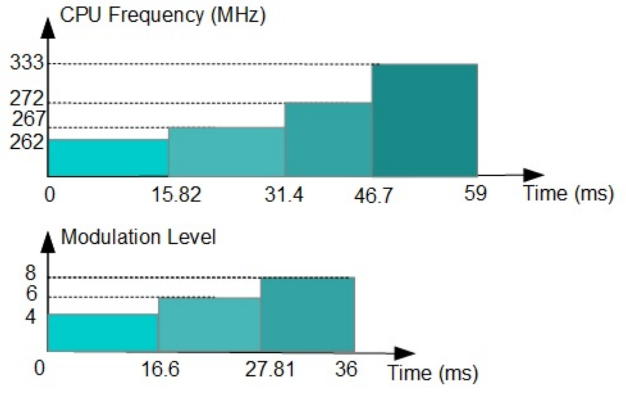


Fig. 3: Speed scheduling example

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 III. 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 j^{th} cycle group as s_j and the modulation level for the i^{th} packet as b_i, the energy optimization problem is written as:

$$\begin{gathered}
Minimize \sum_{j=1}^{W} \Gamma_j^{cp}.C_l.\omega.s_j^{\alpha-1} + \sum_{i=1}^{M} \frac{\rho.\Gamma_i^{cm}}{b_i}\left[C_s.\phi(b_i) + C_e\right] \\
s.t. \sum_{j=1}^{W} \frac{\omega}{s_j} + \sum_{i=1}^{M} \frac{\rho}{b_i R_s} \leq D \\
s_{min} \leq s_j \leq s_{max} \\
b_{min} \leq b_i \leq b_{max}
\end{gathered}$$

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

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

We denote $\phi(b_i) = \phi(\frac{\rho}{R_s t_i^{cm}})$ 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:

$$\begin{gathered}
Minimize \sum_{j=1}^{W} \Gamma_j^{cp} C_l \omega^{\alpha} (t_j^{cp})^{1-\alpha} + \\
\sum_{i=1}^{M} \Gamma_i^{cm} R_s t_i^{cm} \left[C_s \chi(t_i^{cm}) + C_e\right] \\
s.t. \sum_{j=1}^{W} t_j^{cp} + \sum_{i=1}^{M} t_i^{cm} \leq D \\
t_{min}^{cp} = \frac{\omega}{s_{max}} \leq t_j^{cp} \leq \frac{\omega}{s_{min}} = t_{max}^{cp} \\
t_{min}^{cm} = \frac{\rho}{b_{max}} \leq t_i^{cm} \leq \frac{\rho}{b_{min}} = t_{max}^{cm}
\end{gathered}$$

While the energy consumption is a convex function of t_i^{cm} and t_j^{cp}, a couple of observations are in order. The *computation* component of the expected energy, given by the second term of the objective function, monotonically decreases with the increasing computation time (t_i^{cm}), (i.e., with decreasing frequency (s)). However, the *communication* component of the expected energy has two terms, one which increases with the allocated communication time, and another one which decreases. This is to be expected, because while reducing the modulation level reduces the transmit energy, it tends to increase the electronic circuitry energy due to the need to keep the radio on for longer intervals. In other words, there is an *energy-efficient* modulation level, b_e, below which DMS is no longer effective, as observed in [30]. Its value can be determined analytically by setting the first derivative of the radio energy to zero. Consequently, after replacing b_{min} by $b'_{min} = max\{b_{min}, b_e\}$ 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:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{j=1}^{W} \Gamma_j^{cp} C_l \omega^{\alpha} (t_j^{cp})^{1-\alpha} + \sum_{i=1}^{M} \Gamma_i^{cm} R_s t_i^{cm} \left[C_s \chi(t_i^{cm}) + C_e\right] \\
\text{subject to} \quad & \sum_{j=1}^{W} t_j^{cp} + \sum_{i=1}^{M} t_i^{cm} = D \\
& t_{min}^{cp} = \frac{\omega}{s_{max}} \leq t_j^{cp} \leq \frac{\omega}{s_{min}} = t_{max}^{cp} \\
& t_{min}^{cm} = \frac{\rho}{b_{max}} \leq t_i^{cm} \leq \frac{\rho}{b'_{min}} = t_{max'}^{cm}
\end{aligned}$$

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.

V. 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 other design options.

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. We set each task invocation to have a frame deadline and period of $D = 100ms$. In each frame, both the total CPU processing time and total communication time (under maximum frequency and highest modulation level) vary from $10ms$ to $90ms$, respectively. Also, 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, respectively. We call these quantities the *utilization* of computation and communication subtasks. In our experiments, $\Phi_C + \Phi_R$ goes up to 0.9 (i.e., up to 90% of the available frame execution time). Within each frame the *actual workloads* are determined using a uniform probability distribution. Specifically, the actual utilization of the computation sub-task is determined randomly in the range $[\frac{\Phi_C}{W}, \Phi_C]$. Similarly, the actual utilization of the computation sub-component is determined randomly in the range $[\frac{\Phi_R}{M}, \Phi_R]$. We set $M = W = 10$.

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. 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 the other hand, CC2420 and XE1205 radios consume 60 mW and 65 mW, respectively. During our simulation experiments we changed the value of PR by keeping the value of radio power consumption constant and varying the CPU power consumption. We run 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 implemented the following schemes:

1) Joint: The proposed joint DMS-DVS algorithm discussed in Section IV. The slack time is assigned for scaling CPU frequency and modulation level based on the value of power ratio and workload probabilities.

2) 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]. The extra slack time, if any, will remain unused.

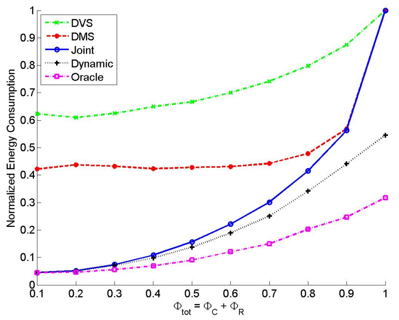


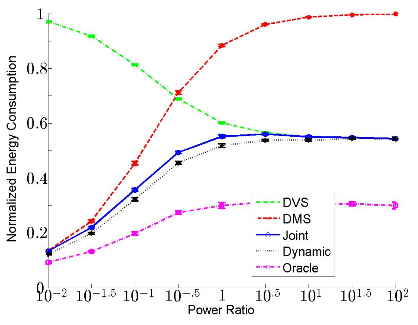
Fig. 4: Impact of total maximum workload utilization

3) DMS-only: This technique uses DVS at the maximum frequency while applying DMS. To do so, the time required to perform the computational workload at the maximum speed is subtracted from the deadline and the 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.

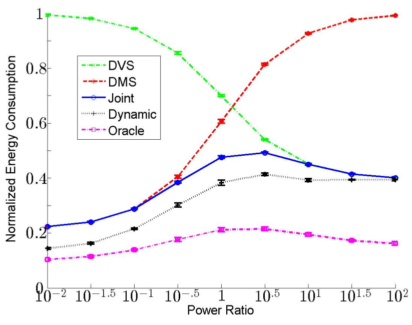
4) Dynamic: This scheme is introduced to model the systems that can, during the execution of the frames, adaptively re-compute optimal modulation levels based on the actual CPU usage. Specifically, at run-time, when the computation subtask completes the optimization problem is re-solved online to determine the optimal modulation level by considering the actual slack time before the deadline and the probabilistic workload profile of the communication sub-task. Note that due to the resource limitations, this online calculation is beyond the capabilities of many current embedded wireless devices; however, we include this algorithm in our comparison to have a more complete evaluation of the design possibilities.

5) 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.

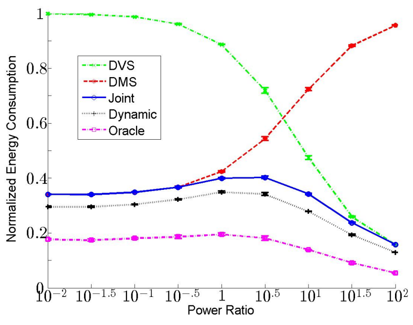
Our results are divided into sections exploring the impact of varying total maximum workload utilization, varying PR (the relative maximum CPU to radio power), varying CPU utilization or radio utilization, as well as determining, for *Joint*, the impact of both the power ratio, PR, and workload utilization. Each data point represents the average energy consumption of 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



(a) $\Phi_C = 0.7$, $\Phi_R = 0.1$



(b) $\Phi_C = 0.4$, $\Phi_R = 0.4$



(c) $\Phi_C = 0.1$, $\Phi_R = 0.7$

Fig. 5: Impact of varying power ratio

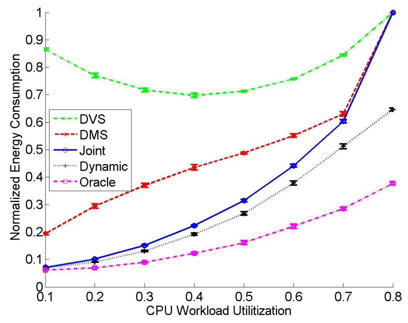
modulation levels.

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, $PR = 10^{-\frac{1}{2}}$ and $\frac{\Phi_C}{\Phi_R} = 2$. As expected, the *Oracle* strategy yields the highest energy savings, since it can clairvoyantly distribute slack between DVS and DMS. *Dynamic* exhibits the next best performance. *Joint* provides a substantial improvement over *DMS-only* and *DVS-only*, by maximum values of 38% and 57% respectively. *Joint*'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.

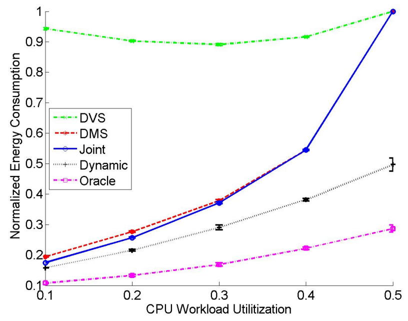
The next experiment aims at finding the operating regions in which *Joint* outperforms both *DVS-only* and *DMS-only* by significant margins. The results show that both the power ratio PR and the relative value of maximum CPU utilization to maximum radio utilization impact energy savings. 95% confidence intervals for each scheme are depicted in the figures. As can be seen, the intervals of different schemes are small enough not to overlap.

Figure 5 shows these results. As can be seen, *Joint* converges towards either *DMS-only* or *DVS-only* at operating regions with extreme imbalances in the power ratio PR. However, *Joint* is quite close to *Dynamic* and substantially improves energy consumption when PR is close to 1.

Figure 5a shows the effect of the power ratio PR for systems with higher computation than communication demand. The maximum improvement of *Joint* is 45% over *DMS-only* and 83% compared to *DVS-only*. *Dynamic* obtains no more than 3% improvement over *Joint*. Figure 5b shows when both CPU and radio have the same share of maximum utilization. For this operating case, *Joint* outperforms *DMS-only* and *DVS-only* by margins up to 58% and 77% respectively. *Oracle* provides an additional 9% improvement. Figure 5c displays the same effect when the application has greater communication workload. In that situation, the improvement of *Joint* over *DMS-only* is maximum of 79% while the improvement over *DVS-only* gets up to 65%. The extra improvement of *Dynamic* increases to at most 6% due to higher radio utilization.



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

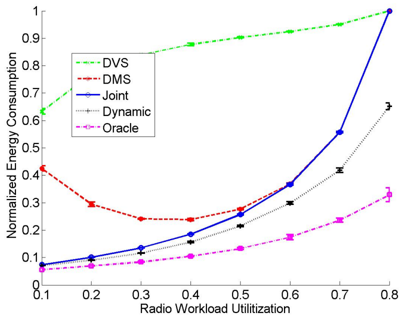


(b) Power Ratio = $10^{-\frac{1}{2}}$, $\Phi_R = 0.5$

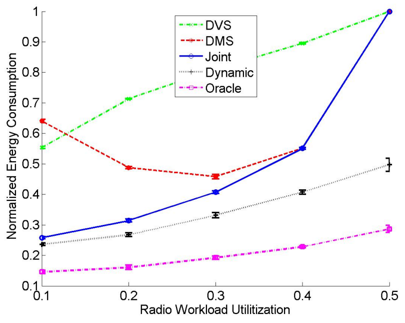
Fig. 6: Impact of varying maximum CPU utilization

Figure 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 $10^{-\frac{1}{2}}$. Maximum CPU utilization varies from 0.1 to $1 - \Phi_R$. With maximum radio utilization of 0.2, the system experiences low total utilization at small values of CPU utilization, which subsequently provides more slack time. As a result, *Joint* starts off with the same performance as both *Oracle* and *Dynamic* as shown in Figure 6a. It converges towards *DVS-only* with increasing CPU utilization. Communication dominates the workload in experiments shown in Figure 6b. The maximum improvement of *Joint* falls within 2% and 76% of *DMS-only* and *DVS-only* respectively. *Dynamic* generates a maximum improvement of

50%. This experiment verifies the importance of DMS scheme that may perform close to *Joint* in applications with high communication workload density.



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



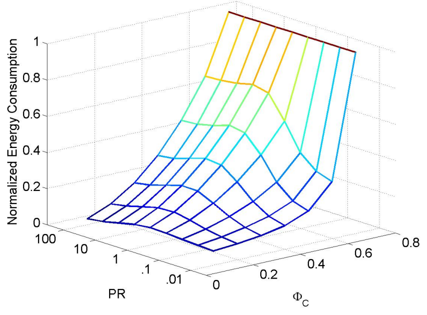
(b) Power Ratio = $10^{-\frac{1}{2}}$, $\Phi_C = 0.5$

Fig. 7: Impact of varying maximum radio utilization

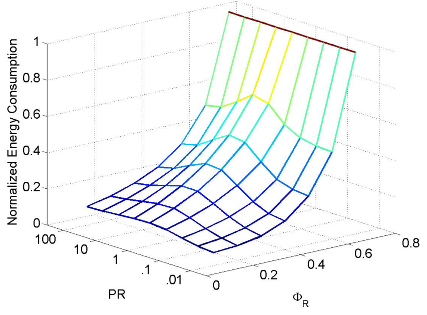
The experiments shown in Figure 7 present the relative performance when varying the maximum radio utilization for power ratio of $10^{-\frac{1}{2}}$. In Figure 7a, while substantially improving performance compared to *DVS-only* (maximum of 70%), the *Joint* scheme approaches *DMS-only* as radio utilization increases. It outperforms *DMS-only* at low radio utilization values within a range of up to 35%. As can be seen, these three schemes 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. Further, high-end systems that are computationally capable of executing *Dynamic* can receive an extra 34% improvement over the *Joint* approach.

The importance of relative computation and communication workload in assigning the slack time is presented in Figure 7b. Here, *DVS-only* starts off with better performance than *DMS-only* but degrades as soon as the radio workload increases.

Figures 6 and 7 show that as total utilization increases and the potential slack time decreases, *Joint* performs more conservatively. The next experiments examine the effect of changing power ratio and either maximum CPU utilization or maximum radio utilization on energy saving. Figure 8 shows that *Joint* is affected more by the growth of maximum utilization of a task component when it has a smaller share in power ratio.



(a) $\Phi_R = 0.2$



(b) $\Phi_C = 0.2$

Fig. 8: Impact of power ratio and workload utilization on *Joint*

VI. 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. We presented a joint DVS-DMS control algorithm that minimizes overall expected energy consumption. 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. This work strongly suggests the desirability of using combined DVS and DMS control algorithms in embedded wireless systems.

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*, 2005.

[2] D. Brunelli, C. Moser, L. Thiele, and L. Benini, "Design of a solar-harvesting circuit for batteryless embedded systems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 11, pp. 2519–2528, 2009.

[3] M. R. Jongerden, A. Mereacre, H. C. Bohnenkamp, B. R. Haverkort, and J.-P. Katoen, "Computing optimal schedules of battery usage in embedded systems," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, pp. 276–286, 2010.

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

[5] C. Schurgers, V. Raghunathan, and M. B. Srivastava, “Power management for energy-aware communication systems,” *ACM Transactions on Embedded Computing Systems*, vol. 2, no. 3, pp. 431–447, 2003.

[6] “IEEE Standard for Local and metropolitan area networks.” http://standards.ieee.org/findstds/standard/802.15.4g-2012.html.

[7] B. Zhang, R. Simon, and H. Aydin, “Harvesting-aware energy management for time-critical wireless sensor networks with joint voltage and modulation scaling,” *IEEE Trans. on Industrial Informatics*, vol. 9, no. 1, pp. 514–526, 2013.

[8] T. Hamachiyo, Y. Yokota, and 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*, 2010.

[9] W. Yuan and K. Nahrstedt, “Energy-efficient soft real-time cpu scheduling for mobile multimedia systems,” in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, 2003.

[10] R. Xu, D. Mossé, and R. Melhem, “Minimizing expected energy consumption in real-time systems through dynamic voltage scaling,” *ACM Transactions on Computer Systems*, vol. 25, no. 4, 2007.

[11] D. Zhu, H. Aydin, and J. J. Chen, “Optimistic reliability aware energy management for real-time tasks with probabilistic execution times,” in *Proc. of the IEEE Real-Time Systems Symposium*, 2008.

[12] P. Pillai and K. Shin, “Real-time dynamic voltage scaling for low-power embedded operating systems,” in *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, 2001.

[13] “3rd Generation Intel XScale Microarchitecture.” http://download.intel.com/design/intelxscale/31505801.pdf.

[14] “AMD Powernow Technology.” http://www.amd-k6.com/wp-content/uploads/2012/07/24404a.pdf.

[15] K. Craig, Y. Shakhsheer, and B. Calhoun, “Optimal power switch design for dynamic voltage scaling from high performance to subthreshold operation,” in *Proceedings of the 2012 ACM/IEEE International Symposium on Low-Power Electronics and Design*, ISLPED, 2012.

[16] Y. Lee *et al.*, “Power-tracking embedded buck x2013; boost converter with fast dynamic voltage scaling for the soc system,” *IEEE Transactions on Power Electronics*, vol. 27, no. 3, pp. 1271–1282, 2012.

[17] J. Park, D. Shin, N. Chang, and M. Pedram, “Accurate modeling and calculation of delay and energy overheads of dynamic voltage scaling in modern high-performance microprocessors,” in *ACM/IEEE International Symposium on Low-Power Electronics and Design*, 2010.

[18] Z. Cao, B. Foo, L. He, and M. Van Der Schaar, “Optimality and improvement of dynamic voltage scaling algorithms for multimedia applications,” *IEEE Transactions on Circuits and Systems*, vol. 57, no. 3, pp. 681–690, 2010.

[19] D. Feitelson, *Workload Modeling for Computer Systems Performance Evaluation*. 0.34 ed., 2011.

[20] M. Yang, Y. Wen, J. Cai, and C. H. Foh, “Energy minimization via dynamic voltage scaling for real-time video encoding on mobile devices,” in *IEEE International Conference on Communications*, 2012.

[21] J. Lorch and A. Smith, “Pace: A new approach to dynamic voltage scaling,” *IEEE Trans. on Computers*, vol. 53, no. 7, pp. 856–869, 2004.

[22] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, “Energy-aware wireless microsensor networks,” *Signal Processing Magazine, IEEE*, vol. 19, no. 2, pp. 40–50, 2002.

[23] G. Sudha Anil Kumar, G. Manimaran, and Z. Wang, “Energy-aware scheduling of real-time tasks in wireless networked embedded systems,” in *Proceedings of the 28th IEEE International Real-Time Systems Symposium*, 2007.

[24] B. Fateh and G. Manimaran, “Energy-aware joint scheduling of tasks and messages in wireless sensor networks,” in *IPDPS Workshops*, 2010.

[25] J. Yi, C. Poellabauer, X. S. Hu, J. Simmer, and L. Zhang, “Energy-conscious co-scheduling of tasks and packets in wireless real-time environments,” in *Proc. of the IEEE Real-Time and Embedded Technology and Applications Symposium*, 2009.

[26] V. Devadas and H. Aydin, “On the interplay of voltage/frequency scaling and device power management for frame-based real-time embedded applications,” *IEEE Transactions on Computers*, vol. 61, no. 1, pp. 31–44, 2012.

[27] L. Santinelli *et al.*, “Energy-aware packet and task co-scheduling for embedded systems,” in *Proceedings of the 10th ACM International Conference on Embedded Software*, 2010.

[28] B. Zhao and H. Aydin, “Minimizing expected energy consumption through optimal integration of dvs and dpm,” in *Proceedings of the ACM Conference on Computer-Aided Design*, 2009.

[29] R. Marculescu and P. Bogdan, “Cyberphysical systems: Workload modeling and design optimization,” *Design Test of Computers, IEEE*, vol. 28, pp. 78–87, July 2011.

[30] Y. Yu, B. Krishnamachari, and V. Prasanna, “Energy-latency tradeoffs for data gathering in wireless sensor networks,” in *INFOCOM 2004. The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, 2004.

[31] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez, “Optimal reward-based scheduling for periodic real-time tasks,” *IEEE Trans. on Computers*, vol. 50, no. 2, pp. 111–130, 2001.

Appendix

In this appendix, we provide the details of the solution to the optimization problem presented in Section IV. 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.

Problem DVMS-D: Case of the Deadline Constraint

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

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

Subject to

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

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

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

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

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

$$\frac{\delta L(t_i^{cm}, t_j^{cp}, \lambda)}{\delta t_j^{cp}} = \Gamma_j^{cp} C_l (1-\alpha)(t_j^{cp})^{-\alpha} + \lambda$$

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:

$$w_i^{cm}(t_i^{cm}) = -\Gamma_i^{cm} R_s \left[C_s \cdot \chi(t_i^{cm}) + C_e + C_s \chi'(t_i^{cm}) t_i^{cm}\right] \quad (6)$$
$$w_j^{cp}(t_j^{cp}) = -\Gamma_j^{cp} C_l (1-\alpha)(t_j^{cp})^{-\alpha} \quad (7)$$

For succint 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:

$$(t_j^{cp})^* = \left(\frac{\lambda^*}{(\alpha-1)\Gamma_j^{cp} C_l}\right)^{-1/\alpha}$$

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

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

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

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 $\frac{e^b}{b}$ 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 $O(M+W)$.

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*:

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

Subject to

$$\sum_{i=1}^{M} t_i^{cm} + \sum_{j=1}^{W} t_j^{cp} \le D$$
$$t_j^{cp} \ge t_{min}^{cp}$$
$$t_i^{cm} \ge t_{min}^{cm}$$

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 [31]. As shown in [31] by manipulating the Karush-Kuhn-Tucker conditions, in that case, the optimal value of the variable which gives the minimum marginal return (according to equations (6) and (7)) at the corresponding lower bound is equal to that lower bound.

This property suggests an iterative solution [31] 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 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 [31], 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 $O((M+W)\log(M+W))$.

Problem DVMS: Combining all the constraints

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

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

Subject to

$$\sum_{i=1}^{M} t_i^{cm} + \sum_{j=1}^{W} t_j^{cp} \le D$$
$$t_{min}^{cp} \le t_j^{cp} \le t_{max}^{cp}$$
$$t_{min}^{cm} \le t_i^{cm} \le t_{max'}^{cm}$$

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 [31]. 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 [31], one can demonstrate that the variable that has the maximum marginal return value (equations (6) and (7)) 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 $O((M+W)^2\log(M+W))$.