Proceedings of the
47th IEEE Conference on Decision and Control
Cancun, Mexico, Dec. 9-11, 2008

Optimal Dynamic Sleep Time Control in Wireless Sensor Networks

Xu Ning and **Christos G. Cassandras**
Center for Information and Systems Engineering
Boston University, Brookline, MA 02446
nx@bu.edu, cgc@bu.edu

Abstract— **We present a dynamic optimization approach to save energy in Wireless Sensor Networks (WSN) at the link level. One of the main sources of energy waste in a WSN is idle listening, i.e., nodes consuming energy to sample an idle channel. Our approach utilizes known traffic statistics and optimally controls the sleep interval between consecutive wake-ups of the receiver so that the expected total energy spent during each transmission is minimized. We derive necessary conditions for optimality that can be numerically solved but may fail to provide the optimal solution. Thus, we also propose an alternative efficient algorithm providing an explicit discretized solution. Simulation results are included to compare fixed sleep times to our dynamic control policy.**

I. Introduction

A Wireless Sensor Network (WSN) is a spatially distributed wireless network consisting of low-cost autonomous nodes which are mainly battery powered and have sensing and wireless communication capabilities [1]. Usually, nodes in such a network share a common objective, such as environmental monitoring or event detection.

Energy in WSN nodes is consumed by the CPU, by sensors, and by radio, with the latter consuming the most [2]. In order to optimize energy usage, it is important to identify the major sources of waste in communication [3]: collision, overhearing, control packet overhead and idle listening. Among these four sources, idle listening (i.e., the receiver staying awake during idle time anticipating an incoming transmission) accounts for the most significant waste of energy in a WSN.

Energy waste due to idle listening can be reduced by adopting a Medium Access Control (MAC) scheme. MAC can be categorized into *scheduled* and *unscheduled* schemes. Scheduled schemes, such as S-MAC [3], maintain a schedule among a small cluster of nodes such that nodes have coordinated transmission. Therefore, nodes can turn off their radio according to the schedule. Unscheduled schemes, on the other hand, try to emulate an "always-on" receiver by introducing additional ad-hoc synchronization. One way to achieve this is to use Low-Power Listening (LPL) [4].

LPL is an approach to save energy at the link level, which consists of a sender and a receiver. The main steps of LPL are:

1) The receiver remains at a sleep state most of the time, and occasionally wakes up to sample the channel in order to determine whether it is busy or idle.
2) When the sender wants to send a message, it begins with an attached signal called the "preamble" to the receiver. The preamble can be viewed as a "wake up signal". After the preamble signal, the sender sends the message.
3) When the receiver wakes up and samples the channel, either of two cases may occur: (i) If the channel is idle, the receiver sets the next wake-up time and sleeps again, (ii) If the channel is busy (preamble detected), the receiver stays on until the message is received. After transmission, the receiver also sets its next wake-up time and sleeps again.

One obvious advantage of unscheduled MAC is its universality, since all transmission controls are transparent to the applications to which it just appears to be an always-on radio. Another advantage is that it does not need advance synchronization (it can, though, benefit from it since a preamble can be shortened when the transmission pair is roughly synchronized). However, many MAC schemes that adopt LPL only use periodic sleep time control for its simplicity. Depending on the tasks they perform, WSNs can be differentiated in terms of continuous monitoring or event-driven operation; in the latter, network activities are triggered by external random events. Since event times are not deterministic, many wake-ups actually take place during times when an event is *not likely* to happen. Using variable preamble LPL, due to its handshaking action, it is not required to have a periodic sleep time control at all, which enables us to control the sleep time carefully and save more energy.

In previous work [4] we proposed a way to utilize network statistical information in aperiodic sleep time control. It was shown that if statistical information about event times is known, we can control the sleep time of the receiver so that it samples the channel more frequently when an event is more likely to happen, and less frequently when it is not. It was also shown that this dynamic sleep time control *dominates* fixed sleep time control in terms of performance expressed as a trade-off between energy and latency. However, we did not attempt to provide an *optimal* control minimizing energy. Instead, we provided a way to calculate the best sleep time given a certain delay constraint, which in turn specifies the energy consumption of the sender. If one seeks to optimize over energy consumption of the link, it is necessary to tune

The authors' work is supported in part by NSF under grants DMI-0330171 and EFRI-0735974, by AFOSR under grants FA9550-04-1-0133 and FA9550-04-1-0208, and by DOE under grant DE-FG52-06NA27490.

the delay constraint in order to achieve optimality.

In this paper, our contribution lies in reformulating sleep time control as a dynamic optimization problem whose objective is to minimize the link energy, i.e., the energy to transmit and receive a message. The model, similar to the one used in [4], incorporates statistical information related to all random events. The optimization problem is solved using Dynamic Programming (DP) to produce an optimal sleep time control policy, in the sense that the energy spent in each transmission is minimized. Exploiting the structure of the problem, we are able to derive an explicit DP algorithm which is of low complexity and hence compute the optimal control policy.

The paper is organized as follows. In Section II, we describe the problem of interest, and establish the DP formulation. Section III describes the special case when inter-event times are exponentially distributed. In Section IV, we derive necessary conditions of optimality, and transform the optimization problem into a pair of differential equations with known initial conditions. In Section V, we discretize the state space and provide an $O(M^2)$ algorithm (M is the cardinality of the discretized state space) to solve the DP problem and obtain an optimal control policy. Simulation results are provided in Section VI and conclusions are given in Section VII.

II. Problem Description and Analysis

We consider a sender-receiver link in a WSN. The sender is a wireless node equipped with some event detector which is driven by external, random events such as body movements in a room or fire alarms. When the sender detects an event or reports its status, it sends a message to the receiver, which is a downstream node in the network. In this link, variable preamble LPL is used. Depending on the specific application, LPL can have different implementations:

1) **"Plain-vanilla"** LPL. This is the simplest LPL where both the receiver's sleep period and the sender's preamble length are fixed. Usually the preamble length is a little bit longer than the sleep period to ensure that the preamble is "picked up" by one of the receiver's channel pollings. B-MAC [5] uses this type of LPL.
2) "**Short preamble**" LPL. In this version, the receiver's sleep period remain fixed. However, there exists some degree of clock synchronization between the sender and the receiver so the sender can estimate within some short interval when the receiver will wake up. Hence, the sender can send a shorter preamble as long as it is long enough to cover the estimated interval. This version is used in WiseMAC [6].
3) "**Variable preamble**" LPL. Although the short preamble version also varies the length of the preamble, it requires clock synchronization between the two peers. In variable preamble LPL, there is no need for such synchronization. Instead, handshaking is used. As shown in Fig. 1, the sender sends "strobed preambles" – intermittent transmission of a preamble signal with gaps for listening. When the receiver wakes up and picks up a preamble signal, it replies to the sender so a handshake is formed. Then, the sender knows for sure the receiver is awake in receiving mode, so the message will be transmitted. X-MAC [7] is based on variable preamble LPL.

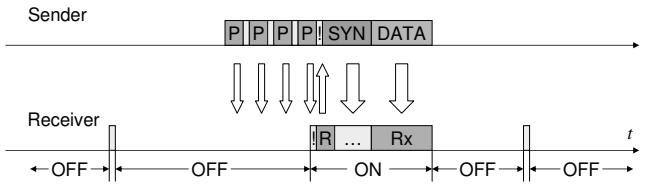

Fig. 1. Low-power listening (LPL) with "strobed preamble". After each preamble packet P sent, the sender listens for a brief period for acknowledgement from the receiver, then send P again if no acknowledgement is received.

The central problem in LPL consists of determining how often the receiver should wake up and sample the channel. Clearly this is a trade-off between the sender and the receiver. On one hand, as the receiver wakes up more often, energy depletes faster because each wake up needs to turn on and off the radio, where a start-up energy cost is incurred. Denote this energy cost by c. On the other hand, if the sleep time between two receiver sampling events is long, the sender will obviously need to send a longer preamble so that it can be picked up by the receiver in its next wake up. Sending a preamble also incurs an energy cost which is proportional to the length (time wise) of the preamble. For simplicity, let the energy cost per unit time of the preamble be 1. Hence, the problem is to determine how the receiver should control its sleep time so that the total energy spent in each message transmission is minimized.

To formulate an optimization problem, we start by describing our model. First, since each channel sampling activity at the receiver is short in time duration, we assume these to be points in the timeline. Second, although in the variable preamble LPL strobed preamble packets are used, we assume that the preamble is a continuous signal whose length is a real positive number. Third, we focus on the preamble and the channel sampling activities, and ignore the energy cost during the transmission of the data payload part of the message since it is not controllable in the scope of our optimization problem. A typical sample path is shown in Fig. 2.

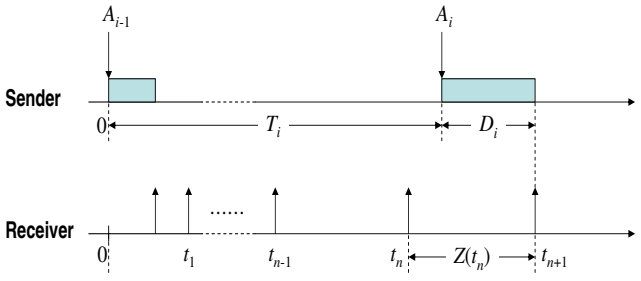

Fig. 2. A typical sample path

We define the "ith event" to be the instant when the transmission of the ith message's preamble starts. In Fig. 2, A_i is the occurrence time of the ith event and $T_i = A_i - A_{i-1}$ is the ith inter-event time. We assume T_i is a random variable with c.d.f. $F(\cdot)$ and p.d.f. $f(t) = F'(t)$. Let t_n be the nth wake-up time of the receiver with respect to time 0, and $Z(t_n) = t_{n+1} - t_n$ is the controllable sleep time to be determined at time t_k. D_i is the preamble time of message i, which is also random. Based on the problem setup and due to the renewal property, we can see that there is no coupling between different events. This allows us to focus on only one event at time A_i and derive an optimal control policy which applies to any other event as well. The optimization problem is formulated as follows:

$$\begin{aligned} \min_{n,t_1,t_2,\ldots,t_n} & E\{nc + D_i\} \\ s.t. \quad & n = \arg\min_k t_k > T_i \\ & D_i = t_n - T_i \end{aligned} \tag{1}$$

This is an atypical optimization problem, partly because the number of sampling events n is a control variable. To illustrate how this works, suppose we are currently at time t_n in Fig. 2. We have to decide the sleep time $Z(t_n)$. After the receiver sleeps and then wakes up at time t_{n+1}, two cases can arise. First, if $T_i < t_{n+1} = t_n + Z(t_n)$, the receiver will pick up the preamble signal. In this case, the cost is

$$G_n = c + D_i = c + t_n + Z(t_n) - T_i$$

In the second case, $T_i > t_{n+1}$. Then, at time t_{n+1}, event A_i has not occurred yet, so the receiver will not pick up the preamble. Hence, it will calculate the next sleep time $Z(t_{n+1})$ and repeat the process. The cost will be:

$$G_n = c + G_{n+1}$$

The probability that the first case arises is

$$P_1 = \Pr(T_i < t_{n+1} | T_i > t_n) = \frac{F(t_n + Z(t_n)) - F(t_n)}{1 - F(t_n)}$$

and the probability that the second case arises is simply $P_2 = 1 - P_1$. Then, the expected cost to receive message A_i, using the control $Z(t_n)$, is:

$$\begin{aligned} E[G_n(Z(t_n)) | T_i > t_n] = & (c + E[D_i | T_i > t_n]) P_1 \\ & + (c + E[G_{n+1}]) P_2 \end{aligned}$$

Now we can clearly see a recursive structure: the decision at time t_n depends on a possible future decision at time t_{n+1} where G_{n+1} depends on $Z(t_{n+1})$. A Bellman equation can be derived by defining $J(t)$ as the *minimum expected cost-to-go* to receive message A_i, that is, the expected energy spent to receive A_i given the fact that A_i has not occurred up to t, the time of any sampling event in Fig. 2 which does not detect the preamble (e.g., t_{n-1} or t_n). $J(t)$ can be expressed as:

$$J(t) = \min_{z>0} \{c + A^0(t,z) P_1^0(t,z) + J(t+z) P_2^0(t,z)\} \tag{2}$$

where z is the sleep time control at time t and

$$\begin{aligned} A^0(t,z) &= E[t + z - T_i | t \le T_i \le t + z] \\ P_1^0(t,z) &= \frac{F(t+z) - F(t)}{\bar{F}(t)} = 1 - P_2^0(t,z) \\ \bar{F}(\tau) &= 1 - F(\tau) \end{aligned}$$

We can see that the system state is represented by t and the minimum expected cost to receive message A_i contains three parts: (i) a fixed cost c to wake up at time $t+z$; (ii) the expected preamble cost $A^0(t,z)$ if A_i occurs during the sleep time; (iii) the future minimum expected cost $J(t+z)$ to receive A_i if A_i does not occur during the sleep time. Here we assume that random variable T_i has support region $(0, +\infty)$. For a distribution with a limited support region $0 < T_i \le T_{\max}$, the boundary condition of (2) is $J(T_{\max}) = c$ and in this case, the allowable control set is limited to $0 < z \le T_{\max} - t$.

By solving the Bellman equation [8] (2) for all possible $t > 0$, one can obtain the *optimal* sleep time control policy $z(t)$.

III. Characterizing the Optimal Solution

In this section we try to characterize the optimal solution to (2). For notational simplicity, define $u = t + z$, which is the wake-up time epoch of the receiver determined at time t. We rewrite the Bellman equation (2) as:

$$J(t) = \min_{u>t} \{c + A(t,u) P_1(t,u) + J(u) P_2 A(t,u)\} \tag{3}$$

where

$$\begin{aligned} A(t,u) &= A^0(t, u-t) \\ P_1(t,u) &= P_1^0(t, u-t) \\ P_2(t,u) &= P_2^0(t, u-t) \end{aligned}$$

Define

$$V(t,u) = c + A(t,u) P_1(t,u) + J(u) P_2(t,u) \tag{4}$$

Let the optimal policy be $u^*(t)$ which is a function of t. Because $u^*(t)$ is the solution to (3), we have:

$$J(t) = \min_{t < u(t) \le T_{\max}} V(t, u(t)) = V(t, u^*(t)) \tag{5}$$

We assume that $F(\tau)$ has a finite support $0 \le \tau \le T_{\max}$, and $f(\tau)$ has a continuous derivative. Since $t < u^*(t) \le T_{\max}$ for any t, either $u^*(t) = T_{\max}$ or

$$\left. \frac{\partial V(t,u)}{\partial u} \right|_{u=u^*(t)} = 0 \tag{6}$$

that is,

$$F(u^*(t)) + \left. \frac{\partial [J(u) P_2(t,u)]}{\partial u} \right|_{u=u^*(t)} = F(t)$$

Differentiating with respect to t on both sides, we obtain:

$$\frac{du^*(t)}{dt} = \begin{pmatrix} f(u^*(t)) + J''(u^*(t)) \bar{F}(u^*(t)) \\ -2J'(u^*(t)) f(u^*(t)) \\ -J(u^*(t)) f'(u^*(t)) \end{pmatrix}^{-1} f(t) \tag{7}$$

For $J(t)$, we have:

$$\begin{aligned}\frac{dJ(t)}{dt} &= \frac{dV(t,u^*(t))}{dt} \\ &= \frac{\partial V(t,u^*(t))}{\partial t} + \left(\left.\frac{\partial V(t,u)}{\partial u}\right|_{u=u^*(t)}\right)\frac{du^*(t)}{dt} \\ &= \frac{\partial V(t,u^*(t))}{\partial t} \\ &= \frac{(t-u^*(t)+J(t)-c)\,f(t)}{\bar{F}(t)} \end{aligned} \quad (8)$$

Differential equations (7) and (8) govern the optimal solution, namely, $J(t)$ and $u^*(t)$. With (7) and (8) we can integrate backwards to find the optimal solution, given appropriate initial conditions. Notice that for $t \to T_{\max}$, we have $u^*(t) = T_{\max}$. Define t_0 such that for $t_0 \le t < T_{\max}$, $u^*(t) = T_{\max}$ and let $I_0 = (t_0, T_{\max}]$. So in I_0,

$$u^*(t) = T_{\max} \quad (9)$$

$$J(t) = V(t, T_{\max}) = c + A(t, T_{\max}) \quad (10)$$

To find t_0, because in $u^*(t) = T_{\max}$ for $t \in I_0$, we have:

$$\left.\frac{\partial V(t,u)}{\partial t}\right|_{u=T_{\max}} < 0$$

that is:

$$F(T_{\max}) + \left.\frac{\partial\left[J(u)\,P_2(t,u)\right]}{\partial u}\right|_{u=T_{\max}} < F(t)$$

so

$$\begin{aligned} 1 - F(t) - cf(T_{\max}) &< 0 \\ t_0 &= F^{-1}(1 - cf(T_{\max})) \end{aligned} \quad (11)$$

I_0 is the interval where equation (6) does not hold, but provides the initial condition needed to integrate (7) and (8) backwards. Note that in (7), the calculation of $du^*(t)/dt$ involves $J(u^*(t))$ and its derivatives. Because $u^*(t) > t$ and we are integrating backwards, all these quantities are already known when we compute $du^*(t)/dt$.

In summary, we have obtained two differential equations (7) and (8) which are necessary conditions for the locally optimal control policy $u^*(t)$ and the associated cost-to-go function $J(t)$. Using initial conditions (9)-(11) we can solve (7) and (8) to find $u^*(t)$ (subject to constraints $t < u^*(t) \le T_{\max}$) and $J(t)$, which are locally optimal solutions. For intervals other than I_0 that $u^*(t)$ happens to be $T_{\max}$ resulting from (7), we calculate the corresponding $J(t)$ with (10). Since $z^*(t) = u^*(t) - t$, we can easily obtain the sleep time control policy at time t. Letting $M = T_{\max}/h$ and h being the step size, solving the differential equations is generally an $O(M)$ task, which is very efficient. However, the drawback is that because (6) is only a necessary condition for the optimality of $u^*(t)$, it may not be the global optimal solution. Numerical stability is another concern as we will see in the numerical example section. This motivates an alternative approach that leads to a globally optimal solution at the expense of increased computational cost.

IV. Discrete Time Model and Dynamic Programming Algorithm

To find the global optimal $u^*(t)$, in what follows we define the DP algorithm which solve the Bellman equation (2) exhaustively. One way to deal with this problem is to discretize the state space. Because the state t is a scalar, the exhaustive algorithm's complexity only depends on the discretization resolution.

Suppose the support region of T is $[0, T_{\max}]$ which is *finite*. We discretize the support region into M time slots and let $h = T_{\max}/M$, then, slot i represents interval $[ih, (i+1)h)$, $i \in \{0, 1, ..., M-1\}$. For any given time t, the corresponding slot i is such that $t \in [ih, (i+1)h)$. Therefore, the discretized state space is $\{0, 1, ..., M-1\}$. In this discrete time model, we only allow the receiver to wake up at time epochs $\{0, 1, ..., M\}$, so the decision space at state i is $\{1, ..., M-i\}$. A decision z at state i means the receiver will wake up at time $(i+z)h$.

For notational simplicity, let $t_i = ih$ and $u = (i+z)$, so that $u \in \{(i+1), ..., M\}$. Denoting by J_i the cost-to-go function at state i, we have

$$\begin{aligned} J_i &= \min_u V_{i,u} \\ &= \min_u \left\{c + A(t_i, uh)\,P_1(t_i, uh) + J_u P_2(t_i, uh)\right\} \\ &= \min_u \left\{ c + \frac{\int_{ih}^{uh}(uh - x)\,f(x)\,dx}{\bar{F}(ih)} + \frac{\bar{F}(uh)}{\bar{F}(ih)} J_u \right\} \\ & u \in \{(i+1), ..., M\} \end{aligned} \quad (12)$$

Now we can solve (12) numerically. The direct, exhaustive algorithm to solve (12) has complexity $O\left(M^3\right)$, because one has to compute $V_{i,u}$ for all possible i and u ,and to compute $V_i(u)$ for a single (i,u) pair is an $O(M)$ task due to the inner integral. However, the computation can be simplified noting that $V_{i,u}$ can be obtained easily from $V_{i,u+1}$ without computing the integral again. To see this, we first, define:

$$f_i = \int_{ih}^{(i+1)h} f(x)\,dx \quad (13)$$

$$\bar{F}_i = \bar{F}(ih) = \sum_{j=i}^{M-1} f_i \quad (14)$$

$$E_i = \int_{ih}^{(i+1)h} x f(x)\,dx \quad (15)$$

all of which can be calculated in advance of the sleep time control with knowledge of $f(x)$. Therefore,

$$V_{i,u} = c + \frac{uh\left(\bar{F}_i - \bar{F}_u\right) - \sum_{j=i}^{u-1} E_j}{\bar{F}_i} + \frac{\bar{F}_u}{\bar{F}_i} J_u \quad (16)$$

Because

$$V_{i,M} = c + \frac{Mh\bar{F}_i - \sum_{j=i}^{M-1} E_j}{\bar{F}_i} \quad (17)$$

$$V_{i,u} = V_{i,u+1} - \frac{1}{\bar{F}_i}\left[\begin{array}{c} uhf_u + h\bar{F}_i - \bar{F}_{u+1} - E_u \\ +\bar{F}_{u+1}J_{u+1} - \bar{F}_u J_u \end{array}\right] \quad (18)$$

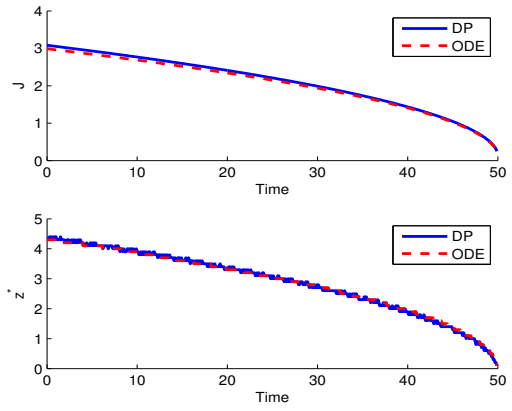

Fig. 3. Optimal sleeping time and cost-to-go function value with uniform interarrival time distribution.

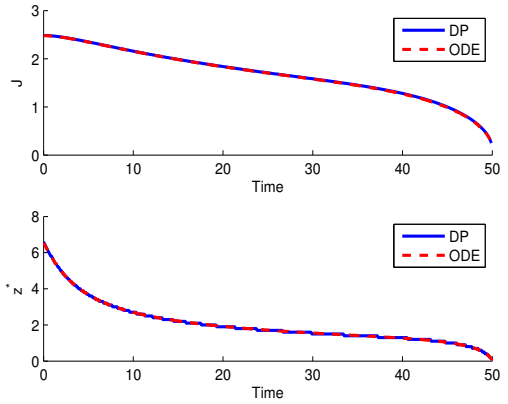

Fig. 4. Optimal sleep time and cost-to-go function value under Weibull distribution.

for a given state i, we can compute each $V_{i,u}$ for all $(i+1) \leq u \leq M$ using (17) and (18) in $O(1)$, providing that J_u is already computed for all u. Finally, for each i, we compute:

$$J_i = \min_{(i+1)\leq u\leq M} V_{i,j}$$
$$z_i^* = \min_{(i+1)\leq u\leq M} V_{i,j} - i$$

Obviously, the total complexity of the above algorithm is $O\left(M^2\right)$. The spatial complexity is actually $O(M)$ because for each i iteration, we can use the same memory space to store $V_{i,j}$. The resulting $\{z_i^*\}$ is the optimal sleep time at each state i.

V. Numerical Results

A. Distribution Examples

Uniform distribution. We consider a uniform distribution for the interarrival time $T \sim U[0, T_{\max}]$ We choose the parameters $c = 0.2$ and $T_{\max} = 50$ (seconds). This means that a channel sampling action costs as much energy as 200 milliseconds of transmission, which is a reasonable figure. Under this distribution, the optimal sleeping time can be calculated using the DP algorithm as well as solving the differential equation (ODE). We also choose the time slot size $h = 0.1$, so $M = 500$. h is also used as the step size in ODE. The result is shown in Figure 3.

We can see that as t increases, the sleep time $z^*(t)$ decreases gradually, and approaches 0 at $t \to U$. Recall that the sleep time at t is calculated based on the fact that "no event has occurred during the last t amount of time". Therefore, as t increases, the possibility that the event takes place in the near future increases as well, which leads to a shorter sleep time. Consider that ODE is a linear time algorithm, it is a better approach here. However, to prevent numerical instability in ODE, the step size h cannot be too large, which limits the lower bound of ODE's computation effort.

Wiebull distribution. The Weibull distribution is a class of distributions to model random events with an increasing hazard rate with respect to the age of the event. We choose parameters $\alpha = 20$ and $\beta = 2$ (the hazard rate is increasing linearly with time). We truncate the distribution at $T_{\max} = 50$ (seconds) to limit the support region. In practical systems this is a reasonable approach. Other parameters are the same: $c = 0.2$, $h = 0.1$, $M = 500$. The results obtained from DP and ODE are shown in Figure 4. For the ODE case, the step size $h = 0.1$ is too large and causes numerical instability so we reduced it to $h = 0.02$.

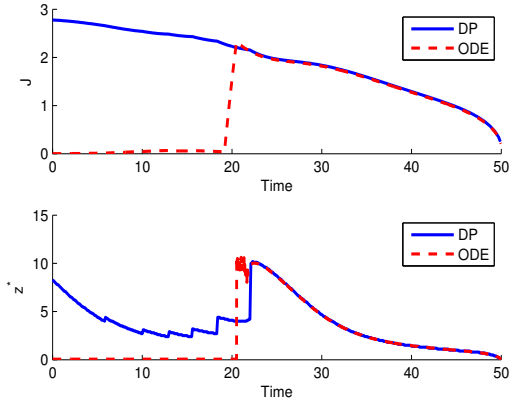

Fig. 5. Optimal sleep time and cost-to-go function value under bimodal Gaussian distribution.

Bimodal Gaussian distribution. A Bimodal Gaussian (BG) distribution is a distribution with two different modes and within each mode a Gaussian distribution. Our parameters are: $\mu_1 = 12.5, \mu_2 = 40, \sigma_1 = \sigma_2 = 5$ where μ_i and σ_i are the corresponding mean and standard deviation of the ith mode. The modal probability is 0.5. The optimal policy obtained is shown in Figure 5:

We first notice that there is a discontinuity in the optimal sleep time curve, generated by DP. From $t = 0$ to around $t = 15$, we see the sleep time is decreasing. However, because the probability of an event occurring in the interval of $[20, 30]$ is relatively small, the optimal sleep time is to

"skip over" the period in order to save energy. We also notice that ODE completely fails to handle the jump, which is expected. The explanation to the occurrence of such a jump is that there exist multiple local minima in $V(t,u)$, so (6) no longer provides a globally optimal solution.

B. Comparison of fixed and dynamic sleep time control policies

To show the advantage of optimal sleep time control, we compare it to the *best* fixed sleep time control policy using simulation. In the simulation, we generate a sample path of 10,000 events, and compare the *average energy spending per message* between the best fixed sleep time control and dynamic sleep time control. The optimal fixed sleep time is obtained through exhaustive search. The comparison results are shown in the following table:

Distrib.	Best Fixed	DP	Save%
Uniform	3.13	2.96	5.50
Weibull	2.67	2.40	10.21
BG-1	3.23	2.68	16.81
BG-2	3.23	2.02	37.55

In the comparison, each distribution takes the same parameters from the previous examples except for the bimodal Gaussian-2 where $\sigma_1 = \sigma_2 = 2.5$. We notice that for the uniform distribution, the energy saving is merely 5%. This is because for a uniform distribution, the tail $P(T < \tau | T > t)$ is still uniform, so not much value can be extracted from the fact that "the age of this renewal process is t". For other distributions, as the p.d.f. "deviates" more from uniform, more energy savings can be achieved. Because we are already comparing with the *best* fixed control, this comparison shows the superiority of dynamic sleep time control over fixed control. Finally, we emphasize that the algorithm itself does not depend on any particular distribution.

C. Computation time experiment

To demonstrate the feasibility of our algorithm, we implemented the computation on a Tmote wireless sensor, manufactured by Sentilla, Inc. The Tmote is equipped with a 8MHz MSP430 16-bit microcontroller, manufactured by Texas Instruments, Inc. We measured the CPU time used to compute the optimal sleep time policy using the ODE and DP which includes equations (14)-(18) (note: f_i's are given or estimated through a separate process). The experiment result of CPU time (in seconds) versus M is shows in the following table:

M	50	100	200	300
ODE	1.5	3	6	9
DP	3	11	45	97

Due to the limitation of onboard memory (10KBytes), we were unable to experiment with larger M. However, the final case of $M = 300$ is more than enough: if the inter-event time does not exceed 60 seconds (typical in WSN because of heartbeat/health/routing messages), this produces a resolution of 0.2 seconds. We believe that a value in the vicinity of $M = 100$ offers a good trade-off between performance and resource usage. In the $M = 300$ case, the CPU time is only 97 seconds which is manageable. Moreover, this computation is performed only once to generate the optimal sleep time policy, unless the inter-event time distribution is changed.

We also notice that ODE is much faster than DP thanks to its linear time growth. However, as is mentioned before, ODE has numerical stability problems and cannot handle cases like the BG distributions.

VI. Conclusions

We have formulated a dynamic optimization problem for saving energy in a transmission control scheme based on a variable preamble technique. We first managed to solve the continuous time Bellman equation, resulting in a pair of differential equations characterizing the optimal solution. Since the differential equations, although efficient to solve, provide only necessary optimality conditions and do not always lead to global optimal solutions, we discretize the system and provided the DP algorithm which is easy to implement and low in complexity. Our numerical results show that our approach fully utilizes statistical information in controlling the sleep time of a wireless sensor node, resulting in substantial energy savings in comparison to the best possible fixed sleep time control. Ongoing work aims at extending this approach to the network level, where the main obstacle is to reformulate the optimization problem subject to the fact that only partial information is available. Another extension is to combine DP and ODE in a hierarchical way such that M can be reduced while retaining a fine grain of control via solving ODE as a fast subproblem.

References

[1] S. Megerian and M. Potkonjak, *Wireless Sensor Networks*, ser. Wiley Encyclopedia of Telecommunications. New York, NY: Wiley-Interscience, January 2003.

[2] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2004, pp. 188–200.

[3] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, June 2004.

[4] X. Ning and C. G. Cassandras, "Dynamic sleep time control in event-driven wireless sensor networks," in *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, CA, USA, December 13-15 2006, pp. 2722–2727.

[5] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2004, pp. 95–107.

[6] A. El-Hoiydi and J.-D. Decotignie, "WiseMAC: An ultra low power mac protocol for multi-hop wireless sensor networks," *Lecture Notes in Computer Science*, vol. 3121, pp. 18–31, Jan 2004, springer-Verlag GmbH.

[7] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks," in *SenSys '06: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, New York, NY, USA, November 1-3 2006, pp. 307–320.

[8] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. MA, USA: Kluver Academic Publishers, Janurary 2007, vol. 1,2.