Duty Cycle Control for Low-Power-Listening MAC Protocols

Christophe J. Merlin and Wendi B. Heinzelman
Department of Electrical and Computer Engineering,
University of Rochester, Rochester NY
{merlin, wheinzel}@ece.rochester.edu

Abstract

Energy efficiency is of the utmost importance in wireless sensor networks. The family of low-power-listening MAC protocols was proposed to reduce one form of energy dissipation—idle listening, a radio state for which the energy consumption cannot be neglected. Low-power-listening (also called channel probing) MAC protocols are characterized by a duty cycle: a node probes the channel every t_i s of sleep. A low duty cycle favors receiving nodes because they may sleep for longer periods of time, but at the same time, contention may increase locally, thereby reducing the number of packets that can be sent. We propose a new approach to dynamically control the duty cycle so that the target rate of transmitted packets is reached, while the consumed energy is minimized. Our approach utilizes control theory and adapts it to the control of t_i for low-power-listening MAC protocols in wireless sensor networks. Results show that this approach can appropriately adjust t_i to the current network conditions.

1 Introduction

Today more than ever, sensor network applications require individual nodes to lower their energy consumption in order to support an application for longer periods of time. Every layer in the protocol stack must reduce its own energy dissipation. *Low-power-listening* (LPL) protocols form a family of MAC protocols that drastically reduce idle listening, a state of the node when its radio is turned on and in receive mode, but not receiving any packets.

In a LPL protocol, nodes probe the channel every t_i s, and if they do not receive any data during this probe, they return to sleep for another t_i s. Aloha with preamble sampling (PS) [3], WiseMAC [4], and B-MAC [10] were among the first LPL protocols to be proposed. All these protocols send data packets with very long preambles so as to ensure that the intended receiver will stay on upon probing the medium. However, the protocols are not adapted to recent radios like

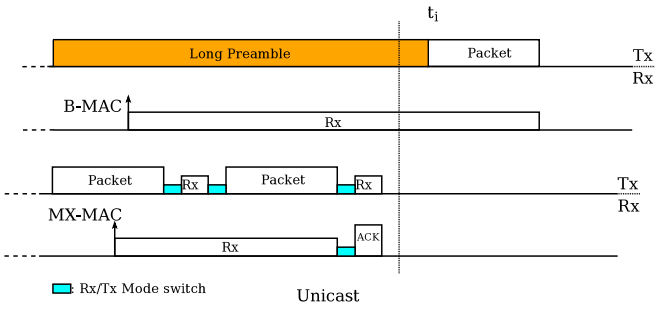


Figure 1. Schedule for B-MAC and MX-MAC.

the IEEE 802.15.4 [6] compliant Chipcon CC2420 [2] radio. Consequently, researchers introduced new compatible LPL protocols such as X-MAC [1], SpeckMac-D [12], and MX-MAC [9]. These protocols are based on repeating either the data packet itself or an advertisement packet, in place of long preambles. The details of the transmission schedules (hereafter "MAC schedule") of B-MAC and MX-MAC are given in Figure 1.

Previous work [9] has shown that longer t_i values favor receiving nodes, because longer t_i values lower a node's duty cycle while switching to *Receive* mode for the same period of time within the duty cycle. On the other hand, nodes that are mostly sending can greatly reduce their energy consumption if the t_i value is low: they can stay in *Sending* mode for shorter periods of time. Consequently, there is a trade-off between the nodes at the two ends of a unidirectional wireless link. In addition, lower duty cycles often cause contention in areas of the network experiencing higher rates of packet transmissions. As Figure 1 shows, only one data packet can be transmitted per cycle, which can cause a node to miss the target rate m^* of packet transmissions.

In [7], Jurdak et al. convincingly argue that a fixed t_i value does not fit WSN deployments where the node locations and traffic patterns are not uniform over the network. Because a fixed t_i value is decided *a-priori*, it would have to be set conservatively to accommodate areas in the net-

978-1-4244-2575-4/08/$20.00

work where traffic is expected to be heavy, thus forcing idle subregions to waste energy.

In this paper, we borrow from control theory to propose a new approach to dynamically adjust the duty cycle based on a small set of parameters. The goal of our method is to minimize the energy consumed by the node with the lowest remaining energy (or the node which the application deems most important), referred to as node $\mathcal{N}$, while exchanging a target number of packets. If $\mathcal{N}$ is mostly sending, lowering t_i (increasing the duty cycle) will have no adverse effect on the target rate m^* of successfully sent packets, and it will reduce the energy dissipation for $\mathcal{N}$, so there is no need for t_i control. However, when $\mathcal{N}$ is mostly receiving, lowering the duty cycle (increasing t_i), while reducing the energy dissipation for $\mathcal{N}$, will cause packets to be dropped. This is the conflict that we propose to arbitrate. Our method can also be extended to control the energy consumed by both the sending and receiving nodes on a wireless link. More generally, we provide a methodological framework that can be applied to control other aspects of the network as well.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 introduces theoretical foundations and expands on them to adapt to our specific problem of t_i control for channel-probing MAC protocols. Section 4 presents simulation results using our adaptive t_i values, and Section 5 concludes this paper.

2 Related Work

Jurdak et al. [7] introduced the idea of adaptive duty cycles in LPL protocols. Because a protocol designer must account for busy regions of the network, a fixed t_i value would have to be set conservatively. Consequently, many parts of the network would waste energy by running at an unnecessarily high duty cycle. Adaptive Low Power Listening, or ALPL, allows areas of the network to run at a lower duty cycle. After forming their routing tree, each individual node can evaluate the number of packets they will transmit per second based on the expected number of packets they will originate and that of their descendant nodes. Contrary to ALPL, our approach does not use a heuristic and adapts the duty cycle to meet the target rate of packets.

The idea of using control theory in sensor networks is not a new one, especially because wireless sensor and actuator networks require such solutions. in our unique approach we optimize the duty cycle for both energy use and packet transmissions, which cannot be easily modeled.

In [11], Vigorito et al. use control theory to adapt the duty cycle of nodes capable of harvesting energy. Maintaining a sufficient power supply level is a non-trivial problem because of changing environmental patterns such as the weather. The authors introduce a model-free approach to adapt the duty cycle in dynamic conditions. Although they set out to control only one parameter in the system (the energy supply level), which constitutes a marked difference from our goals, much of their underlying theoretical foundations are similar to those in the first part of our work.

3 Estimation and Control for Multi-variable Systems

Because low duty cycle schemes tend to create contention and delays, a node wishing to send m^* packets may not be able to do so in a timely manner. Let us consider a one-hop network with various flows among neighbors. Node A wants to send m^* packets to node B, where node B is designated as node $\mathcal{N}$, a critical node for the application, or one with very low remaining energy. Unfortunately, the medium is sometimes occupied by other transmissions. If node A only gets to send $m < m^*$ packets, it may elect to increase its duty cycle. When the duty cycle is larger than its optimal value, node $\mathcal{N}$ wastes precious energy, and may wish to scale back its duty cycle (t_i increased). The control of the duty cycle to send m^* packets is the subject of the first part of this section. We use $t_i(t)$ to designate the time-varying nature of t_i.

We provide more extensive theoretical background at *http://www.ece.rochester.edu/∼merlin/DutyCycleControl/DutyCycleControlURTR.pdf*.

3.1 Background

We start by assuming that the system we wish to represent and control is mostly linear. For instance, the relationship between energy consumption and t_i is linear, as energy consumption grows linearly with the number of probes done per second. Likewise, the number of packets received is mostly linearly related to energy consumption.

The network ("plant") reacts to an input $u(t)$ by producing an output $y(t)$, which it tries to match to a reference $r(t)$. A controller modifies $u(t)$ so as to obtain the desired output $y^*(t)$. In order to do so, the process under control can be defined by its state $x(t)$. A deterministic noisy linear process can be represented in its discrete form as follows:

$$x(t+1) = Ax(t) + Bu(t) + Cw(t) + w(t+1) \quad (1)$$

where $x(t+1)$ designates the value of the system state at time $(k+1)\mathcal{T}$ and w is the noise. $\mathcal{T}$ represents the period between re-evaluations of the control $u(t)$.

For controlling $t_i(t)$, we can set $y(t)$ to $m(t)$ (the number of packets that are successfully sent at time t) and $u(t)$ to the $t_i(t)$ value at time t. The objective value $y^*(t+1)$ becomes $m^*(t+1)$, the desired number of packets to be transmitted at time $t+1$. Consideration must also be given to a second variable, the energy consumed ϵ, which is an incentive to

lower the duty cycle. We note $\epsilon^*(t+1)$ as a target energy consumption at $t+1$.

Because the fundamental characteristics of the system (A, B and C) and its state $x(t)$ cannot be *a-priori* known, the system's output must be estimated using an internal parameter θ and a history of $\{x(t)\}$ (or $\{y(t)\}$) and $\{u(t)\}$ values stored in ϕ.

3.2 The Adaptive Regulator

3.2.1 The Estimator

In [8], Kumar et al. propose an adaptive regulator for linear systems. The system control can be approached by estimating the system first, and using the system model to find the input value that minimizes the predicted output.

In this multi-variable case, we decided to estimate both the number of packets sent and the consumed energy separately. For m and ϵ, the ϕ and θ vectors are:

$$\phi_k^m = \begin{bmatrix} m_k & \dots & m_{k-p} & t_{ik} & \dots & t_{ik-p} \end{bmatrix}^T$$
$$\theta_k^m = \begin{bmatrix} a_0^m & \dots & a_{p-1}^m & b_0^m & \dots & b_{p-1}^m \end{bmatrix}^T$$
$$\phi_k^\epsilon = \begin{bmatrix} \epsilon_k & \dots & \epsilon_{k-p} & t_{ik} & \dots & t_{ik-p} & m_k & \dots & m_{k-p} \end{bmatrix}^T$$
$$\theta_k^\epsilon = \begin{bmatrix} a_0^\epsilon & \dots & a_{p-1}^\epsilon & b_0^\epsilon & \dots & b_{p-1}^\epsilon & c_0^\epsilon & \dots & c_{p-1}^\epsilon \end{bmatrix}^T$$

where $a, b \in \mathbb{R}$ estimator coefficients. We chose $p \gtrapprox 3$, a value that allows the estimate for ϵ and m to be accurate, while being still manageable in limited memory space.

The estimator can be computed using the Normalized Least-Mean-Square Algorithm (NLMS) [5]:

$$\theta(t+1) = \theta(t) + \frac{\mu(t)\phi(t)}{\phi(t)^T\phi(t) + \omega}[y(t+1) - \phi(t)^T\theta(t)] \quad (2)$$

where $\mu(t)$ is a scalar, and ω should be chosen to avoid a division by zero when $\phi(t)^T\phi(t)$ is null. With our notations, $\phi(t)$ is thus the values of the output $y(t) = m(t)$ or $\epsilon(t)$, the command $u(t) = t_i(t)$ and the target $y^*(t) = m^*(t)$ or $\epsilon^*(t)$.

3.2.2 Cost Minimization

The controller should minimize a cost function with a packet loss and an energy component. The controller attempts to minimize the following cost function J:

$$J = (m^{*+} - \hat{m}_{k+1})^2 + K_\epsilon(\epsilon^{*+} - \hat{\epsilon}_{k+1})^2 \quad (3)$$

where $\begin{cases} \hat{m}_{k+1} = \phi_k^{mT}\theta_k^m \\ \hat{\epsilon}_{k+1} = \phi_k^{\epsilon T}\theta_k^\epsilon \end{cases}$, m^{*+} and ϵ^{*+} designate the target values of m and ϵ at time $(k+1)\mathcal{T}$. $K_\epsilon \approx 20$ is a weight given to the cost function in order to indicate a preference to save energy (large K_ϵ) or to strictly meet the number of packets to be sent (small K_ϵ). The control law finds the value of t_i that minimizes J.

Taking the derivative of J at time $k\mathcal{T}$ gives (we omit the k index notation for clarity):

$$t_i = \frac{\theta_p^m(m^{*+} - \sum_{i\neq p}^u \phi_i^m\theta_i^m) + K_\epsilon\theta_p^\epsilon(\epsilon^{*+} - \sum_{i\neq p}^v \phi_i^\epsilon\theta_i^\epsilon)}{(\theta_p^m)^2 + K_\epsilon(\theta_p^\epsilon)^2}$$

where the i-index value on ϕ_i and θ_i are the i^{th} value of these vectors, and u and v are the number of elements in ϕ_k^m and ϕ_k^ϵ ($u = 2p$ and $v = 3p$).

In order to smooth the response of the system, we adopt a conservative update policy $\bar{u}$ for the duty cycle with the following set of rules:

$$\begin{cases} \bar{t}_{ik+1} = \bar{t}_{ik} + \alpha(t_i - \bar{t}_{ik}) \\ \bar{u}_{k+1} = f_\delta^\Delta[\bar{t}_{ik+1}] \end{cases} \quad (4)$$

where $\bar{t_i}$ is the smoothed t_i and

$$f_\delta^\Delta[x] = \begin{cases} \delta & \text{if } x < \delta \\ \Delta & \text{if } x > \Delta \\ x & \text{otherwise} \end{cases}$$

δ and Δ are the minimum and maximum values that t_i can ever take, and can be set to $0.1\ s$ and $4\ s$.

$\alpha \in \mathbb{R}$ is the slope of the update of t_i and helps stabilize the system response, which would otherwise be unstable because of steep variations of the reference $r(t)$ (the desired number of packets for instance) and delays in the feedback. A large α (*i.e.,* close to 1) aggressively updates t_i and incurs oscillations before reaching a determined value. On the other hand, if α is close to 0, no oscillations can be discerned but t_i is slow to reach its eventual value. Poor choices of α may cause energy waste or packet loss. The command used to control the duty cycle is in fact $\bar{u}$ as a smoothed output is critical to a physical network.

3.3 Evaluating the Target Energy

In some cases, the system designer may want to minimize the consumed energy and choose $\epsilon^* = 0$. The risk incurred by this approach is that the duty cycle will tend to be lowered, even below a reasonable value—one that strikes a balance between the number of lost packets and energy consumption. This could be desirable when designing a system that needs to respond faster to lower energy consumption, and that can tolerate repeated packet losses.

In other systems, an acceptable energy consumption value has to be evaluated so that $t_i(t)$ does not consistently increase past a reasonable value. This target energy has critical importance as the system will have a tendency to stabilize around the value of t_i that yields this energy consumption, provided all packets are correctly sent. The control problem thus becomes a linear quadratic tracking ("LQ

Variable initialization:
$\phi^m = \begin{bmatrix} m^* & 0 & 0 & t_i & 0 & 0 \end{bmatrix}^T$
$\phi^\epsilon = \begin{bmatrix} \epsilon^* & 0 & 0 & t_i & 0 & 0 & m^* & 0 & 0 \end{bmatrix}^T$
$\theta^m = \begin{bmatrix} 0.95 & 0.1 & 0.1 & -0.5 & -0.1 & -0.1 \end{bmatrix}$
$\theta^\epsilon = \begin{bmatrix} 0.95 & 0.1 & 0.1 & -0.5 & -0.1 & -0.1 & 0.3 & 0.1 & 0.1 \end{bmatrix}$
for ever **do**
  $m^* = f(\text{packetRate}, \mathcal{T})$
  $\epsilon^* = f(\text{radio}, m^*)$
  θ^m+=$\frac{\mu^m}{r^m}\phi^m(m - \phi^{mT}\theta^m)$
  θ^ϵ+=$\frac{\mu^\epsilon}{r^\epsilon}\phi^\epsilon(\epsilon - \phi^{\epsilon T}\theta^\epsilon)$
  $u = \frac{\theta_p^m(m^{*+} - \sum_{i \neq p}^{u} \phi_i^m \theta_i^m) + \theta_p^\epsilon(\epsilon^{*+} - \sum_{i \neq p}^{v} \phi_i^\epsilon \theta_i^\epsilon)}{(\theta_p^m)^2 + (\theta_p^\epsilon)^2}$
  $\bar{u} = f_{0.1}^5[\bar{u} + \alpha(u - \bar{u})]$
  $\phi^m = \overrightarrow{\phi^m} + \begin{bmatrix} m & \bar{u} \end{bmatrix}$
  $\phi^\epsilon = \overrightarrow{\phi^\epsilon} + \begin{bmatrix} \epsilon & \bar{u} & m \end{bmatrix}$
  Where $\rightarrow$ is a matrix shift operator
  r^m+=$\phi^{mT}\phi^m$
  r^ϵ+=$\phi^{\epsilon T}\phi^\epsilon$
end for

algorithm 3.1: Control pseudo-code for $p = 3$.

tracking") problem where the output of the network must match the energy (and packet delivery) reference.

We chose to evaluate the target energy as the sum of several basic operations (channel probe, packet reception, etc.) for which we precisely measured the energy consumption via a data acquisition board on the Tmote Sky platform. The target energy ϵ^* assumes that each packet is sent every t_i s, and that no energy is wasted on probing a clear channel. It contains no information about other transmissions in the neighborhood as packet loss is taken into account in the first element of J.

3.4 Algorithm for t_i Control

We implemented the previous theoretical foundations in Matlab; Algorithm 3.1 presents the pseudo-code of the controller used to command the network. The initialization of the algorithm variables includes assigning a starting value to the ϕ and θ vectors. ϕ can take the initial values of m^*, t_i and ϵ^*, while θ is initialized with values between -1 and 1. For instance, an increase in t_i translates into a decrease in m and ϵ of node $\mathcal{N}$, and thus the corresponding weights in θ are negative.

In our implementation, we chose an initial $\alpha = 0.01$ and then adjust α to be 0.2 after three iterations of the controller to prevent large oscillations during the first rounds of the estimators. Our network consists of 10 nodes, all in range of one another (the medium can be occupied by only one node at a time). We evaluate the new command $\bar{u}$ every $\mathcal{T} = \frac{5}{\text{packetRate}}$ seconds: for instance, if a node sends packets at a rate of 2 packets per second, the controller will run every 2.5 seconds. The feedback period $\mathcal{T}$ can be increased, although a large value could cause the network adaptation to be sluggish—or worse, instable.

4 Simulation Results

This section presents simulation results for only a limited number of scenarios due to lack of space. The radio behavior was modeled not only after the CC2420 data sheet, but more importantly after the energy use of the whole Tmote Sky platform running a TinyOS implementation. Although we present simulation results, our model closely resembles a real-life deployment, typically within 3% of the measured energy consumption [9]. Here, the term simulation designates an accurate *reconstruction* of the reality.

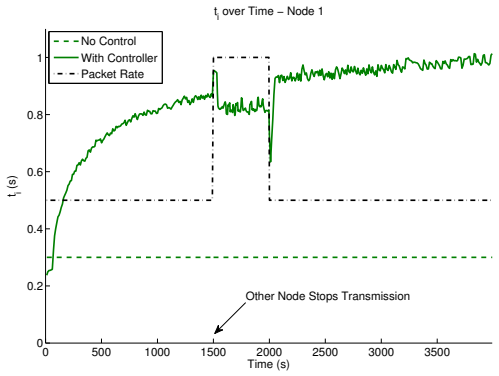
First, we observe the case when two nodes compete for the medium to send packets and only one node can modify its duty cycle. Then, we validate the duty cycle control when more than one node concurrently adjusts their t_i values.

4.1 Lowering the Duty Cycle to Save Energy: Demonstration of Principle

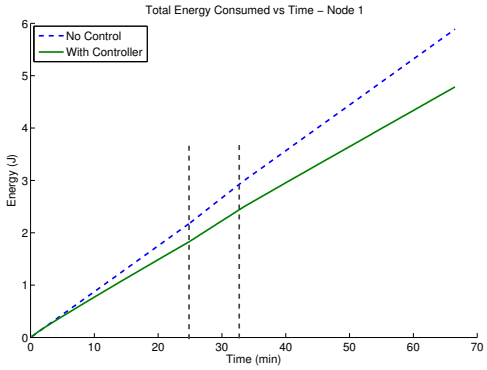
Without the ability to adapt t_i, nodes running a LPL MAC protocol would force designers to select a high duty cycle at deployment to ease contention in busy areas of the network. Consequently, we start with a t_i value of 300 ms, with two nodes sending packets at an initial rate of 0.5 packet per second. We note the evolution of the duty cycle and the energy consumption in Figure 2.

Figure 2(a) presents the evolution of t_i as well as the scenario of the simulation. Because a lower duty cycle can comfortably accommodate concurrent packet rates of 0.5 $pkt.s^{-1}$, the value of t_i increases from 300 ms to around 950 ms in under $1,500$ s (25 min). At this point, the other packet source is turned off, and the packet rate of the remaining node is increased to 1 $pkt.s^{-1}$. The t_i value remains around 950 ms, as this t_i value translates into an energy consumption within close range of the target energy. After $2,000$ s, the packet rate is halved to $0.5\,pkt.s^{-1}$. This allows the duty cycle to decrease further, as t_i goes from 950 ms to 1.1 s, although very slowly. Higher values of α would allow for a more aggressive evolution of t_i, but such values typically induce higher instability and do not fit all scenarios. No packet was lost during this scenario, in spite of the increase in t_i.

Figure 2(b) presents the energy consumed by the $\mathcal{N}$ node, with and without t_i control. t_i control helped reduce energy consumption by up to 19% at the end of the simulation. As the duty cycle keeps decreasing, this number will likely increase.



(a)



(b)

Figure 2. (a) Evolution of $t_i(t)$ as the packet rate varies. (b) Energy consumption under the same scenario.

4.2 Packet Loss Minimization

We now study the other variable of interest by observing the number of lost packets. This example differs from the previous one in the initial value of t_i (now $1.5\ s$) and in the number of neighbors transmitting over time. While it is unlikely that a protocol designer would choose such a high value for t_i in the "fixed" case, this part of our work shows the behavior of our control scheme when packet loss occurs.

Figure 3 shows a decreasing t_i as packets are dropped. Repeated packet losses cause the duty cycle to be increased by a greater amount. The t_i value can be observed to increase slightly between two packet losses in the first $25\ min$ of the runtime as the energy component (approximately equal to the packet loss component) pushes the energy consumption down and the t_i value up. In the second part of the runtime, packet losses become more frequent as the t_i value is unrealistically high compared to the packet rate, until it reaches less than $1\ s$. Design choices could allow for a more aggressive t_i descent, which would prevent the unsightly "oscillations" on t_i, but this would compromise the rate of the t_i increase once the packet rate declines again.

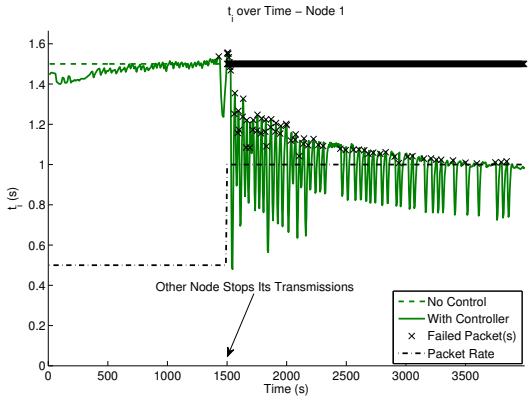


Figure 3. Evolution of t_i over time under a changing scenario.

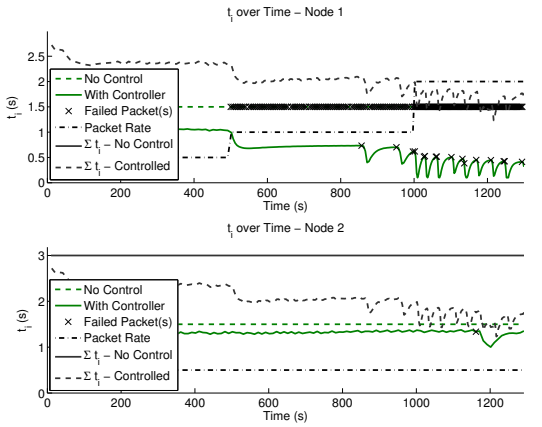
The controlled scheme was able to limit packet loss by 88% over the fixed scenario. As the duty cycle is iteratively modified, the frequency of dropped packets diminishes. However, the increase in packet deliveries is compensated by an increase in energy consumption. Even though the number of dropped packets was cut by over 88%, the additional energy consumption was kept to less than 7% (not shown). The fixed scheme consumes less energy because of two reasons: its duty cycle remains at a low value, and contention around the nodes forces both the sender and the receiver to sleep for longer periods of time instead of transmitting packets—a behavior that results in lower energy consumption.

4.3 Multiple Controllers

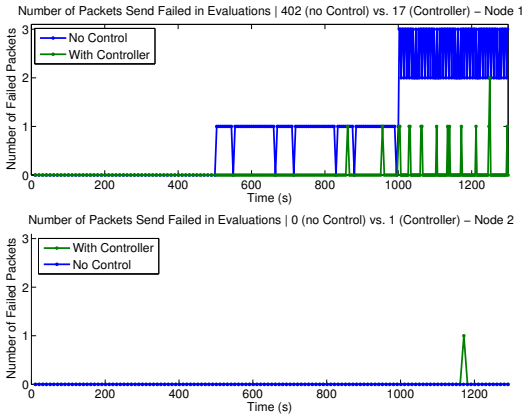
A legitimate concern of t_i control deals with the implementation of several nodes adapting their duty cycle at the same time: the modifications of one should not destabilize the others.

Figure 4 shows that this is not the case as two sending nodes (nodes 1 and 2) correctly adapt their duty cycle to conditions in the local area. The "cumulative t_i" is the sum of the t_i values of the sending nodes and can be seen as a measure of the busyness of a local area.

Figure 4(a) shows that the duty cycle increases when it can no longer accommodate the packet rate (after $1,000\ s$) requested at node 1. Since the packet rate of node 2 is small for the duration of the simulation, its t_i value stays around $900\ ms$ and only decreases after a packet loss. Such a reduction in t_i allows node 1 to drastically reduce the number of dropped packets by as much as 95% as seen in Figure 4(b), although this incurs a small energy consumption



(a)



(b)

Figure 4. (a) Evolution of t_i over time under a changing scenario for the two nodes with duty cycle control. (b) Number of packets lost.

increase (not shown here).

5 Conclusions and Future Work

Low-power-listening MAC protocols show great promise to increase WSN lifetime by reducing idle listening. However, such MAC protocols were typically reserved for networks with low packet rates so as to allow low duty cycles (and greater energy savings).

In this paper, we introduce a control theory method that jointly optimizes the energy consumed at vulnerable nodes, and the number of packets to be transmitted. This results in energy savings on the order of 20%, or in a drastic reduction of dropped packets. This latter property allows the network to respond to sudden bursts of packets as caused by the occurrence of an event, making LPL MAC protocols fit for a greater number of WSN applications. The increase in delivered packets typically comes at a higher premium on energy consumption, although energy savings mean little if the network is unable to serve the application. The proposed t_i control method, which does not require knowledge of a system's physical model, can also be applied to the control of other parameters in the network.

For our future work, we plan to adapt this method of controlling t_i to real-life deployments. This includes providing support for multi-hop networks and for various deployment configurations. We also plan to investigate further the effects of certain variables (such as K_ϵ or α) on the speed and accuracy of the network response.

References

[1] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proc. 2^{nd} ACM Conf. on Embedded Networked Sensor Systems (SenSys'06)*, pages 307–320, 2006.

[2] Chipcon Products from Texas Instruments. CC2420 data sheet, 2.4 ghz ieee 802.15.4 / zigbee-ready rf transceiver.

[3] A. El-Hoiydi. Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks. In *Proc. IEEE Int. Conf. on Communications (ICC)*, Apr. 2002.

[4] A. El-Hoiydi and J. Decotignie. WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks. In *Proc. 1^{st} Int. Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, 2004.

[5] G. C. Goodwin and K. S. Sin. In *Adaptive Filtering Prediction and Control*. Prentice-Hall, 1984.

[6] IEEE Computer Society LAN MAN Standards Committee. Wireless medium access control (MAC) and physical layer (PHY) specifications for low rate wireless personal area networks (LR-WPANs). In *IEEE Std. 802.15*, 2004.

[7] R. Jurdak, P. Baldi, and C. V. Lopes. Adaptive low power listening for wireless sensor networks. In *IEEE Transactions on Mobile Computing*, volume 6, Aug. 2007.

[8] P. Kumar and P. Varaiya. In *Stochastic Systems*. Prentice-Hall, 1986.

[9] C. J. Merlin and W. B. Heinzelman. Network-aware adaptation of mac scheduling for wireless sensor networks. In *Proc. 3^{rd} Conf. on Distributed Computing in Sensor Systems (DCOSS'07 Poster Session)*, June 2007.

[10] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proc. SenSys'04*, pages 95–107, Nov. 2004.

[11] C. M. Vigorito, D. Ganesan, and A. G. Barto. Adaptive control of duty cycling in energy-harvesting wireless sensor networks. In *Proceedings of The Fourth IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'07)*, June 2007.

[12] K.-J. Wong and D. Arvind. Speckmac: Low-power decentralised mac protocol low data rate transmissions in specknets. In *Proc. 2^{nd} IEEE Int. Workshop on Multi-hop Ad Hoc Networks: from Theory to Reality (REALMAN'06)*, May 2006.