

# Sleep in the Dins: Insomnia Therapy for Duty-cycled Sensor Networks

Jiliang Wang, Zhichao Cao, Xufei Mao and Yunhao Liu  
School of Software and TNLIST, Tsinghua University, China  
{jiliang, caozc, xufei, yunhao}@greenorbs.com

**Abstract**—Duty cycling mode is widely adopted in wireless sensor networks to save energy. Existing duty-cycling protocols cannot well adapt to different data rates and dynamics, resulting in a high energy consumption in real networks. Improving those protocols may require global information or heavy computation and thus may not be practical, leading to empirical parameters in real protocols. To fill the gap between the application requirement and protocol performance, we design a light-weight adaptive duty-cycling protocol (LAD), which reduces the energy consumption under different data rates and protocol dynamics. We theoretically validate the performance improvement of the protocol. We implement the protocol in TinyOS and extensively evaluate it on 40 TelosB nodes. The evaluation results show the energy consumption can be reduced by 28.2%~40.1% compared with state-of-the-art protocols. Results based on data from a 1200-node operational network further show the effectiveness and scalability of the design.

## I. INTRODUCTION

Recent advances in Wireless Sensor Networks (WSNs) have fostered a large collection of applications [1] [2]. In those networks, a collection of battery powered sensor nodes are self-organized to form a network, interact with the physical world and perform certain tasks, e.g., data collection. Due to the limited energy budget on wireless sensor nodes, the duty-cycling mode is often used to achieve a long lifetime. In the duty-cycling mode, each node periodically turns on the radio to sense the channel and receive packets. Then the node turns off the radio when there are no packets in order to reduce energy consumption of idle listening.

Due to the importance of duty-cycling mode, a large collection of duty-cycling protocols are developed in WSNs. In *synchronous* duty-cycling protocols [3] [4] [5], the sender and receiver are synchronized, which enables the sender transmit packets right after the receiver wakes up. Synchronous duty-cycling protocols require time synchronization [6] with extra overhead and hence are not flexible and efficient [4]. To overcome those shortcomings, *asynchronous* duty-cycling protocols, e.g., [7] [8] [9] [10], are proposed. In those protocols, each node employs the Low Power Listening (LPL) technique to periodically wake up after sleeping for a certain period (namely *sleep interval*). After waking up, a node stays awake for channel sensing and packet reception (namely *awake time*). With packets to transmit, a node first transmits preambles until the receiver wakes up. Recently, some variant techniques, such

as low power probing (LPP) [11], are proposed to support receiver initiated duty-cycling protocols, e.g., [12].

With those basic designs, there are many works to further improve energy efficiency and support adaptive duty-cycling. For example, MiX-MAC [13] improves the energy efficiency by switching between different duty-cycling MAC protocols. In IDEA [14], a centralized method is proposed to tune the parameters in LPL protocols. In GDSIC [15], a distributed method is presented to achieve energy fairness. In X-MAC [9], energy efficiency is improved by tuning the sleep interval. In [16] [17], heuristic approaches are proposed to improve energy efficiency. In [18], an efficient method is presented to reduce unnecessary awake time due to interference.

Those existing protocols propose promising approaches to improve energy efficiency. However, there exist several problems while applying those protocols to practical WSNs. First, many impacting factors, e.g., data rate, which significantly impact the performance, are not thoroughly addressed in practical designs. Besides, existing works focus on the duty cycle ratio, i.e., the ratio of awake time to sleep interval. However, even for the same duty cycle ratio, different lengths of awake time can lead to different energy consumption. Second, many existing protocols require global information, centralized or heavy computation to improve the performance. Third, in practical protocols, e.g., TinyOS LPL MAC, the awake time is dynamic and depends on the traffic pattern. Such dynamic behaviors, which have a significant impact to system performance [19], are not thoroughly analyzed and addressed.

Due to the existence of those problems, the performance of duty-cycling protocols may significantly deviate from the optimal performance. Such problems are also experienced in a real network CitySee [2], in which 1200 nodes are deployed in the urban area. The duty-cycling mode is adopted to save energy. Based on the collected data, we find that without carefully considering the traffic impact and the awake time, the duty cycle ratio is significantly different from small scale tests. We also find that state-of-the-art protocols with empirical parameters [19] are not adequate to optimize the performance, and also not adaptive for nodes with various traffic patterns in the network.

In this work, we propose a framework for distributed duty-cycling protocol design under different traffic patterns with protocol dynamics. The framework incorporates the awake time and traffic pattern to reduce the energy consumption. We

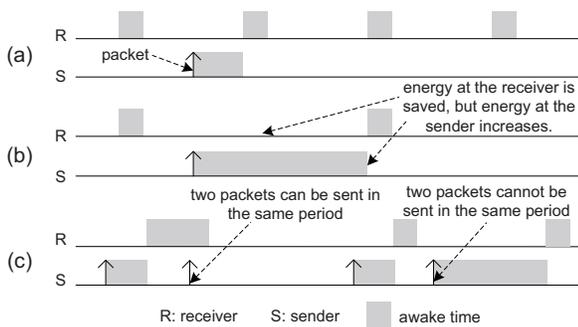


Fig. 1: A motivating example. Both the sleep interval and awake time should be carefully set.

theoretically analyze the performance gain of the proposed framework. Further, as an example, we apply the analysis to the *de facto* duty-cycling protocols in WSNs, i.e., TinyOS LPL MAC [20]. We find that the LPL MAC, though widely adopted by application developers, is inefficient under various network conditions. Our design can significantly improve the performance.

We implement our design LAD in TinyOS [21] and conduct extensive experiments on a network with 40 TelosB nodes. We also examine the performance based on data from a 1200-node network to show the scalability of our design. The contributions are summarized as follows.

- We present a framework to qualitatively analyze the significant impact of traffic patterns and protocol dynamics in duty-cycling protocols.
- We propose a light-weight distributed duty-cycling protocol design (LAD) which can achieve optimal energy efficiency with different data rates and protocol dynamics in real networks.
- We implement the protocol in TinyOS with TelosB nodes. The experimental results demonstrate that our protocol can achieve 28.2%-40.1% performance gain compared to existing duty-cycling protocols.

The remainder of this paper is organized as follows. Section II presents the analysis of the energy consumption in the duty-cycling mode and the analysis on a real protocol implementation. Section III shows our protocol design based on the analysis. Section IV presents the implementation and evaluation results. Section V introduces related work and Section VI concludes this work.

## II. PROTOCOL ANALYSIS

In this section, we first show the basic mechanism of LPL protocol with a simple motivating example. Then we analyze the impact of different parameters to protocol performance. Based on the analysis, we explain why existing adaptive protocols cannot achieve optimal performance and how to achieve this. We also analyze the *de facto* duty-cycling protocol implementation in TinyOS and show that our design can significantly improve the performance.

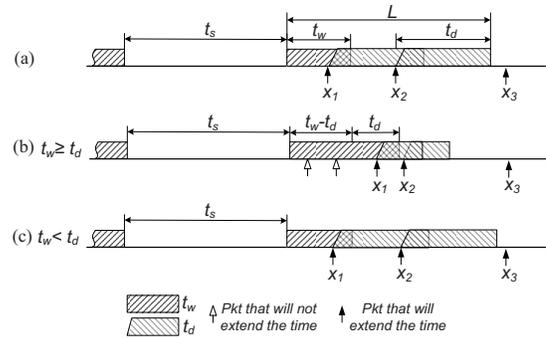


Fig. 2: Illustration of the duty-cycling mechanism.

### A. Background

1) *LPL mechanism*: While there exist a variety of duty-cycling protocols, they share a similar design principle. As shown in Figure 1(a), each node periodically (with a cycle  $t_s$ ) wakes up and senses the channel. If the channel is busy, the node remains awake for a certain period of time  $t_w$ . Upon receiving a packet, the receiver extends the awake time for a certain period  $t_d$  (e.g., TinyOS LPL MAC). With packets to send, a node sends preambles or short strobes until the receiver wakes up.

2) *Motivating example*: Figure 1 shows three examples. Case (a) shows the simplified original LPL protocol. In case (b), with a larger sleep interval, the energy consumption at the receiver is reduced. However, the energy consumption at the sender increases because the sender needs to wait for a longer time until the receiver wakes up. While adjustment on the sleep time can significantly reduce the energy consumption, it is still not adequate. As in case (c), the traffic rate also impacts the energy consumption. If the awake time is large for different traffic rates, the energy consumption at the sender can be significantly reduced since multiple packets can be sent during the same waking up period of the receiver. However, the energy consumption at the receiver will increase. Thus the sleep interval and awake time should be carefully designed to reduce the total energy consumption.

3) *Parameters*: In this work, we focus on analyzing the sender-initiated protocol. Both sender-initiated and receiver-initiated protocols are evaluated in Section IV.

We have the following parameters.

- $t_s$ : the sleep interval.
- $t_w$ : the time a node stays awake after waking up,  $t_w = 0$  means the node will immediately go to sleep when no signal is sensed.
- $t_d$ : the dynamically extended awake time, i.e., a node extends the awake time to  $t + t_d$  after receiving a packet at  $t$ .
- $\tau$ : the overhead to sense the channel.

In our analysis, we assume the data rate is  $\lambda$  and different nodes may have different  $\lambda$ .

## B. Energy Analysis

According to the mechanism, there are two cases for sending a packet in duty-cycling protocols:

- If the receiver is sleeping, the sender should wait until the receiver wakes up. We denote this kind of transmission as *preambled transmission*, e.g.,  $x_3$  in Figure 2(a).
- Otherwise, the packet is transmitted without preambles. We denote this kind of transmission as *non-preambled transmission*, e.g.,  $x_1$  and  $x_2$  in Figure 2(a).

In our analysis, we mainly focus on the radio-on (awake) time, which is the main source of energy consumption on sensor nodes [9][15]. We denote the period from the time a node goes to sleep state to the next time the node goes to sleep state as a cycle, i.e., the period of length  $t_s + L$  in Figure 2(a). Note that the cycle length may not be fixed if  $t_d \neq 0$ . In each cycle, the energy consumption consists of the following two parts [9][15]: (1) radio-on time for receiving packets, including the time for receiving packets and channel sensing; (2) radio-on time for sending packets. To calculate the average energy consumption for each packet in each cycle, we first calculate the expected total radio-on time for sending and receiving packets in each cycle.

1) *Energy consumption for receiving packets*: We first calculate the energy consumption for receiving packets. If  $t_d = 0$ , the radio-on time at the receiver for each cycle is  $t_w$ .  $t_w = 0$  means that a node immediately goes to sleep when no signal is detected [9]. In real protocols the receiver may dynamically extend the awake time upon a reception. We assume there are  $k$  packets to extend the awake time, namely *extending packets*. Denote those  $k$  packets as  $x_1, x_2, \dots, x_k$  and the corresponding receiving time as  $t_1, t_2, \dots, t_k$ . To facilitate the analysis, we denote the packet after  $x_k$  as  $x_{k+1}$  at time  $t_{k+1}$ . Then we calculate the expected radio-on time for two cases,  $t_w \geq t_d$  and  $t_w < t_d$ . We first calculate the probability of  $k$  extending packets. Without ambiguity, we use the notation to denote the corresponding period in the figure as well as the length of the period. Assume in each cycle, the time starts at 0.

*Case 1* ( $t_w \geq t_d$ ): As shown in Figure 2(b), a packet can extend the radio-on time only when it is received after  $t_s + (t_w - t_d)$ . Denote the probability for  $k$  extending packets as  $P_1(k)$  and the number of transmissions in a time interval  $t$  by  $N(t)$ . We have  $P_1(0) = p(N(t_d) = 0)$ . For  $k > 0$ , we have

- $N(t_d) > 0$ : there should be at least one transmission in the time window from  $t_s + (t_w - t_d)$  to  $t_s + t_w$  as shown in Figure 2(b), otherwise  $k = 0$ .
- $t_i - t_{i-1} \leq t_d$  for  $1 < i \leq k$ , otherwise  $t_i$  will not be an extending transmission.
- $t_{k+1} - t_k > t_d$ , otherwise  $t_{k+1}$  should also be an extending transmission in the same cycle.

Then the probability for  $k$  ( $k > 0$ ) extending packets is

$$P_1(k) = p(N(t_d) > 0 \wedge t_2 - t_1 \leq t_d \wedge \dots \wedge t_k - t_{k-1} \leq t_d \wedge t_{k+1} - t_k > t_d)$$

*Case 2* ( $t_w < t_d$ ): Denote the probability as  $P_2(k)$ . This case can be further divided into two cases considering whether

there are packet transmissions in the sleeping time of length  $t_s$ .

*Case 2.1* ( $N(t_s) > 0$ ): There are packet transmissions in the sleep time of length  $t_s$ . The receiver can receive packets after it wakes up and the radio-on time is extended by a length of  $t_d$ . Then this case transforms to Case 1 with  $t_w = t_d$ .

*Case 2.2* ( $N(t_s) = 0$ ): There is no packet transmission in the sleep time of length  $t_s$ . The radio-on time will only be extended if there are packets received after the receiver wakes up, i.e.,  $N(t_w) > 0$ . If  $k = 0$ , we have  $P_2(0) = p(N(t_w) = 0)$ . For  $k > 0$ , the probability can be calculate as follows,

$$P_2(k) = p(N(t_w) > 0 \wedge t_2 - t_1 \leq t_d \wedge \dots \wedge t_k - t_{k-1} \leq t_d \wedge t_{k+1} - t_k > t_d)$$

Till now, we have calculated the probability for  $k$  extending packets. To calculate the expected radio-on time for  $k$  extending packets in Case 1 and Case 2, we first calculate expected inter-packet time between two consecutive extending packets. Given the maximum inter-packet time  $t$ , the expected inter-packet time  $T(t)$  is calculated as

$$T(t) = \int_0^t xp(d = x | N(t) > 0) dx. \quad (1)$$

where  $p(d = x)$  denotes the probability that the inter-packet interval is  $x$ .

For Case 1, as shown in Figure 2(b), the expected radio-on time  $L_1(k)$  with  $k$  packets is calculated as

$$L_1(k) = t_w + kT(t_d) \quad (2)$$

For Case 2, as shown in Figure 2(c), the expected radio-on time  $L_2(k)$  with  $k$  packets is calculated as

$$L_2(k) = \begin{cases} t_d + kT(t_d) & \text{Case2.1} \\ t_w & \text{Case2.2\&k} = 0 \\ T(t_w) + (k-1)T(t_d) + t_d & \text{Case2.2\&k} > 0 \end{cases} \quad (3)$$

Eq.(2) and (3) show that for different data rates, the resulted radio on time are different. Then we can calculate the expected radio-on time.

$$E(L) = \begin{cases} \sum_{k=0}^{\infty} L_1(k)P_1(k) & \text{Case1} \\ \sum_{k=0}^{\infty} L_2(k)P_2(k) & \text{Case2} \end{cases} \quad (4)$$

Note here for case 2 we should calculate the expected radio-on time across different cases.

2) *Energy consumption for sending packets*: The expected energy consumption for sending packets depends on the number of preambled transmissions and non-preambled transmissions. We first calculate the number of non-preambled transmission  $M_i$  and preambled transmission  $M_p$ . For Case 1, the expected number of non-preambled packets is the sum of packets in the time window of length  $t_w - t_d$  and packets after such a time period. For Case 2, the expected number of non-preambled transmission is  $E(M_i) = P_2(k)k$ . We have

$$E(M_i) = \begin{cases} \sum_{k=0}^{\infty} kp(N(t_w - t_d) = k) + \sum_{k=0}^{+\infty} P_1(k)k & \text{Case1} \\ \sum_{k=0}^{+\infty} P_2(k)k & \text{Case2} \end{cases}$$

According to the data rate, the expected number of preambled transmissions is

$$E(M_p) = \lambda t_s. \quad (5)$$

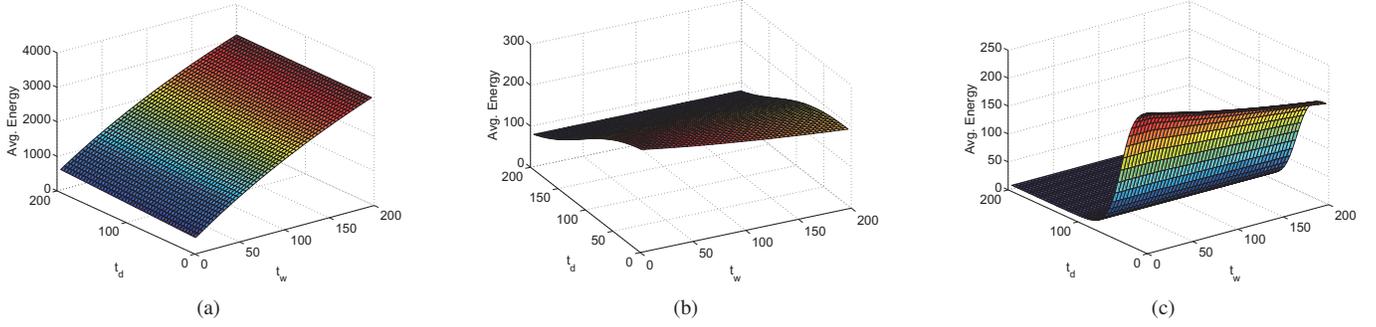


Fig. 3: Performance of duty-cycling for different data rates, with  $\tau = 10$ ,  $\alpha = 1$ ,  $\beta = 1$  and  $\gamma = 1$ . (a)  $\lambda = 0.0001$ , (b)  $\lambda = 0.02$ , (c)  $\lambda = 0.1$  (pkt/ms). For different data rates, using the same parameter setting results in totally different performance. For example, a small  $t_w$  and  $t_d$ , which is preferred in (a), is prohibited in (b).

3) *Average energy consumption*: The energy consumption for each cycle depends the radio-on time for sending and receiving packets, the channel sensing and the energy to send/receive packets. The energy consumption for radio-on time for receiving packets, which is proportional to  $E(L)$ , is calculated as  $\alpha E(L)$ , where  $\alpha$  is a coefficient for energy consumption. The energy consumption for channel sensing is denoted by  $\tau$ . For each preambled transmission, the expected radio-on time at the sender is  $t_s/2$ . For each non-preambled transmission, the extra radio-on time is negligible. Therefore, the expected energy consumption for sending packets can be calculate as  $\beta E(M_p)t_s/2$  with  $\beta$  as a coefficient.

*Theorem 1*: The average energy consumption per packet is calculated as

$$\mathcal{G} = \frac{\alpha E(L) + \beta E(M_p)t_s/2 + \gamma(E(M_p) + E(M_i)) + \tau}{E(M_p) + E(M_i)} \quad (6)$$

Our goal is to optimize  $\mathcal{G}$  for different parameters. We use the equation as the guideline for our protocol design. In the design section, we show how to leverage this equation to improve the energy efficiency.

### C. Example

Assume the traffic follows a poisson distribution. Considering the memorylessness and independence of inter-arrival time, we have

$$P_1(k) = p(N(t_d) > 0) \prod_{i=2}^k p(t_i - t_{i-1} \leq t_d) p(t_{k+1} - t_k > t_d)$$

Denote  $t'_i = t_i - t_{i-1}$ , since the probability density function of inter-arrival time is  $f(t) = \lambda e^{-\lambda t}$ , we have

$$\begin{aligned} P_1(k) &= p(N(t_d) > 0) \prod_{i=2}^k \int_0^{t_d} f(t'_i) dt'_i \int_{t_d}^{+\infty} f(t'_{k+1}) dt'_{k+1} \\ &= (1 - e^{-\lambda t_d})^k e^{-\lambda t_d} \end{aligned} \quad (7)$$

Similarly, we can calculate the  $P_2(k)$ ,  $E(L)$  and finally obtain  $\mathcal{G}$ .

### D. Revisiting Existing Protocols

We revisit the *de facto* duty-cycling protocol implementation, i.e TinyOS LPL MAC, and demonstrate how to leverage the analysis to improve energy efficiency. In TinyOS LPL MAC, the typical channel sensing time is 5~15ms and the typical sleep interval is 500ms. We set  $\alpha = 1$ ,  $\beta = 1$ ,  $t_s = 500$  and  $\tau = 10$ . According to Eq. (6), we calculate average energy consumption  $\mathcal{G}$  for different  $t_w$ ,  $t_d$  and data rate  $\lambda$ . We show the average energy per packet in Figure 3. We find that current parameter settings in TinyOS LPL MAC may lead to very poor performance.

For a low data rate (e.g.,  $\lambda = 0.0001$  pkt/ms), as shown in Figure 3(a), the energy consumption increases when  $t_w$  or  $t_d$  increases. This is because though increasing the awake time reduces the energy consumption at the sender, this increases energy consumption at the receiver. For a low data rate, the reduced energy consumption at the sender is relative small and is defeated by the increased energy consumption at the receiver. In the default TinyOS LPL MAC, the typical value of  $t_d$  is set to 100ms. According to Figure 3(a), such a setting will introduce a significant additional overhead. Thus  $t_d$  should be set smaller.

When the data rate becomes higher (e.g.,  $\lambda = 0.02$  pkt/ms), as shown in Figure 3(b), the energy consumption decreases when both  $t_w$  and  $t_d$  increase. This is because when the traffic is relative high, prolonging the awake time, though increases the energy consumption at the receiver, increases the probability of non-preambled transmissions and thus reduce the energy consumption for the senders. Thus increasing  $t_w$ , previously prohibited, is beneficial. Similarly, prolonging  $t_d$  can also increase the probability for non-preambled transmissions and hence reduce the energy consumption.

When the data rate is even higher (e.g.,  $\lambda = 0.1$  pkt/ms), as shown in Figure 3(c), the energy consumption quickly decreases as the increasing of  $t_d$ . This is because when the data rate is high, the probability of receiving packets, during the extended time of  $t_d$ , becomes very high. When the number of received packets during  $t_d$  increases, the benefit can overcome the overhead. Thus  $t_d$  is a crucial factor to the performance. We should set  $t_w$  and  $t_d$  to larger values.

As a result, we can see that 1) the fixed settings of  $t_w$  and  $t_d$  in TinyOS LPL MAC are not appropriate and the default settings may lead to a high energy consumption, and 2) traditional methods to optimize the sleep interval may not result in an optimal result with the extended time  $t_d$ .

### III. PROTOCOL DESIGN

Based on the analysis result, we present the design of a light-weight distributed adaptive duty-cycling protocol to improve energy efficiency.

#### A. Design Overview

The design consists of three major components, (1) a network estimation component, (2) an online parameter optimization component and (3) an adaptive duty-cycling protocol component. The network estimation component measures the required network status for calculating optimal parameters. Then the parameter optimization component provides the optimal parameters for the duty-cycling protocol. Based on the optimal parameter settings, the adaptive duty-cycling protocol accordingly adjusts the sleep interval, awake time and extending time.

#### B. Network Estimation & Parameter Optimization

To optimize the energy efficiency according to Eq. (6), we need to estimate the parameters  $\lambda$  and  $\tau$ . We estimate the parameter  $\lambda$  using maximum likelihood estimation (MLE). By dividing the time into time slots of length  $w_i$ , we count the number of packets  $k_i$  in the latest  $n$  time slots. Then we obtain the estimation of  $\lambda$  by  $\hat{\lambda}_{MLE} = \frac{1}{nw_i} \sum_{i=1}^n k_i$ . The value of  $\tau$  depends on the channel sensing technique and can be measured offline.

Based on Eq. (6) and the estimated parameters, we calculate the optimal  $t_w$ ,  $t_d$  and  $t_s$ . The challenge is that calculating those parameters according to Eq. (6) introduces a significant computation overhead, which is not applicable for resource limited sensor nodes. To conquer this challenge, we pre-calculate the optimal values of  $t_w$ ,  $t_d$  and  $t_s$  for different  $\lambda$  and store those values on sensor nodes. According to the measured parameters from the parameter estimation component, each node locally searches for corresponding optimal settings of  $t_w$ ,  $t_d$  and  $t_s$  to reduce the computation overhead.

When  $\lambda_{min} \leq \lambda \leq \lambda_{max}$ , we calculate the corresponding values of  $t_w$ ,  $t_d$  and  $t_s$  for the optimal energy consumption. Then we store the optimal values corresponding to discrete values of  $\lambda$  in a table on each sensor node. When  $\lambda < \lambda_{min}$  or  $\lambda > \lambda_{max}$ , we use the parameter settings of  $\lambda = \lambda_{min}$  and  $\lambda = \lambda_{max}$  to approximate those of  $\lambda < \lambda_{min}$  and  $\lambda > \lambda_{max}$ .

#### C. Adaptive Duty-cycling Protocol

To be adaptive to network conditions, the component takes the optimal parameters from the parameter optimization component as input and accordingly adjust the protocol behavior.

First, each node adjusts the sleep interval and awake time according to  $t_s$ ,  $t_w$  and  $t_d$ . Meanwhile, the sender should know the parameter settings of the receiver in order to send packets.

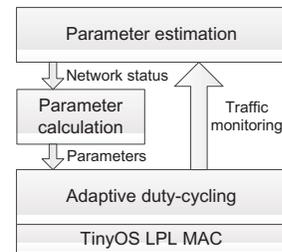


Fig. 4: Implementation of adaptive duty-cycling protocol based on TinyOS LPL MAC.

In our protocol, each node records the sleep interval  $t_s$  for all neighbors. The information is piggybacked in the broadcast or data packets in order to notify other nodes. To increase the probability that the information is received by other nodes, the preamble length of broadcast is set to the maximum length of  $t_s$  in all neighbors. Considering packet losses in real networks, when a sender does not have the sleep interval information for a particular node or the information is stale, the sender will use the maximum  $t_s$  to ensure that the receiver wakes up at least once in a cycle.

### IV. IMPLEMENTATION AND EVALUATION

We implement our protocol in TinyOS 2.1 and evaluate its performance in a network consisting of 40 TelosB nodes. To further validate its scalability and effectiveness, we conduct trace driven simulations based on data from a 1200-node network.

#### A. Implementation

The architecture of the implementation is shown in Figure 4. We build our protocol LAD based on the default duty-cycling protocol in TinyOS. We implement a traffic monitor on top of the TinyOS LPL MAC to record the number of received packets for each time window (currently we set the window size to 1 second). As introduced in Section III, we use the latest 10 windows to estimate  $\lambda$ .

To obtain the optimal parameter settings, as introduced in Section III, we store the optimal parameter settings for different  $\lambda$  on sensor nodes. For  $\lambda < 0.0001$ , we set the  $t_w^{min} = 0$  and  $t_d^{min} = 0$ . For  $\lambda > 0.1$ , we check the space according to optimal value of  $\mathcal{G}$  in Eq. (6) for  $t_w$ ,  $t_d$  and  $t_s$ . We find that increasing  $t_d$  indicates improvement on energy efficiency. However, when  $t_d$  is larger than 100ms, the improvement becomes limited. Thus we set  $t_d^{max} = 100$ ms and  $t_w^{max} = 200$ ms. We  $0.0001 < \lambda < 0.1$ , we discretize  $\lambda$  and calculate the optimal values for  $t_s$ ,  $t_w$  and  $t_d$  with respect to different values of  $\lambda$ . The duty-cycling protocol takes the output from parameter optimization component and accordingly adjust the schedule.

#### B. Evaluation

1) *Methodology*: We evaluate our protocol on a testbed consisting of 40 TelosB nodes. To evaluate the multi-hop

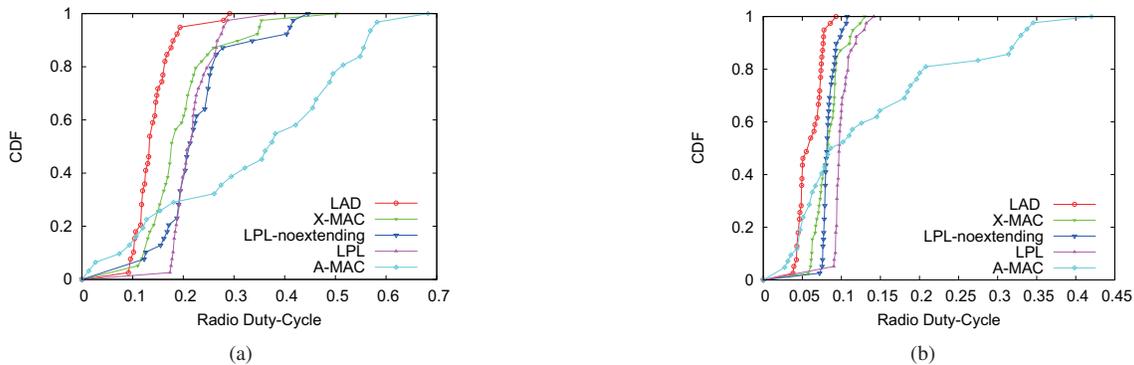


Fig. 5: Duty cycle ratio under different data rates: (a) under a high data rate; (b) under a low data rate.

performance, we incorporate our protocol with CTP [22] protocol for data collection. We compare our design with the following protocols on the testbed:

- TinyOS LPL MAC [20] (LPL) with default settings  $t_s = 500ms$ ,  $t_w = 10ms$  and  $t_d = 100$ .
- TinyOS LPL MAC with minimal  $t_d$  value,  $t_d = 0$  (LPL-noextending).
- Parameter optimization with X-MAC [9].
- A-MAC [12], i.e., the most recent receiver-initiated duty cycling protocol.

We evaluate the performance of those protocols under a relative high data rate (0.25 pkt/s for each node) and a relative low data rate (0.025 pkt/s). For different data rates, we compare the performance of different protocols from the following aspects:

- Duty cycle ratio, the percentage of radio-on time.
- Average energy consumption per packet.
- Packet loss ratio.
- Adaption to different data rates.
- Detailed radio operations.

We further conduct trace-driven simulations with trace from a network consisting of 1200 nodes.

2) *Overall performance*: We first compare the duty cycle ratio for different protocols under different data rates. Figure 5(a) shows the result under a high data rate. We can see that our design outperforms other approaches in terms of duty cycle ratio. More specifically, in our protocol, more than 90% of nodes have a duty cycle ratio lower than 16%. While among other protocols, X-MAC achieves the best performance, because X-MAC can adaptively adjust the parameters. However, the performance of X-MAC is lower than our protocol since X-MAC does not consider the traffic and protocol dynamics. In X-MAC, there are more than 50% of nodes with a duty cycle ratio higher than 15%. Figure 5(b) shows the result for a low data rate. Under a low data rate, the radio duty cycle for all protocols are reduced. In our protocol, more than 80% of nodes have a duty cycle ratio lower than 7%. The average duty cycle improvement to X-MAC with parameter optimization is about 28.8% under a high data rate and 28.2% under a low data rate. The average duty cycle improvement to the default

LPL MAC is about 40.1% under a high data rate and 28.6% under a low data rate.

We also evaluate the average energy consumption per packet. Figure 6(a) shows the average energy consumption per packet under a high data rate. First, we find that under a high data rate, the LPL with delay after receiving ( $t_d > 0$ ) is significantly better than the protocol LPL-noextending ( $t_d = 0$ ). This verifies our observation in Section III that a longer awake time even leads to a lower energy consumption since the time for preambles can be reduced. This coincides with our analysis result in Figure 3(a). In our protocol most nodes have average radio-on time less than 40ms while the best among others has only 50% of nodes with average radio-on time less than 40ms. Figure 6 shows the energy consumption per packet under a low data rate. First, the energy consumption per packet under a low data rate is smaller than that under a high data rate. Under a low data rate, our protocol is still better than other protocols. We can also find that under a low traffic rate, the LPL-noextending becomes slightly better than LPL. This coincides with our result in Figure 3(c).

3) *Packet losses*: Reliability is an important metric for data collection. If the sleep interval and awake time are not appropriate, packets may not be able to be processed in time and thus get lost. We evaluate the packet losses for different nodes. Figure 7 shows the result for different protocols. We can see that under different data rates, our protocol achieves a high reliability. The reliability of A-MAC is relative lower. We investigate the data and find that there are mainly two reasons. First, according to the A-MAC implementation, the sink in A-MAC is not set to be always-on. This causes more packets accumulated near the sink node and results in collisions and overflow. Second, there is no approach provided to adjust the probe time. As introduced in A-MAC [12], probes are easy to collide in A-MAC with a relative high data rate on the testbed. We also test for different network scales and data rates. We find that when the network density is low, the A-MAC performs better and presents similar results with other protocols.

4) *Adaption to different data rates*: We further investigate that how our approach adapts to different data rates. More specifically, we calculate the sleep interval for different nodes

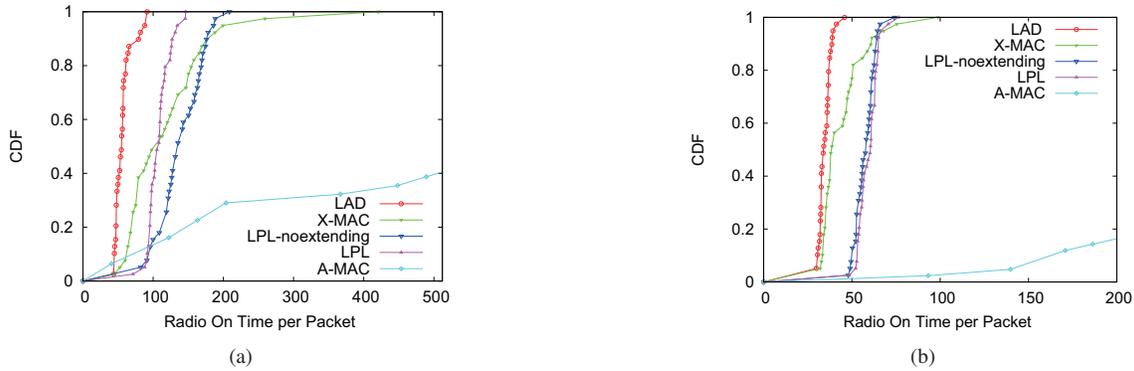


Fig. 6: Energy consumption under different data rates: (a) under a high data rate; (b) under a low data rate.

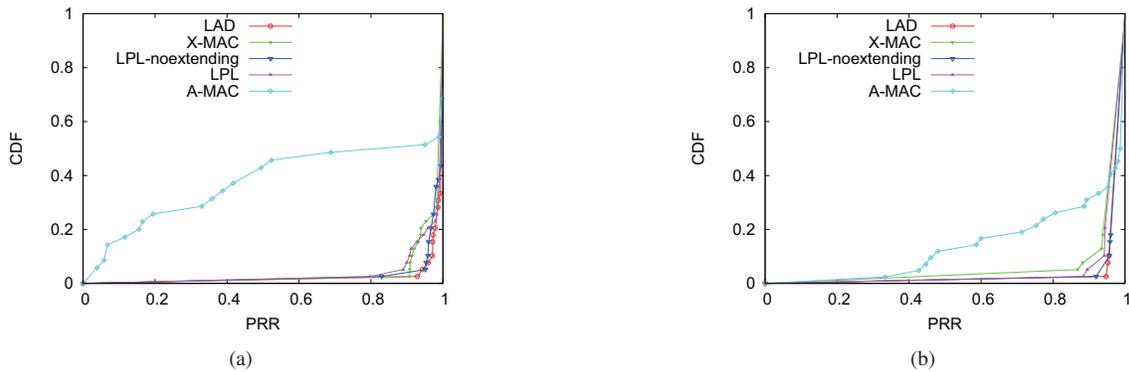


Fig. 7: Reliability of packet transmission for different data rates: (a) under a high data rate; (b) under a low data rate.

with different data rates on the testbed. Figure 8(a) shows the node layout on the testbed. The sink node resides on the left-bottom corner. Figure 8(b) shows the sleep interval distribution for different nodes in our protocol. The darker color indicates a smaller sleep interval. We can see by using our protocol nodes near the sink node with a high data rate have a smaller sleep interval than other nodes. This coincides with our analysis and also further shows that our protocol can adjust the sleep interval according to the data rate.

5) *Radio profiling*: To examine the effectiveness of our protocol, we measure the detailed radio behaviors of each node. To precisely record the radio behavior, we log all radio operations (i.e. radio on/off and packet receiving events) as a tuple  $\langle event, time \rangle$  on the local flash of each node. Then we derive the radio status according to the recorded events. For example, for two consecutive events radio-on and radio-off at time  $t_1$  and  $t_2$ , we derive the period  $[t_1, t_2]$  as a radio-on period. Then we plot the radio status according to the logged events. The data rate in our experiment is set to 4 pkt/s. Figure 9(a) shows the radio operations in the default TinyOS LPL MAC. Figure 9(b) shows the radio operations in our protocol. The upper figures are for the senders and the lower figures are for the receivers. From Figure 9(a), we can see for the default TinyOS LPL MAC there exist many long radio-on time periods for the sender due to the long sleep

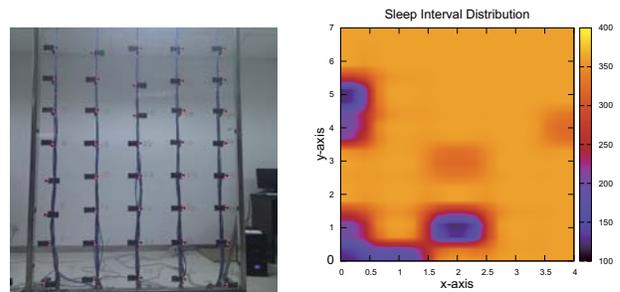


Fig. 8: Node positions and the corresponding sleep interval distribution. The sink node resides at position (0, 0).

interval on the receiver. This is because  $t_s$  and  $t_d$  are not adjusted according to the traffic. Further, Figure 9(b) shows the result for our protocol. By leveraging parameter optimization, the sleep interval and radio-on time can be adjusted according to the traffic. Thus the radio-on time for sending a packet is significantly reduced. While at the receiver side, the radio-on operations are more frequent than that in the default LPL MAC. This is because our protocol considers the energy consumption both at the sender and receiver to optimize the parameters. Therefore, the average energy consumption per packet is significantly reduced.

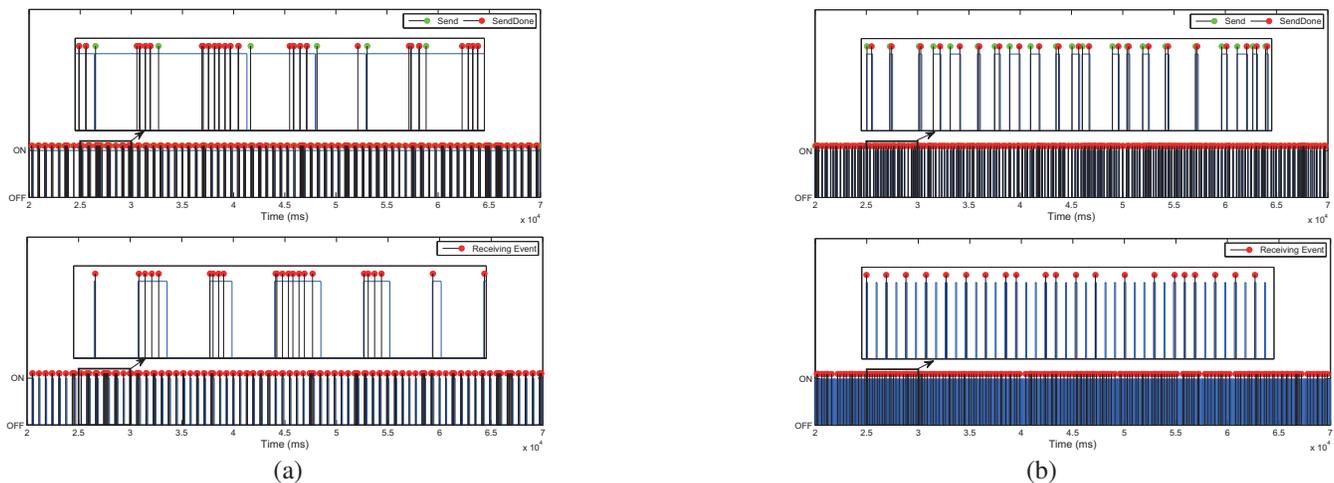


Fig. 9: Radio operations at the sender and receiver. (a) Radio operations for the default LPL MAC. (b) Radio operations for LAD design.

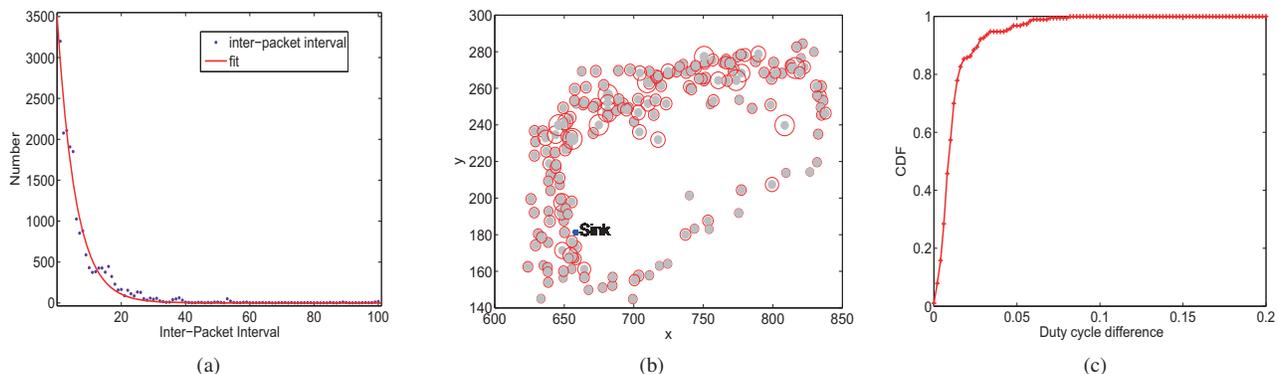


Fig. 10: Trace driven simulation based on data in CitySee network. (a) Real data distribution and the curve fitting result. (b) Duty cycle improvement in a subnet, the red circle denotes the duty cycle calculated from CitySee and the dark disk indicates the duty cycle ratio of our protocol. (c) CDF of the duty cycle improvement.

### C. Trace driven simulations

We further conduct trace driven simulations based on data from CitySee network, which consists of 1200 sensor nodes deployed in the urban area. We use CTP protocol to collect data from the network. Each node in the network transmits 4 data packets back to the sink node every 10 minutes. In the network, we use the TinyOS LPL with  $t_s = 512$ ,  $t_d = 10$  and  $t_w = 10$ .

1) *Performance improvement for different nodes:* We first evaluate the real traffic distribution on each node in the network. In our application, each node records the packet receiving time. Based on the receiving time, we calculate the inter-packet interval to see if the incoming traffic follows poisson distribution. If the traffic follows a poisson distribution, the inter-packet interval should follows the exponential distribution. The result is shown in Figure 10. We also show the curve fitting result for the data. We can see that the real data well fit the exponential distribution. This shows that in the network, the traffic can be approximated with the poisson distribution.

From the collected data, we can calculate the real duty

cycle ratio for each node. Accordingly, we can also use the data rate as the input to our protocol and then calculate the corresponding energy consumption. We compare the duty cycle ratio of LAD to the actual duty cycle ratio in CitySee. Figure 10(b) shows the improvement for nodes at different locations in one subnet. We can see that the duty cycle ratio for most nodes with our protocol is smaller than the duty cycle ratio achieved in CitySee. For nodes near the sink, the improvement is larger because of a higher data rate. We further show the CDF of nodes with duty cycle ratio improvement in Figure 10(c). The duty cycle ratio for more than 80% of nodes can be significantly reduced with our protocol.

## V. RELATED WORK

There are mainly two types of duty-cycling protocols in WSNs. The first type is synchronized duty-cycling protocol, e.g., [3] [4] [5], in which the sender and receiver are synchronized. Protocols of this type may introduce additional computation and communication overhead. Meanwhile, those protocols also have a fixed sleep schedule and are inefficient to handle traffic dynamics [4].

The second type is asynchronous duty-cycling protocol. A representative asynchronous duty-cycling protocol is studied in B-MAC [7]. The sender uses preambles to wake up the receiver. Based on the basic mechanism of B-MAC, many protocols are proposed to improve the performance of B-MAC, such as X-MAC [9], C-MAC [8], Wise-MAC [10], PW-MAC [23] and etc. The basic principles of those protocols are similar. Besides those sender initiated duty-cycling protocols, recently receiver initiated duty-cycling protocols are proposed to reduce the overhead due to collisions in preamble packets. In those protocols, e.g., RI-MAC [11] and A-MAC [12], the receiver will notify the sender to send packets, which is different from the sender initiated protocols in which the sender continuously sending preambles. In this paper, we analyze the mechanism of widely use sender-initiated duty-cycling protocols.

There are also many works proposed to support adaptive duty-cycling and improve the performance. DSF [24] selects a forwarding set to optimize the end-to-end reliability/cost/delay. MiX-MAC [13] improves the energy efficiency by switching between different duty-cycling MAC protocols. IDEA [14] proposes a centralized method to tune the parameters for LPL protocols. In GDSIC [15], a distributed method is proposed to achieve energy fairness. Recently, Sha et al. [18] present an efficient design to improve energy efficiency of LPL in practical networks by addressing the false wake up problem due to interference. pTunes [25] proposes a centralized parameter optimization method which can work for different protocols. In X-MAC [9], sleep interval is calculated to improve energy efficiency. In [16] [17], heuristic approaches are presented to improve energy efficiency. However, as we shown in our analysis, there are various important impacting parameters which are not thoroughly considered in existing works. In this paper, we show that by solely considering the sleep interval and duty cycle ratio, the energy consumption cannot be optimized, especially in practical duty-cycling design with dynamically extended awake time. We need to analyze the impact of different parameters and present a mechanism to optimize energy consumption. So far as we know, such a mechanism is still lacking.

## VI. CONCLUSION

During operation of a 1200-node network, we find that current duty-cycling protocols may lead to a high energy consumption. State-of-the-art protocols cannot efficiently adapt to traffic and protocol dynamics. Thus they are not accurate and adequate to optimize the energy consumption, resulting in many empirical parameters in practical protocols. In this paper, we present a practical adaptive duty-cycling protocol to reduce energy consumption. The protocol minimizes the energy consumption per packet with only local information under various traffic rates and protocol dynamics. We evaluate our approach on 40 TelosB nodes and the results show that our approach can improve the performance by 28.2%-40.1%. Data from a large-scale network deployed in the urban area

also validate the effectiveness of our approach.

## ACKNOWLEDGEMENT

This work is supported in part by NSFC Distinguished Young Scholars Program under grant 61125202, NSFC under grant 61202359, 61373166, 61272426, and China Post doctoral Science Foundation under grant 2012M520013.

## REFERENCES

- [1] X. Wu, M. Liu, and Y. Wus, "In-situ soil moisture sensing: Optimal sensor placement and field estimation," *ACM Transactions on Sensor Network (TOSN)*, vol. 8, no. 4, pp. 1–33, 2012.
- [2] X. Mao, X. Miao, Y. He, X. Li, and Y. Liu, "Citysee: Urban co2 monitoring with sensors," in *IEEE INFOCOM*, 2012.
- [3] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *ACM SenSys*, 2003.
- [4] Y. W. H. J. and E. D., "An energy-efficient mac protocol for wireless sensor networks," in *IEEE INFOCOM*, 2002.
- [5] W. Ye, F. Silva, and J. Heidemann, "Ultra-low duty cycle mac with scheduled channel polling," in *ACM SenSys*, 2006.
- [6] M. Maróti, B. Kusy, G. Simon, and Ákos Lédeczi, "FTSP: The Flooding Time Synchronization Protocol," in *ACM SenSys*, 2004.
- [7] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *ACM SenSys*, 2004.
- [8] S. Liu, K.-W. Fan, and P. Sinha, "Cmac: An energy-efficient mac layer protocol using convergent packet forwarding for wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 5, no. 4, pp. 29:1–29:34, Nov. 2009.
- [9] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks," in *ACM SenSys*, 2006.
- [10] A. El-Hoiydi and J.-D. Decotignie, "Low power downlink mac protocols for infrastructure wireless sensor networks," *Mobile Networks and Applications*, vol. 10, no. 5, pp. 675–690, 2005.
- [11] Y. Sun, O. Gurewitz, and D. B. Johnson, "Ri-mac: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks," in *ACM SenSys*, 2008.
- [12] P. Dutta, S. Dawson-haggerty, Y. Chen, C. Jan Mike Liang, and A. Terzis, "Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless," in *ACM SenSys*, 2010.
- [13] C. J. Merlin and W. B. Heinzelman, "Schedule adaptation of low-power-listening protocols for wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 5, pp. 672–685, 2010.
- [14] G. W. Challen, J. Waterman, and M. Welsh, "Idea: integrated distributed energy awareness for wireless sensor networks," in *ACM MobiSys*, 2010.
- [15] Z. Li, M. Li, and Y. Liu, "Towards energy-fairness in asynchronous low-duty-cycle wireless sensor networks," in *IEEE INFOCOM*, 2012.
- [16] M.-R. Jurdak, S. M.-P. Baldi, and M.-C. V. Lopes, "Adaptive low power listening for wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, pp. 988–1004, 2007.
- [17] C. Merlin and W. Heinzelman, "Cycle control for low-power-listening mac protocols," in *IEEE MASS*, 2008.
- [18] M. Sha, G. Hackmann, and C. Lu, "Energy-efficient low power listening for wireless sensor networks in noisy environments," in *ACM/IEEE IPSN*, 2013.
- [19] Discussion on awake time. [Online]. Available: <http://mail.millennium.berkeley.edu/pipermail/tinyos-help/2008-June/033991.html>
- [20] TinyOS LPL MAC. [Online]. Available: <http://www.tinyos.net/tinyos-2.x/doc/html/tep105.html>
- [21] TinyOS:<http://www.tinyos.net>.
- [22] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *ACM SenSys*, 2009.
- [23] L. Tang, Y. Sun, O. Gurewitz, and D. B. Johnson, "Pw-mac: An energy-efficient predictive-wakeup mac protocol for wireless sensor networks," in *IEEE INFOCOM*, 2011.
- [24] Y. Gu and T. He, "Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links," in *ACM SenSys*, 2007.
- [25] M. Zimmerling, F. Ferrari, T. Voigt, and L. Thiele, "ptunes: Runtime parameter adaptation for low-power mac protocols," in *ACM/IEEE IPSN*, 2012.