

Washington University in St. Louis

Washington University Open Scholarship

All Computer Science and Engineering
Research

Computer Science and Engineering

Report Number: WUCSE-2011-61

2011

Energy-Efficient Low Power Listening for Wireless Sensor Networks in Noisy Environments

Mo Sha, Gregory Hackmann, and Chenyang Lu

Low Power Listening (LPL) is a common MAC-layer technique for reducing energy consumption in wireless sensor networks, where nodes periodically wake up to sample the wireless channel to detect activity. However, LPL is highly susceptible to false wakeups caused by environmental noise being detected as activity on the channel, causing nodes to spuriously wake up in order to receive nonexistent transmissions. In empirical studies in residential environments, we observe that the false wakeup problem can significantly increase a node's duty cycle, compromising the benefit of LPL. We also find that the energy-level threshold used by the Clear Channel Assessment... [Read complete abstract on page 2.](#)

Follow this and additional works at: https://openscholarship.wustl.edu/cse_research



Part of the [Computer Engineering Commons](#), and the [Computer Sciences Commons](#)

Recommended Citation

Sha, Mo; Hackmann, Gregory; and Lu, Chenyang, "Energy-Efficient Low Power Listening for Wireless Sensor Networks in Noisy Environments" Report Number: WUCSE-2011-61 (2011). *All Computer Science and Engineering Research*.

https://openscholarship.wustl.edu/cse_research/64

Department of Computer Science & Engineering - Washington University in St. Louis
Campus Box 1045 - St. Louis, MO - 63130 - ph: (314) 935-6160.

Energy-Efficient Low Power Listening for Wireless Sensor Networks in Noisy Environments

Mo Sha, Gregory Hackmann, and Chenyang Lu

Complete Abstract:

Low Power Listening (LPL) is a common MAC-layer technique for reducing energy consumption in wireless sensor networks, where nodes periodically wake up to sample the wireless channel to detect activity. However, LPL is highly susceptible to false wakeups caused by environmental noise being detected as activity on the channel, causing nodes to spuriously wake up in order to receive nonexistent transmissions. In empirical studies in residential environments, we observe that the false wakeup problem can significantly increase a node's duty cycle, compromising the benefit of LPL. We also find that the energy-level threshold used by the Clear Channel Assessment (CCA) mechanism to detect channel activity has a significant impact on the false wakeup rate. We then design AEDP, an adaptive energy detection protocol for LPL, which dynamically adjust a node's CCA threshold to improve network reliability and duty cycle based on application-specified bounds. Empirical experiments in both controlled tests and real-world environments showed AEDP can effectively mitigate the impact of noise on radio duty cycles, while maintaining satisfactory link reliability.

Energy-Efficient Low Power Listening for Wireless Sensor Networks in Noisy Environments

Mo Sha, Gregory Hackmann, Chenyang Lu
Department of Computer Science and Engineering
Washington University in St. Louis
One Brookings Drive, Box 1045
St. Louis, Missouri 63130, USA
{msha, ghackmann, lu}@wustl.edu

ABSTRACT

Low Power Listening (LPL) is a common MAC-layer technique for reducing energy consumption in wireless sensor networks, where nodes periodically wakeup to sample the wireless channel to detect activity. However, LPL is highly susceptible to *false wakeups* caused by environmental noise being detected as activity on the channel, causing nodes to spuriously wakeup in order to receive nonexistent transmissions. In empirical studies in residential environments, we observe that the false wakeup problem can significantly increase a node's duty cycle, compromising the benefit of LPL. We also find that the energy-level threshold used by the Clear Channel Assessment (CCA) mechanism to detect channel activity has a significant impact on the false wakeup rate. We then design AEDP, an adaptive energy detection protocol for LPL, which dynamically adjust a node's CCA threshold to improve network reliability and duty cycle based on application-specified bounds. Empirical experiments in both controlled tests and real-world environments showed AEDP can effectively mitigate the impact of noise on radio duty cycles, while maintaining satisfactory link reliability.

Categories and Subject Descriptors

C.2.2 [Computer-Communication Networks]: Network Protocols

General Terms

Design; Experimentation; Performance

Keywords

Wireless Sensor Networks; Low Power Listening; CCA Control

1. INTRODUCTION

Clear Channel Assessment (CCA) is a fundamental mechanism in MAC protocols for wireless networks. A CCA check¹ samples the energy level in the wireless channel and considers the channel busy if the energy level is above a

¹CCA, carrier sense and energy detection are used as synonymous in this paper, as supported by [20].

threshold, or idle otherwise. CCA has been commonly used for two important (and orthogonal) purposes. First, it has been used by CSMA/CA protocols to avoid collisions on shared wireless channels, by sampling the channel for activity just before transmission. Second, CCA has been used in Low Power Listening (LPL), a popular MAC-layer approach that enables radio to operate at low duty cycles. Under LPL, every node periodically wakes up to perform CCA. It then stays awake to receive packets if the CCA check detects activity in the wireless channel, or goes back to sleep immediately otherwise. Due to its simplicity and effectiveness, LPL has been a popular approach to energy-efficient MAC protocols in Wireless Sensor Networks (WSNs). A multitude of LPL-based MAC protocols has been developed in recent years [8, 18, 19], and LPL has been implemented by many radio drivers inside sensor operating systems such as TinyOS [1] and Contiki [2].

While the effect of CCA on collision avoidance has been well studied in the literature, its impact on LPL, particularly in noisy environments such as residential and office environments, has received relatively little attention. Applications deployed in noisy wireless conditions are susceptible to frequent *false wakeups*: noise may be detected as legitimate activity on the channel, causing the node to remain awake even when no transmissions occur. False wakeups may significantly increase the duty cycle and energy consumption of the nodes, as shown by our empirical studies in residential environments (see Section 4). This limitation of LPL protocols is becoming increasingly significant as more and more WSNs are being deployed in residential environments, where co-existing wireless devices and electromagnetic equipments cause prevalent and highly variable noise.

To address this important problem, we propose a novel approach that dynamically adjusts the *CCA threshold*, i.e., the energy level threshold used to decide if a channel is active. This approach is motivated by the key observation that nodes may effectively reduce false wakeups by choosing a threshold above the background noise level, but below the level of real transmissions. Specifically, the main contributions of this work are three-fold:

- An empirical study in residential environments that demonstrates the potential benefits of adaptive CCA control based on both normal channel conditions and controlled 802.11n traffic;
- Adaptive Energy Detection Protocol (AEDP), an adaptive protocol that dynamically adjusts a node's CCA

threshold to improve network reliability and duty cycle based on application-specified bounds;

- Discovery of significant shortcomings in the implementation of CCA checks in TinyOS 2.1.1 ² caused by improper selections of key radio parameters, and a systematic methodology to tune these parameters in order to enable efficient CCA checks.

In contrast to previous studies on adjusting the CCA threshold to better avoid collisions in both 802.11 [7] and 802.15.4 [5, 6, 16, 26] networks, this paper investigates the CCA threshold’s role in waking up nodes, with the goal to mitigate the false wakeup problem associated with LPL; to our knowledge, it represents the first systematic study of the CCA threshold’s role in the effectiveness of LPL. The uses of CCA for collision avoidance in CSMA/CA and wakeup in LPL are orthogonal and complementary to each other, as CCA is being used at different times for different goals. Indeed, both forms of CCA adjustment could be deployed simultaneously, by simply maintaining separate thresholds for collision avoidance and LPL.

The remainder of the paper is organized as follows. Section 2 compares our approach with related works. Section 3 describes an overview of LPL. Section 4 presents an empirical study into the effect of noise on LPL behavior, and explores the use of CCA thresholds to control the associated false wakeup problem. Section 5 details the AEDP protocol for dynamically adjusting a node’s CCA threshold in order to minimize false wakeups. Section 6 describes the implementation of AEDP on the TelosB mote platform and analyzes the impact of radio parameters on the effectiveness of CCA. Section 7 presents an empirical evaluation of AEDP in both controlled tests and real-world environments. Finally, we conclude the paper in Section 8.

2. RELATED WORK

Traditionally, CCA functionality has been used in CSMA/CA MAC protocols to avoid collisions on shared wireless channels. A sender performs CCA before transmission. It proceeds with the transmission if the CCA check does not detect channel activity; otherwise it backs off to avoid colliding with an on-going transmission. Numerous studies have explored the impact of the CCA thresholds used for collision avoidance on both 802.11 networks and WSNs [5–7, 16, 26]. Bertocco et al. [5] shows that the CCA threshold is critical, as false negative channel activity detections result in collisions and false positives cause increased latency. Kiryushin et al. [16] studies the real-world impact of CCA thresholds in avoiding packet collisions. Chintalapudi et al. [9] shows that a poor energy detection scheme can lead to significant overhead for listening to the channel and switching the radio between send and receive modes, which may take hundreds of microseconds. Boano et al. [6] shows that tuning the CCA threshold at run time can improve the robustness of existing MAC protocols under interference. Yuan et al. [26] presents that dynamically adjusting CCA threshold can substantially reduce the amount of discarded packets due to channel access failures. Brodsky et al. [7] presents an opposite conclusion based on theories of radio propagation and Shannon capacity and shows that it is possible to choose a fixed CCA

²The shortcomings still exist in TinyOS 2.1.2 officially released on August 20, 2012.

threshold which performs well across a wide range of scenarios since carrier sense performance is surprisingly close to optimal for radios with adaptive bitrate. All these works focus on the impact of CCA on collision avoidance in transmissions rather than its use for wakeup in LPL-based MAC protocols.

In contrast to previous studies on CCA for channel avoidance, this paper investigates the CCA threshold’s role in avoiding the false wakeup problem associated with LPL; to our knowledge, it represents the first systematic study of the CCA threshold’s role in the effectiveness of LPL in achieving low duty cycles in WSNs, especially in noisy environments where traditional LPL protocols are vulnerable to false wakeup problems. Our work is therefore *orthogonal* and *complementary*.

Recently, receiver-initiated MAC protocols have been proposed to avoid the false wakeup problem. Receiver-initiated MAC protocols such as [12, 22] require recipients to transmit probe packets indicating that they are ready for packet reception. As our experiments presented in Section 7.5, AEDP is more energy efficient at low data rates than the state-of-the-art receiver-initiated MAC protocol A-MAC [12], as AEDP avoids the overhead of the probe packets. On the other hand, A-MAC is more energy efficient than AEDP for high data rate applications, where the cost of sending these probe packets are offset by reduced overhead for transmissions. Our work is therefore an alternative sender-initiated approach that is complementary to receiver-initiated MACs for applications with different data rates.

ContikiMAC [11] addresses the false wakeup problem with two targeted optimizations. First, it performs *two* CCA checks spaced slightly apart, allowing it to identify phenomenon too short to be an 802.15.4 transmission. Second, it performs a “fast sleep” optimization that reduces the cost of false wakeups, by detecting patterns of activity and silence which cannot belong to ContikiMAC transmissions. In our testing, we found that our approach’s CCA-threshold-adjustment can effectively avoid false wakeups without these optimizations. Since our approach requires only a *single* CCA check, it induces lower energy cost in low duty-cycle cases where nodes rarely need to wakeup to receive packets. Nevertheless, these approaches are orthogonal, and in particularly challenging environments could be combined to reduce both the likelihood and the energy overhead of false wakeups.

There has been increasing interest in studying the impact of interference on WSNs and enhancing the robustness of MAC protocols in noisy environments. Srinivasan et al. [21] examines the packet delivery behavior of two 802.15.4-based mote platforms, including the impact of interference from 802.11 and Bluetooth. Liang et al. [17] measures the impact of interference from 802.11 networks on 802.15.4 links, proposing the use of redundant headers and forward error correction to alleviate packet corruption. These studies focus on improving the reliability of transmission and do not deal with the false wakeup problem to improve energy efficiency.

3. OVERVIEW OF LPL

Low power listening (LPL) is a common MAC-layer technique for reducing energy consumption in WSNs [19]. Under LPL, nodes periodically wakeup to perform CCA, i.e., to briefly sample the wireless channel for activity. If energy

is detected on the channel, the node remains awake in order to receive a packet (or until some timeout). Otherwise, the node quickly goes back to sleep. To minimize overhead when the network is idle, these periodic wakeups are not synchronized across nodes: that is, the recipient knows the recipient’s wakeup interval but not its wakeup time. Accordingly, before transmitting a packet, the transmitter sends a preamble stream at least as long as the recipient’s wakeup interval; this ensures that the recipient will sample the channel during the preamble. After the preamble, the sender and recipient exchange data packets.

Later LPL-based MAC layers such as X-MAC [8] and BoX-MAC-2 [18] modify this approach by inserting destination address information and periodic gaps in the preamble stream. When a node wakes up, it may decode the destination address and see if it is the packet’s intended recipient. If so, it uses the gaps in the preamble to send an acknowledgment to the sender, which will in turn immediately transmit the payload. If not, the node may go back to sleep immediately. These enhancements significantly reduce the cost of waking up for a packet intended for another node, while also reducing the average cost of unicast packet transmissions by half. BoX-MAC-2 further refines this approach by transmitting the entire data packet in place of the destination address, eliminating the need to explicitly exchange the payload after the recipient has ACKed the preamble.

Quickly and accurately assessing whether the channel is active is a critical component of a LPL-based MAC layer. Modern radios, including all 802.15.4-compliant hardware [15], provide CCA functionality that assists with this procedure. A common method for radios to implement CCA is to provide a digital readout (often a dedicated pin) indicating whether the channel’s energy level currently exceeds some threshold. This particular implementation, known as *energy detection*, is commonly found in low-power radios such as the Chipcon CC2420 and has been identified as a critical feature for WSN hardware design [13]. After waking up the radio, the microcontroller may sample the CCA pin in a tight loop; the node remains awake for packet reception if some minimum number of samples are positive.

4. EMPIRICAL STUDY

This section describes a series of empirical studies that provide the motivation and insights for the design of AEDP. We first measure the false wakeup problem in office and residential environments, followed by a systematic study on the impact of CCA’s energy detection threshold on wakeups in LPL.

4.1 Effects of Wireless Noise

Existing literature on LPL-based MAC layers emphasizes the ability to run applications at an extremely low duty cycle, sometimes as low as 1% [19], in exchange for moderately increasing the cost of packet transmissions. This tradeoff makes LPL well-suited for applications with low-to-moderate data rates. However, noise from other wireless devices can have a dramatic (and often unanticipated) impact on nodes’ duty cycle, significantly reducing system lifetime.

Radios based on the 802.15.4 standard operate in the unlicensed 2.4 GHz band shared by many other devices. Energy detection simply looks for the presence of *some* signal on the wireless channel; it does not distinguish between the system’s own traffic and noise from other devices. To illus-

trate how a false-negative wakeup can considerably increase the cost of a CCA check, we deployed a TelosB mote [10] running TinyOS 2.1.1 [1] in an office environment. The TelosB mote was configured to use the BoX-MAC-2 LPL-based MAC layer, TinyOS’s *de facto* standard LPL implementation. BoX-MAC-2 was in turn configured with a wakeup interval of 2 seconds: i.e., the motes sleep for 2 seconds between sampling the channel for activity. In order to capture the effects of wireless noise, we configured the CC2420 radio to use channel 18, which overlaps with a campus-wide 802.11g network. All other MAC layer and radio parameters were left to their respective defaults.

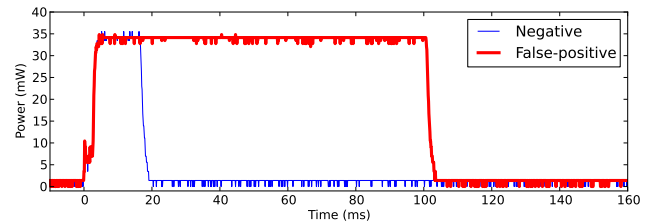


Figure 1: Oscilloscope traces comparing a TelosB node’s energy consumption during a negative (idle) and false-positive (detected) energy detection check.

Figure 1 shows the energy consumption of this mote when performing a single energy detection check, as captured with an oscilloscope. When the channel is idle, the radio is powered on for 19.0 ms; in contrast, when the channel is occupied, the false wakeup causes the radio to remain powered on for 103.4 ms until it eventually times out. Similar results were observed in [12], which found that false wakeups increased the current consumption of a CCA check by 17.3 \times .

An equally important question is how often wireless noise will cause these false wakeups to occur in real-world environments. To measure this phenomenon, we deployed four pairs of TelosB motes on orthogonal channels (11, 16, 21, and 26, respectively) in five different apartments located in different neighborhoods in St. Louis. The motes were deployed for 24 hours in each apartment during the residents’ normal activities. One mote in each pair was configured to transmit 1 packet every minute, and the BoX-MAC-2 MAC layer configured with a wakeup interval of 2 seconds. We augmented TinyOS’s CC2420 radio stack to track the result of each CCA check and the radio “on time”, i.e., the cumulative total time the radio was active during the entire experimental run. The latter data was in turn used to compute each mote’s duty cycle. For the purposes of this experiment, the mote’s onboard CC2420 was again configured with the hardware-default CCA behavior, setting its CCA pin based on an energy threshold of -77 dBm.

Figure 2 plots the false wakeup rate (the proportion of CCA checks resulting in wakeup but no packet reception) of each node in the experiment. From the receiver’s wakeup interval and the sender’s data rate, we expect a nominal duty cycle of 0.17%. However, the false wakeups caused by environmental noise result in substantially higher duty cycles, with an average duty cycle of 1.4% across all four tested channels in all five apartments. In the two worst cases — channel 16 in apartment 5 and channel 26 in apartment 2 — false wakeup rates of 45% resulted in greatly inflated duty cycles of 2.8%.

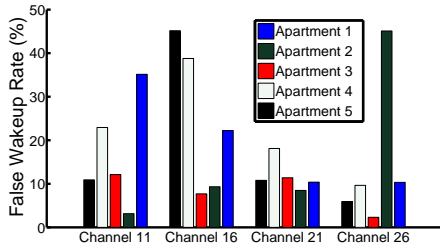


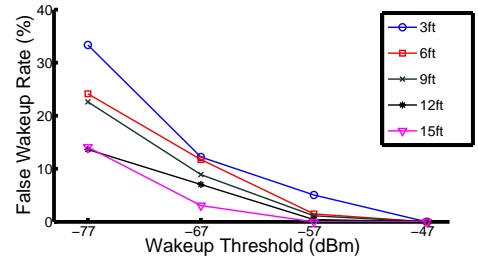
Figure 2: The false wakeup rate of each recipient mote in each apartment.

4.2 Effects of CCA Threshold

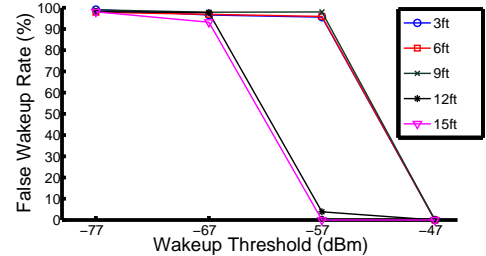
We propose to address the false wakeup problem by adjusting the CCA threshold: that is, the specific energy level used as a binary threshold to determine whether a node should remain awake. In the context of LPL, setting the CCA threshold too low will cause nodes to wakeup to receive non-existent packets. Setting the threshold too high may cause nodes not to wakeup during transmissions, forcing the sender to repeatedly retransmit. We note that adjusting the CCA threshold for LPL has *no* effect on the receiver’s ability to decode packets, so long as the threshold is low enough to wakeup the receiver. Hence, link reliability will only be affected if the threshold is high enough to cause a false-negative energy detection (i.e., a node fails to stay awake to receive a legitimate packet).

As discussed earlier, the CCA threshold also plays a role in the context of collision avoidance. However, adjusting the CCA threshold has a different effect in the context of collision avoidance, where it directly affects the *sender* rather than *receivers*. Setting the threshold too low encourages spurious backoffs, while setting the threshold too high may introduce packet losses from otherwise-avoidable collisions. To distinguish the CCA threshold used by the receiver for LPL from the CCA threshold used by the sender in CSMA/CA, we henceforth refer to the former in this paper as the *wakeup threshold*. This paper focuses on reducing false wakeups by manipulating the wakeup threshold used for LPL. We do not change the CCA threshold used for transmission, an important but orthogonal problem that has been well-studied in literature.

We perform a set of controlled test in an office environment to investigate the potential energy savings from adjusting the wakeup threshold, we deployed five groups of four TelosB motes on channel 16 at varying distances (3–15 ft) from a pair of 802.11n devices (access point+MAC pro laptop) operating on 2.4 GHz band that overlaps with 802.15.4. Each experimental run was carried out for one hour; as before, BoX-MAC-2 was configured with a wakeup interval of 2 seconds. In contrast to the previous experiments, which used the radio-default CCA threshold of -77 dBm, each mote in a group was configured to use one of four different thresholds (-77 , -67 , -57 , and -47 dBm). Signal generated by motes may become part of the background noise when its strength is lower than recipients’ CCA threshold. We intentionally stop motes from generating real transmissions in this set of test, thus we can treat the total wakeup rate as the false wakeup rate.



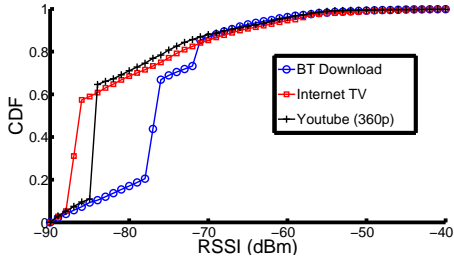
(a) False wakeup rate under office occupants’ normal activities.



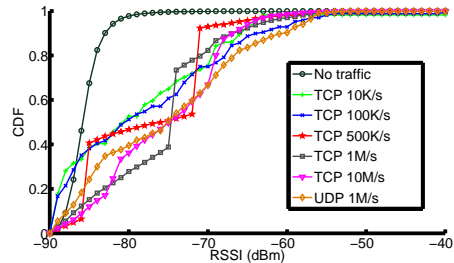
(b) False wakeup rate under controlled (5 Mbps UDP) 802.11n traffic.

Figure 3: The effects of tuning the CC2420’s wakeup threshold on the motes’ false wakeup rate, subject to office occupants’ normal activities and controlled 802.11n traffic. The motes were located 3–15 ft away from the 802.11n router, and were configured to use a threshold ranging from -77 to -47 dBm.

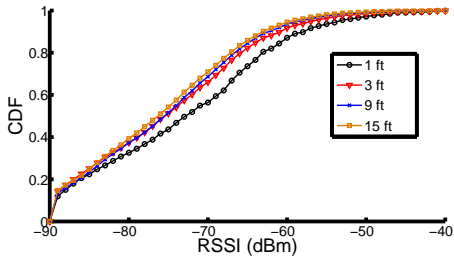
Figure 3(a) plots the recipients’ false wakeup rate under the office occupants’ normal activities in real-world environment. Figure 3(b) plots the false wakeup rate when using LanTraffic V2 [3] to generate a controlled stream of 5 Mbps UDP traffic through the pair of 802.11n devices. Two important conclusions may be drawn from these figures. First, tuning the wakeup threshold provides a powerful opportunity for conserving energy. We observe that the false wakeup rate drops dramatically when increasing the threshold from the radio default of -77 dBm. Under real-world activity as shown in Figure 3(a), the default threshold incurs a false wakeup rate of 14–33%. In comparison, this rate may be reduced to 3–12% by moderately increasing the threshold by 10 dBm, or to 0% by increasing the threshold by 30 dBm. The effects of tuning the threshold are even more pronounced under the higher-bandwidth controlled experiments, as shown in Figure 3(b). At a threshold of -77 dBm, the nodes experience a false wakeup rate no lower than 97.8%, regardless of distance from the pair of 802.11n devices. This rate drops to 0–4% for two of the distances at a threshold of -57 dBm, and to 0% for all distances at a threshold of -47 dBm. Second, the “best” wakeup threshold is highly dependent on external factors such as the 802.15.4 nodes’ vicinity to other devices, and the other devices’ usage patterns and signal strength. Comparing Figures 3(a) and 3(b), we see that increasing the threshold from -77 dBm to -67 dBm significantly reduces the false wakeup rate under normal activities. However, under a sustained 5 Mbps UDP stream, a comparable threshold increase has virtually no impact on the false wakeup rate.



(a) Background noise collected while using different applications over 802.11n.



(b) Background noise collected during controlled 802.11n traffic.



(c) Background noise collected while using different distance from the pair of 802.11n devices.

Figure 4: The CDF of background noise as measured by a nearby TelosB mote, subject to real-world and controlled 802.11n traffic patterns. Except (C), the mote was located 10 ft away from the pair of 802.11n devices.

To further illustrate the challenge of picking a proper wakeup threshold, we deployed a TelosB motes on channel 26 at a distance of 10 ft from the same pair of 802.11n devices. The mote was configured to sample the Received Signal Strength Indication (RSSI) every 2 seconds, providing an estimate of the wireless noise on the channel. Figure 4(a) plots the CDF of the motes' RSSI readings under several different applications; Figure 4(b) plots the same under various types of controlled 802.11n traffic, while Figure 4(c) plots the same under different distance from the pair of 802.11n devices. From the figures, we can see that noise varies from application to application and over time for a given application depending on the distance from interference source.

Hence, picking an appropriate wakeup threshold is not simply a matter of choosing a more aggressive default setting. The minimum threshold needed to avoid noise varies from setting to setting, and even over time depending on the residents' activities. Moreover, selecting too high of a threshold will intuitively cause the receiver to stop waking up for legitimate transmissions, decreasing network reliability.

5. PROTOCOL DESIGN

In this section, we present the design of our *Adaptive Energy Detection Protocol (AEDP)* protocol. At a high level, AEDP tries to meet application-specified constraints on network reliability and wakeup rate. The desired network reliability is specified by the desired ETX, $ETX_{threshold}$, where ETX is the expected number of transmissions needed to successfully send a packet to its destination. The desired wakeup rate, $WR_{threshold}$ can be determined based on the application data rate (and hence the corresponding *true* wakeup rate) plus a small margin for false wakeups allowed by the application. When it is not possible to meet both constraints, network reliability takes precedence, as it is typically more critical than lifetime constraints. We set a default value of $ETX_{threshold}$ to be 5 and a default value of $WR_{threshold}$ to be 5 times of data rate according to the typical low data rate home automation systems.

AEDP maintains three variables at run time: ETX , WR , and WR_L . ETX is the average ETX value over a sliding window (default window size is 15 minutes). WR is the wakeup rate within the same sliding window. WR_L is the cumulative wakeup rate over the whole application lifetime. Note that WR_L reflects the long-term wakeup rate that affects the battery life of the node.

At runtime, AEDP periodically updates these three variables ETX , WR , and WR_L and compares their values against $ETX_{threshold}$ and $WR_{threshold}$. It then computes a new wakeup threshold T based on four different cases, described below.

- **Case 1:** ETX exceeds $ETX_{threshold}$. AEDP attempts to quickly recover by significantly reducing the wakeup threshold. This policy reflects the fact that network reliability constraints are typically more critical than lifetime constraints.
- **Case 2:** ETX meets $ETX_{threshold}$ but WR exceeds $WR_{threshold}$. This case indicates that the current wakeup threshold is too low to achieve the desired wakeup rate. AEDP increases the wakeup threshold by a small amount ΔT to try to meet the application's bound on

wakeup rate. The default value of the tuning step ΔT is set to be 2 dBm.

- **Case 3:** ETX , WR , and WR_L all meet their respective constraints. This case indicates that the current wakeup threshold is meeting the application’s constraints, both in this period and over the application’s lifetime. AEDP aims to find the minimum threshold that does so, as lower wakeup thresholds are potentially more robust to changes in topology and signal strength. Hence, AEDP decreases the wakeup threshold by ΔT .
- **Case 4:** ETX and WR meet their constraints but WR_L does not. Here, AEDP takes no action. Since WR is below $WR_{threshold}$, the wakeup threshold is high enough to meet the application’s wakeup rate constraint in the short term. However, WR_L has still not met the application’s constraint over the long term, so AEDP will not yet start to reduce the wakeup threshold.

In all cases, AEDP constrains the wakeup threshold T to a range $[T_{min}, T_{max}]$. Reducing T too much will cause the node to always be awake, while increasing T too much will cause packet loss (increased ETX). AEDP sets T_{min} to be the noise floor to avoid sustained wakeups, and sets T_{max} to be the minimum Received Signal Strength (RSS) of incoming links, since our experimental results have shown that link reliability degrades heavily when T exceeds the RSS of incoming link (See Appendix). To accommodate topology changes, AEDP periodically resets the wakeup threshold to T_{min} for several periods (a default value of 5 wakeup intervals) enabling node to establish new incoming links with RSS lower than T .

AEDP has several key design features based on the observations in our empirical study. First, AEDP adaptively adjusts energy detection threshold based on changes in network reliability (specifically, ETX) observed at runtime. Second, AEDP performs its computations based solely on local state (WR , WR_L , and ETX), requiring no additional transmissions between sender and receiver. Third, AEDP is a lightweight protocol that only piggybacks a single byte (used to measure ETX) in each existing packet transmission, and introduces no other traffic of its own.

6. IMPLEMENTATION

In this section, we discuss our implementation of AEDP on TinyOS 2.1.1. We first describe the software architecture used by AEDP. We then discuss several key radio parameters that affected the energy efficiency of LPL, and present a methodology for picking these parameters appropriately.

6.1 AEDP Architecture

We implement the AEDP algorithm as a layer situated between the application and MAC layers. This layer consists of three important components. The *Wakeup Rate Monitor* component tracks the wakeup rate WR and computes the cumulative wakeup rate WR_L . The *Link Estimator* component measures the ETX of incoming packets using sequence numbers in each packet, and computes the average ETX value (ETX) over a sliding window. The Link Estimator also measures the RSS of incoming packets, using the minimum average RSS value of all incoming links as the bound

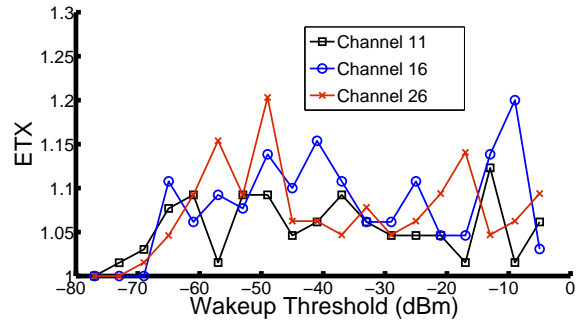


Figure 5: The relationship between wakeup threshold and ETX in the default TinyOS CC2420 stack.

T_{max} . The *CCA Control Engine* component computes and sets the wakeup threshold based on the values ETX , WR and WR_L .

AEDP requires several modifications to the radio stack to support its operations, as listed below. For the purposes of this implementation, we have performed these modifications on TinyOS 2.1.1’s default CC2420 + BoX-MAC-2 stack.

First, we add a *PacketInfo* interface between the MAC layer and Link Estimator to expose the ETX and RSS values of each incoming packet. The Link Estimator buffers the values in sliding windows, calculating the average ETX and RSS values for the variables ETX and T_{max} respectively.

Second, we augment the radio core to count wakeup events. This counter is exposed to the Wakeup Rate Monitor through the *WakeupCounter* interface and used to compute the values of WR and WR_L .

Finally, we add a *CCAcontrol* interface to the radio core to expose the radio’s hardware CCA threshold setting. On the CC2420, this is implemented by writing the new threshold to the radio’s `CCA_THR` register, plus a 45 dBm offset specified by the datasheet [23]. The CCA Control Engine uses this interface to set the newly-computed wakeup threshold T .

6.2 System Parameters

When testing our first implementation of AEDP on the TelosB mote, we were initially surprised to discover that increasing the wakeup threshold had little impact on network reliability. Figure 5 plots the relationship between the wakeup threshold and ETX that we observed in our initial testing, using three different channels and a wide range of threshold values. We initially expected that an excessively high threshold would cause significant packet loss, and a high enough threshold would prohibit the node from receiving packets at all (due to never waking up from sleep). However, in practice, we observed that an overly aggressive threshold only increased the number of retransmissions by a maximum of 20%. Indeed, the node still received packets after setting the threshold to the radio maximum of 82 dBm, or even after modifying the CC2420 stack to *always* put the radio back to sleep regardless of the energy detection result.

From these results, we hypothesized that the radio was fully receiving and decoding entire packets during the CCA check itself. TinyOS’s implementation of BoX-MAC-2 on the CC2420 detects energy by sampling the CCA pin up to 400 times in a tight loop. Modern packet-based radios like the CC2420 are designed to fully decode packets without

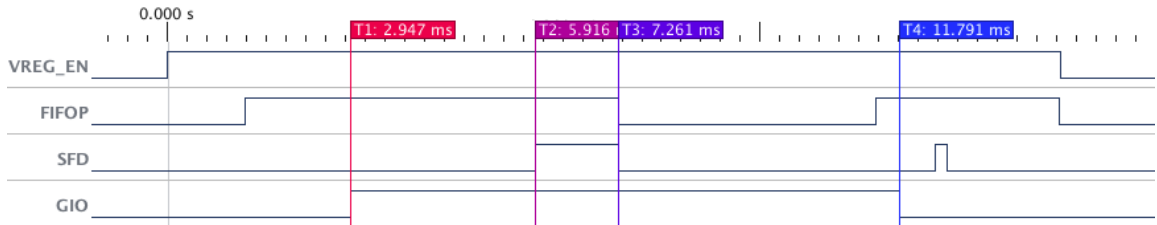


Figure 6: A logic analyzer trace demonstrating the CC2420 fully decoding a packet during the energy detection check. The microcontroller uses the VREG_EN pin to control the CC2420’s power state. The CC2420 uses the SFD and FIFOP pins to signal the beginning (T2) and end (T3) of packet reception, respectively. The GIO pin indicates the duration of the check (T4–T1).

the microcontroller’s intervention, and could decode packets while the microcontroller is occupied by polling the CCA pin.

We confirmed this hypothesis using a logic analyzer to trace the sequence of events inside the radio hardware and radio stack. Figure 6 presents a sample trace that we captured with the CC2420 configured to use the maximum threshold³. At 0 ms, the radio stack begins sampling the wireless channel by powering on the CC2420. The CC2420 is fully powered on at T1 = 2.947 ms, and the radio stack starts energy detection. At T2 = 5.916 ms, the CC2420 signals the beginning of a packet reception; at T3 = 7.261 ms, the CC2420 signals that the packet is fully decoded. The radio stack will not finish energy detection until T4 = 11.791 ms. Indeed, the duration of this check (8.844 ms) is much longer than the on-air time of an 802.15.4 packet (0.59 – 4.24 ms in lab experiments, depending on payload size).

The apparent cause for this lengthy check is a long ACK delay built into TinyOS’s CC2420 driver. After transmitting a packet, the driver waits up to 8 ms for an ACK packet. In our own measurements, this resulted in BoX-MAC-2 leaving the channel idle for 8.3 ms between retransmissions.

In principle, an ACK delay of this length is unnecessary. From the 802.15.4 specification, we can derive a tight bound of 544 μ s on the ACK delay. (Specifically, the recipient must transmit an ACK exactly 192 μ s after decoding the incoming packet’s last bit, and transmitting the ACK packet takes 352 μ s at 802.15.4’s 250 kbps data rate [12, 15].) However, TinyOS disables the CC2420’s hardware auto-acknowledgement feature due to concerns over its reliability [4]. Consequently, packets must pass partway through the recipient’s radio stack before they are acknowledged, adding significant delay.

Nevertheless, we believe that the default ACK delay is overly conservative. In a microbenchmark experiment, we transmitted packets between a pair of TelosB motes with hardware auto-acknowledgement disabled. The transmitter requested an ACK for each transmission, and recorded the delay between finishing a transmission and receiving the corresponding ACK. Out of 2000 transmissions, the transmitter observed a mean delay of 2.2 ms and a maximum delay of 2.4 ms.

³For illustrative purposes, we modified BoX-MAC-2 to mark the duration of the energy detection loop using a GIO pin, and to disable a code branch that short-circuits the loop when the radio starts receiving a packet. We have verified that the CC2420 will still fully decode packets during energy detection, even without these modifications.

Driver	ACK delay
CC2420(cc2420 driver)	8 ms
CC2420(cc2420x driver,most platforms)	1 ms
CC2420(cc2420x driver,micaz platforms)	0.8 ms
CC2520 (most platforms)	1 ms
CC2520(sam3s_ek platform)	0.8 ms
RF230	1 ms
IEEE 802.15.4 specification	0.544 ms
TelosB (lab measurements)	2.4 ms

Table 1: The ACK delays used by various 802.15.4 radio drivers in TinyOS, the ACK delay derived from [15], and the actual ACK delay measured on a TelosB.

This result indicates that an 8 ms ACK delay, and the associated 8.8 ms energy detection length, is excessive. The length of this check contradicts the need for a short, inexpensive energy detection, and arguably even renders the entire check ineffective. From the 20% ETX penalty we observed in our testing, it would have been nearly as effective to simply leave the radio on for 8.8 ms, and ignore the energy detection result. Doing so would have had only a small impact on network reliability, in exchange for *never* incurring a false wakeup.

Instead, for the purposes of implementing and evaluating AEDP, we opt to retain the check but reduce the CC2420 driver’s ACK delay to 2.8 ms (2.3 ms + 0.5 ms guard space). We accordingly modify BoX-MAC-2 to poll the CCA pin up to 115 times, reducing the energy detection duration from 8.8 ms to 2.9 ms. In general, the duration of CCA polling must be longer than the ACK delay to avoid false negatives in energy detection which can heavily worsen the performance.

As we show in Section 7, this modification alone has the effect of significantly reducing the motes’ duty cycle, simply by reducing the cost of energy detection to a fraction of its default length.

Although this modification is specific to the particular radio stack used, it emphasizes the need for a general methodology — such as the analysis performed above — to tune these key radio and MAC layer parameters. Indeed, as shown in Table 1, TinyOS employs three different ACK delays on the sender side, depending on the combination of radio driver, radio stack, and underlying mote platform. None of these three different delays is consistent with the theoretical ACK turnaround time from the 802.15.4 standard, or

with the actual turnaround time measured on the TelosB. Besides energy efficiency, this inconsistency raises concerns about basic interoperability.

7. EVALUATION

To validate the efficiency of our approach in reducing false wakeup rates, we performed a series of controlled experiments and real-world experiments. (1) We first evaluate the capability of AEDP to effectively converge to the desired wakeup threshold. (2) We then performed an experiment where additional transmitters were added to the network at runtime to test AEDP’s resilience to network changes. (3) We evaluate AEDP’s impact on duty cycles at the link level, and compare AEDP’s performance against LPL configurations in a testbed we deployed in a 3-floor apartment building. (4) We compare AEDP against A-MAC, a state-of-the-art receiver-initiated MAC protocol under different data rates. (5) Finally, we evaluate the impacts of AEDP on multi-hop data collection by running AEDP with CTP in a 55-node testbed in an academic building.

In all experiments, we deploy our benchmark applications on top of TelosB motes running the TinyOS 2.1.1 operating system. BoX-MAC-2 is configured with a wakeup interval of 2 seconds: i.e., the motes sleep for 2 seconds between sampling the channel for activity. We use a data rate of 1 packet/5 minutes⁴ for all evaluations except the one in Section 7.5, where we evaluate the performance of AEDP under different data rates.

We emphasize that our experiments changed only the CCA threshold used for wakeup and did not change the threshold used for collision avoidance; hence, improvements in duty cycle are attributed to a reduction in false wakeups rather than retransmissions.

7.1 Self-tuning Wakeup Threshold

We first test the capability of AEDP to automatically adjust its wakeup threshold. For this experiment, we deployed a pair of motes with AEDP on channel 16. We also deployed an 802.11n access point and a laptop producing 1 Mbps of UDP traffic on 802.11 channel 6, which overlaps with 802.15.4 channel 16. We performed two experimental runs: to vary the impact of the interfering 802.11 network on the mote pair, the distance between the mote pair and the 802.11n devices was 10 ft during the first run, and increased to 30 ft for the second run.

Figure 7 illustrates AEDP reactively changing the wakeup threshold based on runtime conditions. During the first experimental run, the receiver mote quickly increases the wakeup threshold to -56 dBm to avoid false wakeups introduced by the nearby 802.11 interferer. At this point, the mote is still unable to meet the application-specified duty cycle, and hence the threshold remains at about -55 dBm for the remainder of the experiment. In the second experimental run, the receiver mote likewise quickly increases the wakeup threshold to -54 dBm. At this point, because the mote is located further away from the interferer, it is able to meet the application-specified duty cycle; hence, it gradually decreases the wakeup threshold in increments of 2 dBm.

⁴The data rate is chosen according to the typical sampling rate of home automation systems (for example, 1 temperature reading every 5 minutes is sufficient for an HVAC system to control ambient temperature).

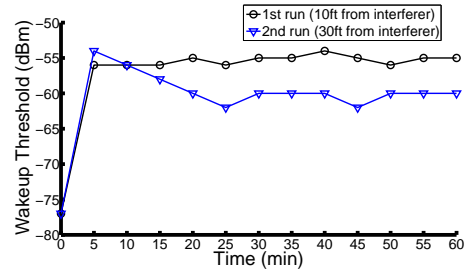


Figure 7: AEDP adapting the wakeup threshold over time.

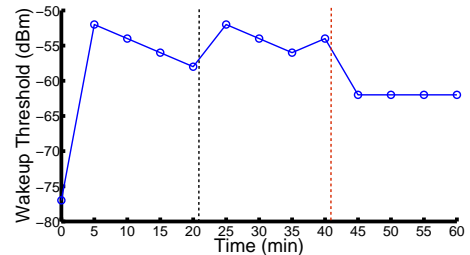


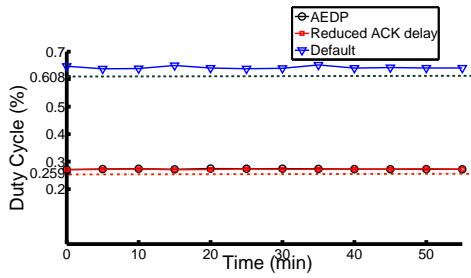
Figure 8: AEDP adapting the wakeup threshold over time when new nodes join the network. A second transmitter joined into the network at 21 minutes (vertical black line) and a third at 41 minutes (vertical red line).

AEDP eventually settles on a threshold between -60 and -62 dBm that closely matches the requested duty cycle, where it remains for the remainder of the experiment.

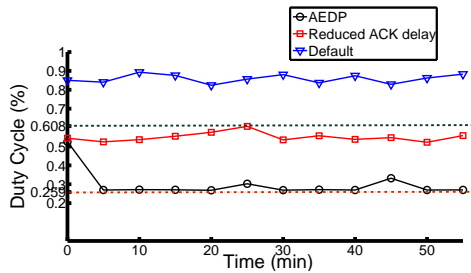
7.2 Adaptation to Network Changes

To test AEDP’s resilience to network changes, we performed an experiment where additional transmitters were added to the network at runtime. We initially deployed a single transmitter mote and a single receiver mote. A second transmitter was added to the network 21 minutes into the experiment, and a third was added at 41 minutes. All three transmitters were configured to send packets to the single receiver node, where we instrumented AEDP to record its wakeup threshold over time.

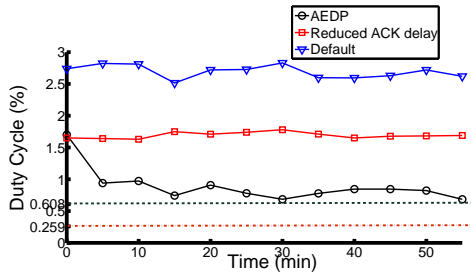
Figure 8 illustrates how AEDP adapted the receiver’s wakeup threshold over the course of the experiment. In order to reduce the false wakeup rate, AEDP quickly increased the wakeup threshold to -52 dBm; this closely matches the -50 dBm RSS of the first transmitter. After AEDP reached its objective false wakeup rate, it began steadily decreasing the threshold until the second transmitter joined at 21 minutes. The second transmitter’s signal strength was slightly higher (-46 dBm) than the existing transmitter. Hence, AEDP responded to the new node by increasing the threshold to -52 dBm, slightly lower than the minimum of both transmitters, and again gradually decreased the threshold over time. At 41 minutes, the third transmitter joined with a significantly lower signal strength at the receiver (-60 dBm) than the previous two transmitters. Benefiting from the periodic wakeup threshold reset process mentioned in Section 5, AEDP adapted by rapidly dropping the wakeup threshold to -62 dBm, again slightly below the minimum signal strength



(a) In a clean environment.



(b) In a residence with residents' normal activities.



(c) In a lab stress test with generated 802.11n interference.

Figure 9: Duty cycle under minimum interference, normal residential activities, and sustained interference. Horizontal lines indicate the theoretical optimal duty cycles of 0.259% (AEDP and reduced-ACK configurations) and 0.608% with default radio configuration.

of all the transmitters. These results demonstrate AEDP dynamically adjusting the wakeup threshold to successfully accommodate network topology changes.

7.3 Impact on Duty Cycles

To explore AEDP's impact on duty cycles, we deployed a pair of motes with a modified radio stack to record the *radio on time* — i.e., the cumulative time the radio was active — on each mote. The precise duty cycle is hence derived from the radio on time and the experiment's length.

We first deployed the pair on channel 26 in an office environment, which we confirmed to be clean with a Wi-Spy spectrum analyzer [24]. We performed experimental runs, for 60 minutes each run, once with the default BoX-MAC-2 configuration and once with AEDP. To isolate the effects of the reduced ACK delay (discussed in Section 6.2) from AEDP's wakeup threshold tuning, we performed a third experimental run which reduced the ACK delay but was otherwise identical to the default BoX-MAC-2 stack.

Figure 9(a) presents the duty cycle under all three experimental runs, broken down into 5 minute windows. In each 5-minute window, the default BoX-MAC-2 configuration activates the radio with an average duty cycle of 0.64%. AEDP consistently reduces this duty cycle over the entire experimental run, by an average of 57.48%. In this clean environment, the false wakeup rate is very low; hence, AEDP achieves a duty cycle within 99.7% of the reduced-ACK configuration.

For comparison, we also plot the theoretical optimal duty cycle for both ACK delay configurations. Specifically, at a data rate of 1 packet/5 min and a wakeup interval of 2 s, the optimal duty cycle is $149 * T_{idle} + (T_p + T_i)/2 + T_p + T_d$, where T_{idle} is the time the radio is active when no energy is detected (11.5 ms under the default configuration, or 4.5 ms with the reduced ACK delay); T_p is the time needed to receive a packet (4.24 ms); T_i is the gap between packets (8.3 ms under the default configuration, or 2.8 ms with the reduced ACK delay); and T_d is the time the radio remains active after receiving a packet (100 ms). Because interference was limited, all experimental runs remained within 7% of their respective optimal duty cycles.

To evaluate AEDP's performance under a more typical deployment, we repeated this experiment in a residential setting. This experiment was carried out under normal wireless condition with residents' regular wireless activity. The mote pair is configured to use channel 16, which overlaps with the residents' 802.11g network. Figure 9(b) plots the results under this experimental setup. We observe that the adjusted ACK delay is responsible for a significant reduction in radio usage, with the average duty cycle in each 5-minute window dropping from 0.86% to 0.55%. However, in the face of typical wireless noise, AEDP's wakeup threshold adjustment has a significant impact on duty cycle. AEDP reduces the duty cycle to an average of 0.30%, resulting in a savings of 45.5% over the tweaked radio stack and 65.1% over the default radio configuration.

Because AEDP is largely able to avoid false wakeups, it comes within 15.8% of the theoretically optimal duty cycle. In contrast, the default and reduced-ACK stacks achieves a duty cycle 41.4% and 112.4% higher than their respective optimal duty cycles.

As a stress test, we repeated the experiment once more in a lab setting under controlled interference, in the form of a laptop and an access point, located 10 ft from the mote pair, generating 1 Mbps UDP traffic over an overlapping 802.11n channel 6, which overlaps with 802.15.4 channel 16.

Figure 9(c) plots the duty cycle under this controlled experiment. Due to the persistent source of interference, the default stack has an average duty cycle of 2.69% while the reduced-ACK stack has an average duty cycle of 1.69%. In contrast, AEDP achieves a duty cycle of 0.89%, a 47.3% reduction over the reduced-ACK stack and 66.9% over the default stack.

Owing to the challenging nature of the wireless environment, all three stacks perform several times worse than their theoretical optimal duty cycles. However, AEDP comes within the closest of its optimal duty cycle: 244% higher than optimal, compared to 342% for the default stack and 552% for the reduced-ACK stack.

7.4 Effects of Signal Strength

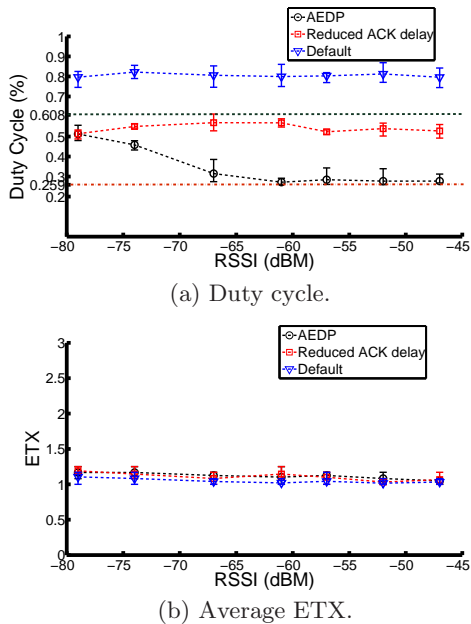


Figure 10: AEDP’s performance on links with diverse signal strengths.

We explored AEDP’s performance on a diverse set of links by selecting 30 links at random from the 380 links detected in a testbed we deployed in a 3-floor residential apartment building. This experiment was carried out under normal wireless condition with four residents’ regular wireless activity. As with the previous experiment, we performed three runs, for 60 minutes each experimental run: one with the default LPL configuration, one with a reduced ACK delay, and one with AEDP.

For the purposes of presentation, we group the 30 links into 7 buckets based on their signal strength, using buckets 5 dBm wide. As shown in Figure 10(a), these links show highly diverse RSS at their respective receivers. For the strongest links ($RSS \in (-65, -45]$), AEDP achieves a duty cycle of 0.28%, close to the theoretical minimum of 0.259%. This represents a 40.3% reduction over the reduced-ACK configuration and 65.1% over the default LPL configuration.

AEDP shows a more moderate — but still significant — improvement in duty cycle on intermediate links ($RSS \in (-75, -65]$). For these links, AEDP achieves a 31.2% reduction in duty cycle over the reduced-ACK configuration and 52.6% over the default LPL configuration.

For the links with the lowest signal strength ($RSS \leq -75$), the RSS is already close to the radio stack’s default wakeup threshold of -77 dBm. AEDP cannot adjust the wakeup threshold below the signal strength, since it sets T_{max} to be the minimum RSS of incoming links to avoid sacrificing network reliability. Hence, AEDP’s 35.7% reduction in duty cycle is attributable only to the reduced ACK length.

As shown in Figure 10(b), the reduced-ACK configuration and AEDP introduced a small number of false-negative energy detection checks which were not experienced under the default stack, since the number of CCA pin polling was reduced from 400 times to 115 times, as discussed in Section 6.2. The reduced-ACK configuration consequently had a 5.5% increase in average ETX (from 1.05 to 1.11) and

AEDP had a 6.7% increase in average ETX (from 1.05 to 1.12). The slight increases in average ETX are in exchange for a proportionally much-larger reduction in duty cycle.

We note that links with the lowest signal strength tend to be highly bursty; while productive for routing, they must be used opportunistically. While AEDP will neither help nor hurt when such links exist, by their nature this will only happen for short bursts during the application’s lifetime. During the periods where moderate-to-strong links are used for routing, AEDP will dynamically increase the wakeup threshold, resulting in significant energy savings.

7.5 Comparison with A-MAC

Receiver-initiated MAC protocols [12, 22] avoid the false wakeup problem by transmitting probe packets when nodes are ready to *receive* data, eliminating the need for recipients to actively sample the channel. Although AEDP and receiver-initiated MAC protocols approach the false wakeup problem from different directions, they share the same goal of extending network lifetime by reducing duty cycle in the face of noisy wireless channels. To understand the effectiveness of these two approaches, we performed a set of experiments comparing AEDP’s performance with that of A-MAC, a state-of-the-art receiver initiated MAC protocol [12]. For this set of experiments, we choose the same set of links from the residential testbed used in Section 7.4, and configured the transmitters to transmit at data rates ranging from 1 packet/2 s to 1 packet/600 s. We performed each experimental run twice, once with AEDP and once with the A-MAC implementation provided by the authors of A-MAC [12]. A-MAC’s radio stack was instrumented to record the radio on time, but was otherwise set to its default configuration. For fairness, we used the *default* parameters for both BoX-MAC-2 in TinyOS 2.1.1 [1] and A-MAC provided by the authors [12]. The only change we made for BoX-MAC-2 is reducing the ACK delay because of the implementation flaw discussed in Section 6.2.

As shown in Figure 11(a), at low data rates (Inter Packet Interval (IPI) within [300, 600] s) AEDP leads to lower duty cycles than A-MAC. For instance, AEDP achieves an average duty cycle of 0.338%, representing a 41.5% reduction over A-MAC (0.578%) when IPI is 300 s. AEDP and A-MAC achieve similar duty cycles at intermediate data rates (IPI within [100, 200] s). In contrast, at high data rates (IPI ≤ 100 s), AEDP leads to a higher duty cycle than A-MAC. For instance, with an IPI of 30 s AEDP achieves an average duty cycle of 0.803%, which is 24.1% higher than A-MAC (0.647%). As shown in Figure 11(b), AEDP introduced a small number of false-negative energy detection checks leading to an up to 16.7% increase in average ETX (from 1.000 to 1.166 when IPI is 400 s) in exchange for a proportionally much-larger reduction in duty cycle at low data rates.

The protocols’ respective advantages at different data rates may be understood by analyzing their respective strategies. Under LPL, senders repeatedly transmit long preambles indicating that they are ready to send data; recipients periodically sample the channel, and turn on the radio if energy is detected. Under receiver-initiated MACs like A-MAC, recipients periodically broadcast beacons announcing that they are ready to receive data; senders keep their radios on waiting for the recipient’s beacon, and then immediately ACK it. In principle, receiver-initiated MACs replace LPL’s short channel sampling with an entire transmission plus a short

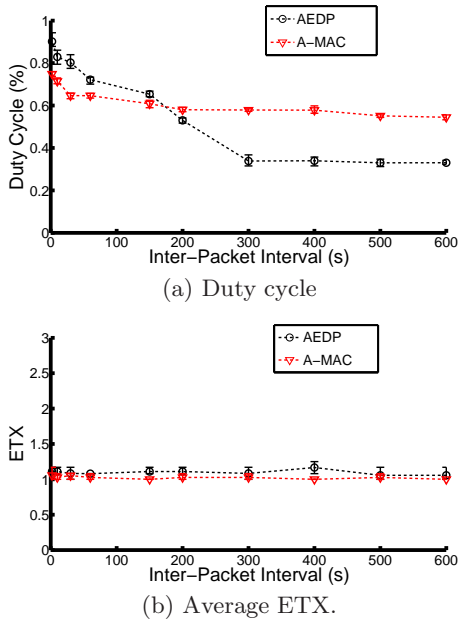


Figure 11: Comparing AEDP and A-MAC with different inter-packet intervals (IPIs)

delay waiting for a response. As discussed in Section 6.2, the default BoX-MAC-2 configuration suffers from an unnecessarily high channel sampling cost of 10.0 ms; in comparison, A-MAC pays a probing cost of 6.2 ms under our oscilloscope measurement. Consequently, previous literature has found that the overhead of receiver-initiated MAC protocols can be even lower than LPL [12]. However, after tuning the energy detection length, AEDP pays a significantly lower sampling cost of 2.9 ms. We note that receiver-initiated MACs *inherently* must pay the overhead of an entire packet transmission; hence A-MAC’s overhead cannot be tuned in this fashion.

Thus, A-MAC has a higher overhead than AEDP at low data rates. However, since A-MAC saves the cost of sending a long preamble, it is able to outperform AEDP at sufficiently high data rates. This result suggests that AEDP is more suitable for low data rate applications, while A-MAC has advantages in high data rate applications. They therefore represent complementary approaches in the design space of low-power MAC protocols in noisy environments⁵.

7.6 Collection Tree Protocol Performance

Finally, we study how well CTP protocol [14] performs over the AEDP. Since AEDP is implemented as a layer situated on top of LPL BoX-MAC-2 MAC layers, running CTP over AEDP is largely a matter of changing configuration wirings. To explore the performance on a large scale, multi-

⁵The pTunes project [27] shows that the performance of MAC protocols are sensitive to their parameters. Optimizing parameters of a MAC protocol is not the focus of this paper. The pTunes system does not support TinyOS and hence cannot be used to select the MAC parameters for our experiments. Nevertheless the experimental study presented in this subsection reveals the general trend of the complementary behavior of AEDP and a receiver-initiated MAC when facing different data rates.

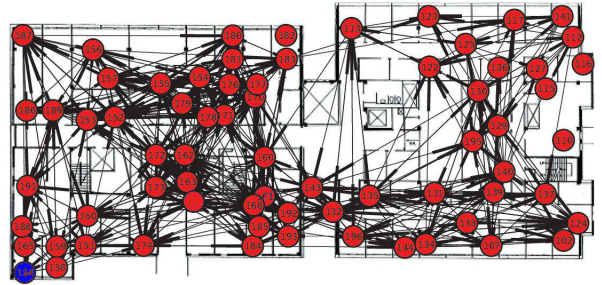


Figure 12: The Testbed topology with a transmission power of 0 dBm. Blue node is a sink node.

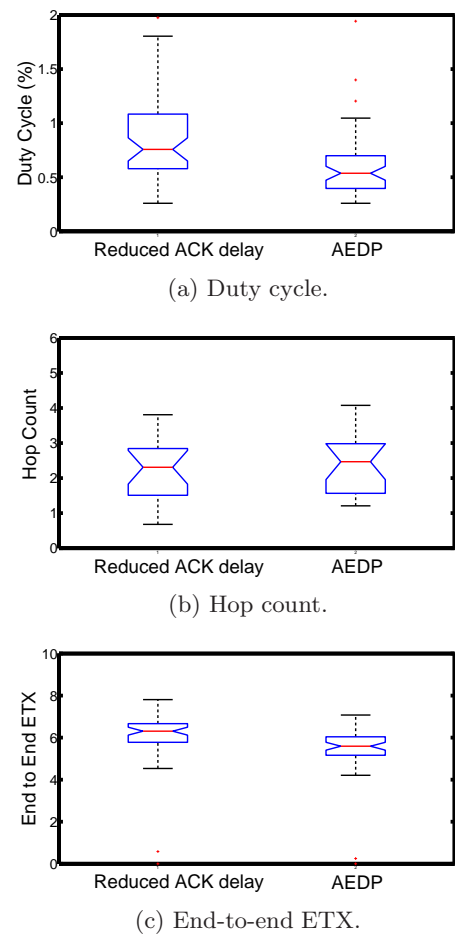


Figure 13: Box-plot comparison between AEDP and LPL BoX-MAC-2 with reduced ACK delay. Central mark in box indicates median; bottom and top of box represent the 25th percentile (q_1) and 75th percentile (q_2); crosses indicate outliers ($x > q_2 + 1.5 \cdot (q_2 - q_1)$ or $x < q_1 - 1.5 \cdot (q_2 - q_1)$); whiskers indicate range excluding outliers.

hop networks, we run the experiments on an indoor testbed consisting of 55-TelosB motes in Jolley and Bryan Hall at Washington University in St. Louis [25]. Figure 12 shows the network topology with transmission power of 0 dBm. Each node produces data at a rate of 1 packet every 5 minutes and all data packets are forwarded to a sink node. We performed two 24-hour experimental runs one with the AEDP and the other with LPL BoX-MAC-2 configuration with the reduced ACK delay. We use the default CTP setting in both two runs. To test the network’s performance in a noisy environment, we set the nodes operating on channel 18 overlapping with the campus Wi-Fi channel.

Figure 13(a)- 13(c) show the box-plots of the duty cycles of all nodes in the testbed and the average hop counts and end-to-end ETX of the routes of all nodes. Since the routes of nodes may change dynamically under CTP, for each node we calculates the average values of hop count and end-to-end ETX during each 24-hour experimental run. As shown Figure 13(a) and 13(c), AEDP reduces the median duty cycle by 35.44% (from 0.79% to 0.51%), while also reducing the median end-to-end ETX by 11.26% (from 6.30 to 5.59). This result shows that AEDA is able to mitigate the impacts of noise on LPL on node duty cycles while simultaneously reducing the multi-hop transmission cost under CTP. As shown in Figure 13(b), AEDP does result in a slight increase the median hop count of the routes (from 2.30 hops to 2.46 hops) as a result of a higher CCA threshold used to filter out noise. The combination of a lower end-to-end ETX and higher hop counts indicate that AEDP was able to filter out weak links affected by noise while still enabling CTP to take advantage of enough good links for low-cost multi-hop communication.

8. CONCLUSION

Maintaining energy efficiency in noisy environments has become an increasingly critical problem as wireless sensor networks are gaining widely deployment in residential and office environments. While LPL has been a popular and effective approach to energy-efficient MAC protocols, false wakeups caused by wireless noise can significantly increase the duty cycle and compromise the benefit of LPL. To address this problem, we first perform an empirical study of the false wakeup problem of LPL in real-world residential environments and find the CCA wakeup threshold is an effective knob for controlling false wakeups. We then propose AEDP, an adaptive protocol that dynamically adjust a node’s wakeup threshold to improve network reliability and duty cycle based on application-specified bounds. AEDP has been implemented on TinyOS 2.1.1 and the TelosB platform. Experimental results from both real-world residential deployments and testbed experiments show that AEDP can effectively maintain low duty cycles in noisy environments and adapt to network changes and links with varying signal strength. We also found AEDP and A-MAC more energy-efficient for applications with low data rate and high data rate, respectively, and therefore provide complementary approaches suitable for different classes of applications.

There are two limitations to AEDP. First, tuning CCA threshold is ineffective for links with low signal strength that can be close to or below the signal strength of noise. In this case AEDP will set the wakeup threshold to the minimum RSS of incoming links and as a result cannot effectively reduce false wakeups caused by noise. This makes AEDP less

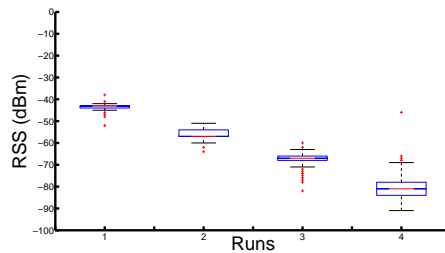


Figure 14: RSS of four runs of experiment. We change distance to achieve different RSS at the receiver.

effective for highly sparse networks connected by mostly long links. Second, our implementation is specific to the particular CC2420 radio stack used. It is important to develop a general methodology — such as the analysis performed in Section 6.2 — to select the key radio and MAC layer parameters. For a new radio stack, developers should firstly measure the ACK delay and then tune the duration of CCA polling accordingly. In general, the duration of CCA polling must be longer than the ACK delay to avoid false negatives in energy detection which can heavily worsen the performance. On the other hand, a long energy detection contradicts the need for a short, inexpensive energy detection, and arguably even renders the entire check ineffective. Therefore, the duration of CCA polling should be slightly longer than the ACK delay.

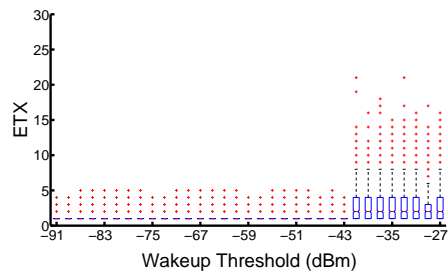
APPENDIX

We run a set of experiments to verify the hypothesis that the maximum wakeup threshold T_{max} should be. We deployed a pair of TelosB motes running TinyOS 2.1.1 in an office environment on a clean channel (channel 26). The TelosB motes were configured to use the BoX-MAC-2 LPL-based MAC with a wakeup interval of 2 seconds. In order to capture the relation between wakeup threshold and link quality (ETX), we configured one mote to generate 1000 transmission in each wakeup threshold, while the wakeup threshold was increased at a step of 2 dBm. We added sequence number into the packet payload, so the receiver could record the ETX for each wakeup threshold based on the sequence numbers. We changed distance to achieve different RSS at the receiver (as shown in Figure 14). We used the reduced ACK delay and 115 times of CCA polling mentioned in Section 6.2.

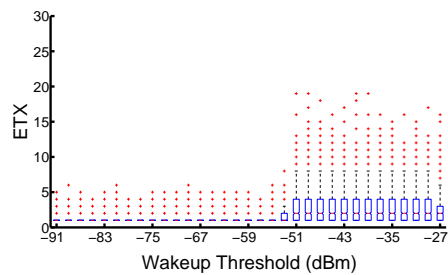
Figure 15 show the relation between ETX and wakeup threshold at different RSS of incoming packets (see Figure 14). As expected, we can see ETX increase significantly when the wakeup threshold exceeds the RSS of incoming packets. This result shows that the maximum wakeup threshold T_{max} should indeed be no higher than the RSS of incoming packets.

A. REFERENCES

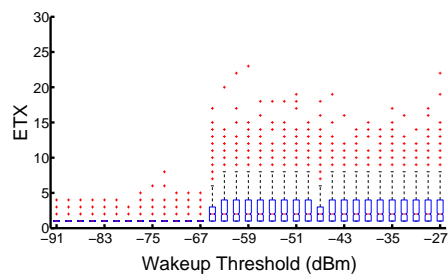
- [1] <http://www.tinyos.net/>.
- [2] <http://www.contiki-os.org/>.
- [3] <http://www.zti-telecom.com/EN/LanTrafficV2.html>.



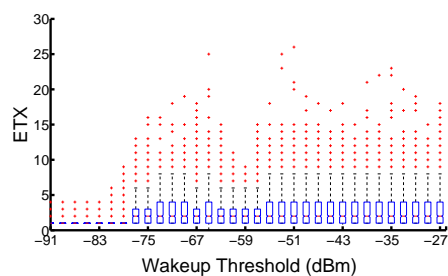
(a) Median RSS is -42 dBm



(b) Median RSS is -55 dBm



(c) Median RSS is -67 dBm



(d) Median RSS is -79 dBm

Figure 15: ETX vs. wakeup threshold with different RSS of incoming packets.

- [4] http://docs.tinyos.net/index.php/CC2420_Hardware_and_Software_Acks.
- [5] M. Bertocco, G. Gamba, and A. Sona. Experimental optimization of CCA thresholds in wireless sensor networks. In *EMC*, 2007.
- [6] C. A. Boano, T. Voigt, N. Tsiftes, and L. Mottola. Making sensornet mac protocols robust against interference. In *EWSN*, 2010.
- [7] M. Z. Brodsky and R. T. Morris. In defense of wireless carrier sense. In *SIGCOMM*, 2009.
- [8] 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*, 2006.
- [9] K. K. Chintalapudi and L. Venkatraman. On the design of mac protocols for lowlatency hard real-time discrete control applications over 802.15.4 hardware. In *IPSN*, 2008.
- [10] Crossbow Technology. TelosB mote platform. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf.
- [11] A. Dunkels. The contikimac radio duty cycling protocol. Technical Report 5128, Swedish Institute of Computer Science, 2011.
- [12] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis. Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless. In *Sensys*, 2010.
- [13] P. Dutta, J. Taneja, J. Jeong, X. Jiang, and D. Culler. A building block approach to sensornet systems. In *SenSys*, 2008.
- [14] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *SenSys*, 2009.
- [15] IEEE. *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, 2006.
- [16] A. Kiryushin, A. Sadkov, and A. Mainwaring. Real-world performance of clear channel assessment in 802.15.4 wireless sensor networks. In *SENSORCOMM*, 2008.
- [17] C.-J. M. Liang, B. Priyantha, J. Liu, and A. Terzis. Surviving wi-fi interference in low power zigbee networks. In *SenSys*, 2010.
- [18] D. Moss and P. Levis. BoX-MACs: Exploiting physical and link layer boundaries in low-power networking. Technical Report SING-08-00, Stanford Information Networks Group, 2008.
- [19] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys*, 2004.
- [20] I. Ramachandran and S. Roy. On the impact of clear channel assessment on mac performance. In *GLOBECOM*, 2006.
- [21] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis. An empirical study of low power wireless. In *ACM Transactions on Sensor Networks*, 2010.
- [22] 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 *Sensys*, 2008.

- [23] Texas Instruments. *2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*.
- [24] Wi-Spy Spectrum Analyzer.
<http://www.metageek.net/>.
- [25] WUSTL Wireless Sensor Network Testbed.
<http://wsn.cse.wustl.edu/index.php/Testbed>.
- [26] W. Yuan, J.-P. M. G. Linnartz, and I. G. M. M. Niemegeers. Adaptive cca for ieee 802.15.4 wireless sensor networks to mitigate interference. In *WCNC*, 2010.
- [27] M. Zimmerling, F. Ferrari, L. Mottolay, T. Voigty, and L. Thiele. ptunes: Runtime parameter adaptation for low-power mac protocols. In *IPSN*, 2012.