X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks[*]

Michael Buettner, Gary V. Yee, Eric Anderson, Richard Han
Department of Computer Science
University of Colorado, Boulder. Boulder, CO [USA]
{michael.buettner, gary.yee, eric.anderson, richard.han}@Colorado.edu

Abstract

In this paper we present X-MAC, a low power MAC protocol for wireless sensor networks (WSNs). Standard MAC protocols developed for duty-cycled WSNs such as B-MAC, which is the default MAC protocol for TinyOS, employ an extended preamble and preamble sampling. While this "low power listening" approach is simple, asynchronous, and energy-efficient, the long preamble introduces excess latency at each hop, is suboptimal in terms of energy consumption, and suffers from excess energy consumption at non-target receivers. X-MAC proposes solutions to each of these problems by employing a shortened preamble approach that retains the advantages of low power listening, namely low power communication, simplicity and a decoupling of transmitter and receiver sleep schedules. We demonstrate through implementation and evaluation in a wireless sensor testbed that X-MAC's shortened preamble approach significantly reduces energy usage at both the transmitter and receiver, reduces per-hop latency, and offers additional advantages such as flexible adaptation to both bursty and periodic sensor data sources.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; D.4.4 [**Operating Systems**]: Communications Management

General Terms

Measurement, Performance, Design, Experimentation, Theory

Keywords

Media Access Protocols, Energy Efficient Operation, Low Power Listening, Adaptive Duty Cycling, Networking, Wireless Sensor Networks

[*]This work was supported by a National Science Foundation (NSF) CAREER Award, project # 1535882, and an NSF ITR grant, project # 1541228.

1 Introduction and Motivation

Energy efficiency is a fundamental theme pervading the design of communication protocols developed for wireless sensor networks (WSNs), including routing and MAC layer protocols. One of the primary mechanisms for achieving low energy consumption in energy-constrained WSNs is duty cycling. In this approach, each sensor node periodically cycles between an awake state and a sleep state. Key parameters that characterize the duty cycle include sleep time, wake time, and the energy consumed during the awake state and the sleep state. The period of a duty cycle is equivalent to its sleep time plus awake time. Given duty cycling sensor nodes, the challenges faced by designers of communication protocols are how to achieve high throughput, low delay, and energy efficiency as nodes are waking and sleeping in the network. This paper focuses on the design of X-MAC, an energy-efficient MAC layer protocol for duty-cycled WSNs, and introduces an optimization to adaptively select sleep times for improved energy consumption and latency.

Standard MAC protocols developed for duty-cycled WSNs can be roughly categorized into synchronized and asynchronous approaches, along with hybrid combinations. These approaches are motivated by the desire to reduce idle listening, which is the time that the node is awake listening to the medium even though no packets are being transmitted to that node. Idle listening has been found in 802.11 protocols to consume substantial energy [8, 14], and therefore must be avoided in WSNs. Synchronized protocols, such as S-MAC [16] and T-MAC [15], negotiate a schedule that specifies when nodes are awake and asleep within a frame. Specifying the time when nodes must be awake in order to communicate reduces the time and energy wasted in idle listening. Asynchronous protocols such as B-MAC [13], and WiseMAC [7], rely on *low power listening* (LPL), also called preamble sampling, to link together a sender with data to a receiver who is duty cycling. Idle listening is reduced in asynchronous protocols by shifting the burden of synchronization to the sender. When a sender has data, the sender transmits a preamble that is at least as long as the sleep period of the receiver. The receiver will wake up, detect the preamble, and stay awake to receive the data. This allows low power communication without the need for explicit synchronization between the nodes. The receiver only wakes for a short time to sample the medium, thereby limiting idle listening. Hybrid protocols also exist that combine a synchronized protocol like T-MAC with asynchronous low power listening [9].

A key advantage of asynchronous low power listening protocols is that the sender and receiver can be completely decoupled in their duty cycles. The simplicity of this design removes the need for, and the overhead introduced by, synchronized wake/sleep schedules. Studies of low power listening have demonstrated its energy-saving capabilities [13, 9].

While the low power listening approach is simple, asynchronous, and energy-efficient, the long preamble in low power listening exhibits several disadvantages: it is suboptimal in terms of energy consumption at both the sender and receiver, it is subject to overhearing that causes excess energy consumption at non-target receivers, and it introduces excess latency at each hop. First, the receiver typically has to wait the full period until the preamble is finished before the data/ack exchange can begin, even if the receiver has woken up at the start of the preamble. This wastes energy at both the receiver and transmitter. Second, the low power listening approach suffers from the *overhearing problem*, where receivers who are not the target of the sender also wake up during the long preamble and have to stay awake until the end of the preamble to find out if the packet is destined for them. This wastes energy at all non-target receivers within transmission range of the sender. Third, because the target receiver has to wait for the full preamble before receiving the data packet, the per-hop latency is lower bounded by the preamble length. Over a multi-hop path, this latency can accumulate to become quite substantial.

This paper proposes a new approach to low power listening called X-MAC, which employs a *short preamble* to further reduce energy consumption and to reduce latency. The first idea is to embed address information of the target in the preamble so that non-target receivers can quickly go back to sleep. This addresses the overhearing problem. The second idea is to use a *strobed preamble* to allow the target receiver to interrupt the long preamble as soon as it wakes up and determines that it is the target receiver. This short strobed preamble approach reduces the time and energy wasted waiting for the entire preamble to complete. We demonstrate through implementation in a wireless sensor testbed that X-MAC results in significant savings in terms of both energy and per-hop latency. Finally, X-MAC can optionally use an automated algorithm for adapting the duty cycle of the nodes to best accommodate the traffic load in the network. We demonstrate the additional savings in energy and latency achieved by this adaptation.

This paper makes the following contributions:

- X-MAC introduces a series of short preamble packets each containing target address information, thereby avoiding the overhearing problem of low power listening, saving energy on non-target receivers.
- X-MAC inserts pauses into the series of short preamble packets, creating a strobed preamble, which enables the target receiver to shorten the strobed preamble via an early acknowledgment, thereby achieving additional energy savings at both the sender and receiver, as well as a reduction in per-hop latency.
- We describe an adaptive algorithm which can be used to dynamically adjust receiver duty cycles to optimize for energy consumption per packet, latency, or both.
- Experimental evaluation validates the performance gains and energy savings of the X-MAC protocol in comparison to a traditional asynchronous duty cycle technique.

In the following, Section 2 describes related work. Section 3 describes the basic X-MAC protocol design. Section 4 presents an optimal algorithm for adapting the receiver's duty cycle, and then presents a practical approximation. Section 5 describes the experimental implementation and evaluation on a testbed of motes. Sections 6 and 7 provide a discussion of future work and our conclusions.

2 Related Work

There are a number of approaches to duty-cycling MAC protocols seen in the literature. These approaches can be broadly divided into two categories: techniques that use some method of synchronization to assure that the wake periods of the nodes are concurrent; and those that have no synchronization requirements and instead depend on an extended preamble and low power listening.

S-MAC [16] is a low power RTC-CTS based MAC protocol that makes use of loose synchronization between nodes to allow for duty cycling in sensor networks. The protocol uses three techniques to achieve low power duty cycling: periodic sleep, virtual clustering, and adaptive listening. The nodes in the network periodically wake up, receive and transmit data, and return to sleep. At the beginning of the awake period, a node exchanges synchronization and schedule information with its neighbors to assure that the node and its neighbors wake up concurrently. This schedule is only adhered to locally, resulting in a virtual cluster, which mitigates the need for system-wide synchronization. Nodes that lie on the border of two virtual clusters adhere to the schedules of both clusters, which maintains connectivity across the network. After the synchronization information is exchanged, the nodes transmit packets using RTS-CTS until the end of the awake period and the nodes then enter sleep mode. In [17], the authors introduce adaptive listening to reduce latency. When a node hears an RTS or CTS from its neighbor, it will wake up briefly at the end of the transmission. If the node is the next hop on the data path, waking up at the end of the transmission will reduce latency as the packet can be forwarded immediately without having to wait until the next scheduled awake period.

T-MAC [15] improves on the design of S-MAC by shortening the awake period if the channel is idle. In S-MAC, the nodes will remain awake through the entire awake period even if they are neither sending nor receiving data. T-MAC improves S-MAC by listening to the channel for only a short time after the synchronization phase, and if no data is received during this window, the node returns to sleep mode. If data is received, the node remains awake until no further data is received or the awake period ends. The authors show that, for variable workloads, T-MAC uses one fifth of the energy used by S-MAC. While this adaptive duty cycling reduces energy usage for variable workloads, these gains come at the cost of reduced throughput and increased latency.

A comparison of duty cycling MAC protocols for WSNs is performed in [9]. Specifically, S-MAC and T-MAC are compared to standard CSMA/CA and LPL. S-MAC and T-MAC are also modified to use low power listening during the awake period, which further decreases the energy consumption of the protocols. While they show that T-MAC in combination with low power listening provides very low power communication, latency is not considered. In addition, T-MAC was not able to handle as heavy a load as LPL and S-MAC due to the early sleeping problem.

B-MAC [13], developed at the University of California at Berkeley, is a CSMA-based technique that utilizes low power listening and an extended preamble to achieve low power communication. Nodes have an awake and a sleep period, and each node can have an independent schedule. If a node wishes to transmit, it precedes the data packet with a preamble that is slightly longer than the sleep period of the receiver. During the awake period, a node samples the medium and if a preamble is detected it remains awake to receive the data. With the extended preamble, a sender is assured that at some point during the preamble the receiver will wake up, detect the preamble, and remain awake in order to receive the data. B-MAC also provides an interface by which the application can adjust the sleep schedule to adapt to changing traffic loads. The method of adaptation is left to the application developer. The authors show that B-MAC surpasses existing protocols in terms of throughput, latency, and for most cases energy consumption. While B-MAC performs quite well, it suffers from the overhearing problem, and the long preamble dominates the energy usage.

WiseMAC [7], which is based on Aloha, also uses preamble sampling to achieve low power communications in infrastructure sensor networks. WiseMAC uses a similar technique to B-MAC, but the sender learns the schedules of the receiver awake periods, and schedules its transmission so as to reduce the length of the extended preamble. To achieve this, the receiver puts the time of its next awake period in the data acknowledgment frame. The next time the transmitter wants to send to that receiver it can begin the preamble only a short time before the receiver will awaken, taking into account possible clock skew. This reduces the energy expended when sending the preamble. In addition, for low traffic loads where the preamble is longer than the data frame, WiseMAC repeats the data frame in place of the extended preamble. Receivers process this data frame and if the node is not the intended recipient it returns to sleep. If the node is the recipient, it remains awake until the end of the transmission and sends an acknowledgment. While WiseMAC solves many of the problems associated with low power communications, it does not provide a mechanism by which nodes can adapt to changing traffic patterns.

In addition, low power listening has been implemented by a number of commercial radios, for example the Chipcon CC2500 [1] and the MaxStream XBee radios [3]. The XBee radio modules allow the user to set the sleep period of the radio and to set the length of the preamble that precedes the data packet. The user must be sure to set the sleep period to a duration shorter than the preamble length in order to be assured that the radio was awakened by the preamble. The Chipcon CC2500 uses a similar mechanism, but it has the added benefit of using a low power radio circuit that listens for the preamble. If in Wake-On-Radio mode, a low power radio circuit is used to intermittently sample the channel for a preamble. If the preamble is detected, the main radio circuit is woken up and the radio receives the data packet.

A variety of techniques have employed a Wake-On-Radio (WOR) approach [14, 12] for energy-efficient communication. These approaches employ a second low power radio as a trigger to wake up the primary radio. These WOR approaches require special hardware assistance.

3 X-MAC Protocol Design

The design goals of the X-MAC protocol for duty-cycled WSNs are:

- energy-efficiency
- simple, low-overhead, distributed implementation
- low latency for data
- high throughput for data
- applicability across all types of packetizing and bit stream digital radios

For many applications, asynchronous duty cycling techniques are preferable to synchronized techniques in terms of energy consumption, latency, and throughput. In part, this is because they do not incur overhead due to synchronization. In addition, asynchronous techniques do not have to share schedule information and only stay awake long enough to sample the medium unless they are receiving or transmitting data. Hence, the awake period can be significantly shorter than that of synchronized methods. With a shorter awake period, asynchronous protocols can wake up more often while still maintaining a low duty cycle. This can also lead to reduced latency and higher throughput. However, the extended preamble begins to dominate the energy per packet as latency tolerance, and thus the receiver's sleep period, increases.

In general, for applications with loose latency requirements, synchronized approaches may be more appropriate. In [9], a modified version of T-MAC is shown to conserve more energy than LPL in the absence of latency requirements. In this study, the period of T-MAC was 610 ms while the sleep time of LPL was 270 μs; this would result in an order of magnitude difference in their latencies. In [13], the authors show that for a 10 hop network B-MAC outperforms S-MAC with respect to energy for latencies under 6 seconds.

For these reasons, X-MAC builds upon the foundation provided by asynchronous duty-cycled MAC protocols. While asynchronous techniques perform quite well, there are a number of problems which, if mitigated, would allow for even more efficient communication. X-MAC is designed to address the following problems of low power listening: over-

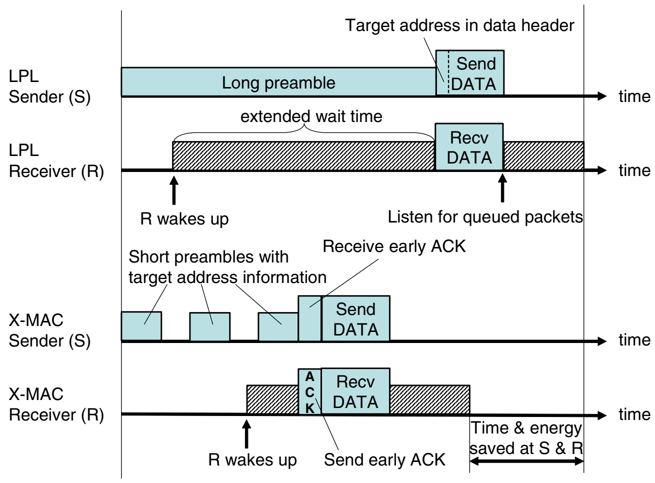

Figure 1. Comparison of the timelines between LPL's extended preamble and X-MAC's short preamble approach.

hearing, excessive preamble and incompatibility with packetizing radios.

3.1 Asynchronous Duty Cycling

A visual representation of asynchronous low power listening (LPL) duty cycling is shown in the top section of Figure 1. When a node has data to send, it first transmits an extended preamble, and then sends the data packet. All other nodes maintain their own unsynchronized sleep schedules. When the receiver awakens, it samples the medium. If a preamble is detected, the receiver remains awake for the remainder of the long preamble, then determines if it is the target. After receiving the full preamble, if the receiver is not the target, it goes back to sleep.

3.2 Embedding the Target ID in the Preamble to Avoid Overhearing

A key limitation of LPL is that non-target receivers who wake and sample the medium while a preamble is being sent must wait until the end of the extended preamble before finding out that they are not the target and should go back to sleep. This is termed as the overhearing problem, and accounts for much of the inefficiency and wasted energy in current asynchronous techniques. This means that for every transmission, the energy expended is proportional to the number of receivers in range. Hence, the energy usage is dependent on density as well as traffic load. This problem is exacerbated by the fact that sensor networks are often deployed with high node densities in order to provide sensing at a fine granularity.

In X-MAC, we ameliorate the overhearing problem by dividing the one long preamble into a series of short preamble packets, each containing the ID of the target node, as indicated in Figure 1. The stream of short preamble packets effectively constitutes a single long preamble. When a node wakes up and receives a short preamble packet, it looks at the target node ID that is included in the packet. If the node is not the intended recipient, the node returns to sleep immediately and continues its duty cycling as if the medium had been idle. If the node is the intended recipient, it remains awake for the subsequent data packet. As seen in the figure, a node can quickly return to sleep, thus avoiding the overhearing problem.

With this technique, the energy expenditure is significantly less affected by network density. The approach of a series of short preamble packets scales well with increasing density, i.e. as the number of senders increases in a neighborhood, energy expenditure remains largely flat. In comparison, as the number of senders increase in each neighborhood of a WSN using LPL, the entire WSN stays awake for increasing amounts of time.

Another advantage of this approach is that it can be employed on all types of radios. Any packetizing radio, such as the CC2420 characteristic of MICAz and TelosB motes, the CC2500, and/or the XBee, will be capable of sending a series of short packets containing the target ID. As we will see later, such universal support across packetizing radios is not true of the traditional extended preamble LPL. In addition, the short preamble packets can be supported across all radios with bit streaming interfaces, e.g. the CC1000 that is found in the MICA2 mote.

3.3 Reducing Excessive Preamble using Strobing

Using an extended preamble and preamble sampling allows for low power communications, yet even greater energy savings are possible if the total time spent transmitting preambles is reduced. In traditional asynchronous techniques, the sender sends the entire preamble even though, on average, the receiver will wake up half way through the preamble. The entire preamble needs to be sent before every data transmission because there is no way for the sender to know that the receiver has woken up. This is one case where more time is spent sending the preamble than is necessary, as illustrated by the extended wait time in Figure 1. Another case occurs when there are a number of transmitters waiting to send to a particular receiver. After the first sender begins transmitting preamble packets, subsequent transmitters will stay awake and wait until the channel is clear. They will then begin sending their preamble, and this occurs for every subsequent sender. Consequently, each sender transmits the entire preamble when in fact the receiver was woken up by the first transmitter in the series.

In the development of X-MAC, we provide solutions for both of these cases. Instead of sending a constant stream of preamble packets, as would most closely approximate traditional LPL, we insert small pauses between packets the series of short preamble packets, during which time the transmitting node pauses to listen to the medium. These gaps enable the receiver to send an *early acknowledgment* packet back to the sender by transmitting the acknowledgment during the short pause between preamble packets. When a sender receives an acknowledgment from the intended receiver, it stops sending preambles and sends the data packet. This allows the receiver to cut short the excessive preamble, which reduces per-hop latency and energy spent unnecessarily waiting and transmitting, as can be seen in Figure 1. Since the sender quickly alternates between a short preamble packet and a short wait time, we term this approach a *strobed preamble*.

In order to guarantee that preambles will be successfully received and that disconnection is avoided, the length of the preamble sequence must be greater than the maximum receiver sleep period. Additionally, the application designer may choose maximum or minimum sleep periods to bound latency and energy consumption, respectively.

In addition to shortening the preamble by use of the acknowledgment, X-MAC also addresses the problem of multiple transmitters sending the entire preamble even though the receiver is already awake. In X-MAC, when a transmitter is attempting to send but detects a preamble and is waiting for a clear channel, the node listens to the channel and if it hears an acknowledgment frame from the node that it wishes to send to, the transmitter will back-off a random amount and then send its data without a preamble. The randomized back-off is necessary because there may be more than one transmitter waiting to send, and the random back-off will mitigate collisions between multiple transmitters. Also, the back-off is long enough to allow the initial transmitter to complete its data transmission. To enable this technique, after the receiver receives a data packet it will remain awake for a short period of time in case there are additional transmitters waiting to send. The period that a receiver remains awake after receiving a data packet is equal to the maximum duration of the senders back-off period, to assure that the receiver remains awake long enough to receive any additional transmitters data packet.

Together, these two techniques greatly reduce excessive preambles, result in the reduction of wasted energy, and allow for lower latency and higher throughput. In addition, both of these techniques are broadly applicable across all forms of digital radios, including packetized and bit stream, because the short time gaps, early acknowledgments, and random back-off can all be implemented in software.

3.4 Packetizing Radios

LPL has a limited ability to support packetizing radios. For example, B-MAC is the default MAC protocol for TinyOS [10] but is incapable of supporting some packet radios such as the Chipcon CC2420. B-MAC was originally developed for bit streaming radios like the Chipcon CC1000, which provides low-level access to the individual bits received by the radio. With these radios, B-MAC can generate long preambles. However, the new generation of sensor motes, such as the MICAz [2], TelosB [4], and iMote [11], make use of the Chipcon CC2420 [1] 802.15.4 radio. Instead of transmitting a raw bit stream, this type of packetizing radio takes as input the payload of the packet, and the radio module inserts its own preamble, header information and CRC. When a packet is received, the radio strips the header, checks the CRC, and if the packet is not corrupted passes the payload of the packet to the microprocessor. While the packet interface reduces the burden on the microprocessor, it limits the ability of the application to precisely control the bits that are sent over the air. Most pertinent, with these radios the application cannot send a preamble of arbitrary length. This precludes the use of LPL protocols that depend on an extended preamble.

For these radios, it is also not possible to mimic an extended preamble by sending a long data packet, which acts as a pseudo-preamble. This is because the receiver will be unable to sample the packet containing the pseudo-preamble, i.e. the packetizing radio will only deliver the packet after it has fully received the entire pseudo-preamble. This defeats the purpose of preamble sampling.

LPL is supported in certain kinds of packetizing radios, such as in the Chipcon CC2500 and MaxStream XBee radios, but only because it is implemented directly in the hardware. In this case, long preambles can be specified because the radio supports this configuration option, unlike the Chipcon CC2420.

In contrast, X-MAC's short strobed preamble is well-suited to all types of digital radios, as mentioned earlier.

4 Adaptation to Traffic Load

While many sensor network applications produce consistent periodic traffic, there is also the need to adapt to variable traffic loads. In addition, different nodes in a multi-hop network may experience different average traffic loads. For example, in a tree topology nodes closer to the base station will forward more data than those closer to the leaves. Nodes with differing traffic loads will consequently have different ideal sleep schedules. Even for a periodic sensing application, it would be difficult for a developer to hand tune all of the nodes' sleep schedules. For applications with time-varying traffic loads, any pre-determined fixed schedule will be sub-optimal.

The performance of a duty-cycling MAC is largely determined by the choice of radio sleep and wake periods for both the senders and receivers. (See Table 3 in Appendix A.) In this section we describe a lightweight algorithm for approximating the optimal sleep and listen periods.

4.1 Optimality

We consider the following metrics for MAC quality: sender and receiver energy consumption and latency. Throughput will be roughly equal to the offered load. The expected energy consumption can therefore be modeled in terms of the durations of the sender and receiver sleep, listen, and transmit periods.

We will show that if the probability, $P_d(t)$, of receiving a packet in any given interval is known then sender and receiver tunable parameters can be set to optimal values. Let P_{Tx}, P_{Rx}, and P_s be the power required to transmit, receive, and sleep, respectively. S_p, S_{al}, and S_d denote the duration of the sender's preamble, acknowledgment listen, and data transmission periods. R_l and R_s denote the receiver listen and sleep periods.

Based on the cycle shown in Figure 1, and assuming uncorrelated packet arrivals and sleep/wake periods, the expected energy to send a packet is given by:

$$
\begin{aligned}
E_s =& (\text{preamble energy + energy per ACK listen}) \\
& * (\text{expected preamble-listen iterations required}) \\
& + (\text{energy to send packet}) \\
=& (P_{Tx}S_p + P_{Rx}S_{al}) \left(\frac{1}{\left(\frac{R_l - S_p}{R_l + R_s}\right)} \right) + S_d P_{Tx} \\
=& \frac{(P_{Tx}S_p + P_{Rx}S_{al})(R_l + R_s)}{R_l - S_p} + S_d P_{Tx}
\end{aligned} \tag{1}
$$

The protocol implementation adds a post-packet-reception delay R_{qpl} to catch queued packet trains. This extra wake time changes the expected number of preamble-listen iterations required, giving:

$$
E_s = \frac{P_{Tx}S_p + P_{Tx}S_{al}}{1 - \left(1 - P_d R_{qpl}\right)\left(1 - \frac{R_l - S_p}{R_s + R_l}\right)} + S_d P_{Tx} \tag{2}
$$

The expected energy to receive a packet is given by:

$$
\begin{aligned}
E_r =& (\text{listen cycle energy + sleep cycle energy}) \\
& * (\text{expected iterations for a preamble to arrive}) \\
& + (\text{energy to send an ACK}) \\
& + (\text{energy to receive packet}) \\
=& \frac{P_s R_s + P_{Rx} R_l}{1 - (1 - P_d(t))^{(R_l + R_s)}} + P_{Tx} R_a + R_d P_{Rx}
\end{aligned} \tag{3}
$$

The expected latency for a single hop is:

$$
\begin{aligned}
Lat =& (\text{duration of preamble + ACK listen}) \\
& * (\text{expected number of iterations required}) \\
& + (\text{duration to send packet}) \\
=& \left(\frac{1}{\left(\frac{R_l - S_p}{R_l + R_s}\right)} \right) * (S_p + S_{al}) + S_d \\
=& \frac{(S_p + S_{al})(R_l + R_s)}{R_l - S_p} + S_d
\end{aligned} \tag{4}
$$

These models lead to the following observations, the derivations of which can be found in appendix B.

THEOREM 4.1. *Energy and latency are both minimized when S_p and S_{al} are set to the lowest values which allow for the preamble to be transmitted and ACK received, respectively.*

For any objective function $f(\cdot)$ which is a function of sender energy, receiver energy, and latency:

THEOREM 4.2. *Optimal receiver sleep and listen times for $\min_{R_s,R_l} f(\cdot)$ depend solely on $P_d(t)$ and device constants.*

For any objective function $f(\cdot)$ consisting of a *convexity-preserving* combination of sender energy, receiver energy, and latency:

THEOREM 4.3. *$\min_{R_s,R_l} f(\cdot)$ can be found by standard convex optimization techniques.*

Thus, given an estimate of $P_d(t)$ and a suitable objective function, the optimal protocol parameters R_s^* and R_l^* can be determined mechanically. While *any linear combination of sender energy consumption, receiver energy consumption, and latency is a suitable objective function*, the remainder of this paper focuses on a single example - optimizing purely for per-packet energy consumption in the one-sender, one-receiver case.

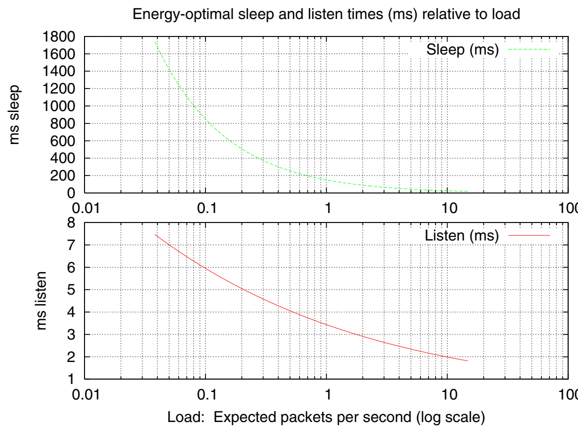

Figure 2. Energy-optimal sleep and listen times

4.2 Approximation

Nonlinear minimization is too demanding a process to be desirable in a sensor networking MAC. That said, the mapping $P_d(t) \rightarrow (R_s{}^*, R_l{}^*)$ is smooth enough to admit lightweight approximations. Figure 2 shows this mapping over a range of packet arrival rates.

The on-node approximation is based on linear interpolation: We pre-compute a table of $P_d(t)$ values and their associated optimal $R_s{}^*, R_l{}^*$ values[6]. In on-line operation, the sensor node uses its estimate $\widehat{P_d(t)}$ to perform a table lookup and interpolates between the closest pre-computed values.

Numerical simulations suggest that this approximation achieves energy efficiency comparable to direct optimization. We chose an interpolation table of 24 exponentially-spaced entries, ranging between 10^{-4} and 10^3 expected packets per second. The energy-efficiency of the optimal and interpolated values of R_s and R_l were then compared for ten thousand values of P_d. Figures 3 and 4 show the results of this experiment: Figure 3 shows the "raw" difference between optimal and interpolated results, and Figure 4 shows the difference as a fraction of the optimal value. The mean difference is 0.45%, and 95th percentile difference is 1.3%.

4.3 Implementation Issues

We found that the timing mechanism used in the implementation of X-MAC was unable to correctly schedule listen periods of less than $\approx 20ms$. This is well above the optimal energy-consumption minimizaing values given in Figure 2 for any load. This produces some inefficiency, but fixing R_l at $20ms$ makes finding optimal values of R_s significantly simpler. The objective function for which we optimized, $f(R_s, P_d) \equiv E_S + E_r$, was the total energy consumed per packet by one sender and one receiver. Figure 5 shows $\frac{\delta f}{\delta R_s}$. The line labeled "0" denotes the set of (R_s, P_d) for which $f' = 0$, which is a local minimum with regard to R_s.

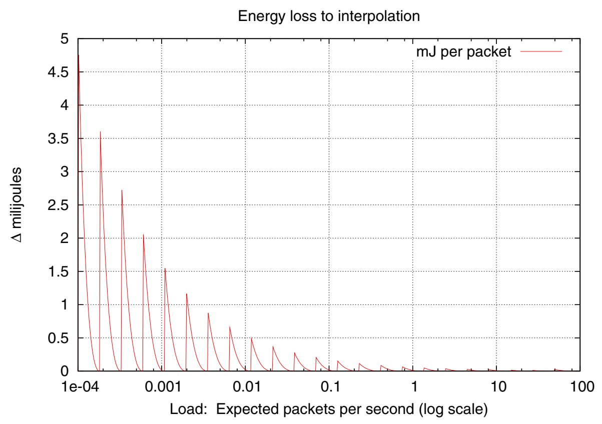

Figure 3. Energy waste per packet due to interpolation.

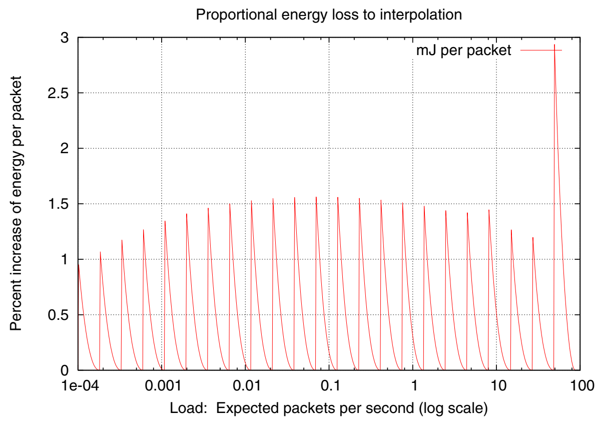

Figure 4. Energy waste as a fraction of optimum.

It is worth noting that there's a large "plateau" around the optimum in which $-1\frac{\mu J}{ms} \leq f' \leq 1\frac{\mu J}{ms}$. For the data rates of interest, the optimal energy consumption is between 10^3 and $10^4 \mu J$ per packet. Thus, within this plateau, a sleep time even 100 ms off of the optimum will do at most 10% worse. An overly long sleep time is more forgiving than a too-short one in terms of energy, but is worse for latency.

4.4 Model Validation

A series of experiments were performed to determine how closely the analytical model of energy consumption matched reality. The analytic model was found to model the receiver energy consumption extremely closely (adjusted R^2 = 0.9975.) For the sender, the model appeared to be close for the number of preamble-listen iterations required, but to overestimate the energy consumption per iteration by a factor of ≈ 1.25. This error may stem from imperfect timing or power measurements, or may represent some aspect of the protocol dynamics not captured by the model. The adjusted R^2 for the sender energy consumption is 0.8487.

The adaptation function used in all of the experiments, as well as for figures 5 and 6, includes an adjustment factor of 0.8 for the per-iteration energy consumption. Figure 6 compares the predicted and observed total energy consumption at several data rates.

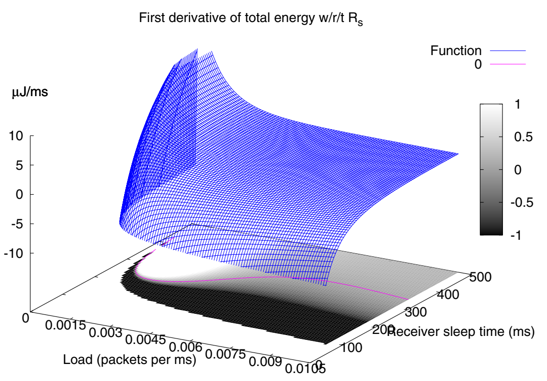

Figure 5. First derivative of energy per packet with respect to receiver sleep time. Shading provides detail for the area around $f' = 0$.

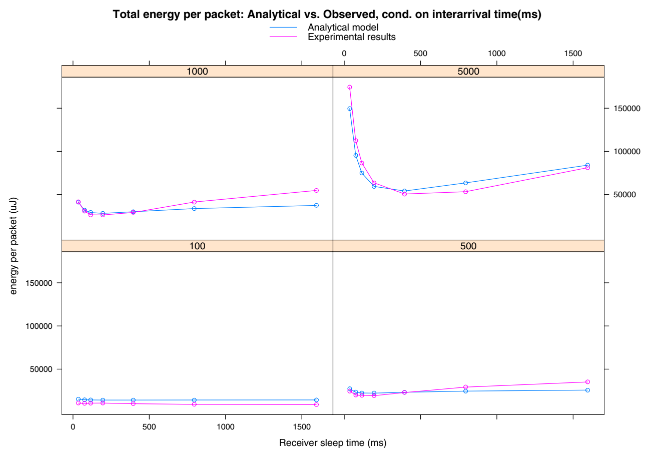

Figure 6. Comparison of analytic model and experimental results for total energy consumer per packet. Subfigures are for different packet inter-arrival times.

4.5 Estimating Traffic Load

The preceding produces near-optimal values when the traffic load $P_d(t)$ is known. An estimate of the instantaneous value, $\widehat{P_d(t)}$ can be derived from the observed packet arrival rate: The likelihood of k packets arriving over a period of $n * t$ can be modeled as a Bernoulli process of n trials with probability of success $P_d(t)$. The most likely value of $P_d(t)$ is that which maximizes the probability of the observed outcome. Applying Bayes' rule, the most likely value of $P_d(t)$ is the maximum on the interval $(0,1)$ of the function:

$$f(P_d|k,n) = \frac{\frac{n!}{(n-k)!k!} P_d{}^k (1-Pd)^{n-k} f(P_d)}{\int_0^1 \frac{n!}{(n-k)!k!} P_d{}^k (1-Pd)^{n-k} f(P_d)\, dP_d} \qquad (5)$$

THEOREM 4.4. $\widehat{P_d(t)} = \frac{k}{n}$ *is an optimal instantaneous estimate of* $P_d(t)$.

In the case where there is no prior knowledge of the probability distribution of $P_d(t)$, equation 19 has its maximum at $\frac{k}{n}$. The derivation is given in appendix B.

A moving estimate can be maintained by any of the standard mechanisms; without knowing the dynamics of application load change it is impossible to identify an optimum.

5 Evaluation

In order to evaluate and demonstrate the correctness and benefits of X-MAC we have implemented the protocol on top of the Mantis Operating System (MOS) [5]. MOS is an open source, multi-threaded operating system developed at the University of Colorado at Boulder for use on wireless sensor networking platforms. An application starts X-MAC by calling an initialization function which takes the minimum sleep time, maximum sleep time, initial sleep time, and initial wake time as parameters.

A second initialization function, spawns a receive thread that wakes and sleeps the radio and handles the adaptation of the sleep periods. The current and boundary values for the wake and sleep times can also be modified directly.

The application thread uses a MOS system call to processes packets in the receive queue, as well as *wor_send()* and *wor_send_skip_preamble()*. The second of the two send functions is only used if the developer is confident that the receiver is listening. To completely sleep the node, the application thread must also put itself to sleep.

5.1 Experimental Setup

For our experiments, deployed an indoor testbed of TelosB motes. The TelosB platform was developed at the University of California at Berkeley and is marketed and sold by Moteiv and Crossbow. The radio used by the TelosB is the Chipcon CC2420, which is an 802.15.4 compliant device, has a data rate of 250kbps, and operates in the 2.4 GHz ISM band. The mote uses an 8 MHz TI MSP430 processor and has 1 MB of external flash.

The current draw of the device was calculated by using an oscilloscope to measure the voltage drop across a 10 Ohm resistor on the power line of the USB cable connecting the node to the desktop. Several baseline values, shown in Table 1, were calibrated for the additional power consumed by the on-board FTDI chip, which is active when the mote is connected to the desktop. These baseline values were checked against moteiv's TelosB data sheet [4]. Similar methods were used to test the power consumption of X-MAC, which will be described later.

State	V (mV)	I (mA)	Calibrated I (mA)
FTDI-Only	197.028	19.7028	——-
RF Off	213.881	21.3881	1.685
RF On	390.213	39.0213	19.319
TX	369.422	36.9422	17.239

Table 1. Observed baseline TelosB current draw using 5V.

As the energy draw of the radio when receiving is far greater than when in idle mode, it is of the utmost importance to achieve a low duty cycle to extend the life of the network. Because the processor consumes an extremely small amount of energy in comparison with the radio, in our evaluation we allow the application thread to run continuously while the radio is turned on and off according to the duty cycle. For these evaluations, the wake time is always 20 ms.

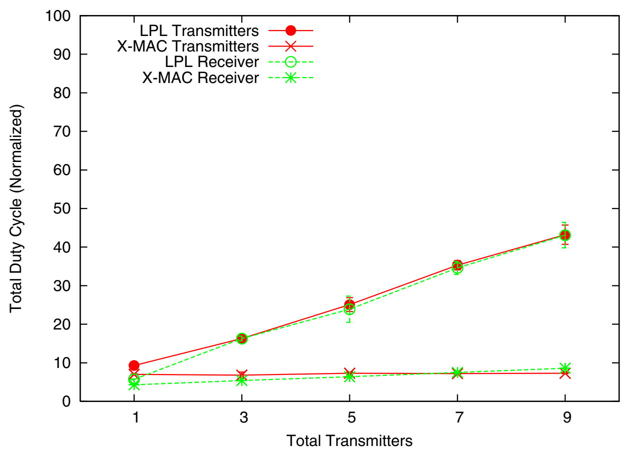

Figure 7. Duty cycles of non-contending senders and receiver and as a function of network density.

As a comparison protocol for X-MAC, we have implemented a basic asynchronous LPL MAC protocol. This protocol is the closest approximation that we could develop using a packetizing radio. When sending, the transmitter sends a stream of preamble packets as rapidly as possible, and after the extended preamble the data packet is sent. There are two differences between X-MAC and the LPL protocol; first, the LPL protocol does not inspect the preamble packets for the target ID so all receivers will remain awake until they receive the data packet; second, with the LPL protocol, transmitters always send the entire extended preamble and receivers do not send an acknowledgement packet to the transmitter. In addition, the adaptation algorithm cannot be applied to the protocol. Although the receiver can adjust its sleep period, the transmitter will not be aware of this change so it will not know to adjust the length of its preamble.

5.2 X-MAC Performance

To evaluate the performance impact of the X-MAC protocol, we performed a number of experiments that test X-MAC without adaptation on simple topologies with no contention. We then introduce contention into the network, and finally evaluate the adaptive optimization for X-MAC.

5.2.1 Duty Cycle Under No Contention

To demonstrate the benefits of the overhearing avoidance and the strobed preamble in X-MAC, we performed an experiment with a varying number of nodes. For this, we set up a star topology consisting of one receiver and up to nine senders where all nodes are within transmission range of each other. Each node sends a packet once every 9 seconds to the receiving node and all nodes have a sleep period and preamble length of 500 ms. The number of senders was varied between 1 and 9, with the transmissions timed so as to avoid contention.

As can be seen in Figure 7, both the X-MAC and LPL duty cycles increase approximately linearly with the number of senders. The difference in slope reflects X-MAC's shorter preambles. Note that fixed overheads such as the minimum

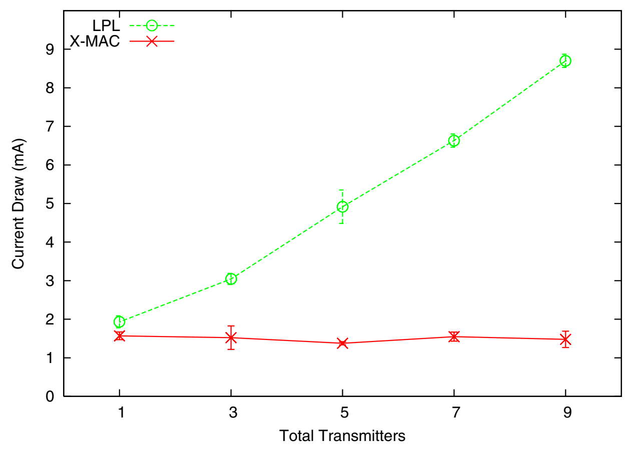

Figure 8. Power consumption per node versus density.

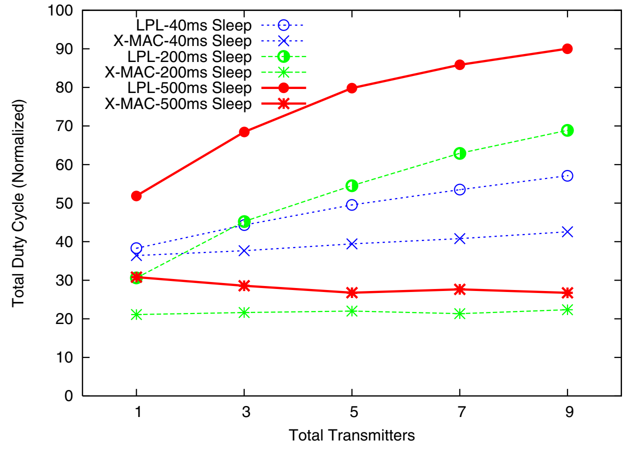

Figure 9. Duty cycle of contending senders, 1 packet per second.

listen time and queued packet listen time reduce the difference.

For this experiment, we show the senders' and receivers' duty cycles separately. In the single-sender case, the receiver's duty cycle is 4.3% for X-MAC and 5.7% for LPL, resulting in a 32.5% increase in energy lifetime of the receiver for X-MAC. On the senders' side, the duty cycle is 7.0% for X-MAC versus 9.3% for LPL. These gains increase as the network becomes more dense.

5.2.2 *Energy Usage*

In order to show the energy savings in terms of actual power consumption, we attach an oscilloscope to one of the transmitting nodes, and repeat the above experiment. We measure the mean current draw in mA. As can be seen in Figure 8, the energy consumption of the LPL protocol increases as network density increases. For X-MAC, energy consumption remains relatively constant as network density increases. These results agree with the previous measurements of the duty cycles.

5.2.3 *Duty Cycle Under Contention*

To show the performance of X-MAC in the presence of contention, we perform a number of experiments similar to the previous duty cycle experiment but with contending transmissions. All packets are sent using a best-effort policy. We vary the number of nodes in the network, and all nodes are within range of each other. All senders generate packets at the same average rate, with randomized jitter to avoid continually synchronized transmissions. As the number of transmitters increases the traffic load increases proportionally. In addition to changing the density of the network, we vary the sleep time of the nodes. We do this for packet send rates of one packet per second, shown in Figure 9, and one packet every ten seconds, shown in Figure 10.

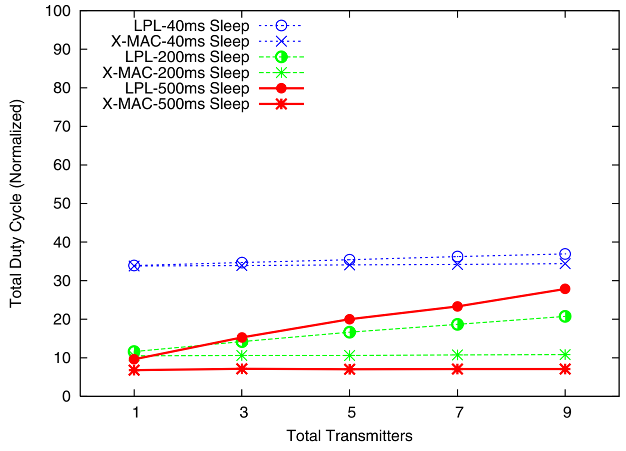

Figure 10. Duty cycle of contending senders, 1 packet per 10 seconds.

The results follow the same trend as the non-contention experiment previously presented. X-MAC uses less energy for nearly all sleep periods and generation rates, and is less sensitive to network density. It should be noted that even over greatly different sleep times (i.e. 200 ms and 500 ms), X-MAC still performs significantly better than LPL. This is seen in the single sender case with a sleep period of 200 ms and a packet generation rate of once every 10 seconds, as seen in Figure 10. In this case, X-MAC uses 10% less energy than LPL.

5.2.4 *Fairness*

We use the variance between senders in the number of data packets generated versus the number of data packets successfully sent as a metric of fairness. Transmit failures occur in LPL when CSMA/CA senses the channel is not clear when it tries to send the data packet. In addition to this, X-MAC fails to send if its preamble was not acknowledged in time (for this test, 5 seconds). The results of this experiment are given in Figure 11. Error bars indicate the standard deviation between the different transmitters, with shorter implying better fairness. X-MAC allows a higher percentage of packets to be sent. In nearly all cases, X-MAC provided better fairness than LPL.

5.2.5 *Transmission Success Rate*

While performing the contention experiments, we also measure how many transmitted packets are successfully received. In Figure 12, we show the results of this experiment when sending one packet per second. The results for the one

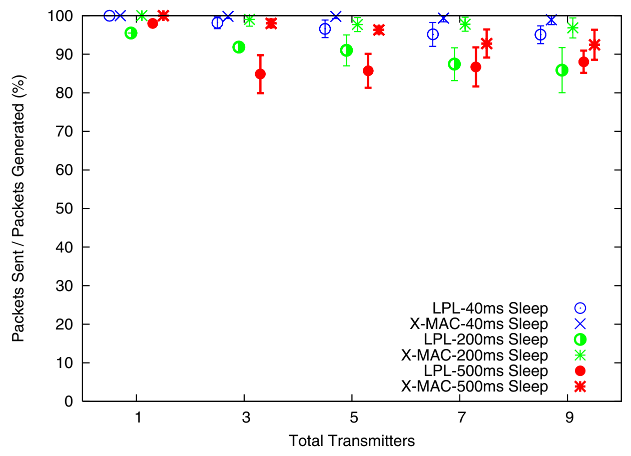

Figure 11. Fairness versus node density, 1 packet per second.

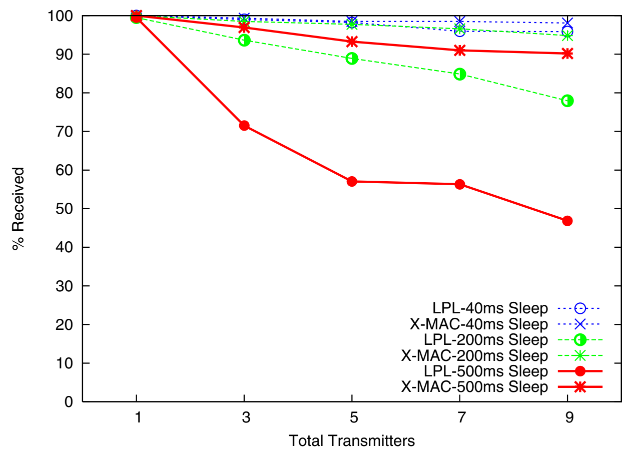

Figure 12. Reception success rate, 1 packet per second.

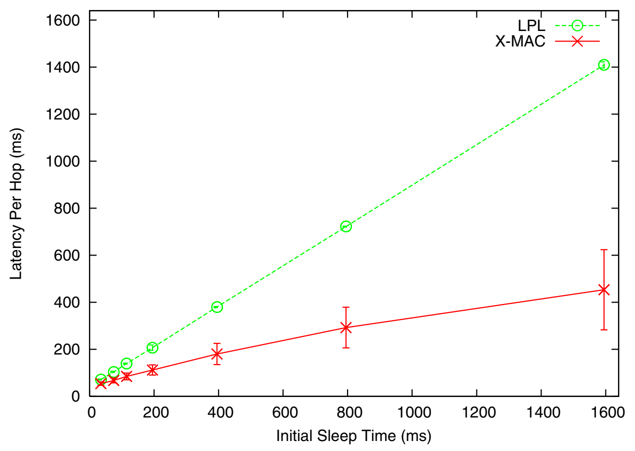

Figure 13. Latency per hop versus initial sleep times.

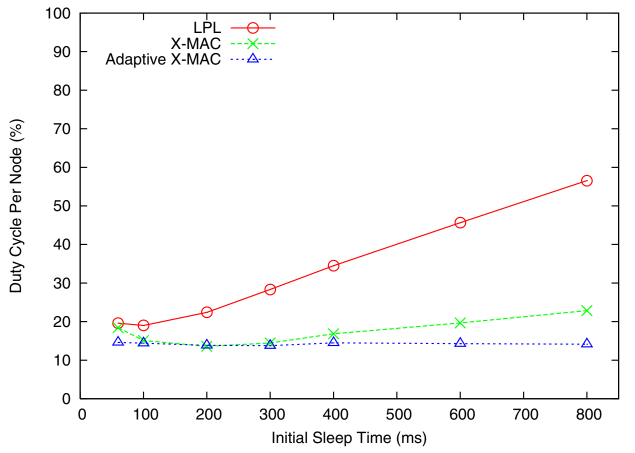

Figure 14. Average duty cycle of all nodes versus initial sleep times, 1 packet per second.

packet per ten second generation rate are omitted, as both protocols received above 90% of the packets for all densities and sleep times.

X-MAC receives approximately 90% or more of the packets for all densities and sleep times. In contrast, LPL loses more packets as density increases. This is due to the longer preamble used by LPL, which results in the channel being saturated with a consequent increase in collisions. This is also evidenced by the fact that LPL performs reasonably well with a 40 ms sleep time, as the time necessary to transmit a packet is reduced. This, however, results in a higher duty cycle and additional energy consumption.

5.2.6 *Latency*

To show the reduction in latency when using X-MAC, we use a chain topology of 8 nodes. We generate packets at one end of the chain at a rate slow enough to ensure two packets are never on the chain at the same time. We then measure the traversal time of various sleep periods using the system clock of the desktop connected to all the motes.

In Figure 13, we show the results of our 7 hop latency test. Analytically, the traversal time should be about half that of LPL, as the receiver will wake up, on average, half way through the preamble. Our results support this, and show that X-MAC reduces latency by approximately 50%.

5.2.7 *X-MAC with Adaptation*

To evaluate the duty cycle performance of X-MAC with the adaptive optimization, we conduct a series of 1-hop experiments under two given traffic rates. For all subjects we measure the average duty cycle of the sender and receiver nodes on a large set of sleep periods. The adaptive variant of X-MAC initially sets its sleep period to the given value, and is allowed to adjust between 220 ms and 2300 ms. We test two traffic rates: 1 packet per second and 0.1 packets per second. For the first test we send 600 packets and for the second test we send 60. These values were selected in order to ensure that each test ran for approximately 10 minutes. Additionally, we only turn on the radio of the sender when it needs to transmit packets in order to fairly focus the results on the amount of energy needed to send and receive a packet. The results of this experiment are shown in Figures 14 and 15.

For each test, X-MAC's adaptive optimization has a lower duty cycle than the static versions of X-MAC and LPL for all sleep times. This is mostly due to the adaptive protocol being

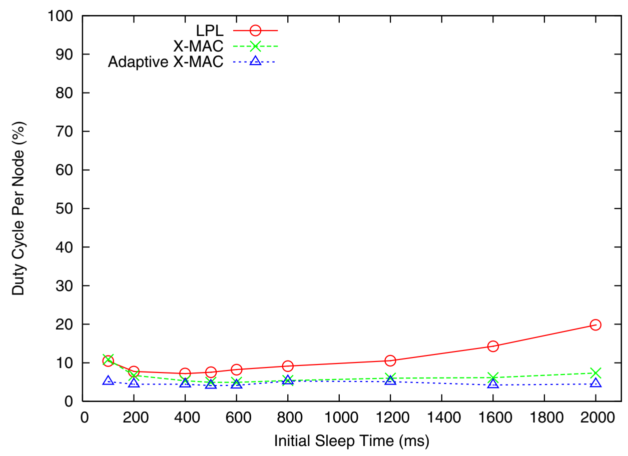

Figure 15. Average duty cycle of all nodes versus initial sleep times, 0.1 packet per second.

able to pick sleep periods between the fixed values given.

We observe that the local minima for the two transmission rates are located at different sleep cycles (roughly 200 ms and 600 ms). This highlights the fact that for static X-MAC and LPL there is no single sleep period that is optimal for all scenarios. The average sleep time selected by the adaptive protocol that has an initial sleep period of 400 ms is given in Table 2. The averages agree with X-MAC's local minima in Figure 6.

Rate (pkts/sec)	Avg Sleep (ms)	Std Dev Sleep (ms)
0.1	592.88	45.64
1	246.1	14.88

Table 2. Average sleep time of X-MAC with adaptive optimization given an initial 400 ms sleep period.

6 Future Work

Our experiments have shown that by using a packetized radio, such as the CC2420, it is possible to achieve significant gains over a basic LPL implementation in energy consumption, latency, and throughput. Moreover, our results have shown that X-MAC's performance gains continually improve as the density of the network increases.

Additionally, we have shown that the adaptive extension of X-MAC introduced in Section 4 is a useful optimization for automatically tuning each node's duty cycle when network loads are not known a-priori. There are other types of traffic, however, for which our adaptive protocol may not be optimal. Further analysis and experimentation of the adaptive optimization are needed to clarify the scenarios (i.e. different types of traffic patterns or application requirements) for which it is appropriate. We have identified areas of potential improvement, which may improve the accuracy of the analytic model.

The 20 ms listen time seen in this paper is an artifact of our implementation and operating system. We are currently working to reduce this listen time to more nearly optimal values.

7 Conclusions

This paper describes X-MAC, a new approach to low power communication in WSNs. X-MAC employs a strobed preamble approach by transmitting a series of short preamble packets, each containing the address of the target receiver. The series of short preamble packets approximates a continuous preamble. Small pauses between preamble packets permit the target receiver to send an acknowledgment that stops the sequence of preamble packets.

Truncating the preamble saves energy at both the transmitter and receiver and allows for lower latency. Non-target receivers which overhear the strobed preamble can go back to sleep immediately, rather than remaining awake for the full preamble as in conventional LPL. This strobed preamble approach can be readily adapted to the packetized radios that are emerging as the standard in today's sensor motes. This paper demonstrated a lightweight algorithm for adapting X-MAC to select near-optimal sleep and listen periods. We verified that X-MAC's strobed preamble approach outperforms traditional LPL by implementing the protocol and performing an array of experiments.

8 Acknowledgments

Special thanks to Brian Shucker, Douglas Sicker, and Dirk Grunwald for their guidance and support. Thanks to our anonymous reviewers and Tobe Yoshito for improving this work with their valuable feedback.

9 References

[1] Chipcon cc 2500 radios. http://www.chipcon.com.

[2] Crossbow micaz motes. http://www.xbow.com.

[3] Maxstream xbee radios. http://www.maxstream.com.

[4] Moteiv telosb motes. http://www.moteiv.com.

[5] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han. Mantis os: An embedded multithreaded operating system for wireless micro sensor platforms. *ACM/Kluwer Mobile Networks & Applications (MONET), Special Issue on Wireless Sensor Networks*, 10(4):563–579, August 2005.

[6] J. Dennis and R. Schnabel. Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall Series in Computational Mathematics. Prentice-Hall, 1983.

[7] A. El-Hoiydi and J. Decotignie. Low power downlink mac protocols for infrastructure wireless sensor networks. *ACM Mobile Networks and Applications*, 10(5):675–690, 2005.

[8] L. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *IEEE INFOCOM*, volume 3, pages 1548–1557, 2001.

[9] G. Halkes, T. V. Dam, and K. Langendoen. Comparing energy-saving mac protocols for wireless sensor networks. *ACM Mobile Networks and Applications*, 10(5):783–791, 2005.

[10] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *Nineth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'00)*, pages 93–104, Cambridge, MA, USA, November 2000.

[11] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 64–75, 2005.

[12] N. Mishra, K. Chebrolu, B. Raman, and A. Pathak. Wake-on-wlan. In *The 15th Annual Interntional World Wide Web Conference (WWW), to appear*, 2006.

[13] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *The Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 95–107, November 2004.

[14] E. Shih, P. Bahl, and M. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *MobiCom*, pages 53–64, 2002.

[15] T. van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *1st ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 171–180, 2003.

[16] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, New York, NY, USA 2002.

[17] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated, adaptive sleeping for wireless sensor networks. *ACM Transactions on Networking*, 12(3):493–506, June 2004.

A Definitions

This appendix defines variables, constraints and equations used in Appendix B.

A.1 Variables

$$
\begin{aligned}
P_{Tx} &\triangleq \text{Power to Tx} \\
P_s &\triangleq \text{Power to sleep} \\
P_{Rx} &\triangleq \text{Power to Rx} \\
S_p &\triangleq \text{Duration of sender preamble} \\
S_{al} &\triangleq \text{Duration of sender ACK listen} \\
S_d &\triangleq \text{Duration of sender data Tx} \\
R_s &\triangleq \text{Duration of receiver sleep} \\
R_a &\triangleq \text{Duration of ACK send} \\
R_l &\triangleq \text{Duration of receiver listen} \\
R_d &\triangleq \text{Duration of receiver data Rx}(= S_d) \\
P_d(t) &\triangleq \text{Packet probability per time } t
\end{aligned}
$$

A.2 Constraints

The following constraints describe the range of variable values for which our model is reasonable:

$$\{P_{Tx}, P_s, P_{Rx}, S_p, S_{al}, S_d, R_s, R_a, R_l, R_d\} > 0 \tag{6}$$

$$
\begin{aligned}
0 \le P_d(t) \le 1 & \quad \text{probability range} && (7) \\
R_l > S_p & \quad \text{preamble reception possible} && (8) \\
S_{al} > R_a & \quad \text{ACK reception possible} && (9) \\
R_d = S_d & \quad \text{No Doppler effect} && (10)
\end{aligned}
$$

A.3 Concrete Equations

Table 3 gives device-specific constants for the Telos B mote.

Variable	**value**	**source**
P_{Tx}	86.2 mW	device-specific
P_s	0.0183 mW	device-specific
P_{Rx}	96.6 mW	device-specific
S_p	1.98 ms	measured
R_a	1.84 ms	measured
S_d, R_d	3.8 ms	application-specific
$P_d \triangleq P_d(1ms)$		application-specific
S_{al}	15.25 ms	Shortest schedulable interval ($S_{al} = R_a$ is optimal.)
R_s	$0 \le R_s$	
R_l	1.84 ms $\le R_l$	

Table 3. Variable values

Substituting these values into equations 2 - 4 gives the following, where time, power, and energy are measured in ms, mW and μJ respectively:

$$E_s = \frac{1188.2}{1-(1-P_d)\left(1-\frac{Rl-1.98}{Rs+Rl}\right)} + 86.2S_d + 2556 \quad (11)$$

$$E_e = \frac{0.0183R_s + 74.4R_l}{1-(1-P_d)^{R_s+R_l}} + 74.4S_d + 14.976 \quad (12)$$

$$Lat = S_d + \frac{0.52\,(R_s+R_l)}{R_l - 0.26} \quad (13)$$

B Proofs

B.1 Theorem 4.1

B.1.1 Sender Preamble Time

The sender preamble duration S_p affects the expected energy to send and the expected latency (equations 2 and 4.) Both take their optimal (minimal) values when S_p is minimized, as will be shown shortly. S_p is bounded from below by the message size / the available bandwidth + processing overhead. The partial derivative of equations 2 and 4 with respect to S_p are given below.

$$\frac{\partial E_s}{\partial S_p} = \frac{(1-P_dR_{qpl})(P_{Tx}S_p + P_{Tx}S_{al})}{(R_s+R_l)\left(1-(1-P_dR_{qpl})\left(1-\frac{R_l-S_p}{R_s+R_l}\right)\right)^2} + \frac{P_{Tx}}{1-(1-P_dR_{qpl})\left(1-\frac{R_l-S_p}{R_s+R_l}\right)} \quad (14)$$

$$\frac{\partial Lat}{\partial S_p} = \frac{(R_s+R_l)\,(S_p+S_{al})}{(R_l-S_p)^2} + \frac{R_s+R_l}{R_l-S_p} \quad (15)$$

It follows from constraints 6 and 8 that $\frac{\partial E_s}{\partial S_p} > 0$ and $\frac{\partial Lat}{\partial S_p} > 0$ for all permissible values. Thus, within the defined bounds, the expected latency and expected sender energy consumption are always reduced by lowering S_p, and expected receiver energy consumption is unaffected.

B.1.2 Sender ACK Listen Time

The sender acknowledge listen time, S_{al} also affects equations 2 and 4.

The partial derivative of equation 2 with respect to S_{al} is given below.

$$\frac{\partial E_s}{\partial S_{al}} = \frac{P_{Tx}}{1-(1-P_dR_{qpl})\left(1-\frac{R_l-S_p}{R_s+R_l}\right)} \quad (16)$$

It follows from constraints 6 and 8 that $\frac{\partial E_s}{\partial S_{al}} \geq 0$ for all feasible values of all variables. Consequently, E_s is always minimized when the lowest permissible value is chosen for S_{al}.

Similarly for the expected latency,

$$\frac{\partial Lat}{\partial S_{al}} = \frac{R_s+R_l}{R_l-S_p} \quad (17)$$

it is always the case that $\frac{\partial Lat}{\partial S_{al}} \geq 0$.

From the preceding paragraphs, it follows that latency, receiver energy consumption and sender energy consumption all take minimal values when S_{al} is minimized.

B.2 Theorem 4.2

Once the device attributes and the parameters with invariant optimal values are fixed, all three objective functions are given by equations 11 - 13. The values of the objectives depend on, at most, $\{P_d, S_d, R_l, R_s\}$.

P_d and S_d can be regarded as unknown constants[1]. Thus, given any particular P_d and S_d and a constrained range of values for R_l and R_s, there exist minimal values E_s^*, E_r^*, and Lat^*.

For any given (P_d, S_d), for each objective or for any combination thereof, there exists a non-empty set of (R_l^*, R_s^*) pairs producing the minimal value of the objective. Additionally, S_d does not appear in the partial derivative of E_s, E_r, or Lat with respect to R_l or R_s. Consequently, the sets of values which minimize those objectives do not depend on the value of S_d. Thus, any objective based on some combination of E_s, E_r, and Lat, can be be written as some $f(P_d, R_l, R_s) : \mathbb{R}^3 \to \mathbb{R}$.

B.3 Theorem 4.3

Consider $f(x) \in \{E_s, E_r, Lat\}$. Each function is convex over the domain x consistent with the constraints given in appendix A.2. All three functions are twice differentiable over this domain, and thus have a well-defined Hessian matrix. For each f, $\forall x \in \mathbf{dom}\, f$:

$$\nabla^2 f(x) \succeq 0$$

Therefore, each f is convex over the appropriate range of R_s and R_l. By extention, any $g(x)$ which consists of convexity-preserving combinations of these fs is also convex. A notable group of these is the set of nonnegative linear combinations of fs.

Any local minimum of $g(x)$ within the appropriate range will therefore also be a global minimum, which means that many non-linear programming techniques can be applied.

B.4 Theorem 4.4

For any given $P_d(t)$, the probability that k packets will arrive over duration nt can be modelled as a Bernoulli process of n trials with probability of success Pd. The frequency mass function for the probability of k "hits" in n trials is given by

$$P_n(k) = f(k,n|P_d) = \binom{n}{k} {P_d}^k (1-P_d)^{n-k} \quad (18)$$

Applying Bayes' rule, we get the following frequency density function:

$$\begin{aligned} f(P_d|k,n) &= \frac{f(k,n|P_d)f(P_d)}{\int_0^1 f(k,t|P_d)f(P_d)\,dP_d} \\ &= \frac{\frac{n!}{(n-k)!k!}{P_d}^k(1-Pd)^{n-k}f(P_d)}{\int_0^1 \frac{n!}{(n-k)!k!}{P_d}^k(1-Pd)^{n-k}f(P_d)\,dP_d} \end{aligned} \quad (19)$$

[1]They can, of course, change value over time.

In the case where there is no prior information about the distribution, that is where $f(P_d)$ is uniform, and $k \geq 0$, equation 19 reduces to:

$$f(P_d|k,n) = \frac{(1-P_d)^{n-k}\, {P_d}^k}{\int_0^1 (1-P_d)^{n-k}\, {P_d}^k\, dP_d} \tag{20}$$

The best estimate $\widehat{P_d(t)}$ is the value of P_d which maximizes $f(P_d|k,n)$. Note that equations 19 and 20 are undefined where P_d is 0 or 1. For the special cases $k=0$ and $k=n$, $f(P_d|k,n)$ has no extrema in the interval [0,1]. Where $k=0$, the maximum is found where $P_d = 0$ and similarly where $k=n$, the maximum occurs where $P_d = 1$. For $0 < k < n$, the following analysis holds:

$$\frac{df}{dP_d} = \frac{k\,(1-P_d)^{n-k}\, {P_d}^{k-1}}{\int_0^1 (1-P_d)^{n-k}\, {P_d}^k\, dP_d} - \frac{(n-k)\,(1-P_d)^{n-k-1}\, {P_d}^k}{\int_0^1 (1-P_d)^{n-k}\, {P_d}^k\, dP_d} \tag{21}$$

$\frac{df}{dP_d}$ is zero where:

$${P_d}^k = -\frac{k\,(1-P_d)\, {P_d}^{k-1}}{k-n} \tag{22}$$

Equation 22 has two solutions: $P_d = 0$ and $P_d = \frac{k}{n}$. As mentioned above, equations 19 and 20 are undefined where P_d is 0. Thus:

$$\widehat{P_d(t)} = \begin{cases} 0 & \text{if } k = 0 \\ \frac{k}{n} & \text{if } 0 < k < 1 \\ 1 & \text{if } k = n \end{cases} \tag{23}$$

This is just $\widehat{P_d(t)} = \frac{k}{n}$ for $0 \leq k \leq n$.