Dynamic Adaptation of MAC Schedules for Wireless Sensor Networks

Christophe J. Merlin and Wendi B. Heinzelman

Department of Electrical and Computer Engineering
University of Rochester
{merlin, wheinzel}@ece.rochester.edu

Abstract. Many recent advances in MAC protocols for wireless sensor networks have been proposed to reduce idle listening, which is costly in terms of energy. New protocols set the burden on the sending node, requiring transmission of long preambles or repeated advertisements for upcoming packets, thereby allowing the sender's neighbors to turn off their radios for longer periods of time. As these "*Low-Power-Listening* MAC protocols" impose a significant drain on the transmitting node, the schedule used for probing as well as the particular probing technique should be well matched to the current network conditions. Our results show that using such network-aware adaptation of the MAC schedule provides up to 30% increase in lifetime for different traffic scenarios compared with using fixed schedules, at no additional overhead.

1 Introduction

Applications for wireless sensor networks are becoming increasingly complex, and they require the network to maintain a satisfactory level of operation for extended periods of time. Consequently, sensor networks have to make the best possible use of their initial energy resources. Both protocol-specific and cross-layer schemes have offered a plethora of energy reducing techniques. In particular, there are several protocols that focus on reducing energy at the data link / MAC layer, which constitutes the scope of this work.

As new sensor platforms have appeared on the market, a simple observation was made that idle listening, far from being negligible, was a major source of energy consumption [1] [2] [3]. MAC protocol design thus had to be re-thought, and a new generation of *Low-Power-Listening* (*LPL*) MAC protocols were introduced as a result. B-MAC [1] and X-MAC [4], among others, use the principles behind Aloha with preamble sampling [5]: the sending node occupies the medium for long intervals of duration t_i s to signal its imminent packet transmission. Receiving nodes are thus allowed to sleep for at most the duration of this preamble (t_i), and they must stay awake when they sense a busy medium until the packet transfer is complete. In this work, we define "MAC schedule" as the pattern of packet transmissions occurring within the t_i interval.

While a MAC protocol may reduce energy consumption for unicast packets, it may at the same time waste energy when applied to broadcast packets. Such

inefficiencies become significant as broadcast packets make up a larger percentage of the total packets sent on a network—for example, in the case of a room monitoring application at night when no events are reported and only protocol administration packets are exchanged. Until now, protocol designers have only adapted their unicast MAC schedules to function in a broadcast configuration without concern for other design considerations. Instead, we propose adopting the transmission schedule that yields the best lifetime taken from a pool of compatible protocols.

Furthermore, changes in the physical radio have prompted researchers to develop new MAC protocols: B-MAC can no longer be implemented on the new IEEE 802.15.4 compliant platforms because these radios require a fixed preamble and packet footers and headers. In this paper, we assume that the target radio is a ChipCon CC2420 [6] 802.15.4 radio, and we make design decisions accordingly—B-MAC is thus not part of the pool of compatible protocols.

The contributions of this paper are threefold:

- We propose switching MAC schedules from a pool of MAC protocols at the transmitter to minimize energy consumption based on parameters such as packet size, whether the packet is broadcast or unicast, and the estimated ratio of transmit to receive packets in the local neighborhood. The protocols are "compatible": the receiver does not need to know what specific schedule is being used, as it always wakes up every t_i s to sense the medium, regardless of which schedule is selected by the transmitter. As a consequence, this protocol called *MiX-MAC*, requires no overhead, and our implementation of this approach shows that lifetime gains can reach up to 30%.
- Because we utilize existing MAC protocols for our pool, we provide a detailed study of two *LPL* MAC protocols, X-MAC and SpeckMAC-D, in a head-to-head comparison, showing the advantages and disadvantages of each approach for both unicast and broadcast packets. We also introduce MX-MAC, a modified version of X-MAC that proves efficient for both broadcast and unicast transmissions.
- Finally, we help build intuition to gauge which assumptions about the MAC are valid by comparing our analytical model with an implementation of the various MAC protocols on the Tmote Sky motes.

This paper continues with a discussion of related work in Section 2. Section 3 then introduces MiX-MAC, the main concept of this work. Section 4 provides simulation results for both broadcast and unicast schedules for the X-MAC and SpeckMAC-D family of protocols. These results help us define the MAC schedules most adapted to specific conditions in the network. Section 5 provides details of our implementation of these MAC protocols on the Tmote Sky motes, as well as results from experiments with this implementation. Finally, Section 6 concludes this work.

2 Related Work

Following Aloha with Preamble Sampling [5], El-Hoiydi et al. introduced WiseMAC [7], a MAC protocol that reduces the preamble length before sending a data packet by exchanging wake-up schedules between neighbors. However, WiseMAC (like B-MAC) cannot be implemented on 802.15.4 radios. It also requires fine time synchronization between nodes, and at a cost that may be difficult to quantify. For these reasons, it was not included as such in our study. Moreover, other work [8] shows that implicit synchronization can be achieved between nodes running some of the protocols studied here, reducing the need for synchronization overhead.

B-MAC [1] with *LPL* was the first MAC protocol to introduce *LPL* schedules for recent radios. Polastre et al. provide a model for *LPL* with strong consideration for the target radio. The authors thoroughly compare B-MAC to S-MAC [2] and T-MAC [3]. To curb limitations imposed on the receiving node to stay awake for the time of the preamble, Polastre et al. propose sending packets with half-sized preambles. Post-B-MAC protocols include X-MAC [4] and SpeckMAC-D [9]. Both protocols are of the channel-probing family and tried to improve the *LPL* scheme presented by B-MAC. Further explanation of these protocols is provided in Section 3. Although more recent, C-MAC [10] uses the same schedule as X-MAC and is therefore included in our work under the same principles that govern X-MAC.

In [9], Wong and Arvind also propose SpeckMAC-B, which is compared, along with SpeckMAC-D, to B-MAC. The SpeckMAC protocol family is intended for miniature motes called specks. SpeckMAC-B stands for *Back off* and replaces the long preamble with a sequence of wake up packets containing the destination target and the time when the data packet will be sent. This allows receiving nodes to sleep for the remainder of t_i and activate just in time for data reception. However, this scheduling supposes fine time synchronization between nodes, which we do not assume in this work. Wong and Arvind develop a model for SpeckMAC protocols and study their impact on the ProSpeckz platform (a larger speck of size one square inch), while comparing them to B-MAC.

In this paper, we build on this past body of work, utilizing the idea of adapting to current network conditions and specifically focusing on adapting *LPL* MAC protocols, which have proven quite energy-efficient for low data-rate wireless sensor network applications.

3 MiX-MAC: A Highly Adaptable MAC Protocol

3.1 Principles of MiX-MAC

No protocol in the *LPL* family outperforms the others over all potential conditions in the network. Selecting a MAC protocol supposes a compromise between excellent performance under certain circumstances (hoped to be the common case), and suboptimal operation otherwise. Various protocols may perform differently according to the broadcast / unicast nature of the packets, the size of

the frames, or whether a node is mostly receiving or sending packets. Adapting the MAC schedule allows optimal performance across those parameters.

We propose creating a pool of MAC schedules that are *compatible* with one another: while the sender may decide which schedule to follow based on the parameters mentioned above, the receiver need not be informed of the changes in MAC schedules. For instance, a sender choosing a certain MAC schedule may expect an ACK frame between packet transmissions; it will thus stay in receiving mode for a given time before it returns to transmitting mode. At the other end of the communication, a receiver simply wakes up periodically (every t_i s), and occasionally receives packets. If a received packet is marked with an acknowledgment request, it immediately sends an ACK frame. Switching between interchangeable MAC schedules guarantees that gains in energy and latency are achieved without any overhead save the computation required to determine the best schedule to use. We call this approach, whereby the MAC schedule is adapted over time, MiX-MAC.

There are many ways in which schedule adaptation can occur. For example, the transmitting node could *learn* about its performance and adapt to performance degradation. However, this requires extra overhead and may take too long to converge to the ideal schedule. Instead, we utilize a small look-up table within MiX-MAC that the transmitting node uses to determine which schedule is best suited for the current node, network and application conditions. This solution proves inexpensive in terms of computation power during runtime. The threshold values dictating a change in the MiX-MAC schedule can be established before deployment using simulation and implementation results. As we will show, inaccuracies in the estimates of current node, network and application conditions do not have a major impact on the performance of MiX-MAC.

Existing MAC protocols were included as part of the pool of compatible MAC schedules: X-MAC [4] and SpeckMAC-D [9], which were introduced around the same time. However, we also add a novel MAC schedule based on a modified version of X-MAC.

3.2 X-MAC: A Short Preamble MAC Protocol

X-MAC Schedule Under the X-MAC [4] schedule, a sender repeats the transmission of an advertisement packet containing the address of the intended receiver. Upon hearing the advertisement packet, the receiver replies with an ACK, which is followed by the transmission of the data packet by the sender. Figure 1 illustrates this process.

In [4], Buettner et al. do not propose implementing X-MAC for broadcast packets. X-MAC cannot broadcast packets as is, as the flow of advertisement packets cannot be answered by an ACK packet. A natural extension to X-MAC is to repeat advertisement packets for t_i (the *inter-listening time*) and then send the data packet; however, receiving nodes have to wait until the completion of the advertisement cycle before they can receive the data packet—and go back to sleep (on average, they must wait for $\frac{t_i}{2} + t_{txPacket}$). In such cases, X-MAC performs

equally to B-MAC, with the added advantage that it can be implemented using fixed preambles.

MX-MAC: A Modified X-MAC for Broadcast Transmissions In [4], Buettner et al. make a convincing case for the energy and latency gains achieved by their proposed X-MAC protocol. Although efficient for unicast packets, this simple scheme is not well suited for broadcast transmissions. One additional drawback to X-MAC is its sensitivity to the hidden node problem and the persistence of a high risk of false positive packet reception acknowledgements. Indeed, early ACKs are sent and received before the data packet is transmitted, which does not guarantee successful reception of the packet.

We propose to modify the MAC schedule of X-MAC by repeating the data packet and waiting for ACK frames between transmissions. A received ACK signifies that the data packet has been correctly received and stops the transmission flow of data packets. This renders the MAC protocol immune to false positive packet receptions.

For broadcast packets, the flow of data packets is still interleaved with periods of listening for acknowledgments. Consequently, multiple receivers of the same packet may wake up, stay in Rx mode until the full reception of a packet, and go back to sleep. Figure 1 illustrates the timeline for this modified X-MAC protocol, called MX-MAC.

3.3 SpeckMAC-D: Repeating the Data Packet

Another *LPL* protocol is SpeckMAC-D [9]. In SpeckMAC-D, if a sender wants to transmit a packet to a receiver, it performs a clear channel assessment (CCA), and if successful, starts repeating the packet for at least t_i seconds. When a receiver wakes up, it checks the medium. If busy, it listens until it has received a full data packet or until it realizes that it is not the intended destination for the packet. Figure 1 illustrates the transmission schedule for SpeckMAC-D.

Although not specified by Wong and Arvind in [9], we can imagine that ACKs be manually sent. If the sender is requesting an ACK, it needs to number the repeated data packets in a way that the receiver knows when to wake up and send an ACK after the sender is done transmitting. However, this small change adds two additional bytes of overhead in each packet in the absence of time synchronization; we consider that for t_i ranging from 0 to 1s, 2 bytes are necessary to number packets (it is not uncommon to schedule more than 256 packet repeats for the smallest packet size and thus 1 byte would not be sufficient). This number has to be updated every time a packet repeat is sent out, forcing the MAC protocol to reload the TxFIFO buffer before switching to Tx mode. Figure 1 illustrates the transmission schedule for this protocol, labeled SpeckMAC-D-ACK.

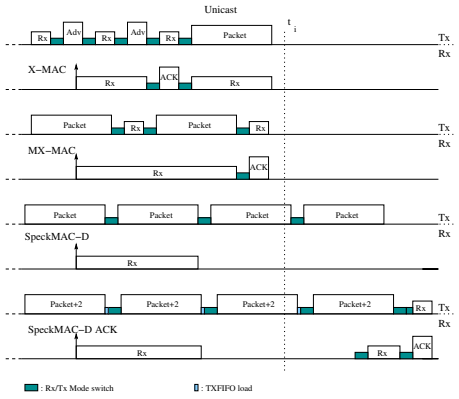


Fig. 1. X-MAC, MX-MAC, SpeckMAC-D and SpeckMAC-D-ACK protocol timelines for unicast packets. t_i is the channel check interval.

3.4 Calculation of MiX-MAC Switching Thresholds

The X-MAC, MX-MAC or SpeckMAC-D schedules perform differently depending on the network scenario and application requirements. MiX-MAC attempts to pick the MAC schedule that yields the best operating points by adapting the MAC schedule, based on a look-up table. Thus, we must populate this adaptation table and find the appropriate switching thresholds. As discussed next, we begin this evaluation with the simulation of an analytical model based on the CC2420 radio data sheet, and verify it with a TinyOS implementation.

4 Determining Optimal Schedules for MiX-MAC Adaptation

This section provides a direct comparison of the *LPL* MAC protocols X-MAC, MX-MAC, SpeckMAC-D and SpeckMAC-D-ACK for both broadcast and unicast packets and for different ratios of transmit to receive and different packet sizes, in order to determine which MAC schedule performs best under different conditions in order to populate a simple look-up table within MiX-MAC for schedule adaptation.

Using the CC2420 data sheet, we modeled the energy dissipation for determined scenarios and simulated the corresponding lifetime of a node following each MAC schedule. This analytical model can be seen as an "ideal" representation of the radio, when not burdened by an operating system. Using this model provides insights into the inner qualities and drawbacks of the various MAC protocols. Section 5 provides a discussion of the implementation of these MAC protocols for TinyOS platforms.

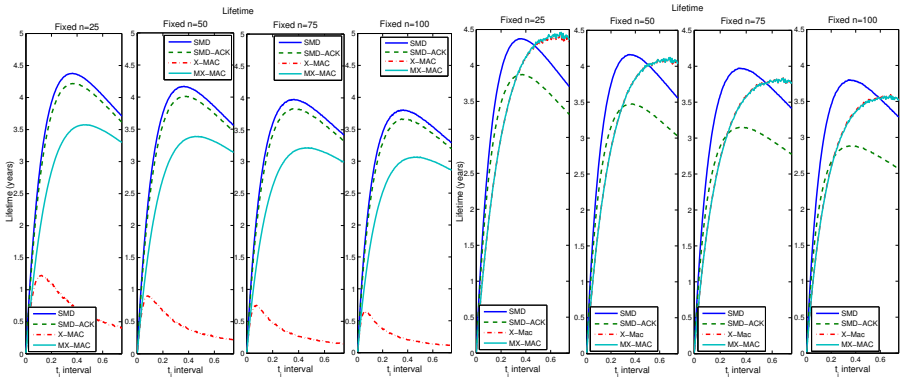


Fig. 2. Node's lifetime for X-MAC, MX-MAC, SpeckMAC-D and SpeckMAC-D-ACK with a fixed n and a rate $r = {}^{1}/_{300}\ pkt.s^{-1}$. All packets sent are broadcast (a) or unicast (b).

4.1 Performance Comparison

We begin by comparing X-MAC, MX-MAC, SpeckMAC-D and SpecMac-D-ACK for a scenario similar to the one described in [1]. One node receives packets at a rate r $pkt.s^{-1}$ from n neighbors. This node sends m packets at rate r to one (unicast) or all (broadcast) neighboring nodes. Unless otherwise specified, m is equal to 1. The lifetime is calculated assuming two AA batteries and the highest transmit power setting (0dBm). Note that the size of the X-MAC advertisement packet is set to 11 bytes, which includes 5 bytes of preamble and 6 additional bytes for headers and footers as standardized by the CC2420 radio.

Broadcast packets In this scenario, all packets are broadcast. Figure 2(a) shows the lifetime given fixed t_i and n for X-MAC, MX-MAC, SpeckMAC-D and SpeckMAC-D-ACK. As is to be expected, SpeckMAC-D performs better than SpeckMAC-D-ACK at all times and for all cases since the latter adds bytes of overhead to the transmission. MX-MAC and X-MAC both exhibit more modest lifetimes than SpeckMAC-D. X-MAC suffers from the problem exposed in Section 3.2: as t_i increases, so do the receive times at the nodes because in broadcast mode, the stream of advertisements cannot be interrupted. MX-MAC does not suffer from the same problem, but it still has a shorter lifetime than SpeckMAC-D, as MX-MAC forces the sending node to spend more time in Rx mode than the latter and Rx mode is more energy costly than Tx mode for CC2420 radios. The "optimal" t_i is approximately 400ms and 150ms for SpeckMAC-D / MX-MAC and X-MAC, respectively.

Unicast packets In this scenario, all packets are unicast. Figure 2(b) shows the lifetime given fixed t_i and n for X-MAC, MX-MAC, SpeckMAC-D and

SpeckMAC-D-ACK. X-MAC and MX-MAC perform similarly, although the results show clearly the impact of packet size. In unicast scenarios, MX-MAC can be considered to be X-MAC with longer packets. When a receiving node running MX-MAC wakes up at a time when it misses the beginning of a new packet, it must stay in Rx mode longer than those running X-MAC. For both, the "optimal" t_i is in the interval [500ms, 750ms]. As m is 1, the main task for the studied node is to receive, causing the optimal t_i to be larger than in $m \approx n$ scenarios. Longer t_i values favor receiving nodes since a packet is almost always available (in both X-MAC and SpeckMAC-D) when they wake up. On the other hand, a longer t_i has the opposite effect in sending nodes since more packet repeats have to be sent (for a fixed packet size).

SpeckMAC-D gives a completely different picture of the evolution of its lifetime as a function of t_i. Most surprisingly, SpeckMAC-D, in spite of repeating data packets for at least t_i seconds without interruption, brings a lifetime comparable to the X-MAC protocols. However, this is achieved for smaller values of t_i such that a sending node would spend much less time in Tx mode. Two observations should be made: the optimal value of t_i is a narrow "sweet spot" and SpeckMAC-D without ACKs puts the burden of acknowledgment on upper layers (not included here). SpeckMAC-D-ACK follows the same pattern, although its lifetime is reduced by increased size and time overhead.

Ratio of transmissions vs. receptions ($m \approx n$) As explained in the previous section, a node's ratio of transmissions to receptions may shift the optimal value for t_i by a considerable amount. Figure 3(a) shows the changes in node lifetime when the node sends approximately as many unicast packets as it receives (the unlikely scenario where the node forwards all received *broadcast* packets is not shown).

X-MAC and MX-MAC still perform equally well, although a small difference can be perceived: the cross-over observed above no longer happens because nodes running X-MAC send many more packets, which, due to the X-MAC schedule, sets the radio in Rx mode for longer periods of time than for MX-MAC. Since advertisement packets are shorter than data packets, X-MAC spends more time switching modes and listening to the medium than MX-MAC. As expected, the optimal t_i value is significantly smaller than before.

For $m \approx n$, the optimal t_i sweet spot is much more narrow. Changes made to t_i by the MAC protocol must be very gradual because the effect of transmitting a packet vs. that of receiving a packet can be very severe outside of a very narrow band within which the MAC protocol must operate.

Packet size Figure 3(b) shows interesting results when the packet size is doubled from 40 bytes to 80 bytes and unicast transmissions are used with $m = 1$. In this case, X-MAC's maximum lifetime is equal to that of SpeckMAC-D. As missing the beginning of a packet transmission requires staying in Rx mode for longer periods of time, X-MAC does better than MX-MAC.

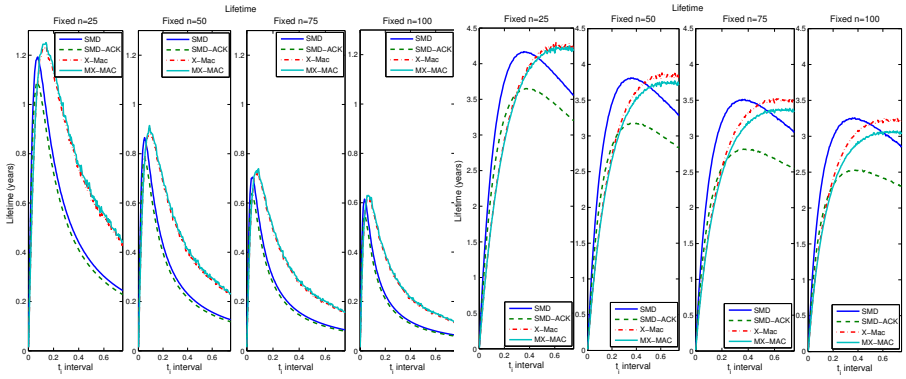


Fig. 3. Node's lifetime for X-MAC, MX-MAC, SpeckMAC-D and SpeckMAC-D-ACK with a fixed n when all packets sent are unicast. Packets are of size 40B and $m = n+1$ (a) or 80B and $m = 1$ (b).

For large packets, choosing X-MAC over MX-MAC can increase the lifetime of individual nodes by up to 30%. It could also be the motivating factor to abandon SpeckMAC-D for X-MAC, which is also less prone to contention. Consequently, packet size is a relevant parameter in choosing the best schedule for a packet transmission—and is free information.

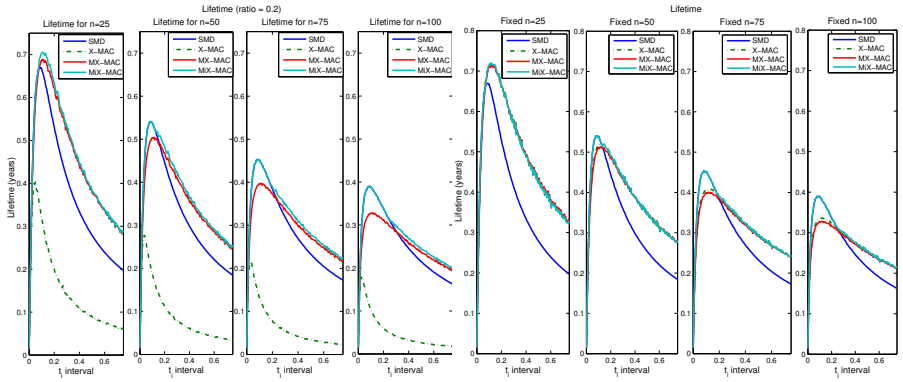
4.2 MiX-MAC Adapts the MAC Schedule to Network Conditions

Picking the right MAC schedule The previous section shows that all MAC protocols sacrifice performance in unicast mode to that of the broadcast mode or vice-versa. MiX-MAC performs well against every combination of parameters.

MiX-MAC adopts SpeckMAC-D's schedule for broadcast packets, and for unicast packets, it uses four axes to decide the appropriate schedule. These include t_i value, packet size, estimated ratio of transmitted vs. received packets, and the ACK requirements determined by the upper level protocols or services.

Resulting lifetime increase Figure 4 presents a comparison of the lifetimes for MiX-MAC, X-MAC and SpeckMAC-D for a packet size of 40 bytes, 20% of the total traffic being broadcast packets (all other packets are unicast) (Figure 4(a)) and all unicast packets (Figure 4(b)). The rate r is $\frac{1}{10}$ packets per second.

These results show that MiX-MAC achieves the upper bound of node lifetime by selecting the best schedule for various scenarios. X-MAC suffers greatly in broadcast mode, even for relatively small proportions of broadcast packets (20%). MiX-MAC helps the MAC protocol obtain the best of all worlds: lifetime gains are obtained over other protocols on a full range of t_i values. This last point is important because in a network where the rate of packet transmissions varies, the optimal value of t_i would naturally vary as was seen previously.



(a) (b)

Fig. 4. Node lifetime as a function of t_i for MiX-MAC, X-MAC, MX-MAC and SpeckMAC-D. The packets are 20% broadcast / 80% unicast (a) and all unicast (b), sent at a rate $r = \frac{1}{10} pkt.s^{-1}$.

5 Implementation in TinyOS

5.1 Reconstruction Model

We chose to accurately evaluate the lifetime of a mote by measuring the energy consumed under various basic operations using a fast data acquisition board. We measured the energy and time spent probing the medium, starting a transmission, sending one frame and switching the radio back to TX mode, stoping a transmission after a successful and failed (only for X-MAC and MX-MAC schedules) transmission, and receiving a packet. Each scenario is then reconstructed by adding the energy expended during each operation.

Our measurements show that it takes a little under 32 μs to send one byte (we measured 31.89 μs exactly), which is confirmed by the CC2420 datasheet. Moreover, our measurements allow us to determine the *turn around* time for each protocol. For SpeckMAC based schedules, the time to revert to TX mode is 772 μs, and it is 1.351 ms for MX-MAC and X-MAC based protocols—as they have to listen for ACK frames between packets. These latter values depend heavily on the TinyOS code and may differ from one programmer to the next.

In order to verify the accuracy of our reconstruction technique for determining node lifetime, we ran several scenarios where we directly measured node lifetime. Based on these experiments, the error of our reconstruction model does not exceed 3%. We have observed that most of the error emanates from the estimation of the idle power, which tends to vary over time due to temperature changes in the acquisition circuit.

5.2 Protocol Design Choices

We tried to optimize as many aspects of the MAC schedule as possible: the time separating two clear channel assessments (CCA) as well as the number of CCAs when sensing the medium, the number of CCAs before a packet transmission, the behavior of a node when detecting a collision, etc. Since our goal is only to *compare* MAC protocols without bias toward one schedule, we endeavored to optimize the behavior of all three MAC protocols. Because all MAC schedules are meant to be compatible, they were implemented by the same TinyOS code. Consequently, all three protocols have the same essential parameters such as the number of CCAs and the time separation between them.

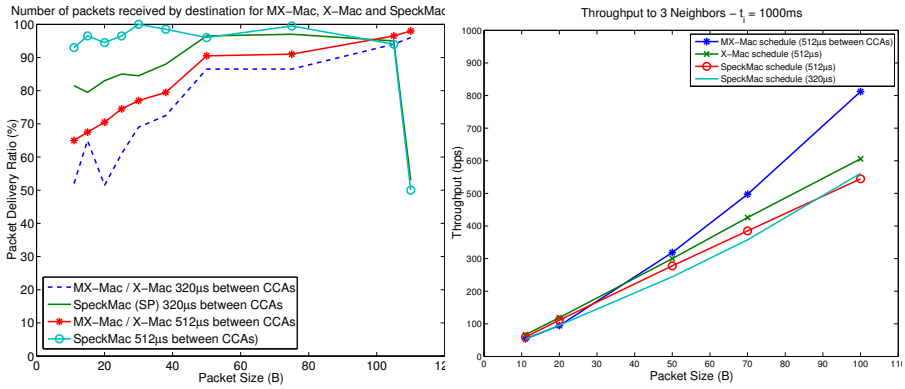
Time Separation Between CCAs Two CCAs are usually sufficient to detect an ongoing transmission, provided they are separated by the correct time. *Turn around* times established in the previous section as well as trial and error experiments helped set the smallest acceptable time separation between CCAs (t_{2CCAs}). During our experiments, we strove to minimize the time separating two CCAs in order to not expend energy, as simulations have shown that medium probes account for most of the energy consumption for low traffic patterns and short t_i values. We also noticed that for a value of 900 μs between CCAs, the delivery reliability of the MAC did not always increase for smaller packets. The reason is that the second CCA would "overshoot" the packet transmission and not bring any advantage.

Figure 5(a) shows the packet delivery ratios for two nodes randomly sending 100 packets to each other, including collisions, missed packets, bad radio states, etc. As the packet size increases, it is generally easier for a receiver to hear a transmission. The dotted line shows a value for t_{2CCAs} that was not retained because of poor reliability.

We found that an acceptable value for t_{2CCAs} is between 320 μs and 512 μs for SpeckMAC, and 512 μs for MX-MAC and X-MAC, which represent the best compromise between energy use in very low traffic networks and fairness to all protocols. For MiX-MAC, which must use compatible parameters for all MAC schedules, we set t_{2CCAs} to 512 μs for all protocols.

For packet sizes close to their maximum value (128B), we found the radio to "jam" under SpeckMAC: the radio would issue RXFIFO overflows because the FIFO was filled before it could be read, and hence the packet delivery ratios in this case dropped significantly.

The X-MAC Design and Choice of Advertisement Packet Size As figure 5(a) shows, the packet delivery ratio for 11B (no payload) packets is only 65% for X-MAC. This prompted the choice of larger advertisement packets (40B), as is supposed by C-MAC [10]. In order to remain compatible with the other protocols, we considered all 40B long packets to be advertisements. The receiver, upon reading a 40B packet from its RXFIFO stays on to receive the subsequent data packet. In other words, when a MAC protocol needs to send a 40B packet, it has to use the X-MAC schedule.



(a) (b)

Fig. 5. (a) Comparison of the packet delivery ratio of the MAC schedules as a function of packet size and time between CCAs. (b) Throughput for t_i values of 1s.

MAC Schedule Compatibility Through design choices, we allowed the three MAC protocols to be compatible. The same TinyOS code can let a mote send and receive packets using the MX-MAC, X-MAC or SpeckMAC schedules.

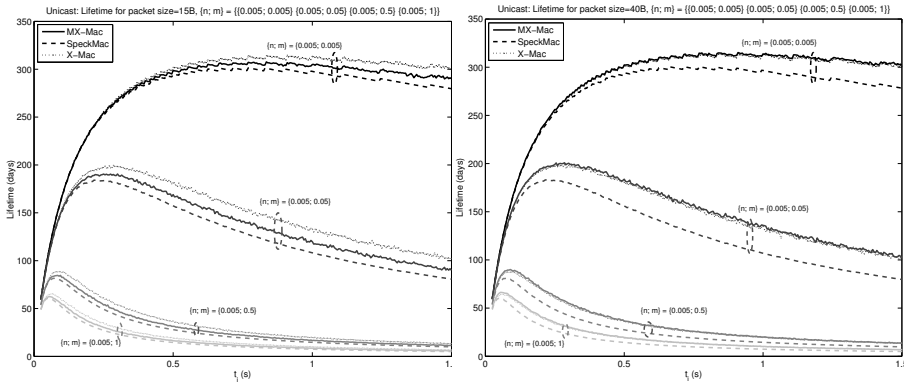
More importantly, the basic principle behind schedule compatibility is that a receiver does not need to know the ongoing schedule, and simply ACKs packets that request it. For MX-MAC and X-MAC, the *acknowledgment request* field must be set to one. If no ACK is requested, the receiver simply turns off after the packet has been received.

5.3 Determination of the Switching Thresholds

In order to populate the MiX-MAC look-up table, we must compare the X-MAC, MX-MAC and SpeckMAC-D schedules and determine which is most appropriate to the current set of parameters on the network.

Reliable Throughput or Goodput In order to evaluate the throughput of the MAC protocols, we let one node send 100 packets to three different neighbors. For MX-MAC and X-MAC, a failed packet transmission was followed by two subsequent attempts to send after 200ms. SpeckMAC does not provide any retransmission mechanism.

Figure 5(b) shows the number of correct bits transmitted in one second. For smaller values of t_i (not shown), X-MAC has the highest throughput. With larger values of t_i and for larger than 20B packets, MX-MAC performs best. This is because MX-MAC is capable of staggering packet transmissions, which compensates for retransmissions. As the packet size increases, fewer retransmissions occur, which increases the throughput by up to 50%. When t_i is too small, MX-MAC cannot fit more transmissions.



(a) (b)

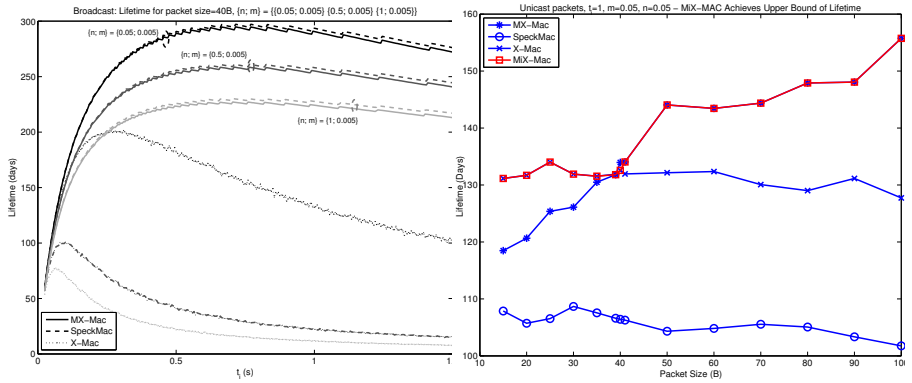
Fig. 6. Comparison of the MX-MAC, X-MAC and SpeckMAC schedules for unicast packets for scenarios where the node is mostly sending. Packets are 15B (a) or 40B (b).

Since X-MAC uses fixed sized advertisement packets, its throughput increases linearly with the data packet size as shown by the figure. For a different reason, SpeckMAC based schedules are also linear: SpeckMAC can send only one packet per t_i period. When the packet size increases, so does the throughput.

These results teach us that MiX-MAC can select the MAC schedule that will yield the best goodput for a certain packet size and t_i value. The results in Figure 5(b) suggest that for $t_i = 1$ s, the X-MAC schedule yields the best throughput for small packets (less than 40B), while the MX-MAC schedule has the best performance for larger packets.

Lifetime for Unicast Packets Figure 6(a) shows the lifetimes for MX-MAC, X-MAC, and SpeckMAC schedules for various $\{n; m\}$ values and 15B unicast packets—n now designates the rate of received packets instead of the number of neighbors as we only have a limited number of Tmotes, and m now designates the rate of transmitted packets. As per the simulations, SpeckMAC does not perform as well as MX-MAC and X-MAC. While this validates the simulations, Figure 6(a) shows that X-MAC performs best for smaller packets as opposed to previously shown. This holds only when the node is mostly sending. This is because the advertisement packet size went from 11B to 40B which increases the chance of being heard during a transmission, and thus prevents retransmissions. At the same time, an increase in packet size increases the energy consumption by only 3-4%. This is because the radio transmits for t_i s, whatever the packet size.

The advantages of X-MAC are reduced further when the data packet size reaches that of the advertisement size because the advertisement packet is no longer easier to hear than the data packet.



(a) (b)

Fig. 7. (a) Comparison of the MX-MAC, X-MAC and SpeckMAC schedules for 40B broadcast packets when the node is mostly receiving. (b) A simple mapping function lets the protocol switch between schedules in order to increase the lifetime.

This section shows that for packets smaller than 40B, and for cases when the node is mostly sending, X-MAC allows the node to increase its lifetime. In other cases, MX-MAC leads to a longer lifetime. If the main concern of the network application is lifetime, MiX-MAC can thus switch between X-MAC and MX-MAC schedules under the conditions of Figure 6. This is possible because while the receiver does not get to pick the MAC schedule, the sender can select the appropriate MAC given current network and neighbor conditions. The receiver does not need to be informed of any changes in MAC scheduling. Based on the packets received, the receiver knows which schedule the transmitter is following.

Lifetime for Broadcast Packets Contrary to unicast packets, one MAC schedule consistently spares the energy of the node, over the range of packet sizes. Figure 7(a) shows that X-MAC performs very poorly, as shown by the simulations as well (Section 4.1). Moreover, the lifetime increase provided by SpeckMAC is modest (2%) when the node is mostly sending (not shown here), and larger (10%) when the node is mostly receiving.

These relatively small lifetime increases hide the fact that with the Speck-MAC schedule, the destination nodes are much more likely to correctly receive the packets (Section 5.2). Thus, using the SpeckMAC schedule for broadcast allows for longer lifetime and more reliable communication.

The results for the broadcast case show that MiX-MAC should always select the SpeckMAC-D schedule: as it will enjoy small gains in lifetime, it will greatly improve packet delivery reliability compared to MX-MAC and X-MAC.

5.4 MiX-MAC Achieves the Upper Bound of Node Lifetime

Implementation example In this section, we present the node lifetime as a function of packet size in Figure 7(b) as an easy-to-read graph of schedule switching benefits. The figure shows a lifetime increase of up to 30% compared to fixed schedules. While the mapping may not be perfect, the error of the point at which schedules are switched stems from the fact that schedules have very similar energy patterns at these packet sizes. The error in schedule switching is thus inherently small.

As can be seen in the figure, depending on the values of $\{m; n\}$, the lifetime may actually increase with the size of the packets sent. This is because a sending node is on for a fixed amount of time, and more reliable communications help avoid retransmissions as shown in Section 5.3.

Effects of erroneous estimates resulting in suboptimal scheduling decisions Since MiX-MAC picks schedules from a pool of existing protocols, erroneous estimates for the optimal t_i would equally affect the performance of X-MAC, MX-MAC, and SpeckMAC-D.

However, if the lookup table were to point to an incorrect schedule, due for instance to an outdated or inaccurate estimate of the number of transmissions over receptions ratio, the network would simply operate at the level of performance of the chosen MAC protocol, without further degradation. For small estimate errors around the points where schedules are switched (the intersection in Figure 7(b)), the difference between the energy consumption of the optimal schedule and that of other schedules is small. A small error around these points cuts lifetime by only a few percentages as our experiment suggests.

Elsewhere, a small estimate error has no effect: because the error is too small to impose a different schedule, the sending node picks the same schedule as the optimum. For very large estimate errors (for which the point of switching schedules is between the estimate and the actual value of a parameter), the resulting performance loss may be significant; however, large estimate errors should be rare by nature.

5.5 Simulation Vs. Implementation Results Comparison

For broadcast packets, our simulations of SpeckMAC-D show a larger lifetime improvement than in the implementation. The difference originates from the time it takes TinyOS to revert to Tx mode after a packet transmission: this results in increased energy consumption. While in the analytical model, the time to switch to Tx mode has a ratio of 4-to-1 from X-MAC / MX-MAC to SpeckMAC-D (736 μs to 192 μs), this ratio is only 2-to-1 in TinyOS. The energy difference is further reduced by longer probes in TinyOS. However, investigation of packet delivery ratio shows that the SpeckMAC-D schedule is more reliable.

For unicast packets, the main difference between simulation and implementation results stems from the size of the advertisement packets sent by X-MAC. Because 11B packets are typically hard to hear for nodes using two CCAs, we

increased the size of advertisement packets to 40B. This shifted the ranges of t_i values and packet sizes for which X-MAC or MX-MAC allowed for longer lifetimes. However, in both sets of results, we observe that the optimal t_i value decreases with an increase in packet transmissions.

6 Conclusion and Future Work

Existing MAC protocols employ identical schedules for both unicast and broadcast packet transmissions or, when impossible, simply modify their "unicast schedule" to work with broadcast packets. For instance, IEEE 802.11 cannot perform an RTS / CTS handshake for broadcast packets, and thus only utilizes CSMA for broadcast packets, regardless of the impact on lifetime or contention.

In this paper, we propose adapting the MAC schedule to node and network conditions to improve performance under a wide range of conditions and for both unicast and broadcast packets. The MAC schedule should be chosen to maximize the lifetime of the network, which includes reducing contention.

The TinyOS implementation brings both validation as well as some contradictions to the simulation results. While most results show similar relative performance, they scale differently as the slowness of TinyOS incurs greater energy consumption. We also demonstrated the feasibility of compatible MAC schedules in a real implementation with no overhead save the memory required for the switching table.

Future work includes researching how to best implement MX-MAC and X-MAC schedules in a lightweight fashion. The number of parameters and metrics may also be increased; However, this will require an efficient way to disperse information within the network.

References

1. Polastre, J., Hill, J., Culler, D.: Versatile low power media access for wireless sensor networks. In: Proc. SenSys '04. (November 2004) 95–107
2. Wei, Y., Heidemann, J., Estrin, D.: An energy-efficient MAC protocol for wireless sensor networks. In: Proc. INFOCOM '02. (June 2002)
3. van Dam, T., Langendoen, K.: An adaptive energy-efficient MAC protocol for wireless sensor networks. In: Proc. SenSys '03. (October 2003)
4. Buettner, M., Yee, G.V., Anderson, E., Han, R.: X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In: Proc. SenSys '06. (November 2006) 307–320
5. El-Hoiydi, A.: Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks. In: Proc. ICC '02. (April 2002)
6. Chipcon Products from Texas Instruments: CC2420 data sheet, 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF transceiver
7. El-Hoiydi, A., Decotignie, J.: WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks. In: Proc. International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS). (July 2004)

8. Merlin, C.J., Heinzelman, W.B.: Node synchronization for minimizing delay and energy consumption in low-power-listening MAC protocols. In: Proc. MASS '08. (September 2008)
9. Wong, K.J., Arvind, D.: SpeckMAC: Low-power decentralised MAC protocol low data rate transmissions in specknets. In: Proc. IEEE International Workshop on Multi-hop Ad Hoc Networks: from Theory to Reality (REALMAN '06). (May 2006)
10. Liu, S., Fan, K.W., Sinha, P.: CMAC: An energy efficient MAC layer protocol using convergent packet forwarding for wireless sensor networks. In: Proc. SECON '07. (June 2007)