EM-MAC: A Dynamic Multichannel Energy-Efficient MAC Protocol for Wireless Sensor Networks

Lei Tang* ltang@cs.rice.edu
Yanjun Sun† yanjun@cs.rice.edu
Omer Gurewitz‡ gurewitz@bgu.ac.il
David B. Johnson* dbj@cs.rice.edu

*Department of Computer Science, Rice University, Houston, TX, USA
†Systems and Applications R&D Center, Texas Instruments, Dallas, TX, USA
‡Department of Communication Systems Engineering, Ben Gurion University, Israel

ABSTRACT

Medium access control (MAC) protocols for wireless sensor networks face many challenges, including energy-efficient operation and robust support for varying traffic loads, in spite of effects such as wireless interference or even possible wireless jamming attacks. This paper presents the design and evaluation of the *EM-MAC* (*Efficient Multichannel MAC*) protocol, which addresses these challenges through the introduction of novel mechanisms for adaptive receiver-initiated multichannel rendezvous and predictive wake-up scheduling. EM-MAC substantially enhances wireless channel utilization and transmission efficiency while resisting wireless interference and jamming by enabling every node to dynamically optimize the selection of wireless channels it utilizes based on the channel conditions it senses, without use of any reserved control channel. EM-MAC achieves high energy efficiency by enabling a sender to predict the receiver's wake-up channel and wake-up time Implemented in TinyOS on MICAz motes, EM-MAC substantially outperformed other MAC protocols studied. EM-MAC maintained the lowest sender and receiver duty cycles, the lowest packet delivery latency, and 100% packet delivery ratio across all experiments. Our evaluation includes single-hop and multihop flows, as well as experiments with heavy ZigBee interference, constant ZigBee jamming, and Wi-Fi interference.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Access Schemes*

General Terms

Algorithms, Design, Performance

Keywords

Sensor networks, medium access control, duty cycling, energy, multichannel, interference, jamming

MobiHoc'11, May 16–19, 2011, Paris, France.

1. INTRODUCTION

Energy efficiency is crucial in wireless sensor networks. To this end, many energy-efficient wireless MAC protocols (e.g., [3, 6, 20–22]) have been proposed that utilize *duty cycling*, a technique in which each sensor node alternates between active and sleeping states, turning its radio on only periodically; the percentage of time a node is awake is referred to as the node's duty cycle. Duty cycling aids in energy efficiency by reducing *idle listening* and *overhearing*, two of the most significant causes of unnecessary energy consumption in wireless sensor networks. Idle listening refers to a node listening to a wireless channel when there is nothing on the channel to receive, and overhearing refers to a node hearing packets not intended for this node.

Wireless sensor network MAC protocols must operate under a number of significant challenges due to the shared nature of the wireless channel. For example, hidden terminals may cause wireless collisions, and heavy wireless usage in one location may create substantial contention for channel access by other nearby sensor nodes, degrading performance for these sensor nodes as well. Many other sources of wireless interference are also possible, including external interference from wireless transmissions by other types of devices, such as Wi-Fi nodes [10]. In some cases, wireless interference may even be deliberate, such as from a malicious node performing an active jamming attack, transmitting continuously in order to block other nodes' access to the channel [24].

In this paper, we present the design and evaluation of a new *multichannel* energy-efficient MAC protocol, called *EM-MAC (Efficient Multichannel MAC)*, that substantially outperforms other sensor network MAC protocols studied, particularly in cases with wireless interference or jamming. EM-MAC is a *predictive*, asynchronous duty-cycling MAC protocol that utilizes the available multiple orthogonal radio channels common with many types of wireless devices, including the IEEE 802.15.4 (ZigBee) radios widely used on sensor node devices. Unlike existing multichannel energy efficient MAC protocols, EM-MAC uses no *control channel* and enables a node to *dynamically* select the channels it switches among for receiving based on the wireless channel conditions it senses. By effectively utilizing the multiple orthogonal radio channels, EM-MAC is able to avoid using individual channels that are currently heavily loaded or are otherwise undesirable such as due to interference or jamming. Furthermore, by not requiring use of a control channel, EM-MAC avoids concentrating control communication on any channel and avoids the performance degradation that would otherwise result if a control channel itself were heavily loaded or experiencing interference or jamming.

EM-MAC achieves high energy efficiency by enabling senders to accurately predict the *wake-up channel* and *wake-up time* of a

receiver. In particular, each time a node using EM-MAC wakes up, it independently selects its own wake-up time and channel according to a *pseudorandom function*, while avoiding undesirable channels on which it has detected high traffic loads or excessive wireless interference, including channels being actively jammed. The independent pseudorandom wake-up scheduling of EM-MAC aims to spread the traffic load to different channels, reducing wireless collisions caused by nodes waking up at the same time on the same channel. EM-MAC efficiently achieves multichannel rendezvous between a sender and a receiver, even given constraints such as variable hardware or operating system latencies and clock drifts: based on the prediction of a receiver's wake-up channel and wake-up time, a sender using EM-MAC wakes up on the receiver's current wake-up channel right before the receiver does, finishes the packet transmission, and quickly goes back to sleeping state, achieving high energy efficiency through minimizing idle listening and overhearing.

We have implemented EM-MAC under TinyOS on a collection of MICAz motes and report the results of experiments comparing the performance of EM-MAC to that of X-MAC [3], WiseMAC [6], Y-MAC [8], McMAC [17], RI-MAC [21], and PW-MAC [22]. Our evaluation includes single-hop and multihop flows, as well as experiments with heavy ZigBee wireless interference, constant ZigBee jamming, and Wi-Fi interference. In these experiments, EM-MAC substantially outperformed the other MAC protocols studied. In all experiments, EM-MAC maintained 100% packet delivery ratio, with the lowest duty cycle and the lowest packet delivery latency among all protocols studied.

Section 2 of this paper discusses related work. In Section 3, we present the design of EM-MAC, including the techniques used by EM-MAC for dynamic channel selection and precise and quick multichannel rendezvous. Section 4 presents the results of our evaluation of EM-MAC under TinyOS on MICAz motes, including single-hop and multihop flows, as well as experiments with heavy ZigBee interference, constant ZigBee jamming, and Wi-Fi interference. Finally, in Section 5, we present conclusions.

2. RELATED WORK

Many energy-efficient sensor network MAC protocols have been proposed that use only a single radio channel for all transmissions; examples include S-MAC [25], B-MAC [13], X-MAC [3], DW-MAC [20], RI-MAC [21], and PW-MAC [22]. The network throughput with these protocols is limited, however, to the capacity of a single channel, and their performance may be further substantially reduced under conditions of wireless interference or jamming. Compared with these protocols, EM-MAC is not only more robust against such conditions but is also more capable of handling large and dynamic traffic loads by efficiently utilizing multiple channels.

The opportunities possible by utilizing multiple orthogonal radio channels in the context of MAC protocols for wireless *ad hoc networks* using Wi-Fi radios have long been recognized [12]; examples include SSCH [1] and McMAC [17]. As these protocols were designed for a network in which nodes are always-on, however, energy efficiency was not a major concern in their design.

In the context of *sensor networks*, researchers have recently also been exploring techniques at the MAC layer to utilize multiple orthogonal radio channels, such as are available in the IEEE 802.15.4 (ZigBee) transceivers widely used in current sensor networks. Example protocols in this area include Y-MAC [8], A-MAC [5], MMAC [18], and CAM-MAC [11]. In these protocols, prior to each data packet transmission, the sender and receiver node first tune to a dedicated *control channel* to negotiate the channel to use for the data transmission. Negotiating first on a control channel makes it easier for a sender to rendezvous with a receiver for transmitting a data packet, but since no data packet transmission would then be possible without a successful negotiation on the control channel, the available bandwidth of the control channel can become a packet transmission bottleneck. Furthermore, if the control channel is subject to heavy interference such as from ZigBee or Wi-Fi traffic [10] or is jammed by an attacker [24], such protocols would be unable to deliver any data packets. In addition, the energy efficiency of these protocols can deteriorate substantially from waiting for the opportunity to negotiate a data channel using the control channel. In contrast, EM-MAC does not rely on a dedicated control channel and is adaptive to dynamic channel conditions, contributing to the high efficiency of EM-MAC and its resilience to wireless interference and jamming.

Like EM-MAC, the MuChMAC protocol [2] does not use a dedicated control channel; each node in MuChMAC independently decides its channel-switching sequences, which can be deduced by its neighbors. However, unlike EM-MAC, nodes in MuChMAC use a fixed set of channels, without adapting to current channel conditions such as interference or jamming on a channel. In addition, MuChMAC divides time into slots, assuming loose global time synchronization and a fixed upper bound on clock drift in attempting to ensure that sender and receiver at least both wake up within the same slot as each other; MuChMAC does not provide any mechanism to recover in the case in which the sender and receiver fail to rendezvous in this way due to larger clock drifts or variable hardware or operating system latencies. In contrast, EM-MAC operates entirely asynchronously, with no reliance on global time synchronization, and provides an efficient mechanism to quickly rendezvous a sender and a receiver after any rendezvous failure occurs.

Some protocols have attempted to make use of multiple channels by assigning different channels to different nodes in a two-hop neighborhood to avoid potential interference; examples of such protocols include MMSN [27] and TMMAC [26], in which time slots are used to coordinate transmissions. To enable such time slots, these protocols assume precise time synchronization in the network, which is not needed in EM-MAC. Similarly, TMCP [23] partitions a network into different trees and assigns a fixed channel to each tree. This design showed high performance in data collection applications, but the fixed channel assignment to each tree is inefficient in handling dynamic traffic and makes the traffic on a tree vulnerable to wireless interference and jamming attacks as discussed above.

Finally, several other energy-efficient MAC protocols have used various forms of predictive wake-up based on pseudorandom number sequences, similar to that used in EM-MAC.

For example, Cao et al. [4] presented an analytical study of different energy-efficient MAC protocol schemes based on globally synchronized time slots: time is divided into a sequence of frames and each frame is divided into a sequence of these slots. In their "Pseudo-random Staggered On" scheme, each node wakes up as a receiver independently in each slot with probability ψ_r (e.g., $\psi_r = 0.01$), where the decision to wake up or not for each slot is determined by comparing the next number in that node's pseudorandom number sequence to this ψ_r threshold. Through the use of the pseudorandom number sequence, the slot in which any node will be awake to receive can be predicted by a sender that knows and follows the state of that node's pseudorandom number generator. Cao et al. also provided a rough sketch of the operation of a protocol called O-MAC based on this scheme, although in

O-MAC, rather than waking up with an independent probability in each slot, the wake-up time of a receiver is simply generated as the slot number within a frame, based on the next number in the pseudorandom sequence. The MAC protocol of the JAVeLEN system [14] uses predicable receiver wake-up timing similar to the "Pseudo-random Staggered On" scheme of Cao et al. [4]; JAVeLEN divides time into globally synchronized slots, and each node independently wakes up to attempt to receive in each slot depending on the comparison of the next number in that node's pseudorandom sequence against a threshold.

In contrast, nodes in EM-MAC operate asynchronously, without globally synchronized time slots, thus avoiding the complexity and overhead of global, network-wide time synchronization. Each node's pseudorandom number sequence in EM-MAC directly provides the *amount of time* between one wake-up by that node and its next wake-up. In addition, whereas JAVeLEN requires specialized hardware with two separate radios, one for sending a "Hail" to an intended receiver and the other for data packet transmission, EM-MAC operates with only a single commodity radio per node; although the specialized radio hardware of JAVeLEN allows it to operate very efficiently in terms of energy consumption, this design limits its use on simpler, more generally available hardware such as the MICAz motes we used in our implementation of EM-MAC.

To our knowledge, EM-MAC is the first *multichannel* MAC protocol to use the pseudorandom number sequence to predict both the receiver *wake-up time* and the receiver *wake-up channel*. In particular, Cao et al. [4] consider only a single radio channel that is shared by all nodes, and although JAVeLEN uses multiple radio channels at the physical layer, at the MAC layer it treats this as a single logical channel and uses its pseudorandom number sequence only for receiver wake-up *time*. The wake-up time prediction in EM-MAC is similar to our own prior work on PW-MAC [22], but PW-MAC is only a single-channel MAC protocol. EM-MAC also improves on the wake-up time prediction scheme from PW-MAC through features such as our *exponential chase* algorithm for efficiently resynchronizing a sender and receiver whose clocks may have drifted apart.

3. EM-MAC PROTOCOL DESIGN

EM-MAC is a multichannel asynchronous duty-cycling MAC protocol. It does not require nodes to synchronize their clocks, does not use a common control channel, and does not explicitly exchange channel and wake-up schedules. Instead, every node independently decides its own *pseudorandom* channel-switching behavior and wake-up times. A sender rendezvous with a receiver by predicting the receiver's wake-up channel and wake-up time based on the sender's knowledge of the state of the receiver's pseudorandom function used to generate its wake-up channels and times.

EM-MAC achieves very high energy efficiency by enabling a sender to precisely and quickly rendezvous with a receiver. A sender in EM-MAC wakes up shortly before the receiver does on the predicted receiver wake-up channel, completes the packet transmission, and quickly goes back to sleeping state, minimizing idle listening and overhearing. EM-MAC does not require special radio hardware. We have implemented EM-MAC on MICAz motes, which have a single half-duplex radio tunable to any of the 16 orthogonal channels of the IEEE 802.15.4 (ZigBee) protocol.

3.1. Overview

Figure 1 shows an example of the operation of EM-MAC, with time progressing from left to right. Here, only two nodes, *S* and *R*, are shown, with sender node *S* sending data packets to receiver node *R*.

EM-MAC is a *receiver-initiated* MAC protocol; a node sends a wake-up beacon to notify potential senders that it is awake and ready to receive data packets. After receiving a wake-up beacon from a receiver *R*, a node *S* that has a data packet for *R* sends it to *R*. *R* sends an ACK beacon to *S* to acknowledge the data packet receipt and to allow another data packet to be sent to *R* by this or another sender; in this example, no other data packet is available, so *S* and *R* quickly go back to sleep. After *R* wakes up for the second time shown, no node has a data packet waiting to send to *R* and *R* quickly goes back to sleep. Finally, after *R* wakes up again, *S* has another packet for *R* and sends it to *R* in response to this beacon from *R*.

In order to reduce wireless collisions caused by nodes waking up at the same time and on the same channel and to distribute the traffic among the available channels, a node in EM-MAC switches among the channels it selects based on its pseudorandom channel scheduling. In addition, each node in EM-MAC pseudorandomly decides its own wake-up times; a wake-up time of a node is determined from the node's previous wake-up time plus its current pseudorandomly chosen wake-up interval. In particular, for each wake-up, a node invokes its pseudorandom number generator twice: once to compute its next wake-up channel and again to compute its next wake-up interval. In our experiments, node pseudorandom wake-up intervals range between 500 ms and 1500 ms.

In Figure 1, *S* has previously learned the time and pseudorandom number generator information of *R* and is able to predict the wake-up channels and wake-up times of *R*. When *S* has a data packet to send to *R*, *S* wakes up on the predicted wake-up channel of *R* right before the predicted wake-up time of *R*, thereby achieving high energy efficiency by minimizing the idle listening and overhearing.

To avoid using congested channels and to be robust against wireless interference and jamming, a node dynamically modifies the set of channels among which it switches, based on the channel conditions it senses; the details of this mechanism are described in Section 3.2.

For energy-efficient resolution of wireless collisions, EM-MAC extends the collision resolution mechanism of RI-MAC [21], in which a receiver notifies potential senders to retransmit their packets using an increased backoff window via a new beacon once a collision is detected. Different from RI-MAC, EM-MAC puts the senders into sleeping state as long as possible during retransmissions. If the collision resolution mechanism does not resolve the collision before the receiver goes to sleep, a sender in EM-MAC goes to sleep and wakes up again to retransmit at the receiver's next predicted wake-up time, if a pre-defined retry limit has not been reached.

EM-MAC can use any pseudorandom function to generate the channel and wake-up schedule for a node. For example, in our implementation on MICAz motes, we chose to use a *linear congruential generator (LCG)*, since LCGs are efficient in computation and storage. LCG generates a pseudorandom number X_{n+1} as $X_{n+1} = (aX_n + c) \bmod m$, where $m > 0$ is the *modulus*, a is the *multiplier*, c is the *increment*, and X_n is the current *seed*; the generated X_{n+1} becomes the next seed. In our implementation, $m = 65536$; each node's a and c are independently chosen following the principles suggested by Knuth [9], such that the LCG will have a full period (i.e., full cycle) for all seed values. The generation of wake-up channels for a node thus generates all possible channel numbers (in a different pseudorandom order each time) before any channel number is repeated. This design contributes to spreading network traffic to different channels. Each node also uses a different initial seed value X_0, based for example on the node's MAC address. With such design, if two nodes happen to

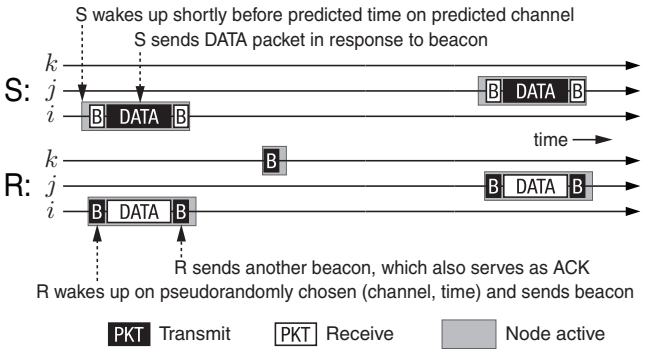


Figure 1: Sender S sends data packets to receiver R using EM-MAC. Only three of the channels are shown here, labeled i, j, and k. At the time of R's second beacon, no node has a packet waiting to send to R.

wake up at the same time and on the same channel, it is unlikely they will do so next time, thus avoiding persistent collisions. The pseudorandom generator used for EM-MAC by a node is private to EM-MAC wake-up scheduling operation so that the sequence of pseudorandom numbers generated is not disturbed by other operations of the node.

A sender predicts the wake-up times and channels of a receiver based on its knowledge of the *prediction state* of the receiver. The prediction state obtained by S for R consists of the multiplier a and increment c of R's pseudorandom number generator, a previous wake-up time and corresponding random seed of R, the channel selection information of R (described in Section 3.2.2), and S's time model for R (enabling S to compute the time of R based on the current time of S; the details of the time modeling used in EM-MAC are described in Section 3.3.1).

If a sender has a packet to send to a receiver but does not have the receiver's prediction state, the sender waits at the first channel. After receiving the wake-up beacon from the receiver, the sender sets a flag in the header of the data packet sent to the receiver, requesting the receiver to embed its prediction state in the ACK packet for the data packet. For a sender unaware of the prediction state of a receiver, its waiting time on a channel for the receiver wake-up beacon should be sufficient to ensure that the receiver will visit the channel at least once during this time. Let N_{ch} and $T_{maxInterval}$ denote the number of channels available and the maximum node wake-up interval, respectively. For MICAz motes, $N_{ch} = 16$. $T_{maxInterval}$ is 1500 ms in our experiments. If the sender does not receive a wake-up beacon from the receiver after waiting $T_{black} + 2 \times N_{ch} \times T_{maxInterval}$ (T_{black} denotes the longest time for which a receiver can skip visiting a channel due to bad channel condition based on the *channel blacklist* mechanism described in Section 3.2), the sender switches to the next channel and repeats the waiting procedure. If the sender is unable to rendezvous with the receiver after waiting on all channels, the sender concludes that the receiver is "unreachable" (e.g., currently powered off or out of wireless transmission range).

After obtaining the prediction state of a receiver, a sender can predict the receiver's future wake-up channels and times. The complete algorithm used by a sender S to predict the wake-up time and channel of a receiver R is given in Section 3.3, but the basic operation of the algorithm is as follows: based on the prediction state of R, S computes every wake-up time and channel of R until it finds a wake-up time of R that is larger than the current time of R plus a small *wake-up advance time*.

EM-MAC can support broadcast operation by sending a broadcast packet to neighboring nodes one-by-one, at each neighbor's individual wake-up time. One method for discovering the neighboring nodes of a node is to let this node periodically stay awake on a channel for $T_{black} + 2 \times N_{ch} \times T_{maxInterval}$ and record the nodes from which it hears a beacon.

3.2. Dynamic Channel Selection

Supplementing the pseudorandom selection of channels described in Section 3.1, each node in EM-MAC uses a dynamic mechanism, based on the channel conditions it senses, to optimize the set of channels over which it switches. With this mechanism, a node avoids choosing channels that are congested or are degraded by interference such as from Wi-Fi devices [10] or from jamming attacks [24]. This dynamic channel selection mechanism spreads the traffic to the available channels and enhances communication robustness, sensor node energy efficiency, and packet delivery efficiency. Our dynamic channel selection algorithm is implemented on MICAz motes, which use a ZigBee radio.

3.2.1. Detecting Channel Conditions

A node in EM-MAC independently collects the channel condition information, as a byproduct of regular transceiving operations, without extra energy consumption. EM-MAC does not send probing packets to determine the channel condition because such proactive channel condition measurement approach increases the node energy consumption and network traffic.

A node maintains for each channel a non-negative "*badness*" metric. When a node in EM-MAC wakes up on a channel to send a wake-up beacon or a data packet, it conducts a CCA (Clear Channel Assessment) check to ensure the channel is idle before beginning the transmission. If the channel is idle, the node sends the packet and decreases that channel's badness metric by 1 (the metric is not allowed to become less than 0). Otherwise, the node conducts a CCA check again after a short random backoff. If the channel is still busy after three such CCA checks, the node increases that channel's badness metric by 2 and goes to sleep. In addition, after a node sends a wake-up beacon, if the CCA check indicates the channel is busy but the node does not receive a valid packet, the node assumes a packet collision may have occurred and resolves the collision by informing the senders to retransmit the packets using an increased backoff window. If collision resolution fails, the node increases its badness metric for this channel by 2. Likewise, if the node sends a data packet but does not receive an ACK for it, the node increases the badness metric of this channel by 2. By updating the channel badness metric in this way, if a channel is congested or many failed transmissions occurred on it, the badness metric of this channel increases; Otherwise, the channel's badness metric decreases.

3.2.2. Multichannel Rendezvous with Blacklisted Channels

Based on the channel badness metrics, a node in EM-MAC selects the set of channels it switches among. Every node maintains its own *channel blacklist* that identifies the channels the node regards as "bad" channels. When the badness metric of a channel is above a threshold C_{bad}, the channel will be added to the node's blacklist.

A node switches among channels based on its pseudorandom channel schedule, except that, if the next pseudorandomly chosen channel is on the node's channel blacklist, the node stays on its current channel (used for its wake-up previous to this one).

The maximum number of channels allowed on a node's blacklist is the total number of available channels minus 1. If all channels

have a badness metric beyond C_{bad}, the "least bad" channel is removed from the blacklist. This limits communication attempts to using this single channel, until other channels are determined to no longer be bad and are removed from the blacklist.

To enable potential senders to learn its blacklisted channels, a node R represents its blacklisted channels using a bitmap and embeds it in its wake-up beacons. Since in our implementation there are 16 channels, the size of the blacklist bitmap is 2 bytes. In other contexts, other protocols have successfully likewise characterized the current status of a channel using a single bit [16].

If R changes its channel blacklist, a sender S learns the updated blacklist of R after receiving a wake-up beacon from R. Before receiving the updated blacklist of R, S may miss a wake-up of R if S waits for R on a channel that R has just added to its channel blacklist. Even in this case, the duty cycle of S should only be slightly increased, as EM-MAC limits the waiting time of S on each channel and puts S into sleep state if it does not receive a wake-up beacon from R at the expected time. The limited waiting time is described in detail in Section 3.3.2.

A node R expires and removes from its channel blacklist the channels that have been on the blacklist for more than T_{black} time and resets the badness metric of such channel to 0. Similarly, a sender S to R also updates its stored channel blacklist from R by expiring and removing channels that have (to S's knowledge) been on the blacklist of R for more than T_{black} time. Once a channel has been removed from the blacklist, a node uses the channel normally. Removing a channel from the blacklist gives a node the opportunity to re-sample the condition of this channel. If this channel is still bad, the badness metric mechanism will cause it to be placed again onto the blacklist.

3.3. Precise and Quick Multichannel Rendezvous

Precise and quick multichannel rendezvous is crucial to high energy efficiency and low delivery latency. EM-MAC achieves this through a comprehensive set of techniques, including an adaptive time modeling technique, the *exponential chase* algorithm, the bounded prediction computation mechanism, and the limited sender waiting mechanism. This section presents the details of these techniques.

3.3.1. Adaptive Time Modeling

Existing work either assumes that the sensor nodes have synchronized clocks or uses a preconfigured clock drift ratio to model the clock rate difference between nodes; however, such mechanisms can lead to senders missing receivers, to a degradation of energy efficiency, and to an increase in packet delivery latency in real sensor networks, as real sensor nodes can have different clock rates that are affected by the environment [7]. For example, WiseMAC models the clock difference between sensor nodes based on an *a priori* preconfigured clock drift ratio. If the preconfigured clock drift ratio is less than the actual clock rate difference between a sender and a receiver, the sender will miss the receiver's wake-ups. If the preconfigured clock drift ratio is much larger than the actual clock rate difference, then the sender will wake up much earlier than the receiver does, unnecessarily increasing the sender's duty cycle and energy consumption.

The adaptive time modeling technique of EM-MAC enables a node S to accurately predict the time of another node R by analyzing observed time information from R and modeling the development of their time difference. A node in EM-MAC does not synchronize its time with other nodes. Instead, a node S models the time of a node R as $y = kx + b$, where x denotes the current time of S, and y denotes the current time of R computed by S. The values k and b denote the clock rate difference and the initial clock difference between the two nodes, respectively. For example, if $k = 1$, we have $y = x + b$, meaning S and R have the same clock rate but have a time difference of b. The time modeling technique of EM-MAC adds no extra energy overhead and does not require special time-synchronization packets, only exploiting the timestamps embedded in prediction states. This technique is simpler, for example, than the wireless sensor network synchronization algorithms reviewed by Sadler and Swami [15], which synchronize the clocks of all nodes in the network based on the time samples containing highly-variable end-to-end packet transmission delays.

When S receives the prediction state from R for the first time, it assumes $k = 1$ and models the time of R using the equation $y = x + b$. If the clocks of S and R run at the same rate, then S does not need to request future prediction state updates from R. If, instead, the clocks of S and R run at different rates, S can detect, when sending a data packet to R, that the actual wake-up time of R differs from its predicted wake-up time. Once this difference approaches S's wake-up advance time, S requests an *on-demand* prediction state update from R by setting a flag in the data packet header; upon receiving the prediction state update from R in the ACK for this data packet, S not only regains its ability to precisely predict R's next wake-up channel and wake-up time but also now has two time samples from R: $y_1 = kx_1 + b$ and $y_2 = kx_2 + b$, in which y_1 denotes the previously received time sample of R and y_2 denotes the newly received time sample of R. Based on these time samples, S is able to compute the value of the parameters k and b, thereby determining how fast the time of R develops relative to the time of S.

Ideally, when S receives the prediction state from R, the time sample in the received prediction state (i.e., y value) is exactly the current time of R. However, hardware latency, operating system latency, carrier sensing, radio backoff, and packet propagation time can cause a gap between when R added its time sample to the prediction state in the packet and when the prediction state is received by S.

To accurately compute the parameters k and b on real sensor hardware, it is important to minimize this gap. We implemented a method on MICAz motes to obtain precise time samples by directly accessing the radio hardware packet transmission RAM buffer and adding the time sample only when the packet is actually being transmitted by the radio. This method removes the error caused by carrier sensing, radio backoff, and operating system latency. Specifically, the hardware interrupt from transmitting the Start Frame Delimiter (SFD) of a packet containing the prediction state indicates that the packet stored in the hardware packet transmission RAM buffer is now being transmitted by the radio. Upon receiving this hardware interrupt, the transmitting node accesses the hardware packet transmission RAM buffer to add its current time to the packet being sent.

3.3.2. Sender Wake-up Time Details

To send a data packet to a receiver, a sender in EM-MAC wakes up before the predicted receiver wake-up time by an amount of time referred to as the *wake-up advance time*, and stays awake at most twice the wake-up advance time (thus centered on the predicted wake-up time). If the sender has not begun to receive the receiver's beacon by this time, the sender goes back to sleep and wakes up at the next predicted wake-up time of the receiver to send the data packet. If the sender is still unable to rendezvous with the receiver in next cycle, it invokes exponential chase algorithm described in Section 3.3.3 to rendezvous with the receiver.

EM-MAC limits the sender waiting time to twice the wake-up advance time in order to maintain a small sender duty cycle even when the sender misses a receiver wake-up. The reason for setting the sender waiting time to twice the wake-up advance time is that the predicted receiver wake-up time can be ahead of or behind the actual receiver wake-up time. The wake-up advance time is a hardware-dependent variable that accounts for the hardware and operating system latency for radio power-up operation, and the time difference between a sender and a receiver. The configuration of the initial wake-up advance time of a sender waiting for a receiver is given in the Section 4.

3.3.3. Exponential Chase Algorithm

The adaptive time modeling technique described in Section 3.3.1 enables a sender to make precise predictions for rendezvousing with a receiver. However, it does not guarantee that a sender can model a receiver time *perfectly* accurately; the actual receiver clock rate and the receiver clock rate modeled by the sender may still be slightly different. If a sender and receiver have not communicated for a long time, the sender may miss a wake-up of the receiver, as even a very small clock rate difference multiplied over a long time can result in a prediction error larger than the normal sender wake-up advance time; if the actual sender or receiver clock rate has changed significantly, for example due to environmental factors, such a prediction error also becomes possible. If a sender misses a receiver wake-up, quickly re-rendezvousing with this receiver is crucial to maintaining a small sender duty cycle and packet delivery latency. EM-MAC introduces the *exponential chase* algorithm to quickly re-rendezvous a sender and a receiver.

After a sender misses a receiver for the second time, the sender invokes the exponential chase algorithm by doubling its current wake-up advance time for this receiver (and thus also the time to wait for the receiver's wake-up beacon after the expected receiver wake-up time). Then the sender computes the receiver's future wake-up channels and wake-up times until finding a wake-up time of the receiver that is at least this wake-up advance time after the receiver's current time. Lastly, the sender wakes up the wake-up advance time before this predicted receiver wake-up time on the corresponding wake-up channel of the receiver, to attempt to rendezvous with the receiver. The sender repeats iterations of this exponential chase algorithm until receiving a wake-up beacon from the receiver. Once the sender successfully rendezvous with this receiver, it resets its wake-up advance time for this receiver to its initial value for future data packets to be sent to this same receiver. Figure 2 shows an example of the operation of exponential chase.

The exponential chase algorithm ensures that a sender is able to rendezvous with a receiver after a finite number of iterations of the algorithm, as the time difference between the sender and the receiver must be finite. Meanwhile, a sender S gives up attempting to send to a receiver R and discards the prediction state of R if S has not been able to rendezvous with R after the wake-up advance time for R in the current exponential chase iteration surpasses T_{giveup} ($T_{giveup} > T_{black} + 2 \times N_{ch} \times T_{maxInterval}$). The reason to limit the sender in this way in the exponential chase algorithm is because the receiver node may have been powered off or be out of wireless transmission range. If, instead, the sender is able to rendezvous with the receiver, after receiving the receiver wake-up beacon, the sender sets a flag in the header of the data packet that it sends to this receiver, requesting the receiver to return its current prediction state in the ACK beacon; the sender thus regains its ability to precisely predict the wake-up times and wake-up channels of this receiver once receiving this prediction state.

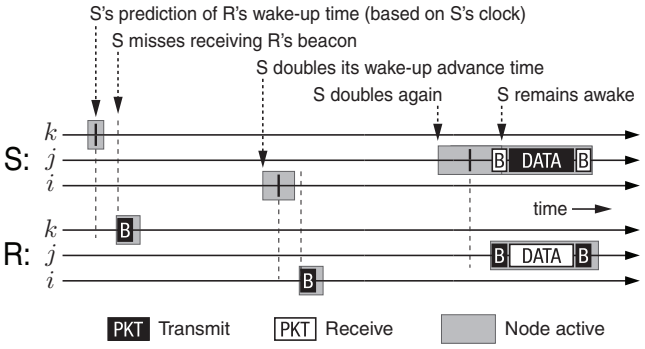


Figure 2: Sender S and receiver R have a large time difference. To rendezvous with R, S exponentially increases its wake-up advance time and the waiting time around the predicted receiver wake-up time, which quickly makes up for the time difference between S and R. Each wake-up time of R predicted by S is marked by a vertical bar during each active time of S.

In EM-MAC, it is not possible for a node S, endlessly, to switch to R's predicted wake-up channel to send a data packet to R, while on each such attempt, at the same time R switches to S's predicted wake-up channel to send a data packet to S. In EM-MAC, two nodes will not wake up at the same time consecutively. In particular, with the limited sender waiting time mechanism described in Section 3.3.2, if S and R fail to rendezvous since they are trying to send to each other, they will switch to sleeping state and at least one of them will wake up before the other on the next attempt, thereby avoiding such endless repeated attempts.

In addition, EM-MAC guarantees that a sender S is able to rendezvous with a receiver R even if they have a large time difference and inconsistent channel blacklist information. A *rendezvous* here means that R sends a wake-up beacon on the channel on which S is awake and waiting, assuming R will send a wake-up beacon after it wakes up. Let $BlackCH(R,R)$ denote the actual channel blacklist of R, and let $BlackCH(R,S)$ denote the channel blacklist of R as known by S.

Case 1: $BlackCH(R,R) = BlackCH(R,S)$ during the period over which S attempts to rendezvous with R. If the time difference between S and R is no larger than T_{giveup}, S will rendezvous with R on a non-blacklisted channel of R after a finite number of iterations of exponential chase algorithm, as illustrated in the example of Figure 2. If the time difference between S and R is larger than T_{giveup}, S is still able to rendezvous with R because T_{giveup} is larger than the time over which R will have visited all non-blacklisted channels. That is to say, during T_{giveup}, R will visit at least once some channel on which S is awake and waiting.

Case 2: $BlackCH(R,R)$ is not always the same as $BlackCH(R,S)$ during the period over which S attempts to rendezvous with R. Because R removes any channel from its blacklist after T_{black}, and since T_{giveup} is larger than T_{black} by $2 \times N_{ch} \times T_{maxInterval}$, R will visit at least once, before the expiration of T_{giveup}, some channel on which S is awake and waiting, even if that channel was initially blacklisted by R. Hence, S will still rendezvous with R.

3.3.4. Bounding Prediction Computation

For existing predictive-wakeup MAC protocols using forms of prediction based on a pseudorandom number sequence (e.g., [2, 17, 22]), if a sender S has not sent a packet to some receiver R for a long time, it can be quite expensive for S to compute the next wake-up time of R. Since values in a pseudorandom sequence must

```
Algorithm PREDICT-WAKEUP(R):
  Output: int ch; {predicted next wake-up channel of R}
  Output: int wakeTime; {predicted next wake-up time of R}
  if S does not have prediction state of R then
    return (ch=RANDOM_CHANNEL(), wakeTime=0); {S wakes up
    now to request the prediction state of R}
  end if
  curTime[R] = CURRENT_TIME(R);{S computes the current time of R}
  if nextWakeupTime[R] > (curTime[R] + wakeAdvance[R]) then
    {if S has computed next wake-up channel and wake-up time of R be-
    fore}
    return (ch=nextWakeupCh[R], wakeTime=nextWakeupTime[R]);
  end if
  while (1) do
    if it is time for R to reset its random number states then
      RESET(randState[R], ID_R);
      continue;
    end if
    randState[R]=RAND_Num(randState[R],ID_R);
    nextWakeupCh[R] = GET_WAKEUP_CHANNEL(randState[R]);
    if nextWakeupCh[R] is a blacklisted channel of R then
      nextWakeupCh[R] = CURRENT_CHANNEL(R);
    end if
    randState[R]=RAND_Num(randState[R],ID_R);
    nextWakeupTime[R]+ = randState[R];
    if nextWakeupTime[R] > (curTime[R] + wakeAdvance[R]) then
      break;
    end if
  end while
  return (ch=nextWakeupCh[R], wakeTime=nextWakeupTime[R]);
```

Figure 3: A node S computes the predicted wake-up channel and wake-up time of another node R.

be generated in-order, all wake-up times and channels of R over this period since S last sent a packet to R must be computed by S.

Since the computational capability of sensor nodes are usually quite limited, EM-MAC introduces a *prediction state resetting* mechanism to bound such potentially expensive prediction computations. With this mechanism, a node resets its pseudorandom number generator state every P seconds. The value P determines the length of a *reset period*. The index of the period corresponding to the time t, denoted as $PI(t)$, is t/P.

Let $T_{previous}(S,R)$ be the last time sender S computed the wake-up channel and wake-up time of a receiver R. With the random number generator resetting mechanism, if $T_{previous}(S,R)$ is before the beginning of the current reset period of R, then instead of computing the next wake-up channel and wake-up time of R from $T_{previous}(S,R)$, S need only compute the random numbers generated by R since the beginning of R's current reset period. This method greatly reduces the prediction computational overhead and bounds the maximum computational overhead to be computing all random numbers generated in a period. S is able to compute all random numbers generated by R in the period $PI(t)$ since S has the prediction state of R and the first random number used for R's wake-up scheduling in the period $PI(t)$ is the $PI(t)$th number generated by R's pseudorandom number generator, with the multiplier of R's pseudorandom number generator being the first seed.

Figure 3 shows the algorithm of computing the wake-up channel and wake-up time of a receiver R.

4. EVALUATION ON MICAz MOTES

To evaluate the realistic performance of EM-MAC, we implemented it on MICAz motes running TinyOS. Each wake-up interval for a node using EM-MAC in our experiments was pseudo-randomly chosen between 500 ms and 1500 ms. On average, a sender generates one new data packet every second. The parameter C_{bad} is configured as 15. T_{black} is set to 100 s and T_{giveup} is set to 150 s. The sender wake-up advance time here is configured as 20 ms, and the random number generator resetting interval is configured as 300 s.

The *duty cycle* metric reported in our evaluation is the percentage of time a node is awake; *sender duty cycle* is the duty cycle of a data packet sender or forwarder, and *destination duty cycle* is the duty cycle of the destination node of a traffic flow. *Packet delivery ratio (PDR)* is the ratio of the number of packets successfully delivered to the destinations to the total number of packets originated by the traffic sources. *Packet delivery latency* is the period between the time a packet is generated by the packet's source node and the time the packet is delivered to the corresponding destination node.

4.1. Performance of Multichannel Rendezvous

To evaluate the multichannel rendezvous performance of EM-MAC, we first studied the time modeling accuracy of the protocol on real hardware. For these experiments, we chose two arbitrary MICAz motes, one as the sender and one as the receiver. For each attempted rendezvous, the sender woke up 20 ms before the predicted receiver wake-up time, on the predicted receiver wake-up channel. The experiments measured the number of times the sender missed the receiver over a period of 6000 seconds. In one of the three experiments, the receiver used the unmodified MICAz clock. In the other two experiments, to create challenging test scenarios, the receiver node's clock was accelerated by 100 ppm and by 200 ppm (parts per million), respectively; it has been reported that clock drift rates for common sensor nodes can normally be between 30 and 100 ppm [19].

Across all three of these experiments, the sender did not miss any of the total of 18,000 receiver wake-ups. Even when the receiver clock was accelerated, the sender was able to accurately predict the receiver time, using the time modeling mechanism in EM-MAC, and to rendezvous with the receiver.

However, the clock rate of a sensor can vary significantly over time, being influenced by environmental factors such as temperature and humidity [7]. For instance, a sensor could be deployed in a location that is hot during the daytime and cold at night. In this case, even with the receiver time modeling technique in EM-MAC, it is possible for the sender to miss the receiver. To address this possibility, upon missing a receiver, the sender in EM-MAC uses the exponential chase algorithm, as described in Section 3.3.3, to re-rendezvous with the receiver.

In order to evaluate the effectiveness of the exponential chase algorithm, in the following set of experiments, we repeatedly increased the receiver clock rate until the sender failed to rendezvous with the receiver when sending a data packet, and then measured how quickly the sender re-rendezvoused with the receiver using exponential chase; in these measurements, EM-MAC's *on-demand* prediction state update mechanism described in Section 3.3.1 was disabled, in order to ensure that a rendezvous failure and the need for exponential chase would eventually occur. Since existing work does not have a mechanism to re-rendezvous a sender and a receiver efficiently on multiple channels after the sender misses the receiver, we implemented two other possible multichannel rendezvous methods and compared them with the exponential chase algorithm of EM-MAC. In one method, which we refer to as *waiting-at-one-channel*, the sender stays awake on its current channel until it again rendezvous with the receiver. In the other method, which we refer to as *rotating-different-channel*, the sender stays awake on

Table 1: Performance of Rendezvous Alternatives

Rendezvous Method	Rendezvous Latency (ms)	Sender Duty Cycle
EM-MAC Exponential Chase	902	6.7%
Waiting-at-one-channel	9095	100.0%
Rotating-different-channel	8153	100.0%

each channel in turn, beginning with its current channel, remaining awake on each channel for a time equal to the average node wake-up interval (1 s, in these experiments) and then switching to the next channel, until the node again rendezvous with the receiver. In these experiments, all three methods use the same 20 ms sender wake-up advance time. Each experiment was conducted 30 times.

The average result for each multichannel rendezvous method in these experiments is presented in Table 1. The right two columns of Table 1 show, respectively, the time it took for the sender to re-rendezvous with the receiver and the sender duty cycle during the process of rendezvousing with the receiver. Compared with the other two multichannel rendezvous methods, the EM-MAC exponential chase algorithm enabled the sender to quickly re-rendezvous with the receiver while maintaining a high energy efficiency.

To further confirm the performance of our exponential chase algorithm, we also conducted another set of experiments in which we deliberately created a range of different amounts of error in the sender's prediction of the receiver's wake-up time, up to 160 ms of prediction error. As the initial sender wake-up advance time is configured as 20 ms, for all sender prediction errors less than this amount, no iterations of the exponential chase algorithm were required for the sender and receiver to rendezvous. Above this amount, each time the amount of error was doubled, one additional iteration of the algorithm was used by our EM-MAC implementation: for errors above 20 ms but below 40 ms, only 1 iteration of the exponential chase algorithm was used; up to 80 ms, only 2 iterations were used; and up to 160 ms, only 3 iterations were used.

Finally, in evaluating the performance of multichannel rendezvous, we compared the performance of EM-MAC, Y-MAC, and PW-MAC in sensor networks with a larger clock rate difference between nodes. In these experiments, one MICAz node is used as the sender and one as the receiver. The clock rate of the receiver node was accelerated by 3000 ppm, which creates a challenging test scenario for these protocols since this clock rate acceleration is significantly higher than the reported clock drift rates of 30 to 100 ppm for common sensor nodes [19].

EM-MAC substantially outperformed both Y-MAC and PW-MAC in these experiments. The duty cycle of both Y-MAC and PW-MAC was more than 3 times greater than that of EM-MAC. The delivery latency of Y-MAC was more than 8 times larger than that of EM-MAC. Compared with their performance without such large clock rate differences (shown later in Table 3 in the "No Interference" column), the performance of Y-MAC and PW-MAC is substantially worse in these experiments, since the large clock rate difference between the sender and the receiver caused the sender with these two protocols to miss the receiver wake-ups and stay awake longer to retransmit the packets, enlarging the delivery latency and sender duty cycle. In contrast, the time modeling technique of EM-MAC enables the sender to accurately compute the receiver clock rate based on the time samples from the receiver, contributing to the small sender duty cycle and delivery latency of EM-MAC, even in such challenging conditions.

Table 2: Performance With Large Clock Rate Difference

Protocol	Sender Duty Cycle	Delivery Latency (ms)	PDR
EM-MAC	5.3%	611	100.0%
Y-MAC	32.8%	5130	100.0%
PW-MAC	19.8%	762	100.0%

4.2. Performance in Multihop Networks

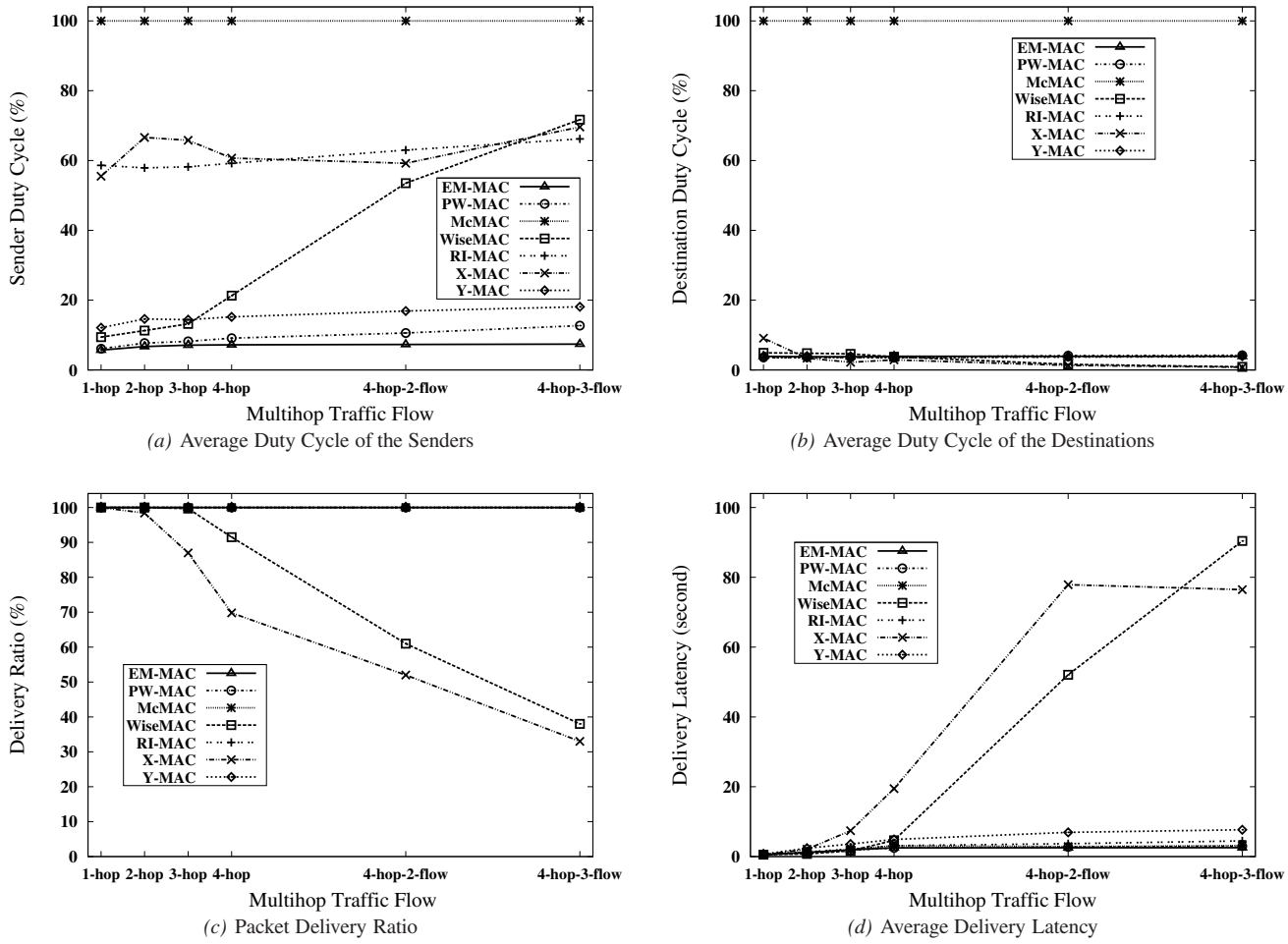
We next compare the performance of EM-MAC with that of two multichannel MAC protocols (McMAC and Y-MAC) and four single channel MAC protocols (PW-MAC, WiseMAC, RI-MAC, and X-MAC) in multihop networks. The experiments were conducted on a testbed of 15 MICAz motes, arranged as 3 rows of 5 nodes each, with all nodes within transmission range of each other so that different traffic flows interfere with one another. Each row of nodes could be operated as a separate traffic flow of up to 4 hops (5 nodes, including the source node); in our experiments, we evaluated each protocol's performance using from 1 to 3 of these traffic flows.

The source of each traffic flow generated one 28-byte data packet every second on average. Each experiment lasted 500 seconds and was conduct three times; the results presented are the average of the three runs. The number of slots in a frame of Y-MAC is 8, the same configuration as that previously published for Y-MAC [8]. The $P_{deviate}$ parameter of McMAC is configured as 1 since McMAC performed best under this configuration in the experiments. The *big slot* length of McMAC is configured as 1 s.

Figure 4 shows the results of these experiments. On the left of each graph are shown the results for a single flow, ranging from 1 hop to 4 hops in length, and on the right of each graph are shown the results for 2 and 3 simultaneous flows, respectively, each of 4 hops in length.

Figure 4(a) shows the sender duty cycle. Other than the destination, every node on a traffic flow is a *sender* since it either generates data packets or forwards data packets. The energy efficiency of a multihop traffic flow is determined mainly by the sender duty cycle rather than by the destination duty cycle since there are multiple senders but only one destination on a multihop traffic flow. The sender duty cycle of McMAC was 100% since nodes in McMAC are always on. EM-MAC achieved the smallest sender duty cycle under all traffic conditions. PW-MAC (a single-channel protocol) ranks second in terms of sender duty cycle, since it also attempts to predict the receiver wake-up time.

As the hop-length of a traffic flow and the number of traffic flows increase, so does the sender duty cycle performance advantage of EM-MAC over the other protocols. When there are three 4-hop flows, the sender duty cycle of EM-MAC was only 58% of that of the second-best-performer, PW-MAC. RI-MAC had a large sender duty cycle because, with RI-MAC, a sender wakes up whenever it has a data packet to send and only switches to sleeping state after the data packet transmission completes. Similarly, X-MAC had a large sender duty cycle. The sender duty cycle of WiseMAC grows as the hop-length and the number of traffic flows increase, since the sender-initiated preambles of WiseMAC lead to wireless collisions and WiseMAC does not have an energy-efficient mechanism to conduct retransmissions. Y-MAC had a larger sender duty cycle than did EM-MAC, since in Y-MAC multiple nodes contend for the same control channel to send their first packet, which often leads to more collisions than EM-MAC, which tries to spread out the contention among different channels. In addition, when collisions occur, Y-MAC does not have an energy-efficient mechanism to retransmit packets.



(a) Average Duty Cycle of the Senders

(b) Average Duty Cycle of the Destinations

(c) Packet Delivery Ratio

(d) Average Delivery Latency

Figure 4: Performance of EM-MAC, PW-MAC, McMAC, WiseMAC, RI-MAC, X-MAC and Y-MAC on a multihop MICAz testbed. The hop-length of the traffic flow varies from 1 to 4 and the number of 4-hop traffic flows ranges from 1 to 3.

Figure 4(b) shows the destination duty cycle. The destination duty cycle of McMAC was 100% since McMAC is an always-on MAC protocol. The destination duty cycle of the other protocols was small (about 5%) because the traffic destination nodes in these protocols only periodically wake up to receive data packets without sending any data packets. When the number of senders increases, the destination duty cycle of X-MAC and WiseMAC decrease and become smaller than that of the other protocols, as packet transmissions from multiple senders cause substantial packet collisions in both protocols and the destination nodes go to sleep when no valid packets are received.

Figure 4(c) shows the packet delivery ratio and Figure 4(d) shows the packet delivery latency in these experiments. EM-MAC achieved the smallest packet delivery latency and 100% packet delivery ratio in all experiments. PW-MAC achieved 100% packet delivery ratio and the second best packet delivery latency. Y-MAC had a larger delivery latency than EM-MAC and PW-MAC. In Y-MAC, once a data packet collision occurs, the sender has to wait until next cycle to retransmit the packet, causing the larger delivery latency. The delivery latency of McMAC is correlated to the $P_{deviate}$ parameter. In these experiments, $P_{deviate}$ is configured as 1; when $P_{deviate}$ was configured between 0.2 and 0.8 as suggested by the authors of McMAC [17], the delivery latency of McMAC was larger. As the number of concurrent transmitters increases, the delivery latency of X-MAC and WiseMAC grow quickly since the sender-initiated preambles generated by these protocols tend to collide under high traffic loads.

EM-MAC and PW-MAC outperformed the other protocols in the experiments shown in Figure 4. The sender duty cycle of EM-MAC is 40% less than that of PW-MAC. The performance advantage of EM-MAC over PW-MAC becomes even larger as the traffic load increases. For example, when the traffic load was doubled and there were three 4-hop traffic flows in the network, both protocols delivered all packets successfully, but the sender duty cycle of EM-MAC was only 33% of that of PW-MAC, and the delivery latency of EM-MAC was only 39% of that of PW-MAC.

EM-MAC achieves a very high energy and packet delivery efficiency mainly due to the following two reasons. First, EM-MAC dynamically optimizes node channel selections, greatly reducing wireless collisions and the energy spent on packet retransmissions. Second, EM-MAC achieves quick and accurate multichannel rendezvous, thereby minimizing sender idle listening and overhearing. In contrast, the other protocols are not as efficient in utilizing the channel resources. With these protocols, when there were many concurrent packet transmissions, a significant amount of wireless collisions occurred, enlarging the delivery latency and sender duty

Table 3: Performance With Wireless Interference and Jamming

Protocol	No Interference			ZigBee Interference			ZigBee Jamming			Wi-Fi Interference		
	Sender Duty Cycle	Delivery Latency (ms)	PDR	Sender Duty Cycle	Delivery Latency (ms)	PDR	Sender Duty Cycle	Delivery Latency (ms)	PDR	Sender Duty Cycle	Delivery Latency (ms)	PDR
EM-MAC	5.7%	573	100.0%	6.3%	625	100.0%	6.4%	675	100.0%	6.4%	670	100.0%
Y-MAC	12.1%	695	100.0%	25.8%	57283	100.0%	100.0%	∞	0.0%	91.0%	252506	35.2%
PW-MAC	6.1%	568	100.0%	39.2%	1506	100.0%	100.0%	∞	0.0%	76.0%	309197	12.0%

cycle since senders had to stay awake to conduct retransmissions. Furthermore, compared with EM-MAC, these protocols are not as accurate in rendezvousing sender and receiver, resulting in a larger idling listening and overhearing.

4.3. Performance with Wireless Interference

Sensor communications are subject to wireless interference, including possible various wireless jamming attacks [24]. In this section, we compare the performance of EM-MAC, Y-MAC, and PW-MAC under ZigBee interference, under ZigBee jamming, and under Wi-Fi interference. Although PW-MAC is a single-channel MAC protocol, it performed the best among the single-channel protocols evaluated in multihop networks in Section 4.2, so we include it as well in our evaluation in this section.

In these experiments, there is one MICAz mote sender and one MICAz mote receiver. We compare the duty cycle and packet delivery ratio of these two nodes in four cases: with no interference, with ZigBee interference, with ZigBee jamming, and with Wi-Fi interference. In the *ZigBee interference* case, there are four additional MICAz transmitting nodes, each independently transmitting a 100-byte packet of random content every 20 ms; each of these packets is sent without carrier sensing, equivalent, for example, to the case of a busy channel with hidden transmitters. In the *ZigBee jamming* case, a single MICAz node transmits back-to-back packets, without carrier sensing, in an attempt to monopolize the wireless channel. In the *Wi-Fi interference* case, one laptop sends Wi-Fi traffic to another laptop connected to a Wi-Fi router. This Wi-Fi traffic flow is generated by the *iperf* network testing tool, which sends TCP traffic as fast as possible between the two laptops; the average TCP throughput achieved by iperf between these two laptops was about 3.6 Mbps. The two MICAz mote sender and receiver being measured were placed 1 meter from the laptop that transmits Wi-Fi traffic; the distance from the Wi-Fi router to the motes was 3 meters, and the Wi-Fi channel overlapped with the ZigBee channel used by the sender and receiver motes. The laptops and the motes were in an open network with background traffic. Each experiment in these four cases lasted 500 seconds and was conducted three times. Table 3 shows the average performance results of EM-MAC, PW-MAC, and Y-MAC in these experiments.

Under Wi-Fi interference, compared with their performance with no interference, the performance of Y-MAC and PW-MAC degraded significantly, because Wi-Fi packets are transmitted with a much higher power level than the power level used for transmitting ZigBee packets and because Wi-Fi transceivers do not recognize ZigBee packets, creating a heavy interference to the nearby ZigBee traffic. PW-MAC uses only one channel. In Y-MAC, a sender does not dynamically choose channels to avoid interfered channels and only has one time slot in each cycle to send to a receiver. Hence, Y-MAC and PW-MAC suffered from a long packet delivery latency under the interference, as the interference often caused a packet to be transmitted many times before it was delivered successfully to the receiver. The substantial increase of the sender duty cycle of these two protocols is because the sender stayed awake for a long time to conduct packet retransmissions.

Under the ZigBee jamming, the packet delivery ratio of Y-MAC was 0, because the jammer monopolized the control channel, leaving the control channel to be busy all the time and causing the sender to be unable to send any packet on the jammed control channel. Y-MAC suffered a high sender duty cycle when ZigBee jamming was present since the sender stayed awake to wait for the opportunity to send the data packets and receive the time synchronization information from the receiver. For similar reasons, the delivery ratio and the sender duty cycle of PW-MAC under ZigBee jamming was 0 and 100%, respectively.

Under ZigBee interference, Wi-Fi interference, and ZigBee jamming, EM-MAC delivered all packets successfully with high energy efficiency and small delivery latency. Compared with the sender duty cycle of Y-MAC and PW-MAC, the sender duty cycle of EM-MAC (at 6.4%) was quite low. The delivery latency of EM-MAC under Wi-Fi interference and ZigBee jamming was less than 1% of that of the other two protocols.

EM-MAC significantly outperformed the other protocols under wireless interference and jamming because its dynamic channel selection mechanism enables a node to detect and avoid using the channels that were very busy or were having a large number of occurrences of packet collisions and transmission failures.

5. CONCLUSION

In this paper, we have presented EM-MAC, an energy-efficient dynamic multichannel MAC protocol for wireless sensor networks. EM-MAC achieves high energy efficiency by enabling senders to accurately predict the *wake-up channel* and *wake-up time* of receivers. EM-MAC uses no control channel and enables a node to dynamically select the channels it switches among based on the channel conditions it senses. By effectively utilizing multiple orthogonal radio channels, EM-MAC is able to avoid using channels that are currently heavily loaded or are otherwise undesirable such as due to interference or jamming. Furthermore, by not requiring use of a control channel, EM-MAC avoids concentrating control communication on any channel and avoids the performance degradation that would otherwise result if a control channel itself were heavily loaded or experiencing interference or jamming. EM-MAC provides precise and quick multichannel rendezvous, even given challenges such as changing clock drift rates between nodes and variable hardware or operating system latencies.

We have implemented EM-MAC in TinyOS and evaluated its performance in single-hop and multihop networks of MICAz motes. Compared with other representative energy efficient MAC protocols, EM-MAC achieved the lowest duty cycle and smallest packet delivery latency, while maintaining 100% delivery ratios in all experiments. In addition, the higher the network traffic loads, the larger the performance margin of EM-MAC over other protocols was observed. For example, when there were 3 concurrent 4-hop traffic flows, EM-MAC at packet sources and all forwarding nodes achieved a low duty cycle that was only 33% of that of

the second-best-performer PW-MAC. At the same time, EM-MAC achieved a very low packet delivery latency, only 39% of that of PW-MAC. We have also evaluated the performance of EM-MAC under Wi-Fi interference and ZigBee jamming. The experimental results show that EM-MAC reduces duty cycle and delivery latency by more than a factor of 10 over other protocols while maintaining a 100% packet delivery ratio. In the experiments where sensors have different degrees of clock drift rates, EM-MAC substantially outperformed other protocols in energy efficiency and packet delivery latency.

Acknowledgments

We thank our shepherd, Craig Partridge, for guiding us through the paper revisions. We also thank the anonymous reviewers for their valuable suggestions on improving the paper.

REFERENCES

[1] Paramvir Bahl, Ranveer Chandra, and John Dunagan. SSCH: Slotted Seeded Channel Hoppingc for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (MobiCom 2004)*, pages 216–230, October 2004.

[2] Joris Borms, Kris Steenhaut, and Bart Lemmens. Low-Overhead Dynamic Multi-channel MAC for Wireless Sensor Networks. In *Proceedings of 7th European Conference on Wireless Sensor Networks (EWSN 2010)*, pages 81–96, February 2010.

[3] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks. In *Proceedings of the 4th ACM Conference on Embedded Network Sensor Systems (SenSys 2006)*, pages 307–320, November 2006.

[4] Hui Cao, Kenneth W. Parker, and Anish Arora. O-MAC: A Receiver Centric Power Management Protocol. In *Proceedings of the 14th IEEE International Conference on Network Protocols (ICNP 2006)*, pages 311–320, November 2006.

[5] Prabal Dutta, Stephen Dawson-Haggerty, Yin Chen, Chieh-Jan (Mike) Liang, and Andreas Terzis. Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless. In *Proceedings of the 8th ACM Conference on Embedded Network Sensor Systems (SenSys 2010)*, pages 1–14, November 2010.

[6] Amre El-Hoiydi and Jean-Dominique Decotignie. WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks. In *Proceedings of the First International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS 2004)*, pages 18–31, July 2004.

[7] Saurabh Ganeriwal, Deepak Ganesan, Hohyun Shim, Vlasios Tsiatsis, and Mani B. Srivastava. Estimating Clock Uncertainty for Efficient Duty-Cycling in Sensor Networks. In *Proceedings of the Third ACM Conference on Embedded Network Sensor Systems (SenSys 2005)*, pages 130–141, November 2005.

[8] Youngmin Kim, Hyojeong Shin, and Hojung Cha. Y-MAC: An Energy-Efficient Multi-channel MAC Protocol for Dense Wireless Sensor Networks. In *Proceedings of the 2008 International Conference on Information Processing in Sensor Networks (IPSN 2008)*, pages 53–63, April 2008.

[9] Donald E. Knuth. The Linear Congruential Method. In *The Art of Computer Programming, Third Edition, Volume 2: Seminumerical Algorithms*, pages 10–26 (Section 3.2.1). Addison-Wesley, 1997.

[10] Chieh-Jan Mike Liang, Nissanka Bodhi Priyantha, Jie Liu, and Andreas Terzis. Surviving Wi-Fi Interference in Low Power ZigBee Networks. In *Proceedings of the 8th ACM Conference on Embedded Network Sensor Systems (SenSys 2010)*, pages 309–322, November 2010.

[11] Tie Luo, Mehul Motani, and Vikram Srinivasan. Cooperative Asynchronous Multichannel MAC: Design, Analysis, and Implementation. *IEEE Transactions on Mobile Computing*, 8(3):338–352, March 2009.

[12] Jeonghoon Mo, Hoi-Sheung Wilson So, and Jean Walrand. Comparison of Multichannel MAC Protocols. *IEEE Transactions on Mobile Computing*, 7(1):50–65, January 2008.

[13] Joseph Polastre, Jason Hill, and David Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proceedings of the Second ACM Conference on Embedded Network Sensor Systems (SenSys 2004)*, pages 95–107, November 2004.

[14] J. Redi, S. Kolek, K. Manning, C. Partridge, R. Rosales-Hain, R. Ramanathan, and I. Castineyra. JAVeLEN — An Ultra-Low Energy Ad Hoc Wireless Network. *Ad Hoc Networks*, 6:108–126, January 2008.

[15] Brian M. Sadler and Ananthram Swami. Synchronization in Sensor Networks: An Overview. In *Proceedings of the 2006 Military Communications Conference (MILCOM 2006)*, pages 1–6, October 2006.

[16] C. Santivanez, R. Ramanathan, C. Partridge, R. Krishnan, M. Condell, and S. Polit. Opportunistic Spectrum Access: Challenges, Architecture, Protocols. In *Proceedings of the 2nd Annual International Wireless Internet Conference (WICON 2006)*, August 2006.

[17] Hoi-Sheung Wilson So, Jean Walrand, and Jeonghoon Mo. McMAC: A Parallel Rendezvous Multi-Channel MAC Protocol. In *Proceedings of the 2007 IEEE Wireless Communications and Networking Conference (WCNC 2007)*, pages 334–339, March 2007.

[18] Jungmin So and Nitin Vaidya. Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using a Single Transceiver. In *Proceedings of the Fifth International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2004)*, pages 222–233, May 2004.

[19] Philipp Sommer and Roger Wattenhofer. Gradient Clock Synchronization in Wireless Sensor Networks. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks (IPSN 2009)*, pages 37–48, 2009.

[20] Yanjun Sun, Shu Du, Omer Gurewitz, and David B. Johnson. DW-MAC: A Low Latency, Energy Efficient Demand-Wakeup MAC Protocol for Wireless Sensor Networks. In *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2008)*, pages 53–62, 2008.

[21] Yanjun Sun, Omer Gurewitz, and David B. Johnson. RI-MAC: A Receiver Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys 2008)*, pages 1–14, November 2008.

[22] Lei Tang, Yanjun Sun, Omer Gurewitz, and David B. Johnson. PW-MAC: An Energy-Efficient Predictive-Wakeup MAC Protocol for Wireless Sensor Networks. In *Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM 2011)*, pages 1305–1313, April 2011.

[23] Yafeng Wu, John A. Stankovic, Tian He, Jiakang Lu, and Shan Lin. Realistic and Efficient Multi-Channel Communications in Wireless Sensor Networks. In *Proceedings of the 27th IEEE International Conference on Computer Communications (INFOCOM 2008)*, pages 1193–1201, April 2008.

[24] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy Wood. The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks. In *Proceedings of the Sixth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2005)*, pages 46–57, May 2005.

[25] Wei Ye, John Heidemann, and Deborah Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of the 21st IEEE International Conference on Computer Communications (INFOCOM 2002)*, pages 1567–1576, June 2002.

[26] Jingbin Zhang, Gang Zhou, Chengdu Huang, Sang H. Son, and John A. Stankovic. TMMAC: An Energy Efficient Multi-Channel MAC Protocol for Ad Hoc Networks. In *Proceedings of the 2007 IEEE International Conference on Communications (ICC 2007)*, pages 3554–3561, June 2007.

[27] Gang Zhou, Chengdu Huang, Ting Yan, Tian He, John A. Stankovic, and Tarek F. Abdelzaher. MMSN: Multi-Frequency Media Access Control for Wireless Sensor Networks. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, pages 1–13, April 2006.