RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks

Yanjun Sun[*] yanjun@cs.rice.edu
Omer Gurewitz[†] gurewitz@cse.bgu.ac.il
David B. Johnson[*] dbj@cs.rice.edu

[*]Department of Computer Science, Rice University, Houston, TX, USA
[†]Department of Communication Systems Engineering, Ben Gurion University, Israel

ABSTRACT

The problem of *idle listening* is one of the most significant sources of energy consumption in wireless sensor nodes, and many techniques have been proposed based on *duty cycling* to reduce this cost. In this paper, we present a new *asynchronous* duty cycle MAC protocol, called *Receiver-Initiated MAC* (*RI-MAC*), that uses receiver-initiated data transmission in order to efficiently and effectively operate over a wide range of traffic loads. RI-MAC attempts to minimize the time a sender and its intended receiver occupy the wireless medium to find a rendezvous time for exchanging data, while still decoupling the sender and receiver's duty cycle schedules. We show the performance of RI-MAC through detailed *ns-2* simulation and through measurements of an implementation in TinyOS in a testbed of MICAz motes. Compared to the prior asynchronous duty cycling approach of X-MAC, RI-MAC achieves higher throughput, packet delivery ratio, and power efficiency under a wide range of traffic loads. Especially when there are contending flows, such as bursty traffic or transmissions from hidden nodes, RI-MAC significantly improves throughput and packet delivery ratio. Even under light traffic load for which X-MAC is optimized, RI-MAC achieves the same high performance in terms of packet delivery ratio and latency while maintaining comparable power efficiency.

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, asynchronous duty cycling, energy, ns-2, TinyOS

SenSys'08, November 5–7, 2008, Raleigh, North Carolina, USA.

1. INTRODUCTION

Nodes in a wireless sensor network (WSN) must typically operate unattended for long periods of time on limited battery capacity, and *idle listening* is one of the most significant sources of energy consumption in sensor nodes. In idle listening, a node waits with its radio turned on, listening for a possible packet to be received even though none has been sent.

Many solutions to the problem of idle listening have been proposed utilizing the technique of *duty cycling* [19, 23]. In this technique, each sensor node turns its radio on only periodically, alternating between active and sleeping states. For example, with a 10% duty cycle, a node has its radio on only 10% of the time, resulting in substantial energy savings. In the active state, a node is able to transmit or receive data, but in the sleeping state, the node completely turns off its radio to save energy.

Contention-based duty cycle MAC protocols in the literature can be roughly categorized into two categories: *synchronous* and *asynchronous*. Synchronous approaches such as S-MAC [23], T-MAC [7], RMAC [8], and DW-MAC [20] synchronize neighboring nodes in order to align their active or sleeping periods. Neighbor nodes start exchanging packets only within the common active time, enabling a node to sleep for most of the time within an operational cycle without missing any incoming packet. This approach greatly reduces idle listening time, but the required synchronization introduces extra overhead and complexity, and a node may need to wake up multiple times if its neighbors are on different schedules.

Existing *asynchronous* approaches such as B-MAC [19], X-MAC [3], and WiseMAC [9], on the other hand, allow nodes to operate independently, with each node on its own duty cycle schedule. Such protocols typically employing low power listening (LPL), in which, prior to data transmission, a sender transmits a *preamble* lasting at least as long as the sleep period of the receiver. When the receiver wakes up and detects the preamble, it stays awake to receive the data. These protocols achieve high energy efficiency and remove the synchronization overhead required in synchronous duty cycle approaches. However, they are mainly optimized for light traffic loads, and we found that they become less efficient in latency, power efficiency, and packet delivery ratio as traffic load increases, due to their long preamble transmissions. WiseMAC attempts to improve efficiency by reducing the duration of preamble transmission, but this improvement requires nodes to maintain a fixed wakeup schedule and depends on frequent, regular communication to the same neighbors.

In asynchronous protocols, preamble transmission in LPL-based protocols may occupy the medium for much longer than actual data transmission. Such long preamble transmission from a sender

could prevent all neighboring nodes with pending data from transmitting their data. As these nodes have to wait until the medium is not occupied, some of them could experience significant delay. This is often the case under bursty or high traffic load such as due to convergecast [25] and correlated-event workload traffic [13], where multiple sensors that have detected the same event send their reports to the sink node or to a node that does data aggregation [11]. As traffic in a WSN can be quite dynamic, depending on the events being sensed and the sensing application and protocols being used, an ideal WSN MAC protocol should perform well under a wide range of traffic loads, including high loads and bursty traffic.

In this paper, we present a new asynchronous duty cycle MAC protocol, called *Receiver-Initiated MAC* (*RI-MAC*). RI-MAC attempts to minimize the time a sender and its intended receiver occupy the medium for them to find a rendezvous time for exchanging data, while still decoupling the sender and receiver's duty cycle schedules as B-MAC and X-MAC do.

RI-MAC differs from prior work in asynchronous duty cycle MAC protocols in how the sender and receiver reach a rendezvous time. In RI-MAC, the sender remains active and waits silently until the receiver explicitly signifies when to start data transmission by sending a short *beacon* frame. As only beacon and data transmissions occupy the medium in RI-MAC, with no preamble transmissions as in LPL-based protocols, occupancy of the medium is significantly decreased, making room for other nodes to exchange data.

We believe this is the first attempt to apply the idea of receiver-initiated transmission to duty cycle MAC protocols for ad hoc wireless sensor networks. By coordinating neighboring nodes using beacons in RI-MAC, a receiver adaptively increases channel utilization as traffic load increases, allowing RI-MAC to achieve high throughput, packet delivery ratio, and power efficiency under a wide range of traffic loads.

The contributions of this work are as follows:

- We present a new asynchronous duty cycle MAC protocol, called *RI-MAC*, employing receiver-initiated transmissions, in order to efficiently and effectively operate over a wide range of traffic loads.
- Due to the receiver-initiated design, RI-MAC not only substantially reduces overhearing, but also achieves lower collision probability and recovery cost than do B-MAC and X-MAC.
- We have implemented RI-MAC in TinyOS and evaluate it in a small testbed network of sensor nodes. We also implemented RI-MAC in the *ns-2* network simulator for evaluations in larger networks.
- RI-MAC significantly improves throughput and packet delivery ratio, especially when there are contending flows such as bursty traffic or transmissions from hidden nodes.
- Even under light traffic loads for which X-MAC is optimized, RI-MAC achieves the same high performance in terms of packet delivery ratio and latency while maintaining comparable power efficiency.

The rest of this paper is organized as follows. In Section 2, we discuss related work in duty cycle MAC protocols for sensor networks. Section 3 presents the detailed design of RI-MAC, including our implementation of it in TinyOS, and Section 4, presents an evaluation of RI-MAC using both *ns-2* simulation and our implementation in TinyOS in a testbed of MICAz motes. Finally, in Section 5, we present conclusions.

2. RELATED WORK

The idea of receiver-initiated transmission in a MAC protocol is not new, but we make the first attempt to combine this idea together with duty cycling in the context of MAC protocols for ad hoc wireless sensor networks, where power efficiency is a major concern. Garcia-Luna-Aceves et al. proposed a receiver-initiated collision-avoidance scheme [12] for general wireless networks, where collision is a major concern but power efficiency is of lesser importance.

Receiver-initiation has previously been applied to sensor networks in the PTIP (Periodic Terminal Initiated Polling) mechanism [10], but only for infrastructure WSNs, where each sensor node is in range of an access point, and access points are assumed to be energy unconstrained. With PTIP, a sensor node periodically wakes up and sends a poll packet to an access point with which the node is associated. If the access point has buffered any packets when the node was sleeping, the access point starts sending those packets to the node upon receiving the poll. The type of WSN assumed for PTIP is very different from a typical ad hoc WSN, where multihop packet delivery can be common and most sensor nodes have limited battery capacity. In addition, the PTIP mechanism was designed only for packets being sent *from* an access point to a sensor node.

Another receiver-initiated mechanism, known as Low Power Probing (LPP), was recently introduced in the Koala system [18]. Koala is designed for reliably downloading bulk data from all sensor nodes, for applications with no real-time requirements. All downloads in Koala are initiated by the gateway or gateways, allowing the nodes to sleep most of the time until the gateway's download initiation. With LPP, each node periodically broadcasts a short probe packet requesting an acknowledgment. If an acknowledgment is received, the node remains active and starts waking up other nodes by acknowledging their probes; otherwise the node goes back to sleep. The LPP mechanism in Koala differs from RI-MAC in both objective and design. In particular, LPP is used in Koala only for waking up all sensor nodes for a download and is not involved in the actual data transfer during a download. As such, features of RI-MAC such as back-to-back data transmission and collision detection and recovery were not discussed in LPP.

In sensor network MAC protocols not using receiver-initiation, B-MAC [19] and X-MAC [3] are representative asynchronous duty cycle-based protocols. In B-MAC, each node periodically wakes up to check if there is any activity currently on the wireless channel. If so, the node remains active to receive a possible incoming packet. Prior to DATA frame transmission, a sender transmits a long "wakeup signal," called a preamble, which lasts longer than the receiver's sleep interval. This policy ensures that the receiver will wake up at least once during the preamble, allowing each node to wake up or sleep based on its own schedule. B-MAC is very energy efficient under light traffic because a node spends only a very short period of time in checking channel activity at each scheduled wakeup time. However, a node with B-MAC may wake up and remain awake due to channel activity, only to, in the end, receive one or more DATA frames actually destined for other nodes.

X-MAC solves this overhearing problem in B-MAC by using a *strobed* preamble that consists of sequence of *short preambles* prior to DATA transmission, as illustrated in Figure 1. In this and similar figures in this paper, the period of time during which a node is active is indicated by a solid gray background, frame reception by a node is indicated by black text on the gray background, and frame transmission by a node is indicated by white text on a dark background. The target address is embedded in each short preamble, which not only helps irrelevant nodes to go to sleep immediately but also allows the intended receiver to send an

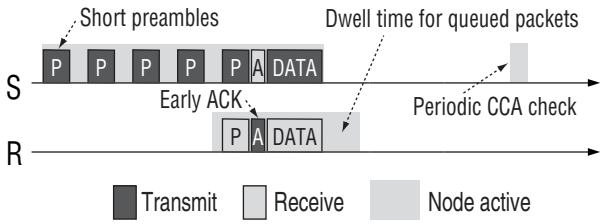


Figure 1: Operation of X-MAC, including the *strobed preamble* and *early acknowledgment*. During a scheduled wakeup time, a node does a CCA (clear channel assessment) check that is longer than the gap between two short preambles.

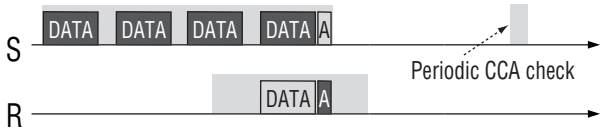


Figure 2: The variation of X-MAC implemented in the UPMA package in TinyOS. The strobed preamble is replaced by a chain of DATA frame transmissions.

early ACK to the sender so that the sender stops preamble transmission and starts transmitting the DATA frame immediately. In this way, X-MAC saves energy by avoiding overhearing while reducing latency almost by half on average. After receiving a DATA frame, a receiver in X-MAC stays awake for a duration equal to the maximum backoff window size to allow queued packets to be transmitted immediately. We refer to this duration as the *dwell time* in the rest of this paper.

The UPMA (Unified Power Management Architecture for Wireless Sensor Networks) package [15] implemented a variation of X-MAC in TinyOS, in which the DATA frame itself is used as the short preamble, as illustrated in Figure 2. This strategy simplifies implementation and helps a sender to determine whether the DATA is successfully delivered from the ACK from the receiver. In the rest of this paper, we refer to this variation of X-MAC as X-MAC-UPMA.

B-MAC and X-MAC achieve high power efficiency under light traffic load, but their preamble transmissions occupies the wireless medium for a long time until DATA is delivered, making them less efficient in case of contending traffic flows. In contrast, a sender in our RI-MAC protocol does not occupy the medium until the intended receiver is ready for receiving, by using receiver-initiated transmission. This property allows RI-MAC not only to achieve comparable performance to X-MAC under light traffic load, but to handle a wide range of traffic loads more efficiently. In addition, the receiver-initiated transmission makes RI-MAC more efficient in detecting collisions and recovering lost DATA frames.

WiseMAC [9] is similar to B-MAC, but a sender in WiseMAC efficiently reduces the length of the wakeup preamble by exploiting the sampling of the schedules of its direct neighbors. In effect, although individual nodes are not synchronized in waking up at the same time as each other, a node does synchronize with its neighbors in learning the wakeup schedules of those neighbors to which it is sending data. To efficiently enable this learning, a node receiving a DATA frame includes in the following ACK frame the remaining time until its next sampling time. With this information, and taking possible clock drifts into account, the sender for its next DATA frame to this receiver estimates when the receiver will wake up next, and starts transmitting its preamble just before then. The resulting shortened preamble greatly helps to save energy and improve channel utilization. However, WiseMAC, as with B-MAC, suffers from the possibility of simultaneous transmissions from hidden nodes, due to the similar preamble sampling techniques they use. In addition, each node with WiseMAC must maintain the same regular wakeup schedule over time, allowing problems such as starvation due to repeated collisions between competing nodes that wake up at the same time over and over again.

Synchronized duty cycle MAC protocols, such as S-MAC [23], T-MAC [7], RMAC [8], and DW-MAC [20], also achieve great energy efficiency in WSNs; so too, hybrid approaches such as SCP [24]. The major difference between RI-MAC and these MAC protocols is that RI-MAC does not require any synchronization, thus saving the overhead and complexity of clock synchronization. Even though no node occupies the medium for a long time in these synchronized duty cycle MAC protocols, it is still difficult for contending flows to finish their transmissions within a single cycle. Specifically, the time window during which transmission is allowed is usually very short in these protocols, as neighboring nodes' wakeup times are synchronized. Once one flow acquires the medium, other flows usually have to wait until next cycle, as their receivers might have gone to sleep when the medium becomes idle. Therefore, RI-MAC has the potential to handle contending flows, and thus bursty traffic, more efficiently and effectively.

3. RI-MAC DESIGN

In this section, we describe the design of the RI-MAC protocol. After an overview of the protocol, we discuss different details RI-MAC's design and conclude with a discussion of how we implemented RI-MAC in TinyOS.

3.1. Overview

Figure 3 gives an overview of the operation of RI-MAC, in which a DATA frame transmission is always initiated by the intended receiver node of the DATA. In RI-MAC, each node periodically wakes up based on its own schedule to check if there are any incoming DATA frames intended for this node. After turning on its radio, a node immediately broadcasts a beacon if the medium is idle, announcing that it is awake and ready to receive a DATA frame. A node with pending DATA to send, node *S* in this figure, stays active silently while waiting for the beacon from the intended receiver *R*. Upon receiving the beacon from *R*, node *S* starts its DATA transmission immediately, which will be acknowledged by *R* with another beacon. Note that this ACK beacon's role is twofold: first, it acknowledges the correct receipt of the sent DATA frame, and second, it invites a new DATA frame transmission to the same receiver. If there is no incoming DATA after broadcasting a beacon, the node goes to sleep, as *S* does later in the figure.

RI-MAC significantly reduces the amount of time a pair of nodes occupy the medium before they reach a rendezvous time for data exchange, compared to the preamble transmission in B-MAC and X-MAC. This short medium occupation time enables more contending nodes to exchange DATA frames with their intended receivers, which helps to increases capacity of the network and thus potential throughput. More importantly, this increase is *adaptive*, by letting a beacon serve both as an acknowledgment to previously received DATA and as a request for the initiation of the next DATA transmission, as discussed in detail in Section 3.2.

In RI-MAC, medium access control among senders that want to transmit DATA frames to the same receiver is mainly controlled by the receiver. This design choice makes RI-MAC more efficient in detecting collisions and recovering lost DATA frames than B-MAC

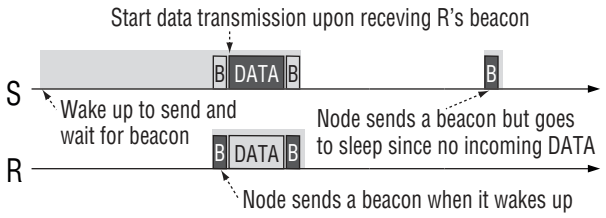


Figure 3: Overview of RI-MAC. Each node periodically wakes up and broadcasts a beacon. When node *S* wants to send a DATA frame to node *R*, it stays active silently and starts DATA transmission upon receiving a beacon from *R*. Node *S* later wakes up but goes to sleep after transmitting a beacon frame since there is no incoming DATA frame.

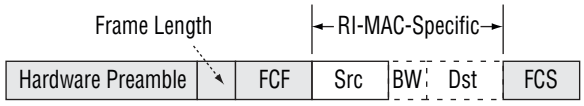


Figure 4: The format of an RI-MAC beacon frame for an IEEE 802.15.4 radio. Dashed rectangles indicate optional fields. The Frame Length, Frame Control Field (FCF), and Frame Check Sequence (FCS) are fields from IEEE 802.15.4 standard.

and X-MAC when the senders are hidden to each other, which can be common in ad hoc sensor networks. As discussed in Section 3.4, after transmitting a beacon, a receiver detects collisions within the duration of the backoff window specified in the beacon, which is much shorter than the delay of a sleep interval needed in B-MAC and X-MAC.

RI-MAC also reduces overhearing, as a receiver expects incoming data only within a small window after beacon transmission. Together with the lower cost for detecting collisions and recovering lost DATA frames, RI-MAC achieves higher power efficiency, especially when the network load increases. Even under light traffic load, which is the worst case for RI-MAC for power efficiency, RI-MAC still shows comparable performance to X-MAC in our simulation and experimental evaluation on MICAz motes. RI-MAC still decouples the sender's and receiver's duty cycle schedules as do B-MAC and X-MAC, which removes the overhead of synchronization compared to synchronous duty cycle MAC protocols.

3.2. Beacon Frames

A beacon frame in RI-MAC always contains a *Src* field, which is the address of the source transmitting node of the beacon. We call a beacon with *only* a Src field a *base* beacon. A beacon can also include two optional fields, depending on the roles the beacon serves: *Dst*, for destination address, and *BW*, for backoff window size. The RI-MAC beacon frame format for an IEEE 802.15.4 radio is illustrated in Figure 4 as an example.

A node that receives a beacon can determine which fields are present in the beacon by looking at the size of the beacon; with an IEEE 802.15.4 radio, size of a beacon is saved in the Frame Length field. A beacon in RI-MAC can play two simultaneous roles: as an acknowledgment to previously received DATA, and as a request for the initiation of the next DATA transmission, as illustrated in Figure 5. After node *R* wakes up and senses clear medium, *R* transmits a base beacon. If the medium is busy, *R* does a backoff and attempts to transmit the beacon later. After receipt of the first DATA frame from *S* in the figure, in the following beacon transmission by *R*, the Dst field is set to *S* to indicate that this beacon also serves as the acknowledgment for the DATA received from *S*. Similar to ACK

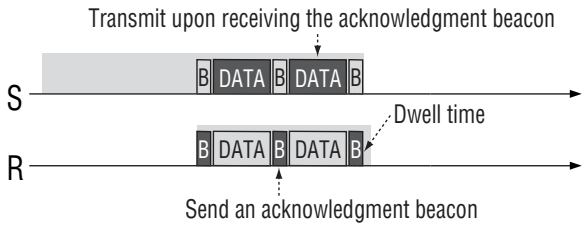


Figure 5: The dual roles of a beacon in RI-MAC. A beacon serves both as an acknowledgment to previously received DATA and as a request for the initiation of the next DATA transmission to this node.

transmission in IEEE 802.11, transmission of this acknowledgment beacon starts after SIFS delay, regardless of medium status. Nodes other than *S* ignore the Dst field in the beacon and treat it as a request for the initiation of a new data transmission. The use of the BW field in a beacon is discussed in detail in Section 3.4.

The duty cycle in RI-MAC is controlled by a parameter called the *sleep interval*, which determines how often a node wakes up and generates a beacon to poll for pending DATA frames. Suppose a sleep interval of L is used in some WSN. After a node generates a beacon, the interval before the next beacon generation is set to a random value between $0.5 \times L$ and $1.5 \times L$. In this way, we attempt to minimize the possibility that beacon transmissions from two nodes become coincidentally synchronized.

3.3. Dwell Time for Queued Packets

After successfully receiving a DATA frame, a node remains active for some extra time in order to allow queued packets to be sent to it immediately, as shown in Figure 5. We refer to this time as the *dwell time*. Unlike in X-MAC, where the dwell time is set to a fixed value of the maximum backoff window, the dwell time in RI-MAC adapts to the number of contending senders. The duration of the dwell time is defined as the BW value from the last beacon plus SIFS and the maximum propagation delay. Since the BW in a beacon is automatically adjusted based on channel collisions observed by a node as discussed in detail next, so is the dwell time. The fewer contending senders and thus the fewer collisions, the shorter the dwell time. This self-adaptation helps RI-MAC using the shortest waiting time possible under light channel contention while avoiding collisions under heavy channel contention.

3.4. DATA Frame Transmissions from Contending Senders

The challenges in handling transmissions from an unpredictable number of contending senders are twofold:

- minimize the active time of a receiver for power efficiency; and
- minimize the cost for collision detection and recovery of lost data, whether or not senders are hidden to each other.

To meet these goals in RI-MAC, a receiver employs beacon frames to coordinate DATA frame transmissions from contending senders, as shown in Figure 6. The BW field in a beacon specifies the backoff window size senders should use when they contend for the medium. If a received beacon does not contain a BW field (i.e., a base beacon), senders for this receiver should start transmitting DATA without backing off. If a beacon contains a BW field, each sender does a random backoff using the BW as the backoff

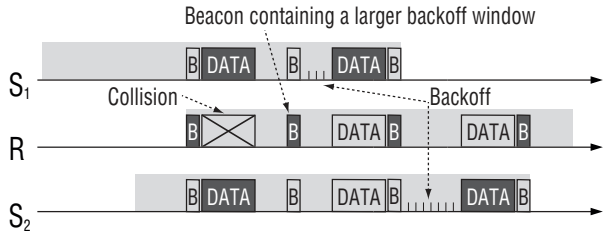


Figure 6: DATA frame transmission from contending senders in RI-MAC. For the first beacon, the receiver *R* requests senders (here, S_1 and S_2) to start transmitting DATA immediately upon receiving the beacon. If a collision is detected, *R* sends another beacon with increased BW value to request that senders do a backoff before their next transmission attempt.

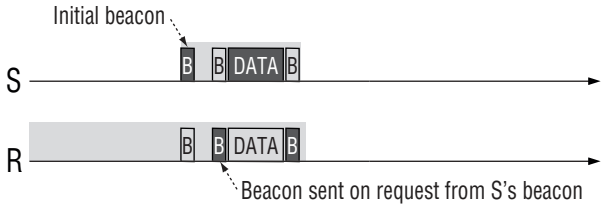


Figure 7: RI-MAC *beacon-on-request*. When node *S* wakes up for transmitting a pending DATA frame, it sends a beacon with the Dst field set to the destination of the pending DATA. If the destination node *R* is already active, *R* in response transmits a beacon to enable *S* to begin DATA transmission immediately.

window size over which to choose the actual backoff. The receiver increases the value of the BW field upon detecting collisions.

If a node cannot start data transmission as soon as it receives a beacon, prior to actual DATA transmission, a sender should make sure that the medium has been idle for at least T_p time using CCA (clear channel assessment) checks. The CCA checks prevent a sender from starting DATA transmission while the intended receiver is generating an acknowledgment beacon to a DATA frame just received from another sender. The time T_p here is set to SIFS plus the maximum propagation delay. If a node needs more time to generate and send an acknowledgment beacon, such as a software ACK used in TinyOS, T_p should be increased correspondingly, as described in Section 3.8.

After waking up, a node always broadcasts a base beacon with no BW field. We made this design choice to optimize RI-MAC for the most common cases of a typical WSN where there is light or no traffic most of the time. By enforcing all senders with pending DATA frames to transmit immediately, we attempt to minimize time for the node to determine whether or not there is incoming DATA. The shorter this duration, the less energy is used at each wakeup. In this way, we attempt to minimize energy consumption if the network is idle most of the time. The duration can be very short, as it is the the maximum round trip propagation delay plus radio switch delay (SIFS in IEEE 802.15.4). If the receiver detects no channel activity within this duration, the receiver goes to sleep immediately. Although a base beacon could lead to concurrent DATA transmissions to a same receiver, we found that they do not necessarily lead to collisions in our experimental implementation on MICAz motes [6], due to the presence of capture effect in the CC2420 radio [4]. This feature makes it possible for one sender to successfully transmit a packet to the receiver even if the transmission overlaps with others, especially when senders have different distances to the receiver (and thus different received signal strengths) [16, 17].

3.5. Collision Detection and Retransmissions

By coordinating DATA frame transmissions at receivers, RI-MAC greatly reduces the cost for detecting collisions and recovering lost DATA frames compared to B-MAC and X-MAC. As a sender can transmit DATA frame only upon receiving a beacon, and since the backoff window size is explicitly controlled by the intended receiver, the receiver knows the maximum delay before a DATA frame's arrival. This delay can be calculated from the BW value in the previous beacon. The receiver need only detect the Start of Frame Delimiter (SFD) to learn of an incoming frame. If no SFD is detected in time, while some channel activity is detected by the CCA (clear channel assessment) check, the receiver will decide that there was a collision and will generate another beacon with a larger BW value. In RI-MAC, this new beacon is transmitted after the longest possible DATA transmission has finished so that all senders' radios are already in receive mode. Prior to transmitting the beacon, a node does a random backoff to avoid possible repeated collisions with beacons from another node.

After detecting a collision, a receiver calculates the new BW value that will be used in the next beacon, by employing some backoff strategy such as binary exponential backoff (BEB) in IEEE 802.11 or Sift [14, 21], depending on the density of a network. BEB is used in our implementation in TinyOS, as we found it adapts to networks of different densities and resolves collisions efficiently in RI-MAC in our evaluations.

As RI-MAC initiates transmissions at the receiver, retransmission in RI-MAC is significantly different from that in sender-initiated approaches such as IEEE 802.11. In RI-MAC, a receiver plays the major role in retransmission control by managing the timing and number of beacon transmissions. If the BW value has reached the maximum backoff window size, or if the receiver keeps detecting collisions after a number of consecutive beacon transmissions, the receive goes to sleep without further attempts. The corresponding senders also become involved in retransmission control, because a sender could miss receiving a beacon either because of collisions or poor channel conditions. Thus, a sender maintains a retry count for each DATA frame. If no beacon has been received from the intended receiver within a time span 3 times as long as the sleep interval, the sender increases the current retry count by 1. In addition, the sender increases this retry count if no acknowledgment beacon is received within the maximum backoff window after the sender transmitted a DATA frame following receipt of a beacon. When the retry count reaches a predefined retry limit, the sender cancels the transmission of the DATA frame.

3.6. Beacon-on-Request

It is possible that the intended receiver node for some sender is already active when the sender wakes up to transmit a DATA frame to it. An optimization, called *beacon-on-request*, is for this sender, after waking up for DATA transmission, to broadcast a beacon following a CCA check, as illustrated in Figure 7. In this beacon, the sender *S* sets the Dst field to the receiver's address, *R*. If the receiver *R* happens to be active, it generates a beacon in response after some random delay longer than the BW announced in the received beacon from *S*. This beacon generated by the receiver on request of the sender allows the sender to transmit the pending DATA frame immediately, rather than waiting until the next scheduled beacon transmission by *R*.

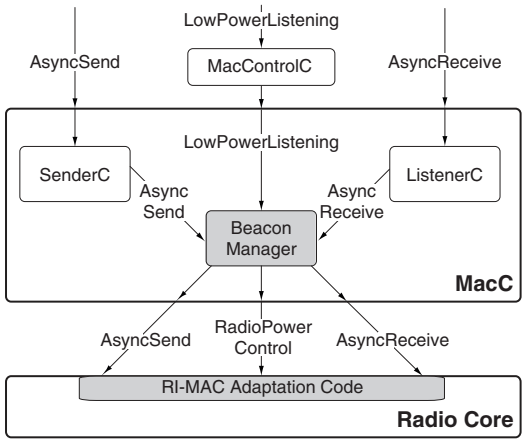


Figure 8: Composition of RI-MAC within the UPMA framework in TinyOS.

3.7. Broadcast Support

RI-MAC can easily support broadcast DATA frame transmission in one of two ways, either by transmitting the DATA as a unicast transmission to each neighbor of the sender node; or by repeatedly transmitting the DATA frame back-to-back for a time equal to the sleep interval, as is done in X-MAC-UPMA. Both approaches have advantages and disadvantages, and a hybrid algorithm that adaptively chooses one of them would likely outperform either single approach. In this paper, we focus on optimizing unicast traffic, and due to space limitations, we leave further investigation and experimental work on broadcast traffic as future work.

3.8. RI-MAC Implementation in TinyOS

We implemented our RI-MAC protocol under the UPMA framework [15] in TinyOS on a network of MICAz sensor motes. The composition of RI-MAC under the UPMA framework in our implementation is shown in Figure 8. We implemented RI-MAC for the CC2420 radio, which is a packetizing radio used in popular MICAz and TelosB motes, although the code can be ported to motes with streaming radios such as the CC1000 [5] as well.

The *BeaconManager* module in Figure 8 performs most of the functionality of RI-MAC, including beacon generation, radio power control, wakeup/sleep scheduling, and retransmission control.

We also added some code to the radio core module of TinyOS, indicated by *RI-MAC Adaptation Code* in the figure. This adaptation code is introduced mainly for two purposes.

First, this adaptation code preloads a DATA frame into the CC2420 TX buffer. In this way, the DATA transmission can start immediately when a desired beacon arrives. This preloading helps to reduce the time a receiver node needs for detecting if there is incoming DATA after a beacon transmission. In our implementation on MICAz motes, after a node sends a beacon, the node needs to wait only 3.75 ms, listening to the medium, in order to detect whether or not there is an incoming packet. A beacon in our implementation is processed entirely in software, as the beacon frame is not supported directly by the CC2420 hardware. With hardware support, this waiting time of 3.75 ms could be further reduced.

Second, the RI-MAC adaptation code starts contiguous CCA (clear channel assessment) checks immediately after a beacon transmission and counts the number of consecutive CCA checks that show busy medium. Suppose that after transmitting a beacon, a packet has not arrived within the expected arrival time that is proportional to the BW field in the beacon transmitted. The node will generate another beacon if the CCA checks indicate busy medium, or will go to sleep otherwise. In particular, on our MICAz motes, if at least 20 consecutive CCA checks indicate busy medium during this time, the RI-MAC adaptation code notifies the BeaconManager of a collision; the BeaconManager then generates another beacon with a larger BW value, if necessary.

As the beacon frame is not part of the IEEE 802.15.4 standard and thus is not directly supported by the CC2420 radio, we turn off hardware *address recognition* in the CC2420 and use a reserved frame type for beacon frames. To minimize our footprint to existing TinyOS code, we use a frame with only the CC2420 header (`cc2420_header_t` in TinyOS) as a beacon. Thus, a beacon is 12 bytes without the preceding hardware preamble, although the size of a base beacon could be implemented to be only 6 bytes, as discussed in Section 3.2.

To account for software processing delays on the MICAz motes, we also adjusted some parameters of RI-MAC in our implementation. A mote may experience some delays before transmitting consecutive packets in the queue, such as post-processing of a transmitted packet, moving a queued packet to the MAC layer, and loading the packet to the hardware buffer. Therefore, in our implementation, we added an extra 10 ms to the dwell time defined in RI-MAC to account for these delays. As an acknowledgment beacon is generated entirely by software in our implementation, T_p, defined in Section 3.4, is set to 2.5 ms, based on our measurements. If a beacon were processed in hardware, this time could be much shorter.

4. EVALUATION

We evaluated RI-MAC both in the *ns-2* network simulator and in an implementation in TinyOS on MICAz motes. We use simulations to explore RI-MAC's performance in a wide variety of networks, especially large network topologies which are hard to deploy and experiment with. As a protocol may not perform in the real world exactly as it does in simulation, for example due to the simplified physical layer models used in *ns-2* [1], we also evaluated RI-MAC in a small testbed network of MICAz motes running TinyOS; our experimental results match our results obtained in simulation and further verify RI-MAC's performance advantages over existing protocols. Since Klues et al. [15] have implemented X-MAC-UPMA on real motes and shown that X-MAC-UPMA outperforms B-MAC and SCP, in this paper, we compared RI-MAC only against X-MAC and X-MAC-UPMA.

4.1. Simulation Evaluation

In our simulation evaluation of RI-MAC, we used version 2.29 of the *ns-2* network simulator, using the standard combined free space and two-ray ground reflection radio propagation model commonly used with *ns-2*. Each sensor node is simulated with a single omnidirectional antenna.

Table 1 summarizes the key parameters we used to simulate the radio of each sensor node. Most of these parameters are from the data sheet of CC2420 radio [4], which is used in popular motes such as MICAz and TelosB. The RSSI sampling delay for CC2420 was reported by Ye et al. [24]; we use this delay as the time for a single CCA (clear channel assessment) check, i.e., the delay before actual transmission starts after a STXONCCA command is strobed [4]. The transmission range and carrier sensing range depend on many factors such as transmission power, antenna, and environment. In *ns-2*, the transmission range and the carrier sensing range are modeled after the 914MHz Lucent WaveLAN radio, which is not typical for a sensor node, but we used these *ns-2* default parameters since measurements have shown that similar proportions of the carrier

Table 1: Simulation Radio Parameters

Bandwidth	250 Kbps	Size of Hardware Preamble	6 B
SIFS	192 μs	Size of ACK	5 B
Slot time	320 μs	CCA Check Delay	128 μs
Tx Range	250 m	Carrier Sensing Range	550 m

Table 2: Simulation MAC Protocol Parameters

	X-MAC	X-MAC-UPMA	RI-MAC
Backoff Window	32	32	0 – 255
Retry Limit	0 or 5	0 or 5	5
Special Frame	Short Preamble	—	Beacon
Special Frame Size	6 B	—	6 – 9 B
Dwell Time	10.5 ms	100 ms	Variable

sensing range to the transmission range are also observed in some state-of-art sensor nodes [2].

Table 2 summarizes the MAC protocol parameters we used in our simulations. Backoff strategy and retransmission have not been explicitly discussed in prior work [3, 15], as X-MAC is optimized for light traffic load. We use 32 as the initial backoff window and 8 as the congestion backoff window, which are the default values used in the UPMA package distributed with TinyOS [22]. In our RI-MAC implementation, a receiver adjusts the BW value in each beacon using a binary exponential backoff (BEB) that takes values of 0, 31, 63, 127, and 255 in our evaluation. The backoff window size for beacon transmission is fixed at 32 slots in RI-MAC.

Although retransmission was not included in X-MAC's original design (none was specified in X-MAC's published design [3, 15, 22]), for fair comparison with RI-MAC in which retransmission is included, we evaluated X-MAC and X-MAC-UPMA both with and without retransmission in our simulations. When retransmission was enabled, we used 5 as the retry limit. The way in which an undecodable signal that is higher than the CCA threshold should be handled was also not explicitly discussed for X-MAC [3], but this occurrence could be common in a large network. Therefore, in our simulated X-MAC, a node turns off its radio if the medium has been idle for a time that is longer than the gap between short preambles. We achieved this by starting a timer that does CCA checks every 20 ms, and each CCA check lasts longer than the gap between short preambles. The time 20 ms was used because that is the wake time used in X-MAC's evaluation [3]. In X-MAC-UPMA, a node that has detected busy medium turns off its radio if no packet is received within 100 ms, according to the code in the UPMA distribution. In our simulated X-MAC-UPMA, similar to the original X-MAC design, only the first preamble in a sequence of short preambles is subject to backoff before transmission (i.e., when the `RESEND_WITHOUT_CCA` option is used in the UPMA package).

In our simulations, a short preamble in X-MAC consists of a Frame Control Field (FCF), destination address, and Frame Check Sequence (FCS). Each of these fields is 2 bytes, resulting in a short preamble of 6 bytes plus the leading 6-byte hardware preamble. A base beacon has the same length and format, except that the address of the transmitting node is in the beacon instead of the destination address. If a beacon also serves as an acknowledgment, or if the BW field is included, a beacon can be 7, 8, or 9 bytes. Dwell time is defined as the maximum backoff window size in X-MAC; we use 10.5 ms, a slightly longer duration, to account for SIFS and propagation delays. The distributed UPMA code uses 100 ms as its default dwell time. Dwell time in RI-MAC is variable, as it is defined as the backoff window for senders (the BW field in a beacon) plus SIFS and propagation delays.

To simplify our evaluation, we do not include routing traffic in the simulations and assume that there is a routing protocol deployed to provide the shortest path between any two nodes. We also ensure that no network used in our simulations is partitioned.

As energy consumption of different radios varies significantly, even in the same radio state [24], we report effective duty cycle in evaluating power efficiency, as done in prior work [3, 15]. The sleep interval for all three MAC protocols is 1 second, and the initial wakeup time of each node was randomized in our evaluation. Note that the sleep interval is an expected value in RI-MAC, as RI-MAC randomizes intervals of sleep time to avoid synchronized beacon transmissions from neighboring nodes. In our evaluation, data payload size was always 28 bytes, the default value in the UPMA package.

We compared X-MAC, X-MAC-UPMA, and RI-MAC in three types of networks: clique networks, a 49-node (7×7) grid network, and random networks. We did not use beacon-on-request in the clique networks, as no multihop communication takes place in these networks; in all other networks, beacon-on-request is used.

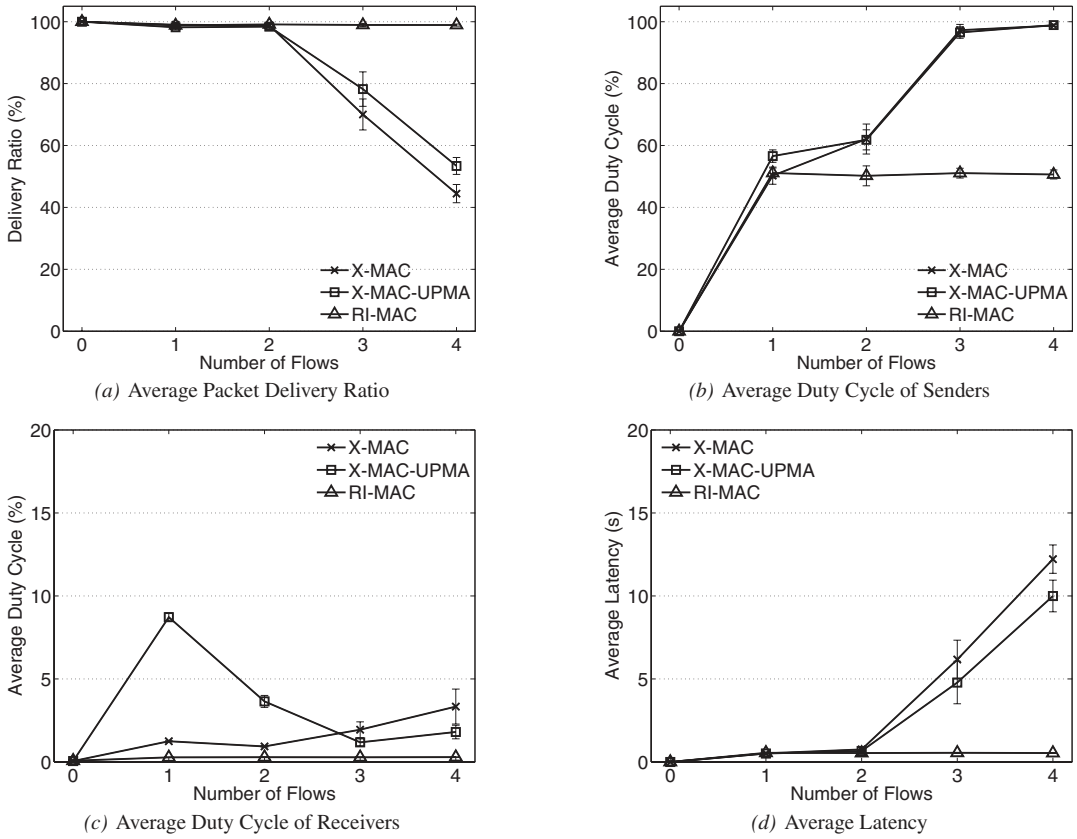
4.1.1. Results in Clique Networks

We discuss first the evaluation of X-MAC, X-MAC-UPMA, and RI-MAC in clique networks, such that all nodes in the network are within transmission range of each other. We varied the traffic load by varying the number of independent flows in the network, with no flow sharing source or destination node with any other flow. In each clique network, the total number of nodes in the network is twice the number of flows. For each flow, the source node starts to generate packets 10 seconds after the beginning of the simulation and generates new packets with an interval between two successive packet generations uniformly distributed between 0.5 and 1.5 seconds. At the beginning of the simulation, each node randomly chooses a time between 0 and 10 seconds as its next wakeup time. In this way, we randomize the wakeup/sleep schedule of each node. The recipient nodes count the number of packets received successfully over the course of 50 seconds. If a packet still resides in any queue or is still being transmitted at the end of the 50-second measurement, the packet is not counted as a delivered packet.

The results for our clique network simulations are shown in Figure 9, where each average value is calculated from the results of 10 random runs. Error bars show the 95% confidence interval. In Figure 9, a value of 0 for number of flows indicates the case in which there is no traffic and just a single node in a network, and thus all energy consumption is due to periodic wakeups of this single node.

Figure 9(a) shows the packet delivery ratios achieved by X-MAC, X-MAC-UPMA, and RI-MAC with increasing number of contending flows in the clique networks. Delivery ratios with RI-MAC are always close to 100%, indicating that total throughput achieved with RI-MAC increases linearly with the increasing traffic load. X-MAC and X-MAC-UPMA deliver most of the given load when there are no more than 2 flows, but their delivery ratios drop quickly beyond 2 flows. This sharp decline is not due to collisions, as all nodes can hear each other. Rather, it is because preamble transmissions in X-MAC and X-MAC-UPMA saturate the network, resulting in a large number of queued packets. When there are 4 flows in a clique network, RI-MAC improves delivery ratio and thus throughput by about 100% compared to X-MAC and X-MAC-UPMA.

The average duty cycles of senders and receivers corresponding to Figure 9(a) are shown in Figure 9(b) and Figure 9(c), respectively. In addition to the improved delivery ratios, RI-MAC saves more energy when there are multiple flows in a clique network,



(a) Average Packet Delivery Ratio

(b) Average Duty Cycle of Senders

(c) Average Duty Cycle of Receivers

(d) Average Latency

Figure 9: Performance comparison in clique networks with contending flows in simulation. The total number of nodes is 1 for 0 flows, and is twice the number of flows otherwise.

compared to X-MAC and X-MAC-UPMA. With 1 flow, senders (Figure 9(b)) show around 50% duty cycle with all protocols, as it takes a sender half a sleep interval to reach its intended receiver, on average. The duty cycles with RI-MAC remain at around 50% with increasing flows, but those with X-MAC and X-MAC-UPMA increase quickly to almost 100% when there are 4 flows. This increase in X-MAC and X-MAC-UPMA is because a sender with pending DATA must do congestion backoff when the medium is occupied by a preamble transmission from another flow. If the corresponding receiver wakes up before the medium becomes idle, the sender must wait until the receiver's next wakeup. If the medium is sensed busy, the sender could go to sleep and to attempt transmission later, but in this approach, latency could be significantly increased without necessarily saving energy.

X-MAC and X-MAC-UPMA each result in a much higher duty cycle than does RI-MAC when there is 1 flow, as shown in Figure 9(c). This higher duty cycle is because of the longer dwell time used in X-MAC and X-MAC-UPMA. In X-MAC, this dwell time is 10.5 ms, roughly a backoff windows of 32 slots, and in X-MAC-UPMA, this dwell time is 100 ms by default. The dwell time in RI-MAC is much smaller with 1 flow. As there is no collision and thus backoff window for senders is always 0, dwell time in RI-MAC is just SIFS plus propagation delay. The duty cycles of receiving nodes decrease with more contending flows in X-MAC and X-MAC-UPMA, as a receiver goes to sleep immediately after receiving packets from other flows.

Despite the high duty cycle at sending nodes, X-MAC and X-MAC-UPMA experience longer latency than does RI-MAC, as shown in Figure 9(d). This latency is mainly because transmission of preambles saturates the medium when there are more than 2 flows. The queuing delay in X-MAC and X-MAC-UPMA results in an average latency that is more than 10 times longer than that with RI-MAC when there are 4 flows.

When the number of flows is 0 in Figure 9, all three protocols show very similar performance, although this is the worst case for RI-MAC compared to X-MAC and X-MAC-UPMA. In this case, a node with RI-MAC has to stay awake each time slightly longer than it does with X-MAC and X-MAC-UPMA. In X-MAC and X-MAC-UPMA, a node needs to listen to the medium for at least SIFS plus the delay for ACK transmission at each wakeup. RI-MAC incurs some extra cost only for the CCA check before a beacon transmission and for detecting incoming signal after the beacon transmission. The difference caused by such extra cost, however, is too small to show clearly in the figure, as all three protocols already show very low duty cycles under very light traffic. As RI-MAC substantially improves throughput and energy efficiency and reduces latency under higher traffic loads, RI-MAC is suitable for a wide range of traffic loads.

4.1.2. Results in a 49-Node Grid Network

In our comparison of X-MAC, X-MAC-UPMA, and RI-MAC in a 49-node (7×7) grid network, each node is 200 meters from its neighbors, and the sink node is at the center.

In our simulations, we used a Random Correlated-Event (RCE) traffic model [20]. This model, based on a correlated-event workload [13], simulates the impulse traffic triggered by spatially-correlated events commonly observed in detection and tracking applications. RCE picks a random (x,y) location for each event. If

Table 3: Average Number of Packets Generated for Each Event under Different Sensing Ranges in the 49-Node Grid Network

Range (m)	100	200	300	400	500
Packets	0.8	3.1	6.4	10.6	15.2

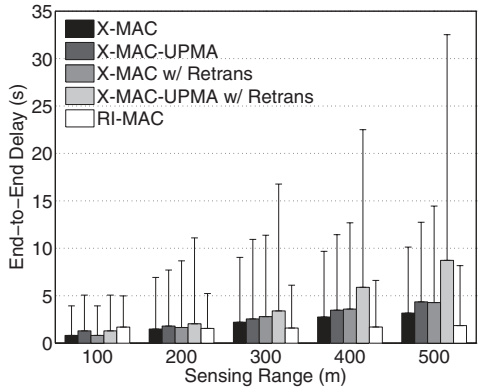
every node has a sensing range R, only nodes that are within the circle centered at (x, y) with radius R generate packets to report this event. We adjusted the sensing rage R to simulate different degrees of workload in the network. We generated a new event once every 60 seconds, and each node having sensed the event sends one packet to the sink node. We varied R from 100 meters to 500 meters; Table 3 shows the average number of packets generated per event. Note that an event triggers at most one packet when R is 100 meters. The lengths of paths traversed by these packets to the sink node range from 1 to 6 hops, with an average of 3.05 hops. In this way, we explore how efficiently X-MAC, X-MAC-UPMA, and RI-MAC handle different degrees of traffic load.

Each simulation run contains unicast packets sent toward a sink node that are triggered by a series of 100 events, and each average value is calculated from the results of 30 random runs. Confidence intervals of the average values are not shown because even 99% confidence intervals are so close to average values that they overlap with the data point markers. The curves labeled *X-MAC w/ Retrans* and *X-MAC-UPMA w/ Retrans* show the results when the original X-MAC and X-MAC-UPMA protocols, respectively, are augmented with retransmission.

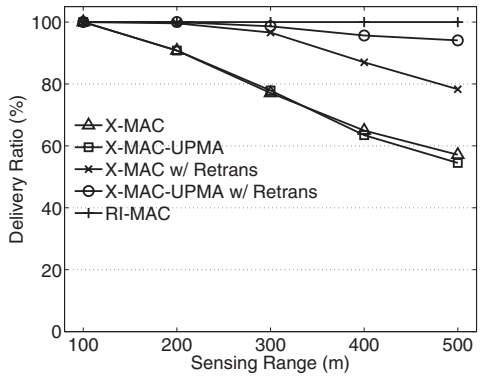
The performance comparison in these grid network scenarios is shown in Figure 10. Figure 10(a) shows the average and maximum end-to-end latency of packets in the RCE model as the sensing range (and thus traffic load) increases. RI-MAC has a much smaller rate of increase than do X-MAC and X-MAC-UPMA, regardless of whether or not retransmission is used. When there are about 15 packets generated for each event (a 500-meter sensing range), RI-MAC reduces average end-to-end delay by 85% compared to X-MAC-UPMA with retransmission, and by around 50% compared to the other protocols.

RI-MAC outperforms X-MAC and X-MAC-UPMA because it greatly increases idle medium time, allowing more competing flows to transmit in single a cycle. End-to-end latency increases when X-MAC and X-MAC-UPMA are augmented with retransmission, due to the added effort to recover packets that would otherwise be lost in collisions. Under the very light traffic load when sensing range is 100 m, X-MAC shows lower latency due to how it handles undecodable signals. For example, consider a chain consisting of nodes A, B and C, where node A can reach B, and B can reach C. Nodes A and C cannot reach each other but can sense each other's transmission. When A sends short preambles followed by a DATA frame to B, node C will remain active after sensing the medium busy, even though no incoming packet can be decoded. If C still has its radio on when B immediately starts forwarding the just-received packet to C, the forwarding will experience less delay. Because C turns off its radio if no packet is successfully received for 100 ms in X-MAC-UPMA, even though the medium is still busy, node C can be either active or sleep when B starts forwarding, depending on when C starts the 100 ms timer. This is why X-MAC-UPMA shows lower latency than does RI-MAC but higher latency than X-MAC under very light traffic load. However, as traffic load increases when sensing range is greater than 100 m, RI-MAC achieves the lowest latency on average due to increased idle medium time.

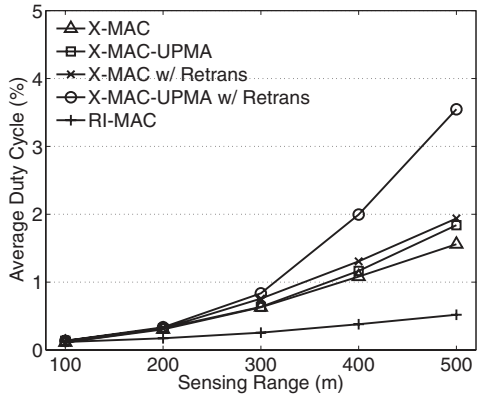
The packet delivery ratios corresponding to Figure 10(a) are shown in Figure 10(b). RI-MAC maintains 100% packet delivery ratio and outperforms X-MAC and X-MAC-UPMA across all sensing ranges. RI-MAC achieves these high delivery ratios mainly by



(a) Average and maximum end-to-end delay versus sensing range.



(b) Delivery ratio versus sensing range.



(c) Average duty cycle of sensors versus sensing range.

Figure 10: Performance for unicast traffic in 49-node (7×7) grid network scenarios in simulation.

efficient collision detection and retransmission control. The delivery ratios with X-MAC and X-MAC-UPMA drop quickly, since the larger the sensing range, the more collisions caused by transmissions from hidden nodes. When X-MAC and X-MAC-UPMA are augmented with retransmission, packet delivery ratio increases. X-MAC with retransmission shows lower delivery ratios than does X-MAC-UPMA due to the lack of an ACK after DATA transmission. If a DATA frame is lost due to collision at a receiver, the corresponding sender has no way to detect the collision and thus the DATA will not be retransmitted.

RI-MAC, in addition to achieving 100% packet delivery ratios, at the same time achieves lower duty cycles. The improved packet

delivery ratios by retransmission in X-MAC and X-MAC-UPMA, however, come at the cost of higher energy consumption, as shown in Figure 10(c). All protocols show larger duty cycles as sensing range, and thus traffic load, increases. However, RI-MAC has a much smaller rate of increase than do the other protocols. For example, when sensing range is 500 m, RI-MAC's duty cycle is only 15% that of X-MAC-UPMA with retransmission and 27% that of X-MAC with retransmission. At the same time, RI-MAC achieves much lower latency and higher packet delivery ratio, as discussed above. With retransmission, X-MAC shows lower duty cycle than does X-MAC-UPMA, mainly due to less retransmission effort because of undetectable DATA collisions.

4.1.3. Results in Random Networks

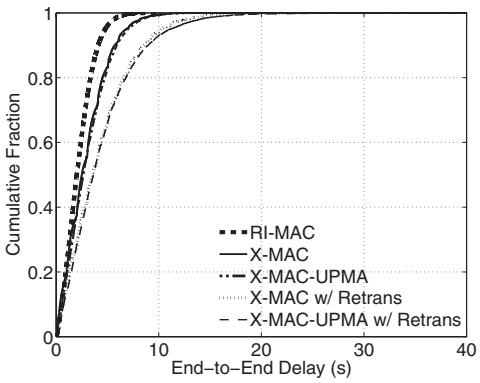
In this set of simulations, we compared RI-MAC, X-MAC, and X-MAC-UPMA in 100 random networks, each with 50 nodes randomly located in a 1000 m $\times$ 1000 m area. For each network, one of these nodes is randomly selected as the sink, and the RCE model with 250-meter sensing range is used to generate 100 events, one every 60 seconds. We conducted one simulation run for each of these 100 networks, with 763 packets on average generated in each run.

The results for these simulations are shown in Figure 11. Figure 11(a) shows the CDF of end-to-end latency for all packets in all 100 runs, Figure 11(b) shows the CDF of packet delivery ratios in these 100 runs, and Figure 11(c) shows the average duty cycles of the sensors. To improve clarity in these graphs, the protocols are listed in each graph's legend, from top to bottom, in the same order as the curves appear in the graph, from left to right. The X-MAC and X-MAC-UPMA curves in Figure 11(a) are almost indistinguishable from each other in the graph, and the curves for these two protocols with retransmissions are likewise almost indistinguishable from each other in this same graph.

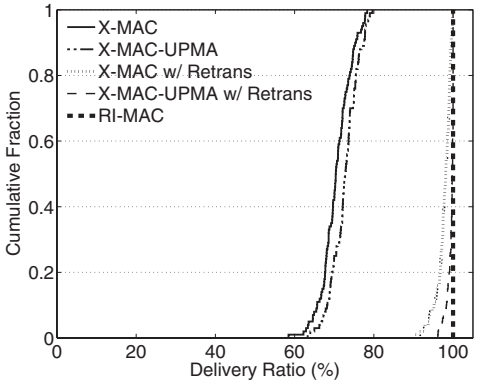
For the same reasons as discusses above, RI-MAC outperforms the other protocols in each of these metrics. For end-to-end latency (Figure 11(a)), the average values for RI-MAC, X-MAC, X-MAC-UPMA, X-MAC with retransmission, and X-MAC-UPMA with retransmission, are 2.21, 2.88, 3.02, 4.19, and 4.40 seconds, respectively. Although the addition of retransmissions in X-MAC and X-MAC-UPMA improves packet delivery ratios by helping to recover packets that would otherwise be lost due to collisions (Figure 11(b)), these retransmitted packets have higher delivery latency than other packets, producing higher average end-to-end latency for these protocol versions. The average packet delivery ratios for X-MAC, X-MAC-UPMA, X-MAC with retransmission, X-MAC-UPMA with retransmission, and RI-MAC are 70.5%, 72.6%, 97.7%, 99.4%, and 100%, respectively. The addition of retransmissions in X-MAC and X-MAC-UPMA also come at the cost of increased energy consumption (Figure 11(c)). The average values for the duty cycles of all sensors for RI-MAC, X-MAC-UPMA, X-MAC, X-MAC-UPMA with retransmission, and X-MAC with retransmission are 0.37%, 0.89%, 0.95%, 1.21%, and 1.23%, respectively. The trends observed in these random networks for each of these three metrics are consistent with those observed in the 49-node (7$\times$7) grid network, discussed above in Section 4.1.2.

4.2. Experimental TinyOS Evaluation

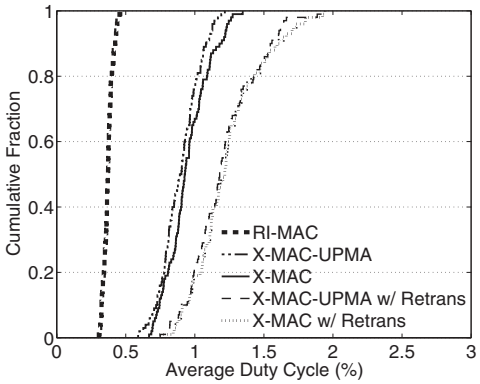
To validate our simulation-based evaluation reported above, and to explore hardware platform-dependent trends and problems, we also compared RI-MAC with X-MAC and X-MAC-UPMA in an implementation in TinyOS on MICAz motes. We implemented RI-MAC under the UPMA framework in TinyOS as described in Section 3.8.



(a) CDF of end-to-end delays



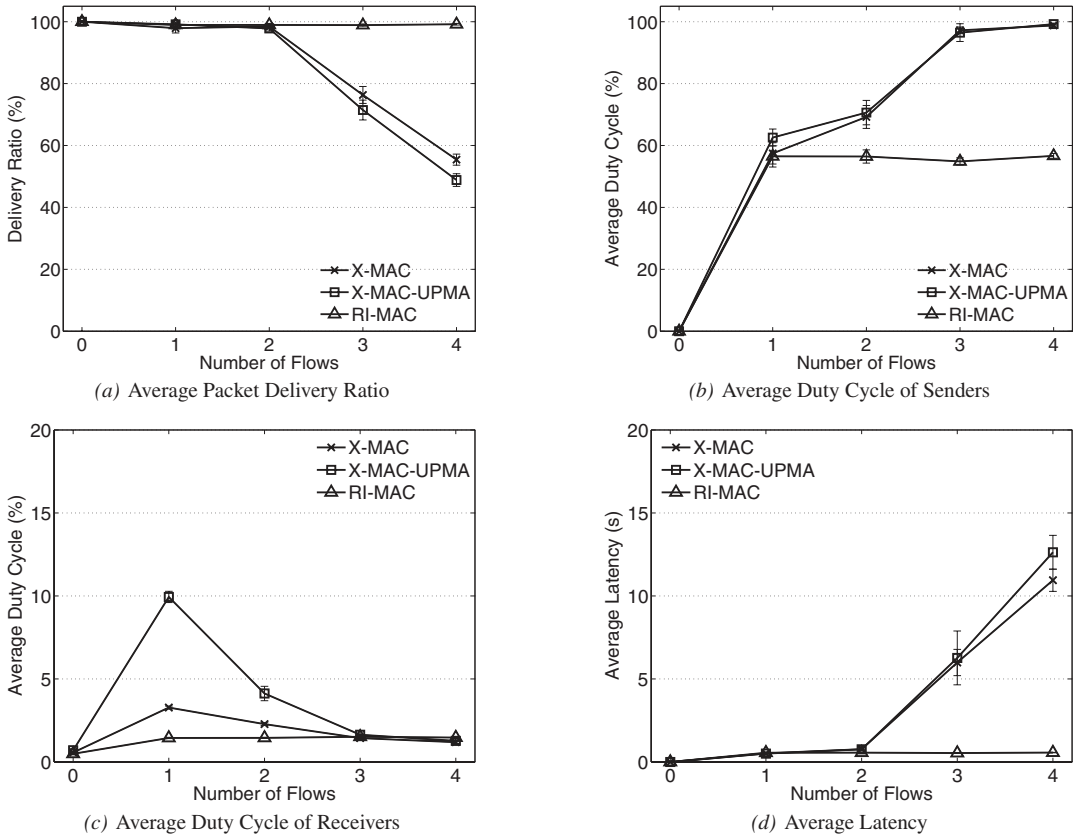
(b) CDF of delivery ratios



(c) CDF of average duty cycles

Figure 11: Performance for random correlated-event traffic in 50-node networks with sensing range of 250 m in simulation.

Although both X-MAC and X-MAC-UPMA use short preambles to achieve LPL, we also implemented X-MAC under the UPMA framework, as X-MAC-UPMA differs from the original X-MAC design in several aspects, as discussed in Sections 2 and 4.1. The configuration of X-MAC is the same as that used in our simulations, except for the continuous CCA check interval, the duration to wait for an ACK after each short preamble transmission, and the dwell time. As the duration of the continuous CCA check interval prior to preamble transmission should be longer than the gap between adjacent short preamble transmissions, the interval is set to the sum of the ACK transmission time, SIFS, and maximum propagation delay in our simulation. As discussed by Klues et al. [15], however, a longer interval is used in the TinyOS implementation in order to account for processing delays and to minimize false



(a) Average Packet Delivery Ratio

(b) Average Duty Cycle of Senders

(c) Average Duty Cycle of Receivers

(d) Average Latency

Figure 12: Performance comparison in clique networks of MICAz motes with contending flows in TinyOS implementation.

negatives. Therefore, we use the default value of 5.25 ms in the X-MAC-UPMA code for X-MAC. For the same reason, the duration to wait for an ACK after each short preamble transmission is set to 4 ms, which is also the default value in the X-MAC-UPMA code. Lastly, the dwell time should also be longer than the backoff window size in X-MAC, in order to account for possible processing delays such as post-processing of a just-transmitted packet, moving a queued packet to the MAC layer, and loading the packet to the hardware buffer. Therefore, we need to extend the dwell time defined in X-MAC to account for these delays. For fair comparison, the extra dwell time for X-MAC is also 10 ms, the same with that in our implementation of RI-MAC. In order to minimize change to underlying radio core of TinyOS, we use a packet that contains only the CC2420 header as a short preamble. Both a short preamble of X-MAC and a beacon of RI-MAC are 12 bytes, although their minimum sizes could be 6 bytes, as discussed in Section 4.1. We use the default configuration of X-MAC-UPMA in our experiments. We did not include beacon-on-request in our RI-MAC implementation, since nodes do not use multihop communication in these experiments.

4.2.1. Results in Clique Networks

In order to verify our simulation models, we present first our experiments on MICAz motes in clique networks; these TinyOS experiments are intended to replicate the simulation experiments we performed for clique networks, discussed in Section 4.1.1. The network configurations and traffic model are the same as we used in simulations. The results are shown in Figure 12 and match closely the trends and results shown earlier in Figure 9 for our clique network simulations.

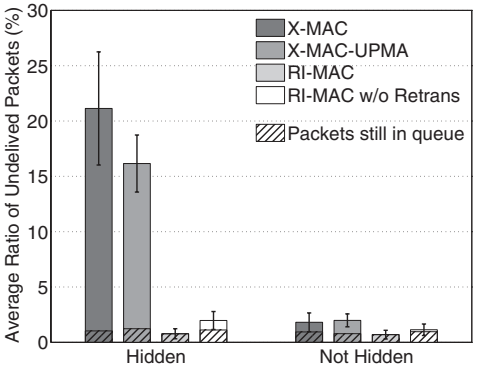
The duty cycles at sending nodes and at receiving nodes (Figures 12(b) and 12(c), respectively) are slightly higher than those in our simulations (Figure 9(b) and Figure 9(c)). This increase is mainly because the MICAz-specific processing delays in software are not simulated. For example, in our TinyOS implementation of RI-MAC, it takes around 3.75 ms for a DATA frame to arrive after a beacon transmission, mainly due to the processing delay of the beacon at a sender in software before it starts transmitting the DATA. In simulation, however, we assume the beacon is handled in hardware, so a DATA frame arrives just SIFS plus some propagation delay after the beacon transmission. In addition, in the TinyOS implementation of X-MAC and RI-MAC, we also add 10 ms to the dwell time from their original design to account for processing delays to handle the transmitted packet and to start transmitting new packets as discussed above. Although our simulation model does not account for these platform-specific delays, our simulation results still agree well with these TinyOS experimental results.

4.2.2. Results in a Network with Hidden Nodes

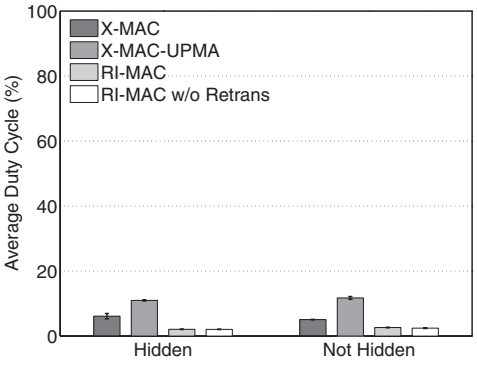
We evaluate here RI-MAC, X-MAC, and X-MAC-UPMA on networks of MICAz motes to determine how efficiently each of them detects collisions and performs retransmission to recover packets lost due to collisions; we chose to evaluate this on our TinyOS implementation rather than in simulation due to the simplified radio model used by *ns-2*.

In this set of experiments, each average value is calculated from the results of 10 experimental runs, in the same way as we did for clique networks. Error bars show the 95% confidence intervals.

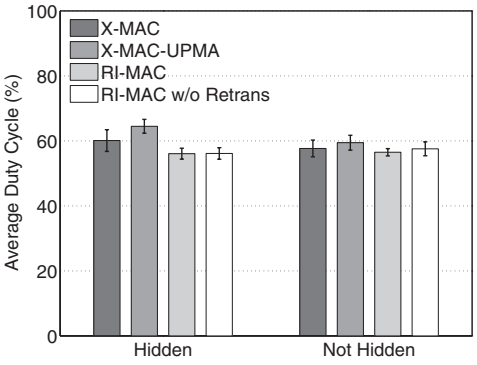
In this evaluation, we experimented with two separate network topologies: one in which hidden nodes were present, and one with



(a) Average Ratio of Undelivered Packets



(b) Average Duty Cycle of The Receiver



(c) Average Duty Cycle of Senders

Figure 13: Performance comparison when two senders are hidden to each other and when they are not in a 3-node network in TinyOS implementation.

no hidden nodes. Specifically, for each topology, we set up a network of 3 motes in which two senders transmit packets to a single receiver node. The distance from each sender to the receiver is the same and is within the transmission range of each sender. In the case with no hidden nodes, the two senders are also within range of each other, whereas in the case in which hidden nodes were present, the two senders are hidden to each other (i.e., the CCA check at each sender almost always indicates a clear channel, even while the other sender is transmitting packets). The two network topologies were otherwise identical. We use the same traffic model as we used for clique networks in Section 4.2.1. To also evaluate how efficiently RI-MAC detects collisions, we included in these experiments a variation of RI-MAC in which a sender does no retransmissions or retries as defined in Section 3.5. We refer to this variation of RI-MAC as *RI-MAC w/o Retrans*.

Results for this set of experiments on MICAz motes are shown in Figure 13. We compare the ratio of undelivered packets for X-MAC, X-MAC-UPMA, and RI-MAC in Figure 13(a). A packet may be undelivered because of collisions; it is also possible that the packet is still in the transmission queue or is being transmitted at the end of experimental measurement period. Therefore, we indicate separately in Figure 13(a) the ratio of undelivered packets for each protocol that are still in the queue (including those being transmitted). In this way, we can evaluate separately how many packets are not delivered due to collisions. The labeling along the *x*-axis in Figure 13(a) indicates whether or not the two senders are hidden to each other.

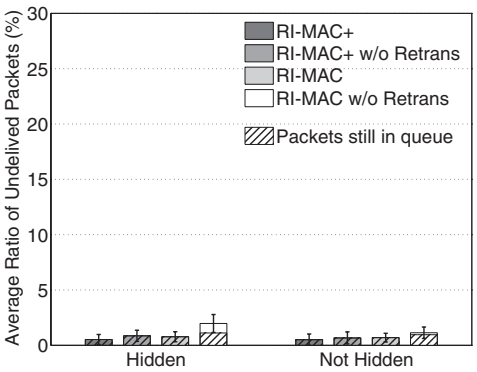
In both network topologies (with hidden nodes present and without), all protocols had a small fraction of undelivered packets still in the queue or still in transmission at the end of the experimental measurement period (Figure 13(a)). With hidden nodes present, X-MAC and X-MAC-UPMA both experienced a much larger number of additional undelivered packets due to other causes, though: about 20% of the generated packets are lost with X-MAC and 15% with X-MAC-UPMA. RI-MAC, on the other hand, experienced almost no such losses with hidden nodes. In order to confirm that these losses with X-MAC and X-MAC-UPMA are likely due to the collisions caused by the hidden node senders, we compared these results to those for the topology with no hidden nodes. In this case, almost all of these additional losses with X-MAC and X-MAC-UPMA were eliminated. With RI-MAC, however, with hidden nodes and without, no packets were lost other than those still in queue, indicating that *no packets are lost due to collisions with RI-MAC*.

In addition to a much higher packet delivery ratio, RI-MAC achieves lower duty cycles both at the receiver and at the senders. The shorter dwell time in RI-MAC is the major reason for the lower duty cycles at the receiver with RI-MAC, as discussed above. Fast collision detection and retransmission with RI-MAC also helps to achieve lower duty cycles at the senders. With X-MAC, if short preambles from the two senders repeatedly collide with each other, each sender can do nothing but retransmit its short preamble. In RI-MAC, the receiver detects the collision quickly and uses a larger sender backoff window to avoid further collisions.

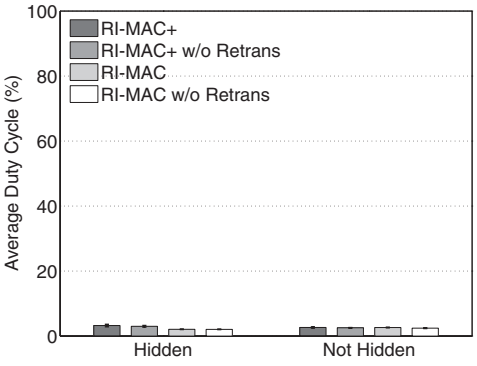
4.2.3. Extra Ending Beacons for MICAz

In Figure 13, the results for *RI-MAC w/o Retrans* are close to those for RI-MAC, except that around 2% of the packets are lost due to collisions. After extensive experimentation, we discovered that this packet loss was caused by a combination of the capture effect and the processing delays on the MICAz motes.

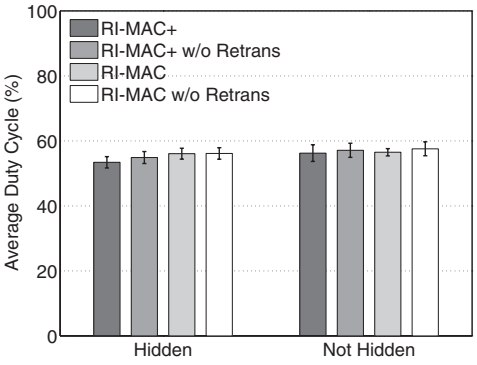
For example, suppose two sender nodes, *A* and *B*, each have 2 packets in their queue to send to receiver *C* before they receive the first beacon without backoff window from *C*. Then *A* and *B* each start transmitting their DATA frames at the same time. Assume *C* receives the DATA from *A* but loses the DATA from *B* due to the capture effect in *C*'s radio. As *C* believes that no collision occurred since it received a DATA frame following its beacon, it sends an acknowledgment beacon without backoff window to *A*. Now *A* and *B* both are allowed to transmit DATA immediately. *B* has a DATA frame already loaded in its hardware buffer that is waiting for an acknowledgment beacon, but *A* has to get a DATA from its upper layer protocol or application and load it to its hardware buffer. Thus, *B* starts transmission immediately, but *A* can only start after this processing delay. If the later DATA transmission from *A* happens to overlap with the acknowledgment beacon transmission from *C* to *B*, *C* will not know that there is a sender with pending



(a) Average Ratio of Undelivered Packets



(b) Average Duty Cycle of The Receiver



(c) Average Duty Cycle of Senders

Figure 14: Effectiveness of using an extra *ending beacon* in RI-MAC in TinyOS implementation.

DATA for it and thus will not generate another beacon. As a result, *A* discards the DATA due to timeout.

This problem happens on the MICAz motes because the CC2420 hardware transmission buffer can hold only one DATA frame. Thus, there is some delay before the queued DATA can be transmitted. If a radio could hold multiple queued packets in its hardware buffer and thus supported back-to-back DATA transmission, this problem would be much less likely to occur.

Although even on the MICAz hardware, this problem occurs only infrequently, to better handle this case, we experimented with adding an extra *ending beacon* to our original RI-MAC design. Suppose a node detects no incoming packet or collisions after the previous beacon transmission. In our original RI-MAC design, this node goes to sleep immediately. With this modification, instead, we let the node send another beacon without backoff window if the node has received at least one DATA frame after waking up in the current cycle. The node treats the beacon in the same way as the first beacon after waking up.

We compared this solution with our original RI-MAC design and show the results in Figure 14. RI-MAC with the extra *ending beacon* modification is referred to as *RI-MAC+*, and the modified RI-MAC without retransmissions is referred to as *RI-MAC+ w/o Retrans*. Figure 14(a) shows the average ratio of undelivered packets, Figure 14(b) shows the average duty cycle of the receiver, and Figure 14(c) shows the average duty cycle of senders. RI-MAC with this modification now does not lose any packets due to collisions, even with retransmission at the senders disabled (*RI-MAC+ w/o Retrans*). RI-MAC is thus very effective in detecting and recovering from collisions, even with the limitations of the real hardware in the MICAz motes.

The receiver with *RI-MAC+* or *RI-MAC+ w/o Retrans* consumes more energy, as shown in Figure 14(b), due to the extra ending beacons, but these beacons help to reduce energy consumption at the senders, as shown in Figure 14(c), as some DATA frames are delivered immediately following the ending beacons rather than waiting until the next cycle.

5. CONCLUSION

In this paper, we have presented *Receiver-Initiated MAC* (*RI-MAC*), a receiver-initiated asynchronous duty cycle MAC protocol for wireless sensor networks. RI-MAC uses receiver-initiated data transmission in order to efficiently and effectively operate over a wide range of traffic loads. To achieve this, RI-MAC attempts to minimize the time a sender and its intended receiver occupy the wireless medium to find a rendezvous time for exchanging data, while still decoupling the sender and receiver's duty cycle schedules.

We evaluated RI-MAC through detailed *ns-2* simulation and through measurements of an RI-MAC implementation in TinyOS in a testbed of MICAz motes. Compared to X-MAC, RI-MAC achieves higher throughput, higher packet delivery ratio, and greater power efficiency under a wide range of traffic loads. Especially when there are contending flows, such as with bursty traffic or transmissions from hidden nodes, RI-MAC significantly improves throughput and packet delivery ratio over X-MAC. In our experimental evaluation in our TinyOS testbed, when there are 4 contending flows in clique networks, RI-MAC improves throughput by 100%, reduces delivery latency by 90%, and reduces duty cycle by 50% at sending nodes compared to X-MAC. In the 3-node TinyOS network with hidden senders, RI-MAC achieves 0 packet loss compared to the more than 15% packet loss in X-MAC. Similar trends were also observed in our *ns-2* simulations for large networks. Even under light traffic loads, for which the X-MAC design was optimized, RI-MAC achieves the same high performance in terms of packet delivery ratio and latency while maintaining comparable power efficiency.

Acknowledgements

We thank Ryan Stinnett for his help during our testbed setup. We also thank the anonymous reviewers and Koen Langendoen, our shepherd, for their valuable feedback that helped to improve this paper. This work was supported in part by the U.S. National Science Foundation under grants CNS-0520280, CNS-0435425, CNS-0338856, and CNS-0325971; and by a gift from Schlumberger. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of NSF, Schlumberger, Rice University, Ben Gurion University, or the U.S. Government or any of its agencies.

REFERENCES

[1] Muneeb Ali, Umar Saif, Adam Dunkels, Thiemo Voigt, Kay Römer, Koen Langendoen, Joseph Polastre, and Zartash Afzal Uzmi. Medium Access Control Issues in Sensor Networks. *Computer Communications Review*, 36(2):33–36, April 2006.

[2] G. Anastasi, A. Falchi, A. Passarella, M. Conti, and E. Gregori. Performance Measurements of Motes Sensor Networks. In *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2004)*, pages 174–181, October 2004.

[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 International Conference on Embedded Networked Sensor Systems*, pages 307–320, 2006.

[4] CC2420 Datasheet. `http://www.ti.com`.

[5] Chipcon. Single Chip Very Low Power RF Transceiver (CC1000 Datasheet), April 2002.

[6] Crossbow MICAz motes. `http://www.xbow.com`.

[7] Tijs van Dam and Koen Langendoen. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of the First International Conference On Embedded Networked Sensor Systems (SenSys 2003)*, pages 171–180, November 2003.

[8] Shu Du, Amit Kumar Saha, and David B. Johnson. RMAC: A Routing-Enhanced Duty-Cycle MAC Protocol for Wireless Sensor Networks. In *Proceedings of the 26th Annual IEEE Conference on Computer Communications (INFOCOM 2007)*, pages 1478–1486, May 2007.

[9] 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), Lecture Notes in Computer Science, LNCS 3121*, pages 18–31, July 2004.

[10] Amre El-Hoiydi and Jean-Dominique Decotignie. Low Power Downlink MAC Protocols for Infrastructure Wireless Sensor Networks. *Mobile Networks and Applications*, 10(5):675–690, 2005.

[11] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networking (MobiCom 1999)*, pages 263–270, August 1999.

[12] J. J. Garcia-Luna-Aceves and Asimakis Tzamaloukas. Reversing the Collision-Avoidance Handshake in Wireless Networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 120–131, 1999.

[13] Bret Hull, Kyle Jamieson, and Hari Balakrishnan. Mitigating Congestion in Wireless Sensor Networks. In *Proceedings of the Second International Conference On Embedded Networked Sensor Systems (SenSys 2004)*, pages 134–147, November 2004.

[14] Kyle Jamieson, Hari Balakrishnan, and Y.C. Tay. Sift: A MAC Protocol for Event-Driven Wireless Sensor Networks. In *Proceedings of the Third European Workshop on Wireless Sensor Networks (EWSN 2006)*, pages 260–275, February 2006.

[15] Kevin Klues, Gregory Hackmann, Octav Chipara, and Chenyang Lu. A Component-Based Architecture for Power-Efficient Media Access Control in Wireless Sensor Networks. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, pages 59–72, 2007.

[16] Andrzej Kochut, Arunchandar Vasan, A. Udaya Shankar, and Ashok Agrawala. Sniffing Out the Correct Physical Layer Capture Model in 802.11b. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP 2004)*, pages 252–261, October 2004.

[17] Jeongkeun Lee, Wonho Kim, Sung-Ju Lee, Daehyung Jo, Jiho Ryu, Taekyoung Kwon, and Yanghee Choi. An Experimental Study on the Capture Effect in 802.11a Networks. In *Proceedings of the the Second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH 2007)*, pages 19–26, September 2007.

[18] Razvan Musaloiu-E., Chieh-Jan Mike Liang, and Andreas Terzis. Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. In *Proceedings of the 2008 International Conference on Information Processing in Sensor Networks (IPSN 2008)*, pages 421–432, April 2008.

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

[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 Ninth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2008)*, pages 53–62, May 2008.

[21] Y.C. Tay, Kyle Jamieson, and Hari Balakrishnan. Collision-Minimizing CSMA and its Applications to Wireless Sensor Networks. *IEEE Journal on Selected Areas in Communications*, 22(6), 2004.

[22] UPMA Package: Unified Power Management Architecture for Wireless Sensor Networks. `http://tinyos.cvs.sourceforge.net/tinyos/tinyos-2.x-contrib/wustl/upma/`.

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

[24] Wei Ye, Fabio Silva, and John Heidemann. Ultra-Low Duty Cycle MAC with Scheduled Channel Polling. In *Proceedings of the Fourth International Conference On Embedded Networked Sensor Systems (SenSys 2006)*, pages 321–334, October 2006.

[25] Hongwei Zhang, Anish Arora, Young-ri Choi, and Mohamed G. Gouda. Reliable Bursty Convergecast in Wireless Sensor Networks. In *Proceedings of the Sixth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2005)*, pages 266–276, May 2005.