Energy-efficient MAC for Broadcast Problems in Wireless Sensor Networks

Lisa DiPippo, David Tucker, Victor Fay-Wolfe, Kevin L Bryan, Tiegeng Ren, William Day, Matthew Murphy, Tim Henry, Shaun Joseph
University of Rhode Island, Kingston, RI, USA
{tucker, dipippo, wolfe, bryank, rentg, wday, murphym, henry, joseph}@cs.uri.edu

Abstract **- This paper describes a medium access control protocol for wireless sensor networks designed for broadcast communication. The RI-MAC (Random Interference Medium Access Control) protocol uses random slot assignment in each MAC frame, along with knowledge of neighbors' transmission schedules to mitigate some of the energy wasting problems with existing MAC protocols. RI-MAC conserves energy, transmits messages fairly among the nodes in the system, and is adaptive to changes in the topology. Preliminary evaluation tests indicate that RI-MAC uses less energy than CSMA.***

I. INTRODUCTION

Energy is a major concern in the development of wireless sensor network applications. On one hand, many sensor networks are expected to function for several months without maintenance. On the other hand, each sensor node has minimal energy resources, typically in the form of a pair of AA batteries. Furthermore, the energy costs of radio use are particularly high, not only for transmission and reception, but also for idle listening [1]. Thus the design of energy-efficient network protocols is key to successful deployments.

In this paper, we discuss the design of a medium access control (MAC) protocol that optimizes for energy conservation. A MAC protocol mediates use of the radio channel among several nodes; it says who is allowed to transmit when. In addition to energy conservation, MAC protocols usually have several other goals. The protocol should be fair: each node should have equal opportunity to communicate with other nodes. The protocol should allow for high bandwidth utilization: the radio channel's time should not be wasted. The protocol should be adaptive to changes in network topology, either due to irregular signals or node mobility.

In addition to the energy challenges described above, protocols must also account for the unstable nature of the radios. Studies have shown that even for immobile nodes, link quality can be poor, can vary with time, and may have irregular propagation patterns (asymmetric links are common) [2].

We take a novel angle on MAC protocol design in deeming energy conservation, fairness, and adaptiveness to be more important goals than bandwidth utilization. This decision reflects the nature of many sensor network applications -- they are long-lived with low duty cycles, and aim to monitor the environment over an extended period of time. To support long lifetimes, energy conservation is crucial.

We introduce RI-MAC, the Random Interference Medium Access Control protocol. Unlike many MAC protocols for sensor networks, ours is not general-purpose, but restricted to a single traffic pattern: multihop broadcast. This restriction enables us to create a MAC protocol that saves more energy. Multihop broadcast has many applications in sensor networks, often in distributing query or code updates from a base station to the entire network [3, 4]. We do not dictate any specific network layer above RI-MAC, except to assume that it is attempting to move data using multihop broadcast, and that all nodes have equal and relatively high throughput. For example, code update applications may need to send many code packets throughout the network. Even though not every node will relay every packet, nodes will be evenly loaded with transmissions and have high throughput.

The rest of the paper is organized as follows. Section 2 describes related work in MAC protocol development and indicates the benefits of RI-MAC over existing protocols for the multihop broadcast traffic pattern. Section 3 describes the details of the RI-MAC protocols, with several examples to illustrate. Section 4 presents a description of our implementation of the RI-MAC protocol for TinyOS in the TOSSIM simulation environment. This section also describes several tests that we performed to evaluate RI-MAC against a CSMA protocol. We finish in Section 5 with conclusions about our work and discussions of future work.

II. RELATED WORK

In this section, we review related work on MAC protocols, and discuss the sources of energy waste in multihop broadcast scenarios. We can divide the sources of energy waste into four categories: (a) transmissions when no node is listening, (b) listening when no node is transmitting ("idle listening"), (c) collisions due to multiple simultaneous transmissions, and (d) protocol overhead -- the exchange of control messages that do not contain application data.

MAC protocols can roughly be divided into contention-based and scheduled protocols. Most contention-based protocols are a variant of CSMA; for example, S-MAC [1] and B-MAC [5]. One problem with CSMA protocols for sensor networks is idle listening. B-MAC supports Low Power Listening (LPL), which aims to overcome the idle listening problem by requiring potential receives to

* This work is supported in part by NSF grant # CNS041030.

periodically wakes up briefly to listen for activity on the radio channel. The implementation of LPL is tightly integrated with the hardware, and is not currently available for most platforms. Under the high throughput broadcast scenario that we are considering, nodes will usually be either receiving or sending. Under this scenario, CSMA is susceptible to the hidden terminal problem -- and thus many messages will collide. Increasing CSMA backoffs can alleviate this; however, the idle listening factor then comes back into play. Other contention-based protocols, such as S-MAC [1] and T-MAC [6], use a request-to-send / clear-to-send handshaking protocol to reduce collisions. However, this scheme is appropriate for point-to-point communications rather than broadcast.

Scheduled protocols tend to reduce collisions, but waste energy in other ways. TDMA protocols often require 2-hop neighbor information to establish schedules. For example, TRAMA [7] uses random access signaling slots to exchange neighbor and schedule information. The messages involved in this create energy waste through overhead. Furthermore, TDMA protocols usually require time synchronization, another source of overhead. This overhead is exacerbated by the irregular and unreliable radio links that are typical of many sensor networks. The Z-MAC protocol is a TDMA/CSMA hybrid. Under high contention, as our broadcast scenario tends to be, Z-MAC acts similarly to TDMA, and thus has many of the problems discussed above. Finally, TSMA protocols [8] are also scheduled, and while they don't ensure collision-freedom, they guarantee certain quality of service. This approach is similar to our RI-MAC work, except that it still requires time synchronization, and doesn't specify sleeping schedules.

III. THE RI-MAC PROTOCOL

We introduce RI-MAC, which eliminates many sources of wasted energy for broadcast problems, while allowing fair channel access for all nodes. Like TDMA, RI-MAC divides time into frames, and frames into slots. We will first explain RI-MAC assuming that all nodes are time-synchronized, and then later relax this assumption.

A. Transmission Schedule

In RI-MAC, each node chooses a random slot in each frame for transmission. Figure 1 shows an example of five nodes (A through E), where each frame has six slots, and the nodes have picked their transmission slots in each frame (marked with T). We assume that each node knows its one-hop neighbors, and the transmission schedules for those neighbors. For this example, we will use the topology in Figure 2. Given the knowledge of its neighbors' transmission schedules, each node fills out its remaining slots as either listening slots (L) or sleeping slots (S) according to the following rules. If exactly one neighbor transmits in a slot, then listen. If no neighbors transmit in a slot, then sleep. If two or more neighbors transmit in a slot, also sleep, as there will only be radio interference.

The resulting schedule can be seen in Figure 3. Notice that in the first frame, node B sleeps during slot 3 because its neighbors A and D both transmit. Also note that a node

NODES	Slots in Frame 1						Slots in Frame 2					
	1	2	3	4	5	6	1	2	3	4	5	6
A			T								T	
B					T			T				
C		T						T				
D			T									T
E						T	T					

Figure 1 – Example Transmission Schedule

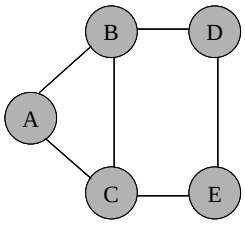

Figure 2 – Example Topology

NODES	Slots in Frame 1						Slots in Frame 2					
	1	2	3	4	5	6	1	2	3	4	5	6
A	S	L	T	S	L	S	S	S	S	S	T	S
B	S	L	S	S	T	S	S	T	S	S	L	L
C	S	T	L	S	L	L	L	T	S	S	L	S
D	S	S	T	S	L	L	L	L	S	S	S	T
E	S	L	L	S	S	T	T	L	S	S	S	L

Figure 3 – Example Schedule

transmits even if a neighbor is scheduled to transmit in the same slot; for example, both B and C transmit during slot 2 of the second frame.

B. Protocol Specifics

Let us address several specific aspects of the protocol and its implementation.

Neighbor Transmission Schedule. We assumed that a node knows each neighbor's transmission schedule. In each packet, we include two data from the sender: its address, and a sequence number indicating where it is in its pseudo-random number sequence. Since a node seeds its own pseudo-random number generator with its own address, these two data can be used by its neighbors to predict the node's transmission schedule. Therefore, once a node hears one packet from a neighbor, it knows that neighbor's entire transmission schedule. To allow nodes to learn its neighbors, RI-MAC has a setup phase of unscheduled listening before entering the main schedule.

Clock Synchronization. Initially we assumed that all clocks were synchronized, and thus frames and slots were aligned. In fact, the RI-MAC protocol and its implementation do not require aligned slots. This changes the rules only slightly: a node only listens if a neighbor's

transmission is not overlapped by some other neighbor's transmission. In each transmission from a neighbor, a timestamp is included, so a node will know the offset of its own slots with its neighbors' slots. Thus, it will be able to predict the overlap of its neighbors' transmissions.

We do not expect clock drift to be a major problem because of the course-grained nature of our scheduling (we need accuracy on the millisecond level). Studies show that clock drift on mica2 motes, for example, is on the order of 1 millisecond over the course of seven hours [9]. However, we do account for clock drift as follows: every time a node receives a message from a neighbor, it updates its internal record of that neighbor's schedule in accordance with the timestamp of that receipt. Therefore, the accuracy of a neighbor's schedule is with respect to the most recently received message from that neighbor.

Energy Conservation. Let's consider the protocol in terms of wasted energy. When a node is sleeping, obviously no energy is wasted. When a node is listening in RI-MAC, no energy is wasted because it will receive a message. The only sources of waste are transmission when a neighbor is not listening, and overhead. Sometimes a sending node's neighbor will not listen if it knows that another of its own neighbors will be sending at the same time. Overhead in RI-MAC occurs through the setup phase when nodes are discovering their neighbors.

C. Analytical Comparison

We can compare RI-MAC with typical TDMA protocols [10] given our three goals: energy efficiency, fairness, and adaptability. RI-MAC is more energy-efficient, because the overhead involved in RI-MAC neighbor discovery is significantly less than the 2-hop neighborhood information required by other TDMA protocols. Further, RI-MAC does not require synchronized clocks, which can be very energy intensive. TDMA protocols can result in unfair schedules due to irregular radio links. As [2] shows, asymmetric links can lead to schedules that are not collision-free. Since TDMA schedules are typically static, a node with a bad schedule may never get channel access. In RI-MAC, the random schedules, chosen in each frame, ensure that all nodes have equal opportunity, regardless of the radio irregularity. Finally, RI-MAC is more adaptive to changing conditions than TDMA because it requires only 1-hop information for its scheduling algorithm, and thus a node can more easily learn about new neighbors and adjust its schedule accordingly.

IV. Implementation and Evaluation

We have implemented RI-MAC for TinyOS and tested it using TOSSIM (the TinyOS Simulator) [11] and its extension for energy profiling, PowerTOSSIM [12]. TOSSIM provides discrete event simulation of TinyOS programs, and includes an implementation of the radio stack for the mica2's CC1000 radio. TOSSIM simulates radio behavior at a low level and implements a CSMA protocol similar to B-MAC for medium access control. We use this as both a point of comparison as well as a basis for our MAC implementation.

A. RI-MAC Implementation Environment

TinyOS's CC1000 radio stack provides two interfaces for sending and receiving messages:

```
interface BareSendMsg {
    command result_t send(TOS_MsgPtr msg);
    event result_t sendDone(TOS_MsgPtr msg,
                                result_t success);
}

interface ReceiveMsg {
    event TOS_MsgPtr receive(TOS_MsgPtr m);
}
```

The user process can call a **`send()`** command, and must handle a **`receive()`** event. Commands are initiated by the user process, whereas events are initiated by the library (in this case, the radio stack) and handled by the user process. The RI-MAC implementation is built on top of the CC1000 radio stack with several modifications. First, backoffs used by CSMA were removed. Second, because RI-MAC is scheduled, priority was given to sends over receives (i.e., the stack sends a packet right away even if it is in the middle of receiving another packet). And third, the scheduling layer on top of the CC1000 radio stack determines send and receive periods, and notifies the user process of them. We preserve the ReceiveMsg interface, but provide a different interface for sending messages:

```
interface BareSendMsgScheduled {
    event TOS MsgPtr sendNow();
    event result_t sendDone(TOS MsgPtr msg,
                                result t success);
  }
```

Instead of calling a **`send()`** command, the user process handles a **`sendNow()`** event when the stack needs a packet to send. The user process can alternatively return NULL to indicate there is no message to send. In terms of control, the radio stack, rather than the user process, decides on the send schedule.

RI-MAC maintains the following 14 to 18 bytes of state information for each neighbor as well as for the mote itself:

- Id: [2 bytes] Id of the neighbor mote.
- Time of Next Packet: [4-8 bytes] World clock time of the neighbor's next predicted transmission.
- Next Slot Number: [2 bytes] Slot in which neighbor will transmit. Computed from the neighbor's random number mod frame size.
- PRNG Sequence Number: [2 bytes] Neighbor's count in the pseudo-random number generator's sequence.
- PRNG State: [4 bytes] Saved state of PRNG based on neighbor's sequence count. This prevents re-initialization of neighbor's PRNG in order to arrive at the current random number in sequence.

Using this data, RI-MAC determines when the mote should sleep and when it should wake for sending or listening. RI-MAC updates each neighbor's PRNG state upon packet reception, or on timeouts after missed packets.

B. CSMA Implementation

We use the CSMA implementation provided by TinyOS, modified so that we can adjust the backoff values. By default, when a mote decides it must transmit, it waits an initial backoff of 7 to 33 milliseconds and then listens to the channel to determine if another mote is transmitting. If the channel is clear, the mote sends its message. If a neighbor is transmitting, the mote performs a congestion backoff of 7 to 115 milliseconds before trying to resend the message.

C. Testing Parameters and Metrics

We performed various experiments to test how well the RI-MAC protocol meets the goals stated in this paper. Due to space limitations, we discuss one specific test here. Other results will be published in the future.

For this test, we used a simple all-to-all network protocol that distributes all of the data to all of the motes. We considered the following questions:

- How does the RI-MAC protocol affect power usage for all-to-all data distribution in a wireless sensor network compared to TinyOS's default CSMA?
- How does the RI-MAC protocol affect the time required to provide all-to-all data distribution compared to CSMA?

This experiment focused on fine-tuning the RI-MAC parameters in order to achieve distribution of data so that each node received at least two thirds of the data values generated by the network. The graph in Figure 4 shows the average power consumption in simulation. RI-MAC was able to achieve nearly as wide a distribution while using a third the power of CSMA. In terms of time, the RI-MAC execution time exceeded that of the CSMA version by a factor of 4. We consider this tradeoff of time for energy to be consistent with the goals of many wireless sensor network applications.

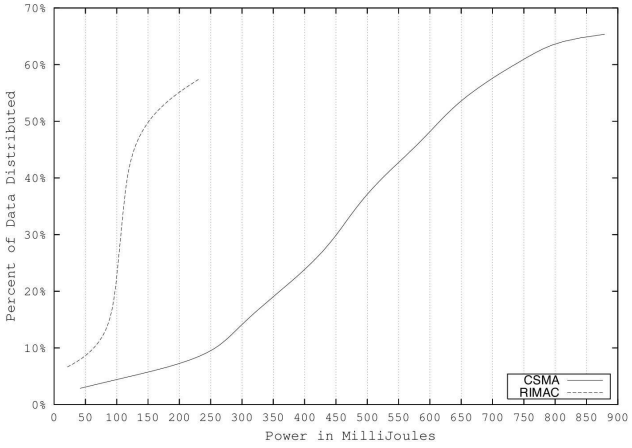

Figure 4 – Experimental Results

V. Conclusions and Future Work

RI-MAC is a MAC protocol designed for broadcast communication in wireless sensor networks. We have shown that it wastes less energy than CSMA when frame size is chosen properly. We have further discussed, analytically, the benefits of RI-MAC over TDMA.

Future work on RI-MAC will address how to compute the frame size at run time. This will allow nodes to adjust their frame sizes based on the number of neighbors they discover, thus positively impacting the energy, time, and reliability tradeoffs. The current implementation of the protocol does not adapt to nodes entering or leaving the network. For nodes exiting, such as when a node fails or is destroyed, the protocol should decide when to stop listening for the particular neighbor. The protocol does not detect when a new node has entered, such as during a second deployment wave. A possible solution is to have regular period where nodes listen and reset their local neighbors connections. Adjusting to nodes entering and leaving the system will be much simpler than in TDMA schedules that require 2-hop neighbor information. Other future work will involve further testing using metrics that can illustrate the fairness, and adaptability of the RIMAC protocol.

References

[1] W. Ye, J. Heidemann, D. Estrin, An Energy-Efficient MAC Protocol for Wireless Sensor Networks, *Proc. of 2002 IEEE InfoCOM*, 2002.

[2] G. Zhou, T. He, S. Krishnamurthy, J. Stankovic, Impact of Radio Irregularity on Wireless Sensor Networks, *Proc. of the 2nd Int. Conf. On Mobile Systems, Apps. and Services*, 2004.

[3] S. Madden, M. Franlin, J. Hellerstein, W. Hong, The Design of an Acquisitional Query Processor For Sensor Networks, *Proc of ACM SIGMOD/PODS 2003 Conference*, June 2003.

[4] P. Levis, N. Patel, D. Culler, S. Shenker, Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks, *Proc. Of 1st Symposium on Network Systems Design and Implementation*, 2004.

[5] J. Polastre, J. Hill, D. Culler, Versatile Low Power Media Access for Wireless Sensor Networks, *Proc. of SenSys 2004*.

[6] T. Van Dam, K. Langendoen, An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks, *Proc. of SenSys 2003*.

[7] V. Rajendran, K. Obraczka, J.J. Garcia-Luna-Aceves, Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks, *Proc. of SenSys 2003*.

[8] I. Chlamtac, A. Farag´o, and H. Zhang, Time-spread multipleaccess (TSMA) protocols for multihop mobile radio networks, *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 804–812, December 1997.

[9] M. Marti, B. Kusy, G. Simon, and A. Ledeczi, The flooding time synchronization protocol, *Proc of SenSys 2004.*

[10] W. Ye, J. Heidemann, Medium Access Control in Wireless Sensor Networks, *USC/ISI Technical Report ISI-TR-580*, October 2003.

[11] P. Levis, N. Lee, M. Welsh, and D. Culler, TOSSIM: accurate and scalable simulation of entire TinyOS applications, *Proc. of SenSys 2003.*

[12] V. Shnayder, M. Hempstead, B. Rong Chen, G. W. Allen, and M. Welsh, Simulating the power consumption of large-scale sensor network applications, *Proc. Of SenSys 2004*