An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks

Tijs van Dam
tijsvd@xs4all.nl

Koen Langendoen
K.G.Langendoen@its.tudelft.nl

Faculty of Information Technology and Systems
Delft University of Technology
The Netherlands

ABSTRACT

In this paper we describe T-MAC, a contention-based Medium Access Control protocol for wireless sensor networks. Applications for these networks have some characteristics (low message rate, insensitivity to latency) that can be exploited to reduce energy consumption by introducing an active/sleep duty cycle. To handle load variations in time and location T-MAC introduces an *adaptive* duty cycle in a novel way: by dynamically ending the active part of it. This reduces the amount of energy wasted on idle listening, in which nodes wait for potentially incoming messages, while still maintaining a reasonable throughput.

We discuss the design of T-MAC, and provide a head-to-head comparison with classic CSMA (no duty cycle) and S-MAC (fixed duty cycle) through extensive simulations. Under homogeneous load, T-MAC and S-MAC achieve similar reductions in energy consumption (up to 98 %) compared to CSMA. In a sample scenario with variable load, however, T-MAC outperforms S-MAC by a factor of 5. Preliminary energy-consumption measurements provide insight into the internal workings of the T-MAC protocol.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication—*MAC protocol*; C.2.5 [**Local and Wide-Area Networks**]: Access schemes; D.4.4 [**Communications Management**]: Message sending

General Terms

Design, Experimentation, Measurement, Performance

Keywords

Ad-hoc, sensor networks, MAC protocol, energy-efficiency

SenSys'03, November 5–7, 2003, Los Angeles, California, USA.

1. INTRODUCTION

Communication in wireless sensor networks can, like most network communication, be divided into several layers. One of those is the Medium Access Control (MAC) layer. This layer is described by a MAC protocol, which tries to ensure that no two nodes are interfering with each other's transmissions, and deals with the situation when they do.

Wireless sensor networks have an additional aspect: as sensor nodes are generally battery-operated, energy consumption is very important. The radio on a sensor node is usually the component that uses most energy. Not only transmitting costs energy; receiving, or merely scanning the ether for communication, can use up to half as much, depending on the type of radio [8].

While traditional MAC protocols are designed to maximize packet throughput, minimize latency and provide fairness, protocol design for wireless sensor networks focuses on minimizing energy consumption. The application determines the requirements for the (modest) minimum throughput and maximum latency. Fairness is usually not an issue, since the nodes in a wireless sensor network are typically part of a single application and work together for a common purpose.

1.1 Communication patterns

It is important to design and test the behavior of MAC protocols based on the kind of traffic they have to handle. We have identified two main *communication patterns* in sensor applications:

Local uni-/broadcast When a real-world event in the network occurs, we expect nodes to perform some in-network processing. This will generally involve local messages being exchanged between neighbors.

Nodes to sink reporting After processing a local event, or just periodically, nodes may want to report something. We expect messages to be directed to one or a few *sink nodes*, which are hooked up to a fixed network or a computer. Messages from different nodes may, or may not, be aggregated along the way. We do not specify an exact routing protocol, but we expect some random variation in message paths—messages flow 'roughly' in the correct direction. In this communication pattern, we see a more or less unidirectional flow of messages through the network.

We explicitly exclude routed, multi-hop communication between random nodes in the network, although this pattern is frequently used to study MAC protocols. After identifying several realistic wireless sensor applications, we have determined that this communication pattern does not occur.

The two basic communication patterns imply that the message rate in the network may vary, both in time and location: events trigger periods of increased activity, and, around sink nodes, the message rate will be higher than at the edge of the network, even when aggregation is used.

1.2 EYES nodes

Wireless sensor hardware is generally cheap and simple. We must consider this when designing a MAC protocol for wireless sensor nodes. As a base for our protocol design, simulations and implementation we use the EYES wireless sensor nodes [13].

The EYES nodes have a Texas Instruments MSP430F149 processor with 2 KB RAM and 60 KB of Flash memory; the 16 bit processor runs at a variable clock rate, with a maximum of 5 MHz. Nodes communicate using a 115 kbps radio (RFM TR1001, 868.35 MHz, hybrid transceiver), and are equipped with a 2 Mb EEPROM memory (AST 25P20V6). The nodes contain multiple interfaces to interact with the outside world, including JTAG, RS232, 2 LEDs, and 16 general purpose I/O pins (8 with ADC capability). The nodes run from 3 V supplied by two AA batteries taking up most of the nodes' volume. Table 1 provides a power breakdown of the processor and radio for active and sleep modes.

CPU	
active (5 MHz)	2.1 mA
sleep	1.6 μA
Radio	
transmit	10 mA
receive	4 mA
sleep	20 μA

Table 1: Power breakdown of EYES nodes.

The capabilities and power consumption of the EYES nodes are quite similar to other prototype sensor nodes, for example, the popular popular Berkeley Motes [14]. We like to stress that memory (2 KB of RAM), next to energy, is a scarce resource. Consequently, a MAC protocol should use as little of it as possible, which limits, for example, the possibility of maintaining elaborate information on neighbors.

1.3 Idle listening problem

Most energy in traditional MAC protocols is wasted by *idle listening*: since a node does not know when it will be the receiver of a message from one of its neighbors, it must keep its radio in receive mode at all times. Consider, for example, a sensor application that requires nodes to exchange messages with their neighbors at an average rate of one per second. Messages are fairly short: they take less than 5 milliseconds to transmit. This results in each node spending on average 5 ms per second on transmitting, 5 ms on receiving a message from another node, and 990 ms on listening while nothing happens. The radio is then doing nothing for 99% of the time.

1.4 Outline

We present T-MAC–Timeout-MAC–, an adaptive energy-efficient MAC protocol for wireless sensor networks that minimizes idle listening, while considering wireless sensor communication patterns and hardware limitations. In Section 2, we will describe some existing energy-saving solutions. Then, in Section 3, we will elaborate on the design of the T-MAC protocol, problems we encountered, and the novel way in which we solved these problems. In Section 4, we will describe our simulation setup, followed by a detailed report of our results. In Section 5 we report the measured energy consumption of a limited T-MAC implementation running on a pair of Eyes nodes.

2. RELATED WORK

There are several solutions addressing the problem of energy waste due to idle listening. In general, some kind of duty cycle is involved, which lets each node sleep periodically. For example, TDMA-based protocols are naturally energy preserving, because they have a duty cycle built-in, and do not suffer from collisions [2]. However, maintaining a TDMA schedule in an ad-hoc network is not an easy task and requires much complexity in the nodes. Keeping a list of neighbor's schedules takes valuable memory capacity. Allocating TDMA slots is a complex problem that requires coordination. Furthermore, as TDMA divides time into very small slots, the effect of clock drift can be disastrous; exact timing is critical.

Another way of energy saving is to use an extra radio—the so-called wake-up radio—, which operates on a different frequency than the radio used for communication [7]. As the wake-up radio is only for waking up other nodes, it needs no data processing and therefore uses much less energy. It does, however, require an extra component on the node and most wireless sensor nodes currently used in research only have a single radio that operates on a single frequency.

Introducing a duty cycle into a contention-based (CSMA) protocol that only uses a single frequency requires some kind of in-band signalling. The well known IEEE 802.11 protocol, for example, has power-saving features, even when working in ad-hoc mode [4]. However, this protocol was designed with the presumption that all nodes are located in a single network cell, while wireless sensor networks will often be multi-hop. Adaptations for multi-hop networks have been proposed, but seem to require more complexity and dynamic state than would generally be available in wireless sensor networks [10].

The TinyOS project [14] includes a sensor-networks specific optimization of the basic CSMA protocol that tackles the idle-listening problem: by sending out a very long preamble, receivers only need to weak up periodically to sense activity. This shifts the cost from the receiver (the frequent case) to the transmitter (the rarer case). The TinyOS approach, briefly discussed in [3], is basically an optimization at the physical layer and may be applied in combination with link-layer (MAC) solutions discussed next.

Another protocol specifically designed for sensor networks is S-MAC [12]. The basic idea of this single-frequency contention-based protocol is that time is divided into—fairly large—frames. Every frame has two parts: an active part and a sleeping part. During the sleeping part, a node turns off its radio to preserve energy. During the active part, it

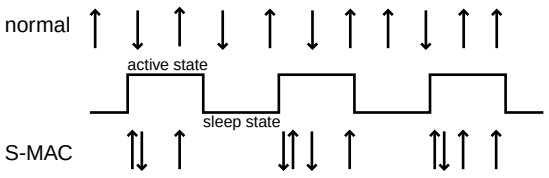

Figure 1: The S-MAC duty cycle; the arrows indicate transmitted and received messages; note that messages come closer together.

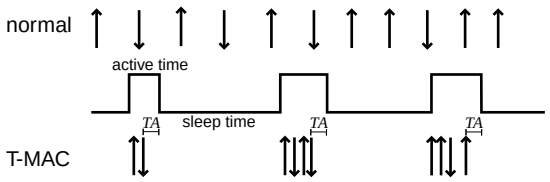

Figure 2: The basic T-MAC protocol scheme, with adaptive active times.

can communicate with its neighbors and send any messages queued during the sleeping part, as shown in Figure 1. Since all messages are packed into the active part, instead of being 'spread out' over the whole frame, the time between messages, and therefore the energy wasted on idle listening, is reduced.

S-MAC needs some synchronization, but that is not as critical as in TDMA-based protocols: the time scale is much larger. Typically, there may be an active part of 200 ms in a frame of one second. A clock drift of 500 μs will not be a problem.

The S-MAC protocol essentially trades used energy for throughput and latency. Throughput is reduced because only the active part of the frame is used for communication. Latency increases because a message-generating event may occur during sleep time. In that case, the message will be queued until the start of the next active part.

3. T-MAC PROTOCOL DESIGN

Energy consumption is the main criterion for our MAC protocol design. We have already identified the problem of idle listening. Other forms of energy waste are:

collisions if two nodes transmit at the same time and interfere with each others transmission, packets are corrupted. Hence, the energy used during transmission and reception is wasted;

protocol overhead most protocols require control packets to be exchanged; as these contain no application data, we may consider any energy used for transmitting and receiving these packets as overhead;

overhearing since the air is a shared medium, a node may receive packets that are not destined for it; it could then as well have turned off its radio.

These other sources of energy consumption are relatively insignificant when compared to the energy wasted by idle listening, especially when messages are infrequent. Consider our example where 99% of the time is spent on idle listening. If, then, the actual transmission and receiving time increases by a factor two—due to collisions and overhead—, idle listening time decreases only from 99% to 98%.

Although reducing the idle listening time, a solution with a *fixed duty cycle*, like the S-MAC protocol [12], is not optimal. S-MAC has two important parameters: the total *frame time*, which is limited by latency requirements and buffer space, and the *active time*. The active time depends mainly on the message rate: it can be so small that nodes are able to transfer all their messages within the active time.

The problem is that, while latency requirements and buffer space are generally fixed, the message rate will usually vary (Section 1.1). If important messages are not to be missed– and unimportant messages should not have been sent in any case–, the nodes must be deployed with an active time that can handle the highest expected load. Whenever the load is lower than that, the active time is not optimally used and energy will be wasted on idle listening.

The novel idea of the T-MAC protocol is to reduce idle listening by transmitting all messages in *bursts* of variable length, and sleeping between bursts. To maintain an optimal active time under variable load, we dynamically determine its length. We end the active time in an intuitive way: we simply time out on hearing nothing.

3.1 Basic protocol

Figure 2 shows the basic scheme of the T-MAC protocol. Every node periodically wakes up to communicate with its neighbors, and then goes to sleep again until the next frame. Meanwhile, new messages are queued. Nodes communicate with each other using a Request-To-Send (RTS), Clear-To-Send (CTS), Data, Acknowledgement (ACK) scheme, which provides both collision avoidance and reliable transmission [1]. This scheme is well known and used, for example, in the IEEE 802.11 standard [4].

A node will keep listening and potentially transmitting, as long as it is in an active period. An active period ends when no *activation event* has occurred for a time *TA*. An activation event is:

- the firing of a periodic frame timer;
- the reception of any data on the radio;
- the sensing of communication[1] on the radio, e.g. during a collision;
- the end-of-transmission of a node's own data packet or acknowledgement;
- the knowledge, through overhearing prior RTS and CTS packets, that a data exchange of a neighbor has ended.

A node will sleep if it is not in an active period. Consequently, *TA* determines the minimal amount of idle listening per frame.

The described timeout scheme moves all communication to a burst at the beginning of the frame. Since messages between active times must be buffered, the buffer capacity determines an upper bound on the maximum frame time.

[1]Through a Received Signal Strength Indication (RSSI) signal from the radio.

3.2 Clustering and synchronization

Frame synchronization is inspired by virtual clustering, as described by the authors of the S-MAC protocol [12]. When a node comes to life, it starts by waiting and listening. If it hears nothing for a certain amount of time, it chooses a frame schedule and transmits a SYNC packet, which contains the time until the next frame starts. If the node, during startup, hears a SYNC packet from another node, it follows the schedule in that SYNC packet and transmits its own SYNC accordingly.

Nodes retransmit their SYNC once in a while. Nodes must also listen for a complete frame sporadically, so they can detect the existence of different schedules. This allows new and mobile nodes to adapt to an existing group.

If a node has a schedule and hears a SYNC with a different schedule from another node, it must adopt both schedules. It must also transmit a SYNC with its own schedule to the other node, to let the other node know about the presence of another schedule. Adopting both schedules means that the node will have an activation event at the start of both frames.

Nodes must start a data transmission only at the start of their own active time. At that time, both neighbors with the same schedule, and neighbors that have adopted the schedule as extra, are awake. If a node would start transmission at the start of a neighbor's frame, it might be transmitting to another, sleeping neighbor. Note that this scheme makes it possible that broadcasts only need to be transmitted once.

The described synchronization scheme, which is called *virtual clustering* [12], urges nodes to form clusters with the same schedule, without enforcing this schedule to all nodes in the network. It allows efficient broadcast, and obviates the need to maintain information on individual neighbors.

The virtual clustering technique is easy to implement. Keeping multiple schedules with a fixed-length active time is more complex, since active times overlap.

3.3 RTS operation and choosing TA

We will now discuss the additional features of the T-MAC protocol that provide optimal tuning.

Fixed contention interval

In contention-based protocols, like IEEE 802.11, nodes wait for a random time within a contention interval after detecting a collision. Only when the air is clear during that time do they restart transmission. Usually, a back-off scheme is used: the contention interval increases when traffic is higher. The back-off scheme reduces the probability of collisions when the load is high, while minimizing latency when the load is low.

In the T-MAC protocol, every node transmits its queued messages in a burst at the start of the frame. During this burst, the medium is saturated: messages are transmitted at maximum rate. A node may expect to be in a fierce fight for winning the medium every time it sends an RTS. An increasing contention interval is not useful, since the load is mostly high and does not change. Therefore, RTS transmission in T-MAC starts by waiting and listening for a random time within a *fixed* contention interval. This interval is tuned for maximum load. The contention time is always used, even if no collision has occurred yet.

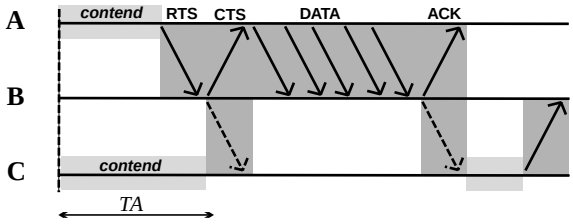

Figure 3: A basic data exchange. Node C overhears the CTS from node B and will not disturb the communication between A and B. *TA* must be long enough for C to hear the start of the CTS.

RTS retries

When a node sends an RTS, but does not receive a CTS back, one of three things has happened:

1. the receiving node has not heard the RTS due to collision; or
2. the receiving node is prohibited from replying due to an overheard RTS or CTS; or
3. the receiving node is asleep.

When the sending node receives no answer within the interval *TA*, it might go to sleep. However, that would be wrong in cases 1 and 2: we would then have a situation where the sending node goes to sleep, while the receiving node is still awake. Since this situation might occur even at the first message of the frame, the throughput would (and did, in our preliminary experiments) dramatically decrease.

Therefore, a node should retry by re-sending the RTS if it receives no answer. If there is still no reply after two retries, it should give up and go to sleep.

Determining TA

A node should not go to sleep while its neighbors are still communicating, since it may be the receiver of a subsequent message. Receiving the start of the RTS or CTS packet from a neighbor is enough to trigger a renewed interval *TA*.

Since a node may not hear, because it is not in range, the RTS that starts a communication with its neighbor, the interval *TA* must be long enough to receive at least the start of the CTS packet (Figure 3). This observation gives us a lower limit on the length of the interval *TA*:

$$TA > C + R + T$$

where C is the length of the contention interval, R is the length of an RTS packet, and T is the turn-around time (the short time between the end of the RTS packet and the beginning of the CTS packet). In our experiments, we used $TA = 1.5 \times (C + R + T)$, which proved to be satisfactory. A larger *TA* increases the energy used.

3.4 Overhearing avoidance

The S-MAC protocol introduced the idea of sleeping after overhearing an RTS or CTS destined for another node. Since a node is prohibited from sending during that time, it can not take part in any communication and may as well turn off its radio to save energy.

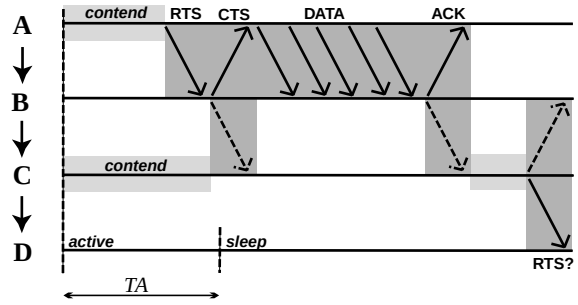

Figure 4: The *early sleeping* problem. Node D goes to sleep before C can send an RTS to it.

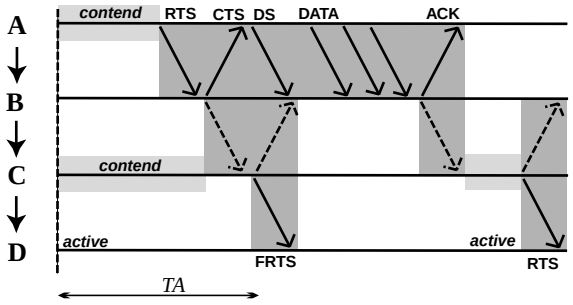

Figure 5: The future-request-to-send packet exchange keeps Node D awake.

In general, overhearing avoidance is a good idea, and it is an option in the T-MAC protocol. However, we have observed in our experiments that, as a side effect, collision overhead becomes higher: a node may miss other RTS and CTS packets while sleeping and disturb some communication when it wakes up. Consequently, the maximum throughput decreases; for short packets by as much as 25%. Thus, although overhearing avoidance saves energy, it must not be used when maximum throughput is (at times) required.

3.5 Asymmetric communication

Preliminary simulation experiments revealed a problem with the T-MAC protocol when traffic through the network is mostly unidirectional, like in a nodes-to-sink communication pattern. This problem is simplified in Figure 4. Each of the nodes A though D in the picture forms a cell with its neighbors. Messages flow from top to bottom, so node A sends only to B, B only to C, and C only to D. Now consider node C. Every time it wants to send a message to D, it must contend for the medium and may loose to either node B (by receiving an RTS packet) or to node A (indirectly, by overhearing a CTS packet from node B).

If node C looses contention because of an RTS packet from node B, it will reply with a CTS packet, which can also be heard by node D. In that case, node D will be awake when the communication between C and B ends. However, if node C looses contention because it overhears a CTS packet from B to A (see Figure 4), C must remain silent. Since D does not know of the communication between A and B, its active time will end, and node D will go to sleep. Only at the start of the next frame will node C have a new chance to send to node D.

Thus for every packet that node C wants to send to node D, it may either succeed or fail (by loosing to node A). Both of these events have equal probability. Failure implies that the frame ends and C can send no more packets. We can therefore calculate that, in this simplified setup, node C has a 50% probability of sending a single packet to node D, a 25% probability of sending two packets (it must succeed twice), etcetera, in each frame.

We call the observed effect the *early sleeping* problem, since a node goes to sleep when a neighbor still has messages for it. In the nodes-to-sink communication pattern, the early sleeping problem reduced the total possible throughput of T-MAC to less than half of the maximum throughput of traditional protocols or S-MAC. In later experiments, we have also encountered this problem at the border of a highly active part of the network. We believe that the problem may occur in any asymmetric communication pattern. We devised two solutions.

Future request-to-send

The first solution is a scheme that we call *future request-to-send*. The idea is to let another node know that we still have a message for it, but are ourselves prohibited from using the medium. It works as follows: if a node overhears a CTS packet destined for another node, it may immediately send a future-request-to-send (FRTS) packet, like node C in Figure 5. The FRTS packet contains the length of the blocking data communication (this information was in the CTS packet). A node must not send an FRTS packet if it senses communication right after the CTS, or if it is prohibited from sending due to a prior RTS or CTS.

A node that receives an FRTS packet knows it will be the future target of an RTS packet and must be awake by that time. The node can determine this from the timing information in the FRTS packet.

As the FRTS packet would otherwise disturb the data packet that follows the CTS, the data packet must be postponed for the duration of the FRTS packet. To prevent any other node from taking the channel during this time, the node that sent the initial RTS (node A in Figure 5) transmits a small Data-Send (DS) packet. After the DS packet, it must immediately send the normal data packet.

Since the FRTS packet has the same size as a DS packet, it will collide with the DS packet, but not with the following data packet. The DS packet is lost, but that is no problem: it contains no useful information.

For the FRTS solution to work, *TA* must be increased with the length of a control packet (CTS), as follows from Figure 5. Implementing the FRTS feature increased the maximum throughput in unidirectional communication patterns by approximately 75%. However, due to the somewhat higher overhead of DS and FRTS packets, energy consumption also increases slightly. One may want to use FRTS packets only if a reasonably high load in a more or less unidirectional communication pattern is expected. Note, however, that when the load is low, the number of exchanged packets, and therefore the increased overhead, is also low.

Taking priority on full buffers

The second solution is a scheme that we call *full-buffer priority*. When a node's transmit/routing buffers are almost full, it may prefer sending to receiving. This means that when a

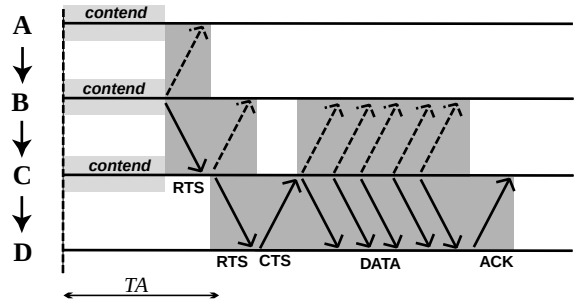

Figure 6: Taking priority upon receiving RTS.

node receives an RTS packet destined for it, it immediately sends its own RTS packet to another node, instead of replying with a CTS like normal. This is depicted in Figure 6. It has two effects. First, the node has an even higher chance of transmitting its own message, since it effectively wins the medium upon hearing a competing RTS; in Figure 6, node C may transmit to node D after losing contention to node B. Thus the probability that the early sleeping problem occurs is lower. Secondly, the full-buffer priority scheme introduces a limited form of flow control into the network, which is advantageous in a nodes-to-sink communication pattern; in Figure 6, node B is prevented from sending until node C has enough buffer space.

We must, however, be careful with the full-buffer priority scheme, since it is dangerous in a high-load situation where communication is not unidirectional. When all nodes in an omnidirectional communication pattern start taking priority, chances of collisions increase rapidly. Therefore, T-MAC uses a threshold: a node may only use this scheme when it has lost contention at least twice. This threshold guarded, in our experiments, the performance in an omnidirectional communication pattern, while still increasing the maximum throughput in a unidirectional pattern.

4. EXPERIMENTS

In our experiments, we compared three protocols: CSMA, S-MAC, and T-MAC. We include CSMA because we consider it as a 'worst case': it has no energy saving features at all. When a node is not transmitting, its radio is set to receive. We include S-MAC because it was, like T-MAC, designed for wireless sensor networks and uses in-band signalling as well.

4.1 Simulation setup and parameters

Our protocol design and evaluation is, for now, based on simulation. In the OMNeT++ discrete event simulation package [11], we have built a realistic model of the EYES wireless sensor nodes [13]. The model has the same limits on clock resolution and precision, radio turn-around and wake-up times, and transmission bit rates as the EYES nodes do. Energy consumption in the model is based on the amount of energy the real nodes use: 20 μA while sleeping, 4 mA while receiving and 10 mA while transmitting a DC-balanced signal [5, 6, 9].

Using these modeled nodes, we have built a network of 100 nodes in a 10 by 10 grid. We have chosen a radio range so that non-edge nodes all have 8 neighbors. We admit that this perfect world is not a realistic setup for most sensor networks, but it serves our simulation needs well, for example, we do not need to know the exact location of each node to generate a workload.

For the nodes-to-sink communication pattern, we used a randomized shortest path routing method: for each message, the possible next hops are enumerated. Next hops are eligible if they have a shorter path to the final destination than the sending node. From these next hops, a random one is chosen. Thus messages flow in the correct direction, but do not use the same path every time. No control messages are exchanged for this routing scheme: nodes automagically determine the next hop. Multi-hop messages travel at least two hops per frame because of T-MAC's in-band signaling: the CTS and ACK messages keep direct neighbors awake.

In our experiments, we tested the S-MAC protocol with a frame length of one second, and with several lengths of the active time, varying from 75 ms to 915 ms. For the T-MAC protocol, we always used a frame length of 610 ms (20000 ticks of a quartz crystal) and an interval *TA* with a length of 15 ms. The T-MAC protocol can optionally be deployed with overhearing avoidance, full-buffer priority, and FRTS. We show the best combination of these options in each experiment.

4.2 Results

We start with a simple benchmark, where network load is homogeneous. We then introduce experiments based on communication patterns and end with a complete, realistic scenario.

Except for the complete scenario, we exercised all experiments with multiple network loads. The load is on the horizontal axes of the graphs. The vertical axes show the energy consumption in the network, since this is our main design criterion.

Since throughput is not completely unimportant, we will end each graph at the point where less than 90% of the messages are correctly received. In a multi-hop pattern, like nodes-to-sink, this means that we want 90% of all messages to finally reach the sink node. We do not use the 90%-restriction for CSMA: since CSMA has no built-in retransmits, it is far less reliable.

The message lengths are data payload sizes, and do not include the MAC header: 4 bytes for CSMA, 6 bytes for S-MAC and T-MAC.

Homogeneous local unicast

(Figure 7) In out first experiment, nodes send packets with some payload (20 or 100 bytes) to one of their neighbors at random. Although this is not a realistic communication pattern, it serves as a 'base case'. For the T-MAC protocol, we used overhearing avoidance, but no FRTS or full-buffer priority mechanisms.

In Figure 7(left), we can clearly see that CSMA provides no energy savings. The energy consumption when idle is 4 mA (the current drawn by the radio in receive mode), slightly going up when more messages are sent.

For the S-MAC protocol, various lines are shown for different lengths of the active time. An additional line is drawn, which connects the S-MAC graphs that use the least energy for each load. So on this line, the S-MAC protocol is tuned to provide at least 90% throughput while using as little energy as possible *for each individual load*. From now, we will only show similarly tuned lines for the S-MAC protocol.

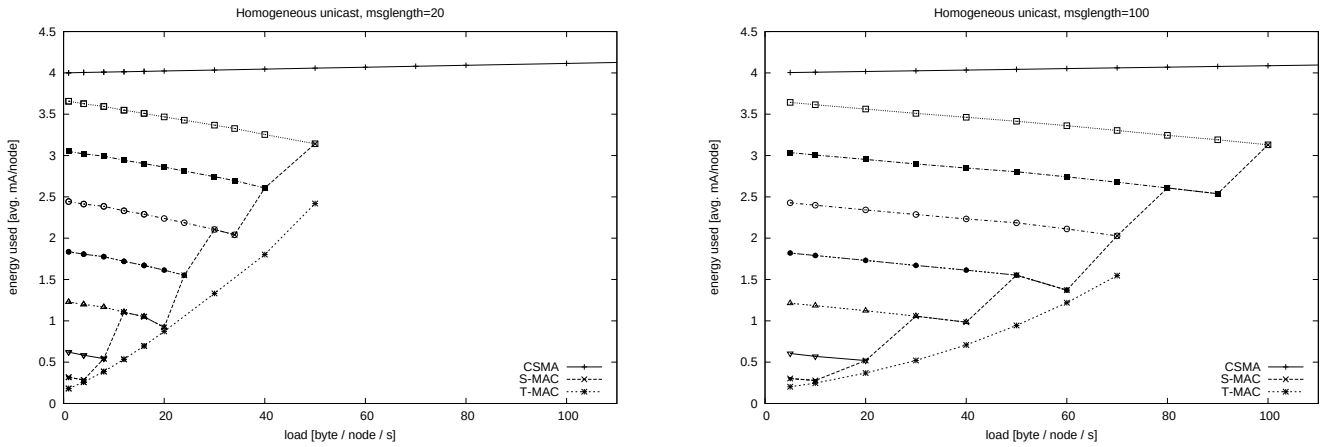

Figure 7: Homogeneous local unicast for small (left) and large messages (right). The numbers on the S-MAC line indicate the length of the active time per 1-second frame.

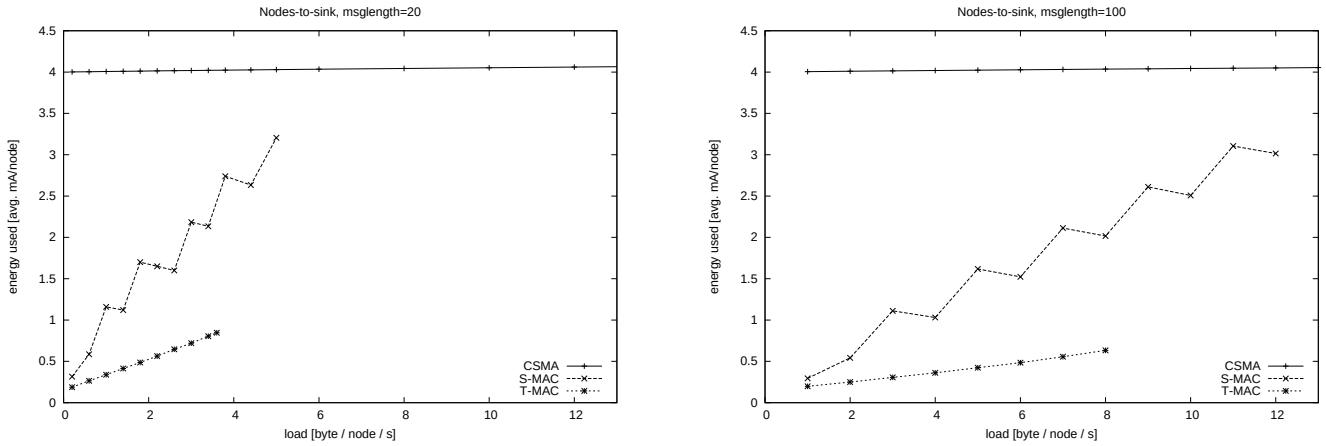

Figure 8: Nodes-to-sink performance for small (left) and large messages (right).

We expect that the homogeneous experiment is the best case for the S-MAC protocol, since the load is constant in both time and location. We see that the T-MAC protocol performs as well as the (per load tuned) S-MAC protocol.

The fact that the T-MAC protocol uses even less energy than S-MAC is due to the fact that we tested the S-MAC protocol only with a limited number of discrete lengths of the active time. To get an optimal parameter value for each load, the S-MAC protocol would require complicated tuning, as it would in real deployment. The adaptive behavior of T-MAC, on the other hand, requires no explicit tuning.

Figure 7(right) shows that that the maximum throughput of the T-MAC protocol with large messages (100 byte payload) is less than that of the S-MAC protocol. This is mainly due to variations on the early sleeping problem as described in Section 3.5.

Nodes-to-sink communication

(Figure 8) In this experiment, nodes send messages to a single sink node at the corner of the network. Messages are routed from node to node with a (slightly randomized) shortest path algorithm. No data aggregation is used. For the T-MAC protocol, we used overhearing avoidance, the full-buffer priority mechanism, and the FRTS mechanism.

Figure 8 shows that T-MAC uses less energy than S-MAC. We expected this, because in the nodes-to-sink communication the load varies with the location of nodes: there is more traffic in the neighborhood of the sink node. This also explains why the absolute load is much lower (factor of 10) for nodes-to-sink than for local unicast: the rate at which the sink can handle incoming messages limits the load individual nodes can generate without congesting the network (around the sink).

As with homogeneous local unicast we see that the maximum throughput of T-MAC is less than that of S-MAC. Our experience with other message sizes and communication patterns is that the maximum throughput of T-MAC is at worst about 70% of S-MAC. We do not worry about too much about T-MAC's reduced maximum throughput, because it only occurs under extreme loads that are best avoided by sensor applications, for example, by aggregating messages.

Early sleeping problem

(Figure 9) In this experiment, we show the effectiveness of the measures addressing the early sleeping problem (Section 3.5). We see that the FRTS mechanism increases maximum throughput by approximately 75% (0.08 vs. 0.14 mes-

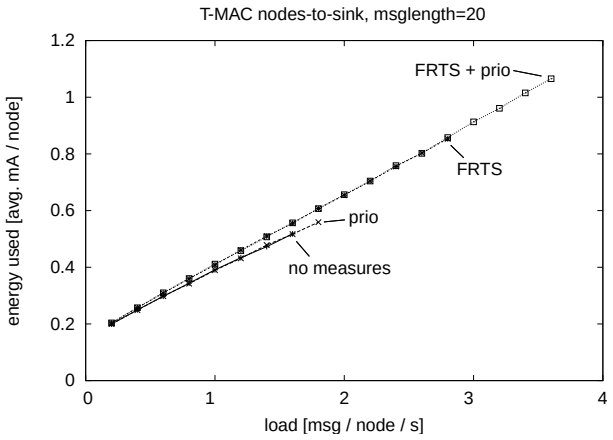

Figure 9: T-MAC options in a nodes-to-sink communication pattern.

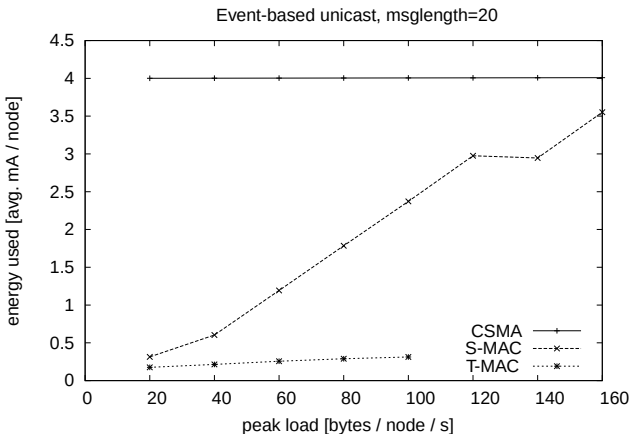

Figure 10: An event-based scenario, where active nodes exchange local unicast messages.

sages per second), at the cost of some energy. Adding the full-buffer priority mechanism adds approximately 30% (0.14 vs. 0.18 messages per second) without the cost of extra energy.

Event-based local unicast

(Figure 10) We now proceed towards a more realistic scenario. In this experiment, events occur in the network with a frequency of one per 10 seconds. Events have an average duration of 5 seconds and affect an area of approximately 9 nodes. These nodes then send local unicast messages to their neighbors for the duration of the event. A neighbor that receives one of these messages replies with a probability of 20%. We performed multiple measurements, with different message frequencies during events. This frequency is on the horizontal axis of the graph (Figure 10). For T-MAC, we used overhearing avoidance but no FRTS and no full-buffer priority.

Figure 10 shows that T-MAC uses much less energy than either S-MAC or CSMA, especially when the message frequency during events increases. However, the maximum frequency that T-MAC can handle is lower than that of S-MAC, like we have seen in the nodes-to-sink communication pattern. Again, T-MAC suffers from the early sleeping problem, because we have relatively many edge nodes.

Event-based local unicast and node-to-sink reporting

(Figure 11) This is an experiment with a complete scenario. When no events happen, nodes exchange local messages of 10 bytes with each other every 20 seconds. They also report to a sink node every 100 seconds. When an event happens (once every 10 seconds, like in the previous experiment), nodes in the neighborhood of the event start sending local unicast messages of 30 bytes, with a rate of 4 per second. They then also send messages of 50 bytes to the sink, once per second. These messages are aggregated in the network.

To be able to handle this kind of traffic, we had to tune S-MAC to listen for at least 715 ms in every second. For the T-MAC protocol, we used overhearing avoidance, but no FRTS and no full-buffer priority.

Figure 11 clearly shows the disadvantage of a fixed-length active part: to be able to handle a short-lived burst of

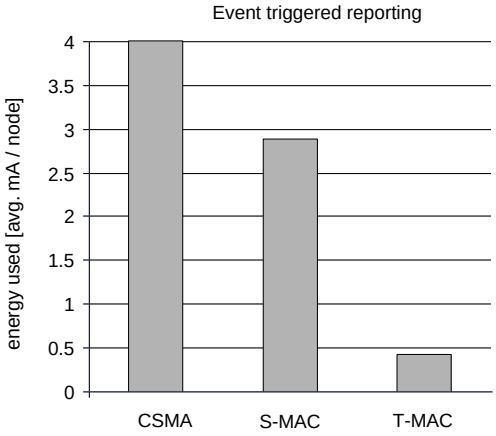

Figure 11: A complete scenario with both periodic reporting and event-based messages.

traffic–even though that happens infrequently–the deployer of S-MAC must choose a long active part of the duty cycle. S-MAC therefore wastes much energy at times when no events happen. By adaptively changing the duty cycle, T-MAC can decrease the used energy in this scenario by a factor of 5.

5. REAL IMPLEMENTATION

After optimizing the T-MAC protocol using simulations, we implemented most of the protocol on the actual EYES hardware [13]. We did not implement all features. To test the effectiveness of the full-buffer-priority and future-RTS schemes, a large-scale experiment is needed, involving a lot of nodes. Since there was no time to do such an experiment, implementation of these features was senseless.

We have also not implemented the possibility to keep multiple schedules yet. Although this is fairly easy to implement, we have only tested T-MAC with single-cluster configurations.

During testing, we noted that the nodes' schedules would drift apart relatively fast. Even though the time-keeping on

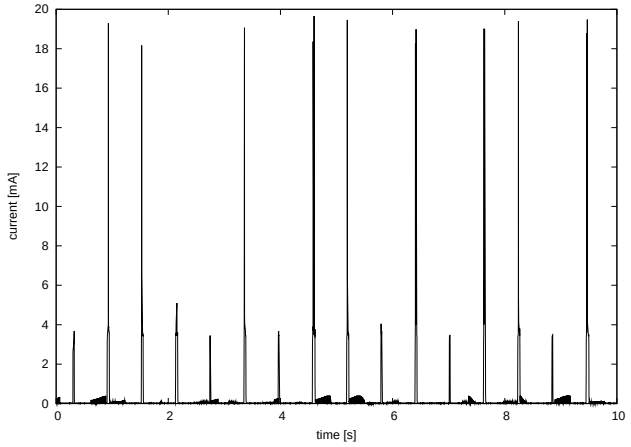

Figure 12: Trace of the electrical current, receiving node, 1 message / second.

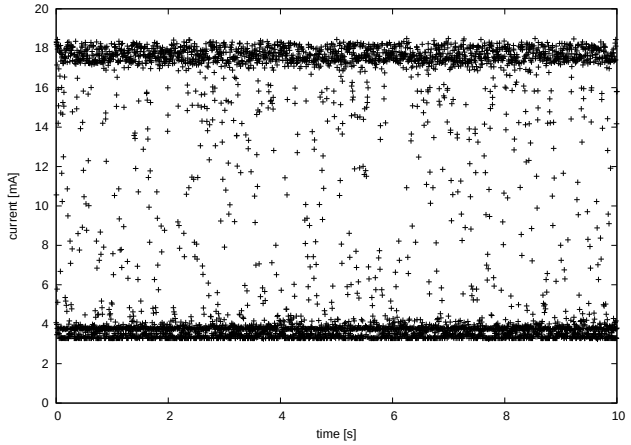

Figure 13: Trace of the electrical current, transmitting node, full speed.

all nodes is based on ticks of a quartz crystal, some of the nodes became unreachable within as little as 10 minutes. We had not simulated this effect.

To solve the drift problem, we used a simple correction scheme: when a node receives a SYNC message that contains almost, but not exactly, its own schedule, the node adjusts its own schedule towards the received schedule. To allow a converging situation, the schedule is only adjusted for 50% of the difference between the two schedules.

The drift correction solved the problem: an experiment showed that nodes were still perfectly synchronized after more than 10 hours.

The final implementation performs well. When sending a continuous stream of data messages, the nodes rarely go to sleep. But when no messages are sent, the indication LEDs on the nodes blink peacefully. The radio is then in the receive mode for 15 ms out of every 610 ms, which is less than 2.5% of the time.

5.1 Energy consumption

After the implementation of the T-MAC protocol, we performed a number of energy consumption experiments. In

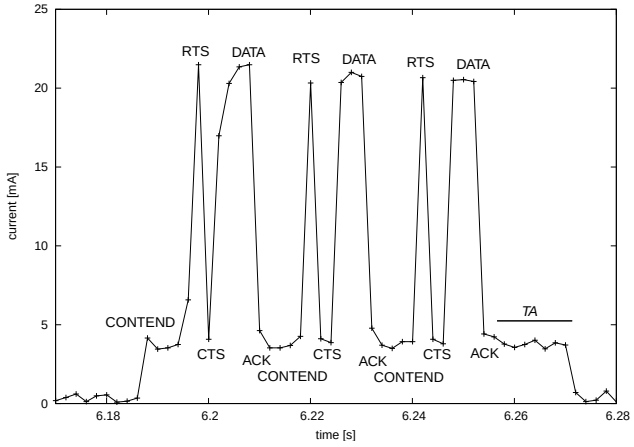

Figure 14: Trace of the electrical current, node successfully transmitting 3 messages.

these experiments, one node was sending, another receiving. The message length was 20 bytes. We measured the electrical current through both the sending and the receiving node using using specialized equipment, which sent the values to a logging computer. After precise calibration, we could measure the current through the nodes with a precision of approximately 15 μA. The sample rate was 500 Hz (limited by the PC software).

In Figure 12 we see a power trace of a node receiving approximately one message per second. There is a transmission in most frames, but the radio is still in sleep mode most of the time. The high spikes are caused by transmitting CTSs. In Figure 13 we see a node that transmits messages at maximum rate. It is clear that this node never sleeps, as it should not. Instead, it toggles between receive (3.75 mA) and transmit (18 mA). Figure 14 shows a closeup of a node transmitting 3 messages during a single frame.

msg / s	transmit mA	receive mA
0	0.138	0.138
1	0.400	0.246
10	1.516	0.890
max	9.590	7.473

Table 2: Average energy consumption of sending and receiving EYES nodes (T-MAC protocol).

Table 2 shows the average electrical current during each experiment. We can see that transmitting nodes use significantly more energy than receiving nodes. This is logical, since transmitting with our radio takes more energy than receiving. More importantly, we see that the idle average current (0.138 mA) is less than 4% of the current of a non-energy saving protocol (which would be between 3.75 and 4 mA). This is well above the theoretical limit (TA/T_{frame} = 15ms/610ms = 2.5%). However, the theoretical limit only takes the *radio* energy consumption into account, not the CPU, EEPROM, RS232 voltage converter and other components. We think that 96% overall energy savings is a very good result.

6. CONCLUSIONS AND FUTURE WORK

To solve the problem of idle listening in a wireless sensor network, we have proposed the T-MAC protocol. T-MAC dynamically adapts a listen/sleep duty cycle in a novel way, through fine-grained timeouts, while having minimum complexity. Simulations have shown that the T-MAC protocol introduces a way of decreasing energy consumption in a volatile environment where the message rate fluctuates, either in time or in location. Implementation of the T-MAC protocol has shown that, during a high load, nodes communicate without sleeping, but during a very low load, nodes will use their radios for as little as 2.5% of the time, saving as much as 96% of the energy compared to a traditional protocol.

We have also identified a problem with the T-MAC protocol (the *early sleeping problem*) and proposed original solutions for this problem (FRTS and full-buffer priority). The trade-off for very low energy consumption is a decreased maximum throughput. We have encountered the early sleeping problem only in a perfect, simulated world. Future work is needed to determine the actual seriousness of the problem and to test the effectiveness of the proposed solutions. Other solutions may well exist.

We have only experimented with a static, non-mobile network. We expect problems when applying virtual clustering to groups of mobile nodes. The virtual clustering technique, proposed in the S-MAC protocol and re-used in the T-MAC protocol, has not been researched thoroughly. Since synchronization of schedules can have a great impact on the energy consumption, clustering is important. Virtual clustering, and multi-hop synchronization in general, is an interesting research subject in itself.

T-MAC's reductions in energy consumption are very promising. This novel protocol is the subject of an ongoing study, and we expect more detailed results in the future.

7. ACKNOWLEDGEMENTS

We like to thank Niels Reijers, the anonymous reviewers, and our shepherd David Culler for providing constructive comments that improved the quality of the paper.

8. REFERENCES

[1] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: a media access protocol for wireless LAN's. In *Conf. on Communications Architectures, Protocols and Applications*, pages 212–225, London, 1994.

[2] P. Havinga and G. Smit. Energy-efficient TDMA medium access control protocol scheduling. In *Asian International Mobile Computing Conference (AMOC 2000)*, pages 1–9, November 2000.

[3] J. Hill and D. Culler. Mica: a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, Nov-Dec 2002.

[4] LAN MAN Standards Committee of the IEEE Computer Society. *IEEE Std 802.11-1999, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*. IEEE, 1999.

[5] RFM. *TR1001 868.35 MHz Hybrid Tranceiver*.

[6] RFM. *ASH Transceiver Designer's Guide*, October 2002.

[7] S. Singh and C. Raghavendra. PAMAS: Power aware multi-access protocol with signalling for ad hoc networks. *ACM SIGCOMM Computer Communication Review*, 28(3):5–26, July 1998.

[8] M. Stemm and R. H. Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. *IEICE Transactions on Communications*, E80-B(8):1125–1131, 1997.

[9] Texas Instruments. *MSP430x1xx Family User's Guide*. SLAU049B.

[10] Y. Tseng, C. Hsu, and T. Hsieh. Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. In *21st Conf. of the IEEE Computer and Communications Societies (INFOCOM)*, volume 1, pages 200–209, June 2002.

[11] A. Varga. The OMNeT++ discrete event simulation system. In *European Simulation Multiconference (ESM'2001)*, Prague, Czech Republic, June 2001.

[12] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *21st Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 3, pages 1567–1576, June 2002.

[13] http://eyes.eu.org/sensnet.htm.

[14] http://webs.cs.berkeley.edu/tos/.