Analyzing MAC Protocols for Low Data-Rate Applications

KOEN LANGENDOEN
Delft University of Technology
and
ANDREAS MEIER
ETH Zurich

The fundamental wireless sensors network (WSN) requirement to be energy-efficient has produced a whole range of specialized medium access control (MAC) protocols. They differ in how performance (latency, throughput) is traded off for a reduction in energy consumption. The question "which protocol is best?" is difficult to answer because (i) this depends on specific details of the application requirements and hardware characteristics involved, and (ii) protocols have mainly been assessed individually with each outperforming the canonical S-MAC protocol, but with different simulators, hardware platforms, and workloads. This article addresses that void for low data-rate applications where collisions are of little concern, making an analytical approach tractable in which latency and energy consumption are modeled as functions of key protocol parameters (duty cycle, slot length, number of slots, etc.). By exhaustive search we determine the Pareto-optimal protocol settings for a given workload (data rate, network topology). Of the protocols compared we find that WiseMAC strikes the best latency versus energy-consumption tradeoff across the range of workloads considered. In particular, its random access scheme in combination with local synchronization not only minimizes protocol overhead, but also maximizes the available channel bandwidth.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; C.3 [**Special-Purpose and Application-Based Systems**]: *Real-time and embedded systems*

General Terms: Performance

Additional Key Words and Phrases: Energy efficiency, performance modeling, sensor networks

---

The work presented in this article was supported by CTI grant number 8222.1 and the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.
Author's addresses: K. Langendoen, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft, University of Technology, Delft, The Netherlands; email: K.G.Langendoen@tudelft.nl; A. Meier, ETH Zurich, ETZ G-77, Gloriastrasse 35, 8092, Zurich, Switzerland; email: a.meier@tik.ee.ethz.ch.

DOI 10.1145/1806895.1806905 http://doi.acm.org/10.1145/1806895.1806905

ACM Reference Format:
Langendoen, K. and Meier, A. 2010. Analyzing MAC protocols for low data-rate applications. ACM Trans. Sensor Netw. 7, 1, Article 10 (August 2010), 34 pages. DOI = 10.1145/1806895.1806905 http://doi.acm.org/10.1145/1806895.1806905

1. INTRODUCTION

Application scenarios for Wireless Sensor Networks (WSNs) often involve battery-powered nodes being active for considerable lengths of time, several months to years, without external control by human operators after initial deployment. With today's hardware platforms drawing tens of milliamperes of current and battery capacity being limited to a few ampere hours, the need for energy management becomes apparent; without it a node would drain its batteries within a couple of days. This fundamental need for energy-efficient operation has drawn the attention to the radio, which is the component of a typical sensor node that consumes most energy.

Duty cycling the radio, that is, repeatedly switching it off for some time, is the only way to achieve the required two orders of magnitude reduction in energy consumption for extending lifetime from days to years. This duty cycling effectively reduces the available bandwidth on the radio channel, and hence limits the amount of data that can be communicated through the sensor network. Note that the reverse does not necessarily hold; applications constraining their payload do not automatically extend the lifetime of a sensor node because current radios consume about as much energy when running idle as when transmitting or receiving data. Only by putting the radio into (deep) sleep does energy consumption reduce to zero (i.e., from the milliwatt to microwatt range), which is the task of the Medium Access Control (MAC) layer driving the radio hardware.

Since the introduction of the canonical S-MAC protocol [Ye et al. 2002], a whole range of energy-efficient MAC protocols supporting low data-rate applications have been developed (see Section 2). These MAC protocols all trade off performance (latency, throughput) for a reduction in energy, but differ in complexity and flexibility to adapt to traffic fluctuations, topology changes, and varying channel conditions. Simple protocols are often based on random access (CSMA), while more complex protocols organize channel access according to some predefined schedule (TDMA).

Developers of long-running applications must be careful in selecting the MAC protocol that suits their needs best, so that they can squeeze the most out of the limited hardware resources. As such it is essential to understand how MAC protocols operate in specific conditions (data rates, traffic patterns, interference levels, etc.). At the moment, however, there is no reference framework for doing so. Typical MAC protocols have been demonstrated to outperform S-MAC , but with different simulators, hardware platforms, and workloads, making it very difficult to assess their behavior in another context. Comparative studies like Halkes et al. [2005] shed some light on the relative performance of a few protocols, but again only for a limited number of deployment scenarios. The latter aspect can be addressed by analytical models capturing relevant system parameters, as for example the analytical model for the energy consumption of the data-link layer by Zhong et al. in [2004], which models the network traffic and the radio characteristics. However, Zhong et al. discussed very few and meanwhile outdated MAC protocols (ALOHA, CSMA) while advanced MAC protocols, as considered in this article, contain a number of internal parameters that must be taken into account too. For optimal performance, it is simply not sufficient to compare MAC protocols running with their standard settings. Instead the whole parameter range for all protocols have to be compared against each other.

This article addresses the exploration void for the case of low data-rate applications where energy efficiency is needed the most. We present analytical models of state-of-the-art MAC protocols that are driven by a set of context parameters (e.g.radio characteristics, data rate, and network topology) as well as internal MAC-protocol parameters (e.g.duty cycle, slot length, and number of slots). To avoid the intricacies of modeling retransmissions and the interplay with other protocol layers, we do not take collisions or other sources of packet loss into account. This rules out a small class of applications that exhibit bursty behavior, but covers the majority of low data-rate applications and allows that the models can be kept rather simple in nature. This has three advantages. First, it makes our analytical approach scalable in the sense that analyzing various protocols is feasible. Second, it provides fairness for the comparison of the protocols, as with the rather simple parameters we avoid getting lost in minor model details while losing sight of the big picture. Last, we can evaluate MAC protocols in a large number of different settings. This article presents a few results from such a design-space exploration, showing that reducing idle listening and overhearing, and managing clock drift, are keys to achieving energy efficiency for low data-rate applications. However, the true value of the work lies in the analytical models, which are made available to the research community so individuals can use them to select the MAC protocol that favors their specific requirements and conditions.

The remainder of this article is structured as follows. Section 2 presents a brief overview of MAC protocols especially developed for WSNs. Section 3 introduces the modeling framework, followed by a few example models of state-of-the-art MAC protocols in Section 4. These models are analyzed in Section 5 for a set of delay-insensitive monitoring applications in typical network topologies, where energy efficiency is the prime consideration. Most MAC protocols were originally designed for being used with a byte-stream radio. In Section 6 we discuss how the MAC protocols and the modeling framework can be adapted for being used with packet-based radios. Section 7 concludes the article.

2. ENERGY-EFFICIENT MAC PROTOCOLS

The primary concern of WSN-specific Medium Access Control protocols is to switch the radio into sleep mode; otherwise energy would be wasted due to so-called *idle listening* by nodes waiting for potential incoming traffic. Other sources of overhead that should be avoided include *overhearing* of messages

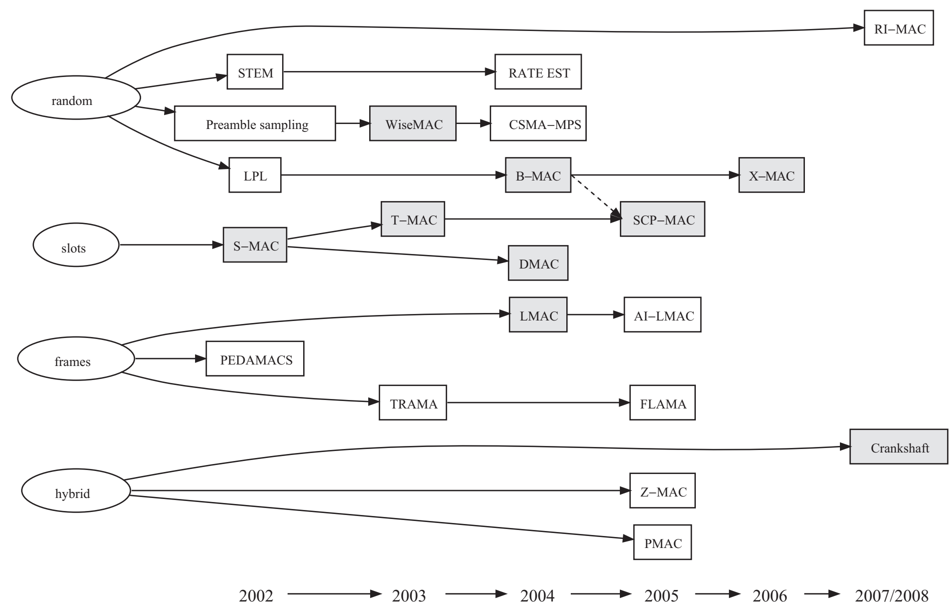


Fig. 1. Taxonomy of MAC protocols according to organization and historic development; the protocols analyzed in this article are highlighted.

destined to other nodes, *unnecessary sending* if the receiver is not listening, *collisions*, for example, due to hidden terminals, and *protocol overhead* for medium reservation and clock synchronization. An additional complication for the low data-rate applications considered in this article is that amortizing overheads, for example, by piggybacking protocol information on data messages, becomes rather difficult when applications only send out a message every few seconds, or even minutes.

All energy-efficient MAC protocols duty cycle the radio, at the expense of reducing bandwidth and increasing latency. The reduction in throughput is of little concern since commonly used radios like the TI CC1000 offer ample bandwidth (76.8 kb/s) compared to what most applications need ($\ll$ 1 kb/s). The latency increase, on the other hand, is much more of a concern especially when targeting duty cycles of less than 1% in a multihop scenario a message may encounter a delay up to a complete sleep interval (1 s or more) on every transfer, resulting in long end-to-end latencies.

A whole range of WSN-specific MAC protocols has been developed, each with its own tradeoff between latency, throughput and energy savings. In general, we distinguish three classes of protocols, depending on how strict access to the channel is organized. Figure 1 shows a taxonomy of MAC protocols along this classification of *random*, *slotted*, and *frame-based* (TDMA) access, as well as the historic development within each class. Note that Figure 1 is an excerpt from an elaborate survey of energy-efficient MAC protocols available online.[1]

[1] `https://apstwo.st.ewi.tudelft.nl/~koen/MACsoup/`.

The class of *random access* protocols puts no restrictions on when a sleep/active cycle is taking place. Neighbors therefore do not need to coordinate their cycles and consequently wake up independently of each other. This avoids the overheads and bookkeeping associated with running a time synchronization protocol, but leaves it up to the sending nodes to arrange a rendezvous with the intended receiver whenever it wakes up. An effective way is to stretch the length of the standard preamble to cover one complete sleep interval, which assures that the receiver (when polling the channel for activity) will detect a signal and eventually detect a start symbol, followed by the true message. This technique is known as *low-power listening* (LPL) [Hill and Culler 2002] or *preamble sampling* [El-Hoiydi 2002], and was subsequently refined by the B-MAC protocol [Polastre et al. 2004], which added a user-controlled sleep interval, and by the WiseMAC protocol [El-Hoiydi and Decotignie 2004], which tracks the phase offsets of neighbors' schedules allowing senders to transmit a message just in time with a short-length preamble saving energy and bandwidth. The X-MAC protocol [Buettner et al. 2006] in turn enhances B-MAC by adapting it for packet-based radios sending out streams of packets instead of one long preamble.

The *reverse* approach to LPL, originally proposed with the Piconet project [Bennett et al. 1997], was recently redesigned for sensor networks as the RI-MAC protocol [Sun et al. 2008]. Instead of listening periodically, beacons are sent at regular intervals, indicating that the node is ready to subsequently receive a message. This avoids sending a long wakeup preamble (the sender has to listen for the receiver's beacon instead) and shortens tranmission times considerably; a drawback is that beacons interfere with ordinary traffic as well as with each other. A hardware approach to arrange a rendezvous is to use a second, low-power radio to send a wakeup signal, which will prompt the receiver to power up its primary radio to listen for the message that follows shortly. The idea of using a wakeup radio was explored by STEM [Schurgers et al. 2002] and RATE EST [Miller and Vaidya 2004] in simulation. Since we are not aware of any hardware implementation in regular use, we do not consider wakeup radios in the remainder of this article.

The class of *slotted access* protocols requires nodes to synchronize on a global notion of time, which is then organized as a sequence of slots. This organization allows nodes to collectively iterate through a sequence of active/sleep cycles. In the simplest case of the S-MAC protocol [Ye et al. 2002], nodes spend a fixed amount of time in active and sleep mode, that is, they wake up at the beginning of each slot and go back to sleep after a fixed-length interval. Within an active period, nodes follow the classic CSMA with collision avoidance (RTS/CTS signaling) approach to gain access to the channel. To account for variations in traffic, both in time and location, T-MAC [van Dam and Langendoen 2003] introduces a simple timeout mechanism to adapt the length of the active period to the actual load. At the start of each slot, nodes listen for a short period (around 10 ms) to see if there is any communication to engage in; if not, they switch back to sleep mode. This allows saving a lot of energy over S-MAC running at a duty cycle that matches the load at the busiest node in the network. The SCP-MAC protocol [Ye et al. 2006] even manages to reduce the length of the active period to just 1-2 ms by orchestrating senders to resolve contention before the receivers poll the channel (see Section 4.2). A down-side shared by all slotted protocols is that communication is grouped at the beginning of each slot, raising the chances for collisions, hence limiting their dynamic range to low traffic rates only.

The class of *frame-based access* protocols groups slots into frames, and eliminates contention by precisely scheduling who is allowed to send in which slot. Computing these (periodic) schedules is rather difficult. Simple (distributed) policies lead to overprovisioning, as in the case of LMAC [van Hoesel and Havinga 2004]; complicated policies taking actual traffic loads into consideration induce great complexity and relatively large memory footprints for maintaining neighbor states, making protocols like TRAMA [Rajendran et al. 2003] hard to use in practice.

Lately, a number of hybrid protocols have been proposed that try to combine the best of all domains, basically by putting a TDMA-overlay structure on top of CSMA/CA. This combination employs the flexibility of random access with a (much) lower chance of collision. The P-MAC [Zheng et al. 2005], Z-MAC [Rhee et al. 2005], and Crankshaft [Halkes and Langendoen 2007] are example protocols from this hybrid category, with Crankshaft being used as the representative in the remainder of this article.

3. MODELING FRAMEWORK

Given the wide variety of MAC protocols, it is important to understand their (relative) performance, such that the best protocol can be selected for a specific deployment. The focus in this article is on long running, low data-rate applications, which already corners the set of possible operational conditions that has to be considered when evaluating MAC protocols. Nevertheless, a thorough exploration should consider variation in workload (application traffic), radio and channel characteristics, and network topology (e.g., node density).

We take an analytical, model-based approach to allow for a fast evaluation of a number of MAC protocols in a rather large space of operational conditions. In particular we present models for the network topology, radio hardware, application traffic, and nine MAC protocols. These MAC models are discussed in the next section. Here, we present the other three models that capture the essential operating conditions that are varied when evaluating the MAC protocols in Section 5.

3.1 Application Characteristics

We distinguish two kinds of applications, namely, event-based and periodic reporting. In many monitoring type of applications, nodes simply send status reports on a regular basis to a central sink node for (offline) processing and storage. Example deployments include observing the nesting habits of Storm Petrels by monitoring the presence of birds in individual burrows [Mainwaring et al. 2002], recording the change of light intensity at various heights in a Redwood tree [Tolle et al. 2005], and observing the microclimate (temperature and

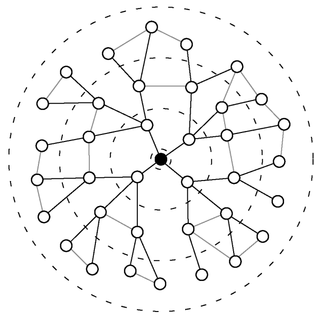


Fig. 2. Sample spanning tree with the sink at level 0 and a depth of 3.

humidity) in a potato field [Goense et al. 2005]. The typical communication pattern that emerges from periodic reporting is a spanning tree with traffic flowing from the leaves to the data sink at the root (see Figure 2). Depending on the specific deployment, data capturing may be synchronized (network-wide snapshots), and data may be aggregated at intermediate nodes. For simplicity we consider unsynchronized data capture without aggregation only, but extending the models to include these features has proven to be straightforward.

The class of event-based applications shows a much more erratic communication pattern as network activity is triggered by some external event. For example, in the case of a surveillance system, the detection of an intruder prompts the forwarding of an alarm message along some route to the sink. Also, many monitoring applications can be optimized to report only significant changes (bird enters/leaves a burrow) instead of continuously streaming messages with redundant data (the presence of the bird).

For both classes of applications, we are interested in the amount of energy consumed, because that determines the lifetime of the network (i.e.,the time until the first node runs out of energy), and hence the feasibility of the deployment. In the case of event-based applications, almost all energy is spent on keeping the network alive as the data rate is close to zero, assuming applications where events are rare. In the case of periodic reporting, additional energy is spent on forwarding data packets to the sink, with nodes close to the sink typically handling more traffic. A MAC model should take both classes into account and incorporate system-level traffic (for keeping the network alive) as well as application-level traffic.

The second performance metric of interest is latency. In many event-based scenarios (end-to-end) latency is bounded by application requirements, for example, an intrusion detection must be reported at the sink within a few seconds. Depending on the MAC protocol, there can be a rather large difference between average latency and worst-case latency. For example with LMAC , a TDMA-style protocol in which each node owns a time slot, the average delay at each hop is half the frame length, but in the worst case the send slot is just missed and the message is delayed for (almost) a complete frame. Reporting worst-case latency, however, has little merit as in real life collisions and external interference rule out any strict guarantees to begin with. We therefore only model the average latency of the MAC protocols concerned.

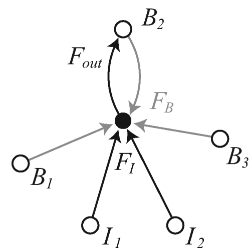


Fig. 3. A MAC model's local view; black arrows indicate traffic flowing through; grey arrows indicate interference from background nodes.

As the focus of this article is on low data-rate applications, throughput is of no concern, other than that MAC protocols should not be driven into overload leading to collisions and queue overflows (which we do not model). This will be safe-guarded by adding boundary constraints on the amount of traffic flowing through the network. As a final note, we like to remark that, to keep the analysis tractable, we ignore the impact of external interference, that is, random packet loss is not considered. As a consequence, the MAC models do not need to consider retransmissions, which greatly simplifies the models.

3.2 Traffic Model

To keep a MAC model as simple as possible, we abstract away the actual network topology by detailing for every node what *input* traffic it is handling, and what *background* traffic is potentially bothering it being sent out by its other neighbors. Note that background nodes may be peers at the same level of the tree, as well as nodes at levels below and above that of the node itself. The latter group includes the parent node forwarding a node's outgoing traffic (among others) further up in the tree. Figure 3 illustrates the local view of a MAC model, which is sufficient to express a node's energy consumption using the particular traffic parameters from Table I. The network lifetime can then be obtained by simply iterating over all nodes and recording the maximum energy consumption for the traffic parameters at each node.

The traffic model embeds the spanning tree of the application on top of a raw network topology by specifying for each node the set of input (child) nodes I and the set of overheard (background) nodes B. This allows for accurate modeling of specific, irregular deployment scenarios, as well as for modeling regular topologies like grids and the ring structure of Figure 2.

We detail now the set of equations of the ring model being used in the MAC performance analysis in Section 5. In particular, we construct a spanning tree in the network that is based on shortest-hop routing to the sink located in the center. Assuming a uniform node density on the plane and a unit disk graph communication model, there are $C + 1$ nodes on the unit disk. Hence all nodes are in communication range with a fixed number of C neighbors. The nodes are grouped into rings according to their distance d (minimal hop count) to the sink (d=0). The first ring contains C nodes, from which we can derive the node

Table I. Traffic Model (for Node N) with Typical Parameter Values (The topology information is encoded by means of sets I and B (cf. Figure 3), satisfying $C = |I| + |B|$. Note that all parameters are node-specific, but that the indices (superscripts) have been omitted for clarity whenever possible (e.g.we write I instead of I^N).)

Parameter	Description	Value (Range)
P	Payload [byte]	32
F_S	Sampling frequency [#pkts/node/min]	0.01 – 10
C	Connectivity (#neighbors)	4 – 16
F_I	Input nodes' aggregate report frequency	$\sum_{n\in I} F_{out}^n$
F_B	Background nodes' aggregate report frequency	$\sum_{n\in B} F_{out}^n$
F_{out}	Output frequency	$F_I + F_S$

density, and subsequently the number of nodes N_d in ring d:

$$N_d = \begin{cases} 1, & \text{if } d = 0, \\ Cd^2 - C(d-1)^2 = (2d-1)C, & \text{otherwise.} \end{cases}$$

Knowing the number of nodes in each ring allows us to compute the average number of input links of a node at level d as the ratio between N_{d+1} and N_d:

$$|I_d| = \begin{cases} 0, & \text{if } d{=}D, \\ C, & \text{if } d{=}0, \\ N_{d+1}/N_d = (2d+1)/(2d-1), & \text{otherwise,} \end{cases}$$

where D denotes the maximum distance to the sink. Note that the number of input links is independent of the node density. Knowing the number of input nodes and the nodes' basic sampling frequency (F_S), we can compute the output frequency:

$$F_{out}^d = \begin{cases} F_S, & \text{if } d{=}D, \\ F_I^d + F_S = |I_d| F_{out}^{d+1} + F_S, & \text{otherwise.} \end{cases}$$

This iterative formula can be reduced to

$$F_{out}^d = F_S(D^2 - d^2 + 2d - 1)/(2d-1), \tag{1}$$

which gives us a closed formula for the input rate (using $F_I^d = F_{out}^d - F_S$):

$$F_I^d = \begin{cases} F_S D^2 C, & \text{if } d = 0, \\ F_S(D^2 - d^2)/(2d-1), & \text{otherwise.} \end{cases} \tag{2}$$

To arrive at the aggregate background traffic we assume that the B nodes in Figure 3 on average generate the same load (F_{out}^d) as the node itself:

$$F_B^d = |B_d| F_{out}^d = (C - |I_d|) F_{out}^d. \tag{3}$$

With Equations (1)–(3) determining the node that consumes the most energy requires just D evaluations (evaluate one node per ring) of a MAC model, whereas brute-force (node by node) processing would involve CD^2 evaluations. When considering real-world, irregular topologies, similar speedups can

Table II. Radio Model with Typical Parameter Values for Radios Used in Sections 5 and 6

Parameter	Description	CC1000	CC2420	RFM TR1001
Type	Byte-stream vs. packet based	Byte	Packet	Byte
R	Rate [kbyte/s] (after channel encoding)	2.40	31.25	57.50
$T_{powerup}$	Turn radio on into RX or TX [ms]	2.10	2.40	0.5
T_{cs}	Time [ms] to turn the radio on and to probe the channel (carrier sense)	2.45	2.60	0.53
θ	Frequency tolerance [ppm]	30	30	30
L_{pbl}	Minimal preamble length [byte]	6	4	2.5

be achieved by focusing on a few bottleneck nodes through specifying their individual parameters (F_I, F_{out}, F_B, C, etc.).

3.3 Radio Model

Besides the traffic parameters, the evaluation of a MAC model requires input about the radio hardware that is, or will be, used in the specific deployment scenario. Since the objective of this article is to compare the relative performance of various MAC protocols, we can leave out many details. In particular, we are not computing an absolute energy consumption level, but only the effective duty cycle (i.e., the fraction of time the radio is switched on— regardless of whether it is in transceiving, idle listening, or powering up mode). This allows us to omit exact energy consumption numbers and only focus on the timing aspects. As such, a radio can be modeled with just three parameters: the time needed to power it up (i.e.,to transit from sleep into active mode), its data rate, and the time needed to do a carrier sense (including power-up).

An important issue with low data-rate applications is that clock drift becomes an issue for advanced MAC protocols that arrange nodes to wake up at precise moments in time to minimize energy consumption; without any data traffic the clocks of different nodes may drift apart and need to be accounted for. Therefore, we include a parameter θ that captures the precision of the (external) quartz crystal that determines the timing of the underlying (radio) hardware. An overview of the radios and their parameters used in this study are depicted in Table II. It should be noted that we assume a byte-stream radio for our MAC models, since most protocols were originally designed for them. It is, however, discussed in detail in Section 6 how the models are applicable to packet-based radios.

4. MAC MODELS

There are many parameters influencing the performance of a MAC protocol. First, there are the (external) radio and network parameters as introduced in the previous section. Second, there are the internal MAC parameters, such as duty cycle, number of slots, and polling time. In this section we show how we model the energy efficiency and latency of the protocols based on both the internal and external parameters.

We modeled nine different protocols, taking a selection of the whole MAC design space as indicated in Figure 1. An overview of these nine protocols and their internal parameters is provided in Tables III and IV for reference. Discussing

Table III. Internal (Configurable) Protocol Parameters

Protocol	Parameter	Description	Value (Range or Set)
B-MAC	T_w	Sample/polling period [s]	[0.02, 2]
Crankshaft	L_{data}^{max}	Maximum data length [byte]	32
	T_{sync}	Time between sync packets [s]	[12, 60]
	N_u	Number of unicast slots per frame	[4, 32]
	N_b	Number of broadcast slots per frame	2
D-MAC	N_{sleep}	Number of sleep slots	[6, 100]
	T_{sync}	Time between sync packets [s]	[60, 600]
LMAC	L_{data}^{max}	Maximum data length [byte]	{32,64,128,256}
	N_{slots}	Number of slots per frame	32
S-MAC	DC	Duty Cycle [%]	[0.1, 10]
	$T_{discover}$	Discovery interval [s]	360
	T_{active}	Active phase/slot [s]	[0.02, 0.1]
SCP-MAC	T_w	Sample/polling period [s]	[0.02, 2]
	T_{sync}	Time between sync packets [s]	[12, 60]
T-MAC	T_{slot}	Duration of a single slot [s]	[0.15, 10]
	T_{sync}	Time between SYNC packets [s]	100
	$T_{discover}$	Discovery interval [s]	360
WiseMAC	T_w	Sample/polling period [s]	[0.02, 2]
X-MAC	T_w	Sample/polling period [s]	[0.02, 2]
	T_{al}	Acknowledgement listen period [ms]	0.95

Table IV. Implementation-Specific (Fixed) Protocol Settings for Headers and Contention Windows (CW), the Latter Having a Slot Time of $T_{Slot}^{CW} = 0.62\,\text{ms}$

Protocol	Control packet	Size [byte]	CW [#slots]
B-MAC	L_{hdr}	9	15
	L_{ack}	$9 + L_{pbl}$	
Crankshaft	L_{hdr}	11	15
	L_{ack} (Sync)	$9 + L_{pbl}$	
D-MAC	L_{hdr}	10	15
LMAC	L_{hdr}	$7 + L_{pbl}$	–
S-MAC	L_{ctrl}	$8 + L_{pbl}$	15
SCP-MAC	L_{hdr} (Sync)	10	7 + 8
	L_{ack}	$8 + L_{pbl}$	
T-MAC	L_{ctrl}	$8 + L_{pbl}$	15
WiseMAC	L_{hdr}	7	15
	L_{ack} (Sync)	$9 + L_{pbl}$	
X-MAC	L_{hdr}, L_{ack}	$9 + L_{pbl}$	15
	L_{ps}	$5 + L_{pbl}$	

each protocol in detail would take up too much space, so we only present the most sophisticated from each category: LMAC (scheduled), SCP-MAC (slotted), WiseMAC (random access with wakeup prediction), and Crankshaft (hybrid). The remaining five protocols, namely S-MAC (slotted), T-MAC (slotted with timeout), D-MAC (slotted with convergecast), B-MAC (random access), and

X-MAC (packet-based random access), are briefly discussed in online Appendix A (available at the ACM Digital Library).

Given that for long-running applications most of the time (energy) will be spent in the operational phase of a MAC protocol, our models ignore any initialization procedures, which might be rather complex as in the case of setting up a TDMA schedule. Another important simplification follows from the assumption that messages do not get distorted (due to interference) or lost (due to collisions), so retransmissions do not need to be modeled. Collisions are unlikely for low data-rate traffic, but we do include boundary conditions to safeguard against the improper selection of protocol parameters, for example, a MAC protocol that samples the radio channel every second cannot possibly receive two packets per second.

In the following discussions of the four advanced MAC protocols (LMAC , SCP-MAC , WiseMAC , and Crankshaft), we first provide a short description of the protocol, then discuss the synchronization (clock drift) aspects, and finally provide equations for the average h-hop latency and energy efficiency (duty cycle).

4.1 LMAC

The first protocol that we consider is the Lightweight MAC [van Hoesel and Havinga 2004] protocol featuring a self-organizing TDMA scheme that organizes time into frames containing N_{slots} slots, each as illustrated in Figure 4. Every node owns one slot in which it sends out a header (to mark its occupancy), possibly followed by a data payload either addressed to a specific recipient (unicast) or to all nodes in range (broadcast). Consequently, a node must listen (i.e.,perform a carrier sense) in all slots other than its own to check for incoming data. To allow for collision-free transmissions and spatial reuse of slots, a header includes a list of all occupied slots in the owner's 1-hop neighborhood; after merging the occupancy information of its neighbors, a new node joining the network can select a free transmission slot within its 2-hop neighborhood. This distributed, interference-free slot selection mechanism obviates the need for explicit acknowledgment messages, which are left out from the LMAC protocol to save energy; the recovery from external interference is left to the upper layers in the protocol stack.

4.1.1 *Synchronization.* Synchronization is performed with every header that is sent, that is, in every occupied slot. In the worst case, a node hears one header per frame; hence, a sender and receiver can drift at most $2\theta T_{frame}$ apart (one running ahead, the other running behind). For efficiency the (receiving) nodes perform only a carrier sense, so the slot owner has to guard for the maximum clock drift by sending out a stretched preamble. Since it is unknown who is running ahead, the guard time must be twice the maximum drift:

$$T_{guard} = 4\theta T_{frame}. \tag{4}$$

where $T_{frame} = N_{slots} \cdot T_{slot}$.

Depending on the length of the slot T_{slot}, only a certain amount of payload can be transmitted at once. Therefore, if the slot should fit a payload of L^{max}_{data},

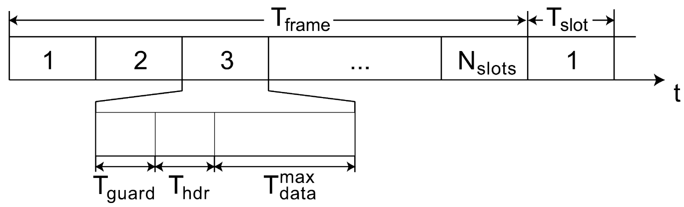


Fig. 4. The frame structure of LMAC.

the length of the slot results in

$$T_{slot} = T_{guard} + T_{hdr} + L_{data}^{max}/R. \tag{5}$$

4.1.2 *Latency.* After receiving a message, a node must wait for its own slot for forwarding. The message is therefore sent in one of the next $N_{slots} - 1$ slots, which results in an average latency of $T_{hd} = (N_{slots} - 1) \cdot T_{slot}/2$. This is based on the assumption that no queues occur at any given node due to the low data rate of the application. For the first hop, however, the message could be triggered right after the beginning of the owned slot and hence the message is delayed for $T_{init} = (T_{frame} + T_{slot})/2$ on average. Finally, for the last hop only part of the transmission slot is occupied when the payload is shorter than L_{data}^{max}. For a message that needs to be forwarded h hops, this results in an average latency of

$$\begin{aligned} L(h) &= T_{init} + (h-1) \cdot T_{hd} - (L_{data}^{max} - P)/R \\ &= (h \cdot T_{frame} - (h-2) \cdot T_{slot})/2 - (L_{data}^{max} - P)/R. \end{aligned} \tag{6}$$

4.1.3 *Energy Efficiency.* The efficiency, or duty cycle, is assessed by considering the different sources individually. LMAC requires performing a carrier sense (E_{cs}) in every slot but the owned one. In every frame, C neighbors are sending a (guarded) header that is overheard (E_{hdr}), after which the radio can be switched off in most cases; only the incoming traffic F_I for the node itself is received (E_{rx}). Energy is spent also by transmitting (E_{tx}) a header in every frame (which needs to be guarded for potential clock drift) and the payload if there is data to send.

$$\begin{aligned} E_{cs} &= (N_{slots} - 1) \cdot T_{cs}/T_{frame}, \\ E_{hdr} &= C \cdot (T_{guard}/2 + T_{hdr})/T_{frame}, \\ E_{rx} &= F_I \cdot P/R, \\ E_{tx} &= (T_{powerup} + T_{guard} + T_{hdr})/T_{frame} + F_{out} \cdot P/R, \\ E &= E_{cs} + E_{hdr} + E_{rx} + E_{tx}. \end{aligned} \tag{7}$$

4.1.4 *Parameter Constraints.* Every node has its own transmission slot, and hence the bottleneck is at the nodes having the most packets to send, that is, the bottleneck nodes are the ones next to the sink. In order to avoid queues, we set the bottleneck bandwidth to 50%, that is, having a packet in every second (owned) slot only:

$$F^1_{out} \cdot T_{frame} < 1/2. \tag{8}$$

4.2 SCP-MAC

The scheduled-channel-polling MAC [Ye et al. 2006] protocol combines low-power listening (LPL) with a global synchronized channel access, especially designed for low duty-cycle operation. All nodes in the network wake up at a regular interval T_w and perform a synchronized carrier sense. A sender node has to contend for the channel *before* the scheduled wakeup, so the receivers do not waste energy on listening to a complete contention window. The synchronized channel poll must be guarded to account for possible clock drift, which results in a prolonged wakeup preamble (tone).

To limit the length of the senders' contention window (T_{cw1}), SCP-MAC employs a second contention window after the wakeup time to handle the (increased) chance of multiple senders picking the same slot initially. This two-stage contention resolution policy allows for two short contention windows instead of a single long one. After this second contention window (T_{cw2}), the packet is sent and acknowledged.[2]

4.2.1 *Synchronization.* SCP-MAC requires that at least one message is sent every T_{sync} in order to keep the neighboring nodes synchronized, which results in a guard time of $T_{guard} = 4\theta T_{sync}$. If the data rate is too low, additional synchronization messages have to be sent:

$$F_{sync} = \begin{cases} 0, & \text{if } F_{out} > 1/T_{sync}, \\ 1/T_{sync}, & \text{otherwise.} \end{cases} \tag{9}$$

4.2.2 *Latency.* A message is generated somewhere during the wakeup interval before the first contention window, which results in an average delay of $T_w/2$ at the first hop. For every additional hop, the packet will be delayed for another T_w. At the last hop, the time required for the packet transfer sequence needs to be taken into account.

$$L(h) = T_w/2 + (h-1) \cdot T_w + T_{cw1} + T_{guard} + T_{cs} + T_{cw2}/2 + T_{msg}, \tag{10}$$

where $T_{msg} = T_{hdr} + P/R + T_{ack}$ is the time required for sending the packet header, the payload, and the acknowledgment.

4.2.3 *Energy Efficiency.* The energy consumption (duty cycle) of SCP-MAC is calculated by adding up its sources. The nodes are required to perform a (synchronized) carrier sense E_{cs} in every slot to check for a potential message. If a message is sent (E_{tx}), the node has to guard for the receiver's clock drift, has to perform a carrier sense in the second contention window, and has to transfer the message as illustrated in Figure 5. A receiving node (E_{rx}), on

[2]SCP-MAC offers several options, namely, an RTS/CTS handshake with acknowledgment, and an acknowledged and an unacknowledged data transfer. Considering the low data rate and the short payload size in sensor networks, the handshake is a big overhead. So we decided to use the acknowledged service.

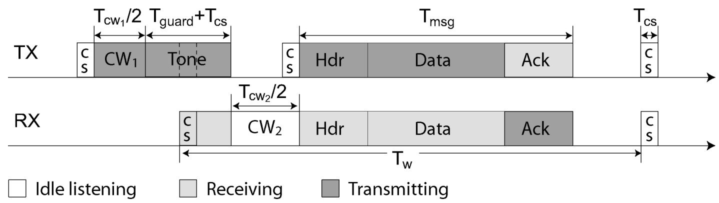


Fig. 5. The channel access policy of SCP-MAC .

the other hand, will listen for half the guard time on average, half of the second contention window, and the message. For the messages that are overheard (E_{ovr}), that is, sent to another node, the radio is switched off after receiving the header. The synchronization messages are handled the same way as data messages. However, only a header without an acknowledgment is broadcast (E_{stx}) and received (E_{srx}).

$$\begin{aligned}
E_{cs} &= T_{cs}/T_w, \\
E_{tx} &= F_{out} \cdot (T_{cw1}/2 + T_{guard} + T_{cs} + T_{msg}), \\
E_{rx} &= F_I \cdot (T_{guard}/2 + T_{cw2}/2 + T_{msg}), \\
E_{ovr} &= F_B \cdot (T_{guard}/2 + T_{cw2}/2 + T_{hdr}), \\
E_{stx} &= F_{sync} \cdot (T_{cw1}/2 + T_{guard} + T_{cs} + T_{hdr}), \\
E_{srx} &= C \cdot F_{sync} \cdot (T_{guard}/2 + T_{cw2}/2 + T_{hdr}), \\
E &= E_{cs} + E_{tx} + E_{rx} + E_{ovr} + E_{stx} + E_{srx}.
\end{aligned} \tag{11}$$

In contrast to LMAC , the length of the guard time can be influenced through a protocol parameter (T_{sync}). Sending a synchronization message entails overhead (i.e., adds E_{stx} and E_{srx}), but reduces the length of the guard interval (i.e., decreases E_{tx}, E_{rx}, and E_{ovr}). When the data rate of the application (F_s) is known, the optimum synchronization interval can be determined by taking the derivative of Equation (11) with respect to T_{sync}, and setting that to zero. In our evaluations, however, we simply tried a range of alternatives (see Table III) and selected the one yielding the best efficiency.

4.2.4 *Parameter Constraints.* The bottleneck is at the sink node, which has to receive all (data & sync) messages injected into the network. In order to avoid hidden terminal collisions at the sink, there should only be one message every fourth slot. Note that we used a tighter bound (1/4) on the maximum bandwidth than for LMAC (1/2), which rules out hidden terminals by design (i.e., ensures conflict-free transmissions in a 2-hop neighborhood). An additional constraint on the parameter settings is that a complete message sequence must fit into one slot.

$$\begin{aligned}
(F_I^0 + |I^0| \cdot F_{sync}) \cdot T_w &< 1/4, \\
T_{cw1} + T_{guard} + T_{cw2} + T_{msg} &< T_w.
\end{aligned} \tag{12}$$

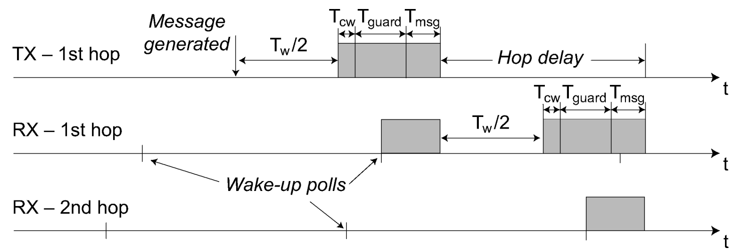


Fig. 6. WiseMAC latency estimation: the sending node has to wait for $T_w/2$ on average before starting to transmit the preamble and message.

4.3 WiseMAC

The Wireless Sensor MAC [El-Hoiydi and Decotignie 2004] is, like SCP-MAC , a refinement to the periodic carrier sense of LPL. With WiseMAC , the nodes wake up independently from each other at a periodic interval T_w. Instead of sending a long preamble, WiseMAC maintains a table with the neighboring nodes' poll schedule (updated individually with every packet that is sent), which allows sending a short wakeup preambles only. The nodes' clock drift is compensated for by dynamically adapting the length of the wakeup preamble according to the maximal possible clock drift since the last message exchange. If no information about a neighbor's poll schedule is available, WiseMAC falls back to LPL's long wakeup preamble. However, instead of just sending the long wakeup preamble, WiseMAC is sending consecutive data packets. Therefore, all neighboring nodes, except the intended receiver, will only receive a truncated first packet plus the header of a second one. The intended receiver must wait for the complete sequence before it can acknowledge the data.

4.3.1 *Synchronization.* WiseMAC updates the polling schedule of the neighbor with every received acknowledgment. In particular no special synchronization messages are required to be sent. Unlike SCP-MAC , the guard time that compensates for the clock drift is adapted dynamically: from the moment the last message exchange takes place, the guard time increases up to the LPL's long wakeup preamble.

$$T_{guard} = \max(4\theta/F_{out}, T_w). \tag{13}$$

4.3.2 *Latency.* WiseMAC determines the starting point of the preamble based on the estimated wakeup time of the receiving node, subtracting half the dynamically adapted guard time and the contention window.[3] On average, the sending node has to wait for $T_w/2$, before starting to transmit the wakeup preamble and message, as illustrated in Figure 6.

$$L(h) = h \cdot (T_w/2 + T_{cw} + T_{guard} + T_{msg}). \tag{14}$$

where $T_{msg} = T_{hdr} + P/R + T_{ack}$.

[3]The contention window, also referred to as the *medium reservation preamble*, is included to prevent multiple senders with the same guard time from transmitting at the same time.

4.3.3 *Energy Efficiency.* The energy consumption can best be approximated by analyzing its sources individually. There is the idle listening (E_{cs}), which requires switching on the radio every wakeup interval in order to perform a carrier sense. The energy consumption for sending a message (E_{tx}) depends on the length of the preamble, consisting of the contention window and the guard time, and the time to transfer the message and the acknowledgment. A receiving node will on average listen to the second half of the guard time and take part in the message transfer sequence. Not all messages transmitted by background nodes are overheard; only those in progress when a node polls the channel are observed. The probability of overhearing a message is thus related to the length of an actual message sequence in relation to the length of the wakeup interval: $p_{ovr} = (T_{cw}/2 + T_{guard} + T_{msg})/T_w$. Furthermore, WiseMAC sends consecutive data packets instead of a long preamble. So only a part of the first, and the header of a second data packet will be overheard.

$$\begin{aligned}
E_{cs} &= T_{cs}/T_w, \\
E_{tx} &= F_{out} \cdot (T_{cs} + T_{cw}/2 + T_{guard} + T_{msg}), \\
E_{rx} &= F_I \cdot (T_{guard}/2 + T_{msg}), \\
E_{ovr} &= \begin{cases} F_B \cdot p_{ovr} \cdot ((T_{hdr} + P/R)/2 + T_{hdr}), & \text{if } T_{cw}/2 + T_{guard} > T_{hdr} + P/R, \\ F_B \cdot p_{ovr} \cdot ((T_{cw}/2 + T_{guard})/2 + T_{hdr}), & \text{otherwise,} \end{cases} \\
E &= E_{cs} + E_{rx} + E_{tx} + E_{ovr} + E_{sync}.
\end{aligned} \tag{15}$$

4.3.4 *Parameter Constraints.* There is a bottleneck at the sink node. However, the randomly distributed wakeup slots of WiseMAC provide a natural way of increasing the available bandwidth, that is, the sink does not have to share its slot with neighboring nodes as with synchronized protocols. Therefore the traffic at the sink can be higher with WiseMAC than with SCP-MAC , that is, having a message in every second wakeup slot (16). A second constraint needs to ensure that the message sequence and the contention window fit into one slot (17).

$$(F_I^0 + |I^0| \cdot F_{sync}^1) \cdot T_w < 1/2, \tag{16}$$

$$T_{cw} + T_{msg} < T_w. \tag{17}$$

4.4 Crankshaft

The Crankshaft [Halkes and Langendoen 2007] protocol is a hybrid MAC that combines scheduled with contention-based access. Time is divided into frames consisting of N_b broadcast and N_u unicast slots. Every node is required to listen to *all* broadcast slots and to *one* of the unicast slots, which is assigned based on its MAC address modulo N_u. A node that needs to transmit a packet to a particular node has to wait for this node's unicast slot and content for it using the Sift [Jamieson et al. 2006] contention resolution scheme. The contention resolution is required since several nodes might want to send a packet in a particular slot (but not necessarily to the same node). Even though the data traffic is divided into several unicast slots and contention resolution is performed, collisions might still occur or data packets are not received due to other interference. Therefore, data packets are acknowledged. Furthermore, Crankshaft explicitly provides a special mode for the usually line-powered sink. Since energy consumption is of no concern, the sink listens into all unicast slots. A message to the sink can therefore be sent in any unicast slot, which increases the receiving bandwidth of the sink substantially. Note that the access control of Crankshaft basically reduces to that of SCP-MAC when operating with broadcast slots only ($N_u = 0$).

4.4.1 *Synchronization.* The Crankshaft protocol requires every node to send a synchronization message every T_{sync} in one of the broadcast slots to keep the network synchronized. It should be noted that the ordinary data transfer cannot be used to keep the network synchronized, as these messages are sent in unicast slots and, hence, not received by all neighboring nodes. In order to reduce the energy consumption for idle listening, the nodes do only perform a carrier sense in their slots leaving the transmitting node to guard for potential clock drift.

$$T_{guard} = 4\theta T_{sync}. \tag{18}$$

As with LMAC , the slot length T_{slot} limits the payload that can be transmitted. If the slot should fit a payload of L_{data}^{max}, the length of the slot results in

$$T_{slot} = T_{cw} + T_{guard} + T_{hdr} + L_{data}^{max}/R + T_{ack}, \tag{19}$$

and therefore the frame length in

$$T_{frame} = (N_b + N_u) \cdot T_{slot}. \tag{20}$$

4.4.2 *Latency.* The latency for Crankshaft is very similar in nature to that of LMAC . There is an initial delay $T_{init} = (T_{frame} + T_{slot})/2$, followed by waiting at each hop until the next forwarder's slot shows up taking $T_{hd} = T_{frame}/2$ on average. For the final transmission to the sink, the node has only to wait for the next unicast slots, that is, has to wait for $N_b/N_u + 1$ slots on average before the message can be sent.

$$\begin{aligned} L(h) &= T_{init} + (h-2) \cdot T_{hd} + (N_b/N_u + 1) \cdot T_{slot} - (L_{data}^{max} - L_{data})/R \\ &= (h-1) \cdot T_{frame}/2 + (N_b/N_u + 3/2) \cdot T_{slot} - (L_{data}^{max} - L_{data})/R. \end{aligned} \tag{21}$$

4.4.3 *Energy Efficiency.* Crankshaft requires performing a carrier sense (E_{cs}) in one unicast slot and all broadcast slots per frame. A node sending a message (E_{tx}) has to contend for the channel and guard the clock drift before starting the actual message transfer sequence. If a message is received (E_{rx}), the node will on average overhear half of the guard time only. Sending (E_{stx}) and receiving (E_{srx}) synchronization messages is very similar to ordinary data messages except that they only consist of a header. Data messages are only overheard if a neighboring node has the same unicast slot assigned. Since slots are assigned based on MAC addresses, which we assume to be randomly distributed, the number of nodes sharing the same slot follow a binomial distribution. That is, the probability that n out of $|B|$ nodes share a particular node's unicast slot is

$$P_r(X = n) \;=\; \binom{|B|}{n} p^n (1-p)^{|B|-n}, \tag{22}$$

where $p = 1/N_u$.

When calculating the number of neighbors that are overheard, we do not assume the worst-case of all $|B|$ neighbors having the same slot assigned. Instead we estimate the number of overheard neighbors N_{ovr} according to the paradigm $P_r(X \leq N_{ovr}) < 0.9$, that is, the number of slot collisions (N_{ovr}) is in 90% of the cases less than the expected value.

$$\begin{aligned}
E_{cs} &= (N_b + 1) \cdot T_{cs}/T_{frame}, \\
E_{rx} &= F_I \cdot (T_{guard}/2 + T_{msg}), \quad \text{where } T_{msg} = T_{hdr} + P/R + T_{ack}, \\
E_{ovr} &= N_{ovr} \cdot F_B/|B| \cdot (T_{guard}/2 + T_{hdr}), \\
E_{tx} &= F_{out} \cdot (T_{cs} + T_{cw}/2 + T_{guard} + T_{msg}), \\
E_{srx} &= C \cdot (T_{guard}/2 + T_{hdr})/T_{sync}, \\
E_{stx} &= (T_{cw}/2 + T_{guard} + T_{hdr})/T_{sync}, \\
E &= E_{cs} + E_{rx} + E_{ovr} + E_{tx} + E_{srx} + E_{stx}.
\end{aligned} \tag{23}$$

4.4.4 *Parameter Constraints.* Due to the special sink mode of Crankshaft , the bottleneck is only at the sink node if the number of incoming links exceeds the number of unicast slots (24). Otherwise, the bottleneck is at the nodes next to the sink (25). Then there must be enough broadcast slots for sending the synchronization messages (26). Similarly to LMAC , a message should only be received in every second slot.

$$F_I^0/N_u \cdot T_{frame} \;<\; 1/2, \tag{24}$$

$$(F_I^1 + N_{ovr}^1 \cdot F_B^1/|B^1|) \cdot T_{frame} \;<\; 1/2, \tag{25}$$

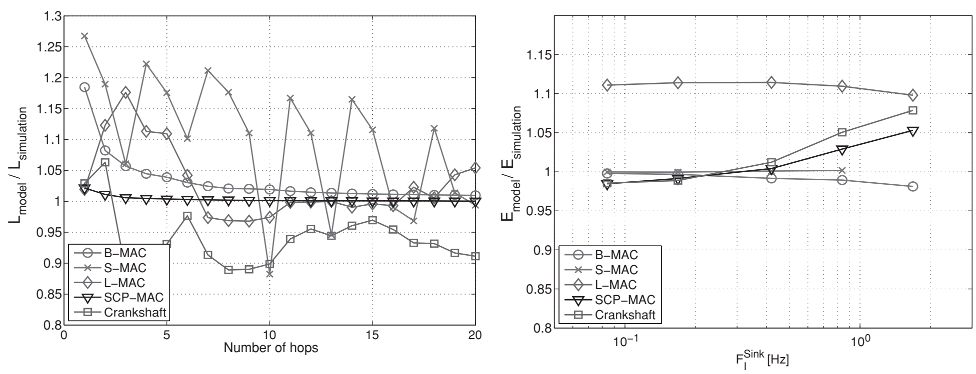
$$C/N_b \cdot F_{sync} \cdot T_{frame} \;<\; 1/2. \tag{26}$$

5. ANALYSIS

In the previous section we have modeled the main characteristics of four advanced energy-efficient MAC protocols (LMAC , SCP-MAC , WiseMAC , and Crankshaft); an additional five models of well-known, protocols (S-MAC , T-MAC , D-MAC , B-MAC , and X-MAC) are provided in online Appendix A in The ACM Digital Library. In this section we analyze the fundamental latency-efficiency tradeoff of the individual protocols, as well as how they compare to each other. First, however, we provide validation results demonstrating the soundness of the MAC models when compared to detailed, and time-consuming, simulations. We also detail the tuning process for obtaining the best performance of a MAC protocol given a set of external conditions and constraints.

5.1 Validation

The art of modeling is to “make everything as simple as possible, but not simpler” (Albert Einstein). Our MAC models are rather simple and abstract away



(a) Latency (b) Energy efficiency

Fig. 7. Comparing model and simulation results (CC1000 radio, ring topology, $C = 8$, $D = 4$).

many implementation details. The concern is then if the models are accurate enough to capture the essential performance characteristics. To answer this question we compared the latency and efficiency of those protocols (B-MAC , S-MAC , LMAC , SCP-MAC , and Crankshaft) that we had a detailed simulation code available for through prior work in the area of comparing WSN-specific MAC protocols [Halkes et al. 2005; Halkes and Langendoen 2007]. The underlying simulator, an OMNeT++-based discrete-event simulator, uses an SNR-based reception model to determine which packets are dropped due to contention and external interference. This reception model in combination with the MAC protocols was proven to be rather accurate; most simulation results are within 5% of actual delivery ratios and energy consumption numbers obtained on a 24-node testbed when the measured RSSI values are made available to the SNR-based channel model [Halkes and Langendoen 2009].

Figure 7 shows the ratio of the MAC performance models over the outcome of the corresponding simulations assuming no external interference. Each simulation result was the outcome over a series of 10 runs with a different random seed to average out the nondeterministic effects introduced by channel access policies, collisions, and the like. Figure 7(a) shows that the end-to-end latency as predicted by the models is usually within 10% of the value determined by simulation. The ratio for S-MAC is more erratic, and can be attributed to the model fixing (rounding down) the number of hops that can be made in one active period (cf. Equation 29); in reality the number of hops depends on the choice of waiting times in the contention windows introducing a certain amount of variability that is not accounted for.

Figure 7(b) shows that efficiency (duty cycle) as predicted by the models is generally also within a 10% margin when compared to simulation results, with very good accuracies for low data rates ($F_I^{Sink} < 0.1\,\text{Hz}$). LMAC is the exception, with the model always being too pessimistic by roughly 12% of the true efficiency. Unfortunately, we have not been able to identify the source of this discrepancy, nor can we simply correct for it as the overshoot depends on the network topology. Nevertheless with the majority of errors within 10% we feel that the MAC models strike a good balance between complexity and accuracy.

5.2 Protocol Optimization

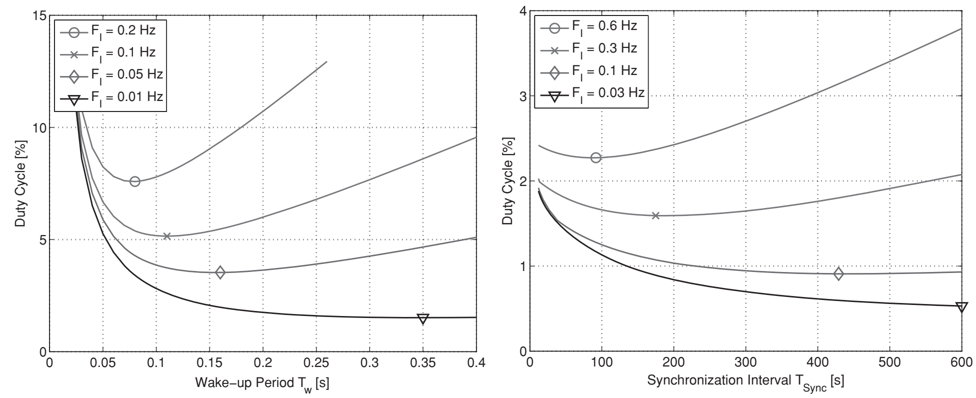
The exact behavior of the MAC models depends on the settings of some protocol-specific parameters as listed in Table III. For example, the performance of LMAC strongly depends on the number of slots in a frame (N_{slots}) and the length of an individual slot ($L_{hdr} + L_{data}^{max}$). The value ranges for these internal protocol parameters are derived (centered around) the settings provided in the original protocol descriptions and an earlier simulation-based comparison study [Halkes et al. 2005]. The notable exception is the setting of the synchronization interval (T_{sync}) of SCP-MAC , which is varied between 12 and 60 s, because the advocated range of 300–3600 s in Ye et al. [2006] yielded worse results in our evaluation. We attribute this large discrepancy to a different view on the effectiveness of synchronization messages.[4] Table IV provides implementation details regarding the length of the protocol headers, control messages, and contention window (if applicable) of each protocol.

Since the optimal settings of the internal protocol parameters depend on the external conditions (e.g.traffic rate, network density, radio characteristics), comparing MAC models is not completely straightforward. In the analysis presented in this section we adopt a simple, exhaustive method that, given a set of external conditions, iterates over all combinations of parameters considered for a given MAC protocol (cf. Table III). For these settings we compute the performance metrics of interest, for example, latency and energy efficiency, and then prune those settings that are inferior to so-called Pareto points [Deb 2001], which offer in this case either lower latency for the same energy efficiency, or higher energy efficiency for the same latency, or provide both lower latency and higher energy efficiency. The end result of this multiobjective optimization process is that we find those parameter setting that offer the best tradeoffs in the latency, energy efficiency and data-rate space under consideration.

As an example, consider the plots in Figure 8 that illustrate the optimization process for B-MAC and Crankshaft . B-MAC allows trading off a more frequent channel polling (controlled through T_w) for a shortened wakeup preamble. As shown in Figure 8(a), both a very short and a very long sampling period will result in a high duty cycle (i.e., low efficiency), but their causes differ. While a short polling period will increase the energy consumption for idle listening, a long one will increase the energy consumption for transceiving the wakeup preamble. This results in an optimal sampling time T_w, being dependent on the data load in the network. Furthermore, the polling period cannot be chosen arbitrarily large, as indicated by the topmost line ($F_I = 0.2\,\text{Hz}$) discontinued at $T_w = 260\,\text{ms}$; the polling period limits the maximum amount of traffic in the network as denoted in Equation (43).

Crankshaft requires sending special messages to keep the network synchronized. As illustrated in Figure 8(b), the interval T_{sync} of these messages can

[4]We argue that a node should receive a synchronization message from *all* neighbors within one period, while Ye et al. [2006] stated it is sufficient to receive one message from *any* neighbor; this relaxed constraint, however, is not enough for synchronizing nodes in sparse network topologies like linear chains. Furthermore this ensures fairness for the comparison, since SCP-MAC is now following the same synchronization policy as the other protocols that maintain a global structure.



(a) B-MAC (b) Crankshaft

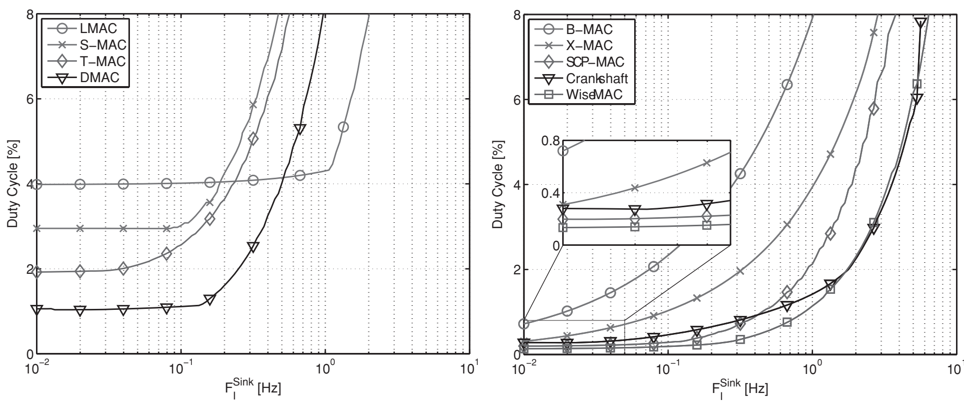
Fig. 8. Optimization of MAC parameters with respect to data load (F_I). The markers indicate the optimal operating points of the protocols for different data loads.

be optimized with respect to the data load. For high data rates, the synchronization interval should be chosen rather short, which results in a shortened guard time for all messages sent (see Equation (18)). For low data rates, on the other hand, it does not pay off to send synchronization messages at a high rate, since the potential savings for the shortened guard time are minimized. Crankshaft not only allows us to parameterize the synchronization interval, but also to adapt the number of unicast slots N_u being used. Hence, for minimizing the duty cycle for a given data load, both parameters N_u and T_{sync} need to be considered, which results in a two-dimensional parameter optimization.

5.3 Data Load Versus Energy Consumption

In a first experiment we studied the impact of the data load on the energy consumption of the nine MAC protocols that we modeled. The data load as it arrives at the sink is a function of the number of nodes in the network and the sampling rate. In this experiment we kept the network size (and topology) fixed and varied the rate F_S at which messages were injected into the network. For ease of understanding, though, we report the aggregate rate of the incoming traffic at the sink (F_I^{Sink}), which directly shows the load at the bottleneck in the network. As with the validation experiments, the network was structured as a set of four rings ($D = 4$) with a uniform density of eight neighbors per node ($C = 8$), resulting in a network size of 108 nodes. For the radio model, the popular CC1000 radio was used (see Section 3.3 for specifications), and we assumed perfect links (i.e., external interference is not taken into account). Unless overruled explicitly, these network and radio settings were also used in the other experiments discussed in the remainder of this study.

Figure 9 shows the individual Pareto fronts for the fundamental data-load versus energy-consumption tradeoff. It consists of two plots each having the (optimized) duty cycle on the vertical axis, and the increasing data load on



(a) Slot-based protocols (b) CP-based protocols

Fig. 9. Energy-load tradeoff (after Pareto optimization of MAC parameters).

the horizontal axis. The left plot features the "slot-based" protocols having the receiving nodes listen for a long period (S-MAC , T-MAC , and D-MAC) or into many slots (LMAC). The right plot, on the other hand, features the "CP-based" protocols, that is, the channel polling ones that periodically check for activity. Comparing the two plots clearly shows the advantage of the CP-based protocols where receiving nodes spend energy only in the case of ongoing activity; the difference is especially large for aggregate data rates below 1 message/10 s.

The slot-based protocols do all have a quite high offset for very low data rates, that is, a lot of energy is spent even when almost no data is communicated through the network. A consequence of this "hot" idle mode is that a certain traffic load can be accommodated for free, as indicated by the initial flatness of the curves. Once the data load crosses a certain threshold (around 10^{-1} Hz for S-MAC , T-MAC , and D-MAC ; around 10^{0} Hz for LMAC), default parameter settings need to be adjusted to handle, the increased traffic to the best of the protocol's capabilities. The reason that S-MAC is less efficient than T-MAC and D-MAC in idle mode is a direct outcome of the minimal active period which is the longest for S-MAC and the shortest for D-MAC . LMAC , on the other hand, spends a lot of energy in idle mode due to its large synchronization overhead required to maintain the TDMA structure. The advantage of this structure becomes apparent with higher data rates, showing a much better energy efficiency than the other slotted protocols.

The CP-based protocols consume significantly less energy in idle mode since the nodes only perform short carrier sensing and do not have to listen into long slots. One might anticipate that, for very low data traffic, SCP-MAC and Crankshaft perform worst in the class of CP-based protocols since they incur the overhead of maintaining a slotting structure. However, the results in Figure 9 show that this overhead already pays off compared to B-MAC and X-MAC , for very little traffic. This is due to the parameterization of the polling interval, which can be set to a very large value when a structure is maintained. For

B-MAC and X-MAC , on the other hand, a long polling interval also results in very long messages (preambles) when transmitting; hence the optimized polling interval is shorter for the unstructured CP-based protocol variants. WiseMAC exhibits the best energy efficiency for very low data rates. This can be attributed to its design of having the nodes synchronize on a per-link basis without the necessity of maintaining an expensive global structure. When comparing WiseMAC with B-MAC and X-MAC , a much longer polling interval can be chosen, since this does not imply an increased message length.

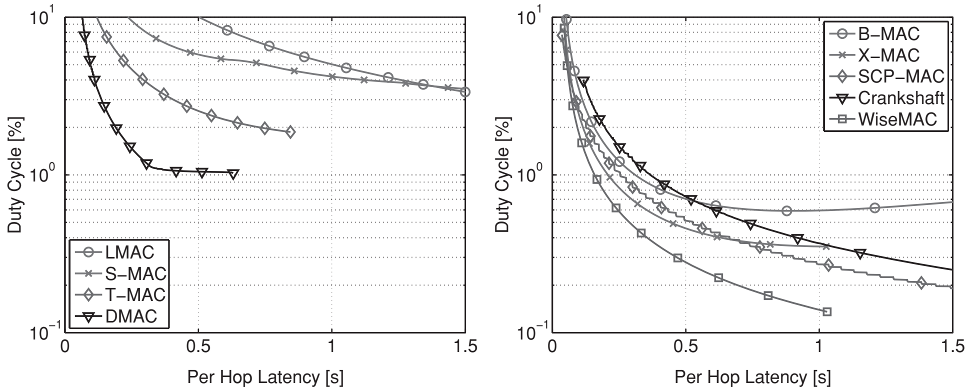
The energy consumption of SCP-MAC increases the fastest once the data rate at the sink exceeds 1 message/10 s. This is due to its global synchronization, grouping all communication activity into a single slots. This greatly reduces the available radio bandwidth and further results in frequent overhearing. This overhearing is especially expensive due to the two-contention-window scheme of SCP-MAC , which results in overhearing the second contention window for all nodes. It would therefore be more energy efficient to have a single contention window using the Sift [Jamieson et al. 2006] contention resolution scheme. WiseMAC and Crankshaft , on the other hand, use the complete channel bandwidth, which allows for an increased energy efficiency for higher data rates. For Crankshaft this is achieved by orchestrating to different slots, while WiseMAC inherently balances the channel activity by having random drifting channel-access times for the different nodes.

For the highest data rates, Crankshaft shows about the same energy efficiency as WiseMAC . This can be attributed to the special sink mode of Crankshaft that spreads the load evenly across all unicast slots instead of using just one slot. By itself this does not change the energy spent on sending and receiving, but the reduced pressure allows for a different setting of the internal protocol parameters of Crankshaft ; in particular, it may operate with fewer, larger slots per frame, reducing the energy spend on carrier sensing. This gives Crankshaft an advantage over WiseMAC , which must select a relatively short wakeup interval (T_w) to meet its boundary condition of handling at most one message every two wakeup slots of the sink (cf. Equation (17)). The impact of having a dedicated sink mode is further detailed in Section 5.6.

5.4 Energy Consumption Versus Latency

In the second experiment, we studied the trade off between energy consumption and latency. This is of particular importance for event-based applications such as burglar alarms that rarely exercise the sensor network, but do need a fast response. The low-latency requirement forces, for example, B-MAC to select a much shorter wakeup period T_w than is necessary for handling the near-zero data rate. Figure 10 shows the fundamental trade off between average per-hop latency and energy consumption (duty cycle) for a 6-hop event message injected into an idle network. In order to ensure that the topology is being maintained, we assume that every node sends a status message to its parent node every 10 min, checking for its availability.

The energy-latency trade off is related to the efficiency plot discussed in the previous section. In particular, the high offset in the energy consumption of the



(a) Slot-based protocols

(b) CP-based protocols

Fig. 10. Energy-latency trade off (after optimization of MAC parameters).

slot-based protocols can also be observed in Figure 10(a), limiting the minimal energy consumption to a duty cycle of 1% at best (D-MAC). The message latency depends a lot on the protocol design. Especially the TDMA structure of LMAC delays the message greatly due to the rather long frame time, whereas the staggered wakeup slots of D-MAC pay off well. However, it needs be considered that D-MAC is likely to result in a largely increased delay in the case of a link error, since there is no effective way of signaling the higher levels in the tree of a pending retransmission.

The CP-based protocols depicted in Figure 10(b) show the possibility for very low duty cycles if latency cutbacks are possible. WiseMAC stands out with its superior energy-latency trade-off. The reason is twofold: first, WiseMAC has been already shown to operate very energy efficiently for low data rates in the previous section. Second, due to the random access times of the nodes, the average waiting time for the parent to wake up is $T_w/2$. This is in contrast to SCP-MAC , which also operates very energy efficiently for very low data rates, yet delays the message by T_w at every hop. Overall, this results in SCP-MAC roughly having a doubled latency compared to WiseMAC . A similar trend holds for the frame structure of Crankshaft , which delays the message due to the long frame time (analog to LMAC). The delay of B-MAC and X-MAC is quite different, despite their similar design. This is attributed to the strobed preamble of X-MAC , which reduces the average preamble length and message delay by a factor of 2. Note that for both B-MAC and X-MAC the latency-efficiency curve levels off for high message delays. This is due to the regular status messages that are sent every 10 min. Furthermore, it can be observed that the curves for WiseMAC and X-MAC are limited to a maximum latency of about 1 s. The underlying cause is the polling period T_w having an upper bound of 2 s (see Table III).

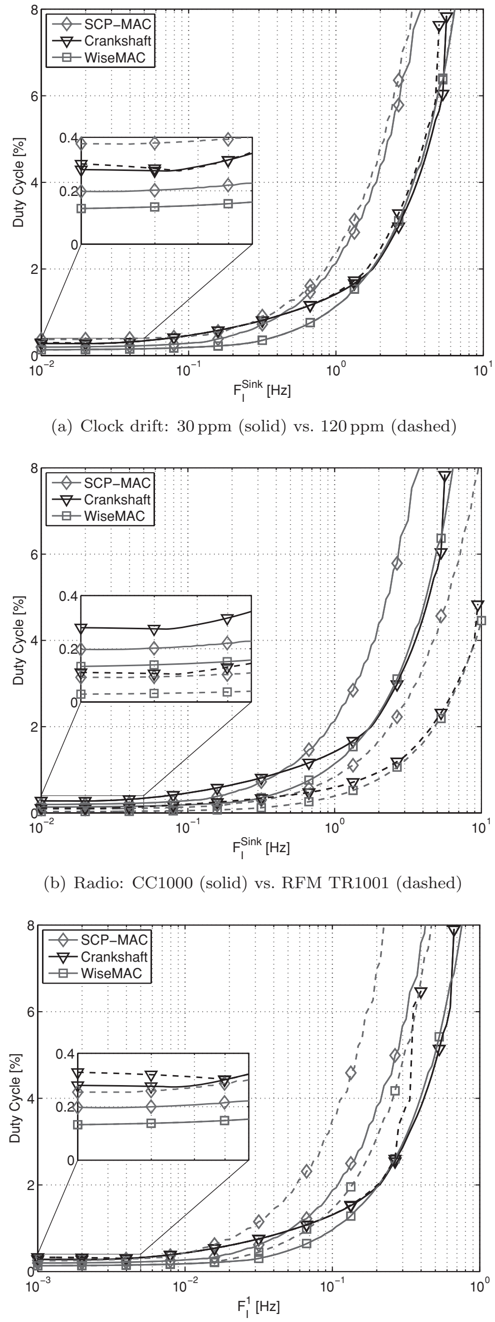
It should be noted that Figure 10 plots the *average* latency, while some real-time applications might want to consider the *worst-case* latency. For B-MAC and SCP-MAC these are very similar, but for WiseMAC , Crankshaft , and X-MAC the worst-case latency is in fact almost doubled. Depending on the application at hand this may, or may not, change the picture for selecting the most suitable protocol.

5.5 Sensitivity

The optimization presented before tunes the MAC-protocol parameters for the most energy-efficient operation in a specific setup, that is, for a specific set of network characteristics, radio parameters, etc. The following sensitivity analysis shows how the most energy-efficient protocols, namely, Crankshaft , SCP-MAC , and WiseMAC , are influenced by changes in the setup.

The three most energy-efficient protocols are all based on periodic channel polling combined with some form of synchronization; whereas WiseMAC synchronizes with each node individually, Crankshaft and SCP-MAC are based on globally synchronized slots. The three protocols have in common that they guard for potential clock drift, making it likely that the node's clock drift parameter θ impacts the energy efficiency. This effect is depicted in Figure 11(a), which shows the energy efficiency of the protocols for clock drift settings of 30 (default) and 120 ppm. (We did experiment with 60 ppm initially, but none of the protocols was significantly affected.) WiseMAC only shows one (rather thick) line, indicating its robustness against clock drift, which is a consequence of the dynamically adapted guard time of WiseMAC . SCP-MAC shows some sensitivity to clock drift. Especially for low data rates, the duty cycle is almost doubled (0.20% vs. 0.38%). This is attributed to the long synchronization interval in combination with a quadrupled guard time, resulting in long guard times overheard by all nodes in the network. For higher data rates, the guard time is not the dominating factor anymore, since the network is tightly synchronized due to the frequent traffic. Crankshaft is less affected by the clock drift than SCP-MAC . This is rather surprising, since both protocols require global synchronization. Crankshaft minimizes overhearing of the data traffic with its slot assignment, which explains the smaller offset for higher data rates. For low data rates, there is almost no difference for the different clock drifts. Detailed inspection of the protocol parameters settings for this low data traffic showed that the prolonged guard time results in an increased frame time (cf. Equation (20)) (3.5 s vs. 10.9 s). This is still sufficiently short to accommodate all traffic in the network and allows compensating for the prolonged guard time by fewer channel polls.

The analysis so far is based on the CC1000 radio transceiver, featuring multichannel operation but having limitations in the available bandwidth and rather long switching times. Figure 11(b) depicts the impact of using the fast RFM TR1001 radio transceiver; the solid curves plot the default (CC1000) performance, whereas the dashed, more efficient curves derive from the faster (TR1001) radio. Using a faster radio (0.5 ms vs. 2.10 ms "switch on" time and 5.75 kb/s vs. 2.4 kb/s bandwidth) impacts the energy demands of the protocols with improved efficiency for all data rates. Overall, the faster switching time is most beneficial for low data rates where most energy is spent on polling the channel, or rather on turning the radio (CC1000) on *before* probing the



(a) Clock drift: 30 ppm (solid) vs. 120 ppm (dashed)

(b) Radio: CC1000 (solid) vs. RFM TR1001 (dashed)

(c) Network density: 8 (solid) vs. 16 (dashed) neighbors

Fig. 11. Sensitivity analysis.

channel. The faster transmission rate is most beneficial for higher traffic rates where overhearing of headers and (partial) messages has a larger impact on the overall energy consumption.

The network model assumes a constant node density in the network. However, it is likely that a real deployment shows areas with an increased node density. This would increase the number of neighbors and therefore the number of messages that can be potentially overheard. Figure 11(c) shows this effect, featuring the energy consumption of the protocols when doubling the number of nodes—hence with 8 (solid lines) and 16 (dashed lines) neighbors. Note that the x-axis plots the input frequency of the bottleneck nodes next to the sink (F_I^1) to ensure that the same amount of traffic is handled in both cases; when focusing on the aggregated input rate at the sink (F_I^{Sink}) as before, the send rate would have to be halved, making for an unfair comparison (i.e.,protocols becoming seemingly more efficient for higher densities due to handling proportionally less traffic). All protocols show increased energy consumption in the denser network; however, the amount varies. SCP-MAC shows a large increase in energy consumption since all messages (i.e.,preambles and headers) are overheard. Crankshaft , especially designed for high-density networks, is hardly affected (up to around 0.3 Hz) due to its unicast slots, minimizing the probability of overhearing messages. For very low data rates, however, the duty cycle of Crankshaft is slightly increased. This is attributed to the increased number of synchronization messages overheard due to the increased number of neighbors. WiseMAC is not based on a global schedule and is synchronized with specific neighbors (i.e., parent and children) on an individual basis, resulting in shorter guard preambles compared to Crankshaft or SCP-MAC . This explains WiseMAC being only affected for higher data rates, as the probability of overhearing increases with higher node density and data rate.

5.6 Bottleneck Sink

In the typical data-gathering scenario studied in this article, the sink becomes the bottleneck because it has to handle the most traffic. Therefore, any optimization in the access mode of the sink (and its immediate neighbors) is likely to have a large impact. Crankshaft includes an optimization in which the sink is always listening in contrast to ordinary nodes following an active/sleep duty cycle (at slot level). This increases the effective bandwidth to the sink, allowing Crankshaft to operate more efficiently, and causing it to challenge the energy efficiency of WiseMAC for high data rates (see Section 5.3).

The idea of employing such a special (always on) sink mode for other MAC protocols as well is rather attractive, but not always straightforward to implement or may even not be applicable at all. For instance, the concept of LMAC of scheduling send slots already gets *all* nodes to listen in on every slot, ruling out additional listening by the sink.[5] In the case of slotted protocols (S-MAC,

[5]A related optimization to LMAC [Chatterjea et al. 2004] is to allocate more slots to the immediate neighbors of the sink, which increases the effective bandwidth to the sink, but does not reduce the overhearing overhead as Crankshaft does.

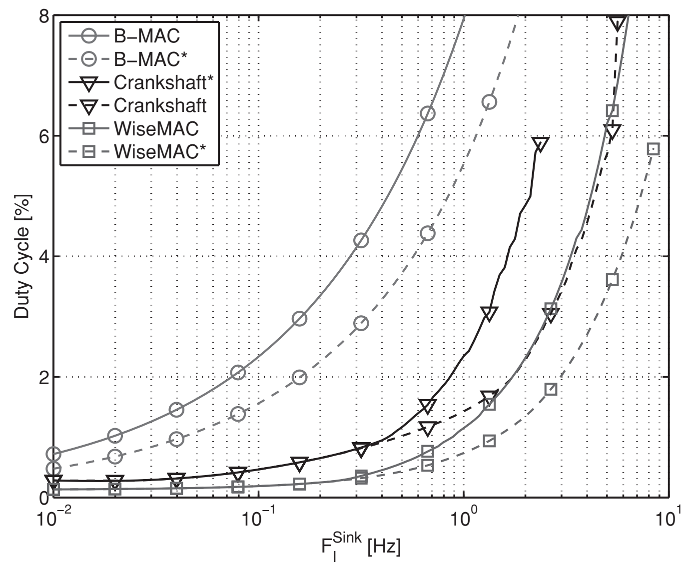


Fig. 12. Benefit of having the sink keeping its radio switched on (dashed line) over ordinary duty cycling (solid line). The variants marked with an asterisk represent the adapted protocols.

T-MAC, D-MAC, and SCP-MAC), it would be possible to have nodes send messages to the sink “outside" the normal active period in a slot when the sink is always listening, but at a considerable increase in complexity (additional timers and bookkeeping), rendering it less attractive. For the class of random-access protocols (B-MAC and WiseMAC), however, only a minor modification is required to take advantage of sink that is always listening. As suggested in Polastre et al. [2004], the nodes next to the sink are no longer required to send a stretched wakeup preamble to meet a specific channel poll. This minimal optimization greatly reduces the transmission energy for nodes next to the sink and also reduces the channel load at the sink, reducing overhearing overheads.

To determine the impact of a special sink mode on various protocols, we have adapted the models for Crankshaft , B-MAC , and WiseMAC (see online Appendix B in the ACM Digital Library). We did not adapt any of the slotted protocols, because they would require a major protocol redesign and have not proven to be very energy efficient to start with. Figure 12 shows the gain in energy efficiency that results from optimizing the protocols to keep the sink listening at all times. All three protocols benefit from a special sink mode, but Crankshaft benefits the most. Its performance without the optimization rapidly deteriorates when the data rate increases, and the resulting bottleneck of all data passing through one slot causes it to violate the basic parameter constraints for data arriving faster than two messages per second.

WiseMAC and B-MAC follow the same channel access strategy to benefit from the increased resources of the sink. Nevertheless their impact differs largely: B-MAC benefits a lot, rather independently of the data rate, whereas

WiseMAC only shows a substantial gain for higher data rates. This can be explained by the very energy-efficient operation of WiseMAC for low data rates, spending most energy on regular channel polls, which makes the energy saved on transmitting messages of little importance. For higher data rates, on the other hand, the energy consumption for transmitting and overhearing messages is a nonneglectable factor, allowing WiseMAC to benefit from the shortened preamble. In particular, WiseMAC even outperforms Crankshaft for higher data rates, making this variant the overall most energy-efficient protocol.

5.7 Broadcast Versus Unicast

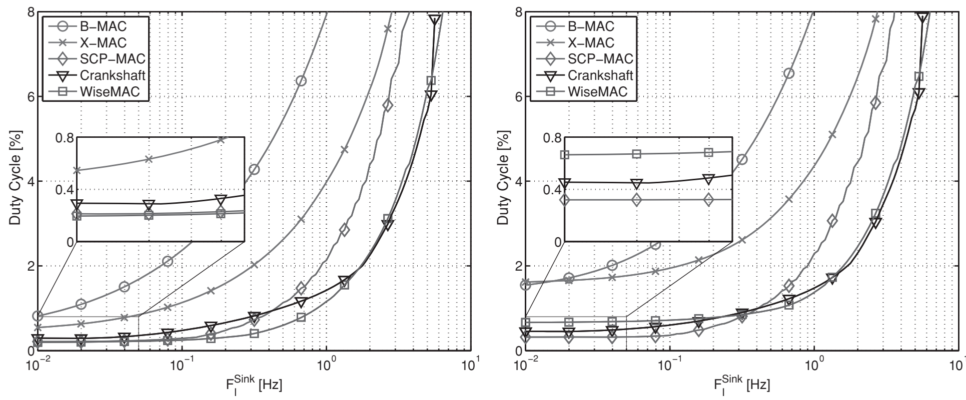
The common traffic pattern for low-power data gathering is unicast. Nevertheless, broadcasts are sent at times, that is, for announcing a change in the topology. In the following, we analyze how such broadcasts will impact the contention-based MAC protocols.

The impact of the energy consumption differs greatly for broadcast messages, as certain protocols are better suited for sending them. Crankshaft with its special broadcast slots and SCP-MAC with its single slot for all communication are very well suited to broadcast traffic. For B-MAC , the difference between a unicast and a broadcast is negligible, since the long wakeup preamble is waking up all neighbors anyhow. X-MAC and in particular WiseMAC highlight a reduced wakeup preamble for unicast messages, which has to be extended to the long wakeup preamble of B-MAC for broadcast traffic. For receiving broadcast messages, X-MAC and WiseMAC differ greatly: WiseMAC sends a packet stream which allows the receiver to switch off the radio after receiving one complete packet out of the packet stream. With X-MAC , on the other hand, the receiving node has to stay awake until the end of the wakeup strobe, waiting for the packet to be sent.

In order to analyze the impact of sending additional broadcast messages to the regular unicast traffic, the models were adapted according to the discussion above. If the broadcasts are limited to one per hour, per node, as shown in Figure 13(a), not much cost is added compared to the solely unicast case as depicted in Figure 9(b). If, however, the broadcast frequency is increased to once every 5 min, as shown in Figure 13(b), a clear impact is observed for low data rates. Especially WiseMAC and even more X-MAC are significantly affected by the additional broadcasts. On the other hand, SCP-MAC and Crankshaft are only moderately influenced. Especially Crankshaft is very well suited for combined unicast and broadcast traffic, due to its dedicated unicast and broadcast slots.

6. PACKET-BASED VERSUS BYTE-STREAM RADIOS

Most of the discussed protocols are designed for byte-stream radios like the CC1000 and the RFM TR1001 analyzed in the last section. However, state-of-the-art platforms tend toward packet-based radios such as the IEEE 802.15.4 compliant CC2420 used for the MicaZ and the TelosB node. With such a radio, the packet is first copied into a dedicated radio buffer. After receiving a



(a) One broadcast every hour

(b) One broadcast every 5 min

Fig. 13. Additionally to the unicast traffic (cf. Figure 9(b)), every node sends broadcasts.

trigger, the radio sends the packet autonomously, in particular also adding the preamble and CRC checksum. For the CC2420 the length of this preamble can be set between 4 and 16 bytes, which is generally too short to accommodate low-level MAC synchronization techniques like, for instance, the long wakeup preamble of B-MAC and SCP-MAC 's guard preamble and contention resolution "tone."

As shown in Ye et al. [2006], a long preamble can be replaced by a stream of packets. The granularity of such an artificial preamble depends on the minimal packet size and the data rate (16 bytes and 250 kb/s, respectively for the CC2420), which results in a minimal packet transmission time of 0.51 ms. The gap between two consecutive packets can be reduced to 30μs. It is therefore possible to imitate an arbitrarily long preamble with a granularity of 0.54 ms. This allows adopting the protocols discussed in this study to packet-based radios with minimal loss compared to a byte-stream radio. In exchange, the MAC implementation does not have to burden the microcontroller with transceiving the byte stream, checking for a start-frame delimiter, and performing a CRC check.

Adapting the models for packet-based radios is straightforward. For instance, if the preamble exhibits a certain granularity, the model has to account for this as shown with the packet-based X-MAC model (cf. Equation (45)). For the popular CC2420 radio, this makes only a small difference, due to the fine granularity (0.54 ms) that can be achieved. We therefore analyzed the energy versus data-rate tradeoff as well for the CC2420 radio, using the existing models. Since the general trend is very similar to those for the CC1000 and TR1001 radios, we do not show the plots but briefly discuss the results for the CP-based protocols.

For very low data rates, the energy consumption of the MAC protocols for the CC2420 radio is very similar to that of the CC1000 (see Figure 9(b) for $F_I^{Sink} < 0.1$). This can be attributed to the fact that both radios have similar switching times (see Table II) and the fact that the regular channel polls are the main source of radio activity. For higher data rates, it is mainly X-MAC , WiseMAC , and Crankshaft that benefit from the increased bandwidth, which allows them to minimize the energy consumption in the same order as the faster TR1001 radio does (see Figure 11(b) for $F_I^{Sink} > 1$).

7. CONCLUSIONS

The fundamental need for energy-efficient operation has been a driving force behind the development of many WSN-specific medium access control protocols. Each protocol strikes a different balance between performance (latency, throughput) and energy consumption; all claiming to be more efficient than the canonical S-MAC protocol, but evaluated with different workloads, simulation environments, and hardware platforms, making it hard to assess the true benefits of the individual MAC protocols.

In this article, we have taken an analytical approach to answering the question "which protocol is best?" given a set of external conditions including radio hardware characteristics, network topology, and workload. Our study focuses on low data-rate applications for which the energy-efficient operation of the MAC protocols is most critical as there is little room for amortizing overheads. To keep the analysis tractable, we did not model MAC-level retransmissions, but simply simply included specific boundary conditions safeguarding the contention-free operation of each protocol.

For each of the nine MAC protocols considered (B-MAC , Crankshaft , D-MAC , LMAC , S-MAC , SCP-MAC , T-MAC , WiseMAC , and X-MAC), we have modeled their latency and energy efficiency as a function of external parameters (radio hardware, network topology, etc.) as well as key, internal parameters (duty cycle, slot length, number of slots, etc.). Despite their simplicity, the models' performance predictions match well with detailed simulation results with typical validation experiments.

An extensive exploration of the MAC protocols (iterating over different data rates, clock drifts, network densities, and radio characteristics) has revealed a number of important protocol features that, collectively, warrant a very efficient handling of low data-rate applications. First, the timing uncertainty introduced by clock drifts is best handled (guarded) at the sending node as that avoids any overhead at the receiver and other nodes overhearing the message (one vs. many). Second, (randomized) channel polling reduces idle listening to a great extent when compared to more structured approaches organizing time into slots (and frames). Third, taking advantage of the line-powered sink node by keeping it on at all times allows 1-hop neighbors to short-cut sending procedures and alleviates the bandwidth bottleneck in the network; usually the reduced pressure on the (bottleneck) nodes surrounding the sink allows for a more efficient setting of the internal protocol parameters, reducing overall energy consumption. Finally, protocols like T-MAC and SCP-MAC that cluster communication (at the beginning of a slot) suffer from overhearing overheads when compared to protocols like B-MAC , Crankshaft , and WiseMAC that spread (randomize) traffic over time; an added disadvantage is that protocols must operate with conservative settings to avoid contention.

Although announcing an absolute winner is impossible, we did observe that the WiseMAC protocol showed a remarkable consistent behavior across a wide range of operational conditions, always achieving the best, or second-best performance. The true value of the work presented in this article, however, lies in the analytical models, which are made available to the research community. These models do not only allow individuals to select the MAC that favors their specific requirements, but also allow MAC-protocol designers to quickly check the impact of design choices and allow a fair and easy comparison with existing protocols.

ACKNOWLEDGMENTS

We thank Gertjan Halkes, Matthias Woehrle, and the anonymous reviewers for proofreading draft versions of this article.

REFERENCES

BENNETT, F., CLARKE, D., EVANS, J., HOPPER, A., JONES, A., AND LEASK, D. 1997. Piconet: Embedded mobile networking. *IEEE Pers. Comm. 4*, 5, 8–15.

BUETTNER, M., YEE, G., ANDERSON, E., AND HAN, R. 2006. X-MAC: A short preamble MAC protocol for duty-cycled wireless networks. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*. 307–320.

CHATTERJEA, S., VAN HOESEL, L., AND HAVINGA, P. 2004. AI-LMAC: An adaptive, information-centric and lightweight MAC protocol for wireless sensor networks. In *Proceedings of the 2nd International Conference on Intelligent Sensors, Sensor Networks and Information Processing*.

VAN DAM, T. AND LANGENDOEN, K. 2003. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys)*. 171–180.

DEB, K. 2001. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, U.K.

EL-HOIYDI, A. 2002. Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks. In *Proceedings of the 1st IEEE International Conference on Communications (ICC)*.

EL-HOIYDI, A. AND DECOTIGNIE, J.-D. 2004. WiseMAC: An ultra low power MAC protocol for multihop wireless sensor networks. In *Proceedings of the 1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*. Lecture Notes in Computer Science, vol. 3121. Springer-Verlag, Berlin, Germany, 18–31.

GOENSE, D., THELEN, J., AND LANGENDOEN, K. 2005. Wireless sensor networks for precise Phytophthora decision support. In *Proceedings of the 5th European Conference on Precision Agriculture* (5ECPA).

HALKES, G., DAM, T. V., AND LANGENDOEN, K. 2005. Comparing energy-saving MAC protocols for wireless sensor networks. *Mobile Netw. Appl. 10*, 5, 783–791.

HALKES, G. AND LANGENDOEN, K. 2007. Crankshaft: An energy-efficient MAC-protocol for dense wireless sensor networks. In *Proceedings of the 4th European Conference on Wireless Sensor Networks (EWSN)*. 228–244.

HALKES, G. AND LANGENDOEN, K. 2009. Experimental evaluation of simulation abstractions for wireless sensor network mac protocols. In *Proceedings of the 14th IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*.

HILL, J. AND CULLER, D. 2002. Mica: a wireless platform for deeply embedded networks. *IEEE Micro 22*, 6, 12–24.

VAN HOESEL, L. AND HAVINGA, P. 2004. A lightweight medium access protocol (LMAC) for wireless sensor networks. In *Proceedings of the 1st International Workshop on Networked Sensing Systems (INSS)*.

JAMIESON, K., BALAKRISHNAN, H., AND TAY, Y. 2006. Sift: A MAC protocol for event-driven wireless sensor networks. In *Proceedings of the 3rd European Workshop on Wireless Sensor Networks (EWSN)*. 260–275.

LU, G., KRISHNAMACHARI, B., AND RAGHAVENDRA, C. 2004. An adaptive energy-efficient and low-latency MAC for data gathering in sensor networks. In *Proceedings of the International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*.

MAINWARING, A., POLASTRE, J., SZEWCZYK, R., CULLER, D., AND ANDERSON, J. 2002. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Application (WSNA)*. 88–97.

MILLER, M. AND VAIDYA, N. 2004. Minimizing energy consumption in sensor networks using a wakeup radio. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*.

POLASTRE, J., HILL, J., AND CULLER, D. 2004. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*. 95–107.

RAJENDRAN, V., OBRACZKA, K., AND GARCIA-LUNA-ACEVES, J. 2003. Energy-efficient, collision-free medium access control for wireless sensor networks. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys)*. 181–192.

RHEE, I., WARRIER, A., AIA, M., AND MIN, J. 2005. Z-MAC: A hybrid MAC for wireless sensor networks. In *Proceedings of the 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys)*. 90–101.

SCHURGERS, C., TSIATSIS, V., GANERIWAL, S., AND SRIVASTAVA, M. 2002. Optimizing sensor networks in the energy-latency-density design space. *IEEE Trans. Mobile Comput. 1*, 1 (Jan.), 70–80.

SUN, Y., GUREWITZ, O., AND JOHNSON, D. 2008. Ri-mac: A receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys)*. 1–14.

TOLLE, G., POLASTRE, J., SZEWCZYK, R., CULLER, D., TURNER, N., TU, K., BURGESS, S., DAWSON, T., BUONADONNA, P., GAY, D., AND HONG, W. 2005. A Macroscope in the Redwoods. In *Proceedings of the 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys)*. 51–63.

YE, W., HEIDEMANN, J., AND ESTRIN, D. 2002. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 21st Conference of the IEEE Computer and Communications Societies (INFOCOM)*. Vol. 3. 1567–1576.

YE, W., SILVA, F., AND HEIDEMANN, J. 2006. Ultra-low duty cycle MAC with scheduled channel polling. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*. 321–334.

ZHENG, T., RADHAKRISHNAN, S., AND SARANGAN, V. 2005. PMAC: An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 65–72.

ZHONG, L. C., RABAEY, J. M., AND WOLISZ, A. 2004. An integrated data-link energy model for wireless sensor networks. In *Proceedings of the IEEE International Conference on Communications*. Vol. 7. 3777–3783.

Received July 2007; revised August 2009; accepted March 2010