Preprint of a book chapter in "Medium Access Control in Wireless Networks, Volume II: Practice and Standards" edited by H.Wu and Y. Pan, to be published by *Nova Science Publishers* in 2007

MEDIUM ACCESS CONTROL IN WIRELESS SENSOR NETWORKS

KOEN LANGENDOEN*

Abstract. This chapter provides a broad overview of the MAC protocols especially developed for sensor networks. These MAC protocols differ from typical WLAN access protocols in that they trade off performance (latency and throughput) for a reduction in energy consumption to maximize the lifetime of the network. This is in general achieved by duty cycling the radio, and it is the MAC layer that controls when the radio is switched on and off. An important consequence is that a MAC protocol needs to be aware of its neighbors' sleep/active schedules, since sending a message is only effective when the destination node is awake. An obvious solution is to have all nodes synchronize on one global schedule, so no separate neighbor state is required, which maps well onto the resource limitations of typical sensor nodes. However, grouping communication into small (active) periods increases the chance on collisions, hence, other forms of organization have been proposed. This chapter surveys, and details the historic development of, the three most common styles of medium access control for wireless sensor networks: random, slotted, and frame-based organization.

1. Introduction. The research area of Wireless Sensor Networks, WSNs for short, is driven by the ongoing advances in digital circuitry leading to ever smaller computing systems. In their visionary "Smart Dust" paper [16], Pister et al. proposed to capitalize on this trend by combining sensors, a micro controller, and a radio into a tiny sensor node. Multiple nodes could then self-organize into a wireless network and collectively report information about their environment opening a whole new range of applications. Examples include unobtrusive habitat monitoring of wildlife, ad-hoc deployments for disaster management, precision agriculture, and tracking of goods and objects, to name a few.

Although diverse in nature, the proposed application scenarios share a number of characteristics and constraints that define the research area of WSNs. First, nodes must operate for a number of years to make applications economically viable. This puts severe constraints on energy consumption, because nodes are usually battery powered and changing batteries is not an option. Second, also from a cost perspective, sensor networks must function autonomously without (much) external control. Third, the network must be resilient to errors of all kinds; nodes may die when running out of energy; radio communication may be distorted by external interference; and low-cost sensors may malfunction and produce erroneous readings. Finally, data – generated periodically or sparked by an external event – has to be relayed to a *sink* (gateway) node for further processing and for generating an appropriate response, respectively.

The need for *energy-efficient* operation of a wireless network of resource-scarce devices has prompted the development of novel protocols in all layers of the communication stack. Given that the radio is the most power-consuming component of a typical sensor node, large gains can be achieved at the link layer where the MAC protocol is controlling the usage of the radio. Therefore, a whole range of energy-efficient MAC protocols have been developed taking into consideration, and advantage of, the application characteristics outlined above. These WSN-specific MAC protocols typically trade off classical performance parameters (throughput, latency, and fairness) for a reduction in energy consumption to maximize the lifetime of the network. Each MAC protocol has its own policy for switching off the radio leading to a different trade-off. Basic protocols implement a fixed duty cycle, while others adapt to changes in traffic over time and place; whether or not the additional reduction in energy consumption outweighs the increase in complexity depends on the particular application

* Delft University of Technology, The Netherlands (K.G.Langendoen@tudelft.nl).

and channel conditions at hand. In this chapter we will classify the major trends in MAC design for energy efficiency, and detail the historic advances within each class. We do not provide a thorough performance analysis, but include enough hints for end users to select an appropriate MAC for their (next) WSN deployment.

This chapter is structured as follows. First, we provide background information regarding the resource limitations and deployment scenarios of wireless sensor networks (Section 2). Next, we outline the fundamental issue of how to (not) organize nodes in a network to address idle listening, the major source of energy consumption in WSNs (Section 3). Then, we will discuss various protocols from each class of organization: Random access (Section 4), Slotted access (Section 5), and Frame-based access (Section 6). Finally, we will outline current trends (Section 7) and draw conclusions (Section 8).

2. WSN characteristics. Wireless sensor networks are in many aspects quite similar to Mobile Ad Hoc Networks (Chapter 14) and Wireless Mesh Networks (Chapter 15), but two distinct characteristics call for a different approach. First, the need for energy-efficient operation severely constrains the capabilities of individual sensor nodes; processing, memory, and communication are limited resources, much more so than in mobile devices like laptops and PDAs. Second, WSN deployment scenarios highly structure the communication between nodes in the network; in particular, communication between two arbitrary nodes in the network, being part of many ad hoc and mesh scenarios, does not occur in WSNs where most information is relayed either between neighbors or to/from the sink. We will now discuss both defining characteristics – resource limitations and communication patterns – in more detail.

2.1. Resource limitations. The standard disclaimer "your mileage may vary" certainly applies to the device capabilities of different sensor node platforms developed over the last 10 years. Table 2.1 gives the key performance parameters of four sensor nodes developed out of commodity components. (The numbers are taken from the data sheets and need to be taken with a grain of salt as manufacturers tend to be optimistic and differ in their terminology and interpretation [3].) Clearly, there is an increasing trend in raw performance; CPUs become faster and provide more memory, and radios transmit at higher bit rates. Interestingly enough, the power consumption of the complete system (CPU+radio) stays more or less constant around 100 mW. The net effect is that as technology progresses over time you can get much more work done running from the same set of batteries. The effective lifetime of a sensor node, however, heavily depends on how much time it spends in sleep state (with the CPU and radio powered off); running a sensor node flat out will drain a pair of AA batteries (3000 mAh) in about 100 hours, or just 4 days. This demonstrates that power management is a must, especially at the MAC layer since the radio uses up a large fraction of the total energy consumed by a sensor node.

A recent trend in the sensor network community is to move towards a two-tiered architecture with clusters of resource-limited sensor nodes, like the Mica-2 and Tmote Sky platforms, being serviced by a backbone network of more powerful *and* more costly nodes, like the Imote2. In this chapter we do not consider tiered architectures like Tenet [9], but focus instead on flat systems with one type of nodes, which collectively form a multi-hop network. As such we will ignore the "luxury" Imote2 and assume that typical sensor nodes have rather limited resources due to being composed out of cheap, low power components. The resources of interest are:

- **CPU:** 8-bit processors are common, but 16-bit ones are gaining popularity, with clock rates in the range of 1-10 MHz. Experience has shown that this is

TABLE 2.1
Performance specifications of sensor network platforms

	René 1999	**Mica-2 2002**	**Tmote Sky 2005**	**Imote2 2007**
CPU	**ATMEL 8535**	**ATmega128L**	**TI MSP430**	**Intel PXA271**
	8-bit, 4 MHz	8-bit, 8 MHz	16-bit, 8 MHz	32-bit, 13-416 MHz
	36 μW sleep	36 μW sleep	15 μW sleep	390 μW sleep
	60 mW active	60 mW active	5.4 mW active	$\geq$ 31 mW active
Memory	512 B RAM	4 KB RAM	10 KB RAM	32 MB RAM
	8 KB Flash	128 KB Flash	48 KB Flash	32 MB Flash
Radio	**RFM TR1000**	**CC1000**	**CC2420**	
	10 Kbps	76 Kbps	250 Kbps	
	2 μW sleep	100 μW sleep	60 μW sleep	
	12 mW receive	36 mW receive	63 mW receive	
	36 mW xmit	75 mW xmit	57 mW xmit	
	0.5 ms setup	2 ms setup	1 ms setup	

enough to run a MAC protocol driving a simple, byte-level radio and leaves some room for application processing, but not much more. With packet-level radios like the CC2420 running more demanding applications is certainly possible.

- **Memory:** the amount of Flash memory available for storing the program code is generally enough, although the MSP430's 48 KB is a bit tight. The real problem is the amount of RAM available for program data; the 4 KB (ATmega128L) or 10 KB (MSP430) forces developers to minimize the memory footprint of their software.
- **Radio:** compared to today's WLAN standards (55 Mbps and up), the bandwidth provided by sensor node radios (10-250 Kbps) is very low indeed. However, most WSN applications need only a fraction of that as will be detailed below, so in fact, bandwidth is not an issue. What does matter is the rather poor performance in terms of range (10s of meters) and link quality. This is caused by simple modulation schemes being sensitive to noise and poor (integrated) antennas showing irregular reception patterns. Another important factor for MAC design is the setup time needed to switch the radio from sleep into receive/transmit mode. This time is largely spent on waiting for oscillator circuits to stabilize, effectively consuming precious energy while doing nothing. As a consequence, switching a radio into sleep mode only saves energy when doing so for a long time. This, in turn, may delay sensor data from being injected into the network, effectively increasing the end-to-end latency.

The resource limitations imposed by the need to consume little energy constrain the type of applications that can be supported. This is reflected in the network traffic generated by typical WSN applications, which is limited to a few, simple communication patterns detailed in the next section.

2.2. Communication patterns. A common characteristic of many WSN application scenarios is that sensor nodes are deployed to just monitor the environment and relay (preprocessed) data to a sink node for further processing. In particular, when sensor nodes detect a significant effect, they are not expected to respond themselves. The prime reason being that often some physical action is required like sounding an alarm, adjusting some valves, or stopping an intruder, which would drain the batter-

ies and increase the form factor significantly. A single sink can only serve a limited number of sensor nodes, so large deployments may include multiple sinks each serving a patch of the nodes. For convenience, and without loss of generality, we will restrict the discussion to a single sink from now on.

Two classes of monitoring applications can be distinguished. *Periodic monitoring* of key parameters and *event-based reporting* of outliers. Examples of the periodic reporting class include the observation of nesting patterns of storm petrols at Great Duck Island [22], measuring light intensities at various heights in a redwood tree [30], and logging temperature and humidity in the canopy of potato plants for precision agriculture [10]. Examples of event-based reporting include the localization of a sniper based on analyzing the sound of a gun shot [29], and the "A Line in the Sand" intrusion detection system [2].

Monitoring applications do not need to transfer large amounts of data; typical rates are in the order of 1-200 bytes per second. In the case of periodic reporting, simple delta coding yields a huge reduction in data since physical parameters like temperature do not change that fast. In the case of event-based reporting, rather infrequent KEEP-ALIVE messages guaranteeing system integrity dominate traffic with an occasional burst in activity when an interesting event occurs. An important consequence of the low data rate is that messages are rather short, with typical payloads of 25 bytes and less. The most important characteristic of monitoring applications, however, is the highly structured style of inter-node communication, which is limited to three basic communication patterns known as *flooding*[1], *convergecast*, and *local-gossip* (see [17]).

- **Flooding:** one of the tasks of the sink node is to control the operation of the multi-hop network of sensor nodes. As such the sink needs to communicate information to all nodes, for example, to change an application parameter or to upload a new code image (bug fix). In principle a flood is initiated by the sink broadcasting a message to all its immediate neighbors, who in turn forward the message to all their neighbors by re-broadcasting it. In practice, however, not all messages are re-broadcast; duplicates are filtered out and transmissions may be suppressed to address the broadcast storm problem (see Chapter XX). The end result being that all nodes have received a copy of the original message of at least one neighbor. By recording the identity of this neighbor, a spanning tree can be setup to provide every node with a route to the sink.
- **Convergecast:** in general sensors report their findings, either periodically or triggered by an event, to the sink along a spanning tree. Since messages are small and need to travel across multiple hops, the overheads become quite large, especially for periodic reporting. Therefore, aggregating messages within interior nodes in the spanning tree pays off. In the best case, only one piece of information needs to be forwarded (e.g., the maximum room temperature) and in the worst case, two (or more) messages can still be coalesced to share a common header. A down-side of aggregation is that it implies waiting (or very careful synchronization) since messages can only be forwarded when all children have reported, which increases end-to-end latency. Fortunately, many WSN applications are rather delay tolerant. Another characteristic of convergecast is that nodes around the sink have to handle more messages than

[1] Originally named broadcast in [17], but we use the term flooding to stress the network-wide nature and avoid confusion with (local) broadcast reaching only the immediate neighbors.

nodes at the edges of the network (i.e., the leaves of the spanning tree). This imbalance makes convergecast rather complicated to handle efficiently at the MAC layer. Aggregation may alleviates the problem by reducing traffic to fewer, and longer messages, but hardly ever cures it completely. Therefore, a MAC protocol should be prepared to handle more traffic around the sink.

- **Local gossip:** the third communication pattern is only employed by sophisticated applications that refine raw data before sending it to the sink. By sharing (gossiping) data with immediate neighbors, nodes can easily filter out false alarms caused by faulty readings (majority vote) or derive additional information like speed and direction of a moving target instead of reporting mere presence. An additional advantage is that after reaching consensus, only one node needs to report back to the sink, eliminating many individual messages travelling over multiple hops. Depending on the situation, neighbors may be addressed individually (unicast) or collectively (broadcast). Although broadcast is attractive from the sender's perspective, we will see below that from the MAC's perspective it is difficult to implement efficiently, that is, without consuming much energy.

In the case of periodic monitoring, convergecast is the dominating communication pattern, while for event-based monitoring the gossiping of KEEP-ALIVE messages consumes most network resources. In both cases, flooding is used rather infrequently, although link and node failures leading to broken spanning trees may be countered by periodically running the setup procedure. Therefore, when discussing the implications of the joint traffic load on the MAC layer in the next section, we will restrict the discussion to the convergecast and local gossip patterns.

3. MAC design space. The driving force behind WSN research is to develop systems that can operate unattended for years, which calls for robust and energy-efficient solutions both at the hardware and software level. Since the radio is the component of a sensor node that consumes most energy, it should be managed carefully. Usually one has to be prepared to pay a price in terms of performance for the desired reduction in energy consumption. Fortunately, many WSN applications are rather undemanding and can get by with low bandwidth and long end-to-end latency. In addition, the resource limitations imposed by typical node hardware call for solutions that require minimal processing and have a small memory footprint. These considerations limit the design space of medium access control. Nevertheless many WSN-specific MAC protocols have been proposed, each shooting for a different trade-off between energy consumption and performance. All protocols, address the same sources of overhead and can be conveniently grouped into three main classes based on the degree of organization between nodes.

3.1. Sources of overhead. When running the standard IEEE 802.11 (CSMA/CA) protocol developed for Wireless LANs (see Chapter XX) on a sensor network with little traffic, much energy is wasted due to the following sources of overhead:

- **Idle listening:** only a fraction of the available bandwidth is needed for communication, but without any further information a MAC protocol cannot tell when a message will be sent. Therefore, the radio must be kept on at all times or a node would miss some of the messages being sent to it. This so-called idle-listening overhead is *the* main source of energy waste as typical radios consume much more energy in receive mode (even when no data is arriving) than in sleep mode (cf. Table 2.1).

- **Overhearing:** another effect of always listening for incoming traffic, is that a node will receive *all* messages including those that concern its neighbors only. Overhearing these messages is simply a waste of energy, and becomes problematic in dense networks with many nodes inside the reception range of a node. Dense deployments are not uncommon because the sensing range of many physical parameters (e.g., temperature) is much smaller than the communication range.
- **Collisions:** although senders are using random back-off within a contention window, collisions can still occur because the switch between carrier sense and transmit takes time. Also the overhead of the RTS/CTS handshake to implement collision avoidance is considered prohibitive in comparison to the small, 25-byte WSN payloads leaving the hidden-terminal problem unaddressed. The usual cure of retransmitting messages may actually degrade performance because of the additional traffic causes more collisions, in turn triggering even more retransmissions, and cascading into total collapse in the worst case.
- **Traffic fluctuations:** traffic generated by WSN applications often fluctuates in time (event-based reporting) and in place (convergecast). The resulting peak loads may drive the network into congestion, or alternatively enforce the use of long contention window (overprovisioning). In either case, energy consumption rises to undesired levels.
- **Protocol overhead:** MAC headers and control messages are considered overhead because they do not contain useful application data, yet consume energy. In the case of WLAN traffic these costs can be amortized, but the small WSN payloads shift the boundary considerably, which essentially rules out sophisticated protocols that exchange detailed information.

Most of these overheads are incurred by other contention-based protocols too, although the relative importance may vary. For example, collisions can be remedied at the expense of protocol overhead.

The alternative of using a schedule-based approach (i.e., TDMA) may seem rather attractive at first glance because idle-listening, overhearing, and collisions simply do not occur; after having received the traffic schedule it is clear in which slots a node should receive and transmit. The problem, however, is the price to be paid in terms of reduced flexibility leading to overprovisioning, protocol overhead, and complexity. Dynamically changing the number of slots in a frame is infeasible, which forces the choice of some upper bound leading to overprovisioning. Collision-free slot assignment implies a large memory footprint for storing the state of the nodes in the two-hop neighborhood. These concerns show that by organizing nodes many of the classic sources of overhead can be avoided, but only at the expense of introducing new ones.

3.2. Organization. The classic S-MAC paper by Ye et al. in 2002 [33], in which they introduce Sensor MAC, inspired the development of a whole string of energy-efficient MAC protocols. More than 50 have been documented with all of them addressing the sources of overhead listed above, and with many claiming their own letter (S-MAC, T-MAC, B-MAC, etc.). Table 3.1 shows an excerpt from the resulting MAC alphabet soup [18] including a "collision" for RMAC. To understand the differences between, and commonalities of, the different WSN-specific MAC protocols, we will classify them according to how nodes organize access to the shared radio channel. This classification is a simplification of [19] focusing on the organizational aspect only.

TABLE 3.1
The MAC alphabet soup; an excerpt taken from [18] classified by organization

Acronym	**Full name**	**Organization**
AI-LMAC	Adaptive Information-centric LMAC	frames
B-MAC	Berkeley MAC	random
⋮	⋮	⋮
RMAC	Randomized adaptive MAC	frames
RMAC	Reliable MAC	random
S-MAC	Sensor MAC	slotted
T-MAC	Timeout MAC	slotted
⋮	⋮	⋮
X-MAC	X-MAC	random
Z-MAC	Zebra MAC	hybrid

We distinguish three different classes of organization. The simplest being *random access*, in which nodes do not organize time and contend for access to the radio channel. To reduce idle listening, protocols in this class shift the costs from the receiver to the sender by extending the MAC header (i.e., the preamble), allowing nodes to check the channel periodically and sleep most of the time. This elegant, low-level approach will be detailed in Section 4.

A slightly more complex organization is to divide time into slots. *Slotted access* requires nodes to synchronize on some common time of reference such that they can wake-up collectively at the beginning of each slot, exchange messages when available, and then go back to sleep for the remainder of the slot. S-MAC is the prime example from this class of slotted access, and improves on CSMA/CA by implementing a fixed duty cycle and overhearing avoidance, see Section 5.

The most complicated class schedules the channel access in great detail. Time is divided into frames containing a fixed number of slots. *Frame-based* protocols differ in how slots are assigned to nodes. Classic TDMA (Chapter XX) has an access point controlling this for a single cell. LMAC [32] on the other hand employs a distributed slot-selection mechanism that self-organizes a multi-hop network into a conflict-free schedule. LMAC and other approaches will be discussed in Section 6.

The increase in the degree of organization allows for tighter control of who is communicating when, but at the expense of being less flexible to accommodate changing conditions. Therefore, several hybrid protocols have been developed aiming at combining the best of both worlds. Such combinations of random and frame-based access will be discussed last (Section 7).

Figure 3.1 shows the historic development within each class of protocols. We will use these "time lines" as the basis for discussion because they demonstrate how a basic idea can be progressed through a small series of optimizations into a state-of-the-art protocol.

4. Random access. This class of CSMA-style protocols does not restrict when nodes may access the channel. This provides a lot of flexibility to handle different nodes densities and traffic loads, so nothing has to be decided before deployment and dynamic changes (e.g., a node joining the network) can be accommodated easily. Also, nodes need not synchronize their clocks, making these protocols rather simple. The down side of this relaxed, random access approach is that lots of energy is often wasted due to idle listening and collisions.

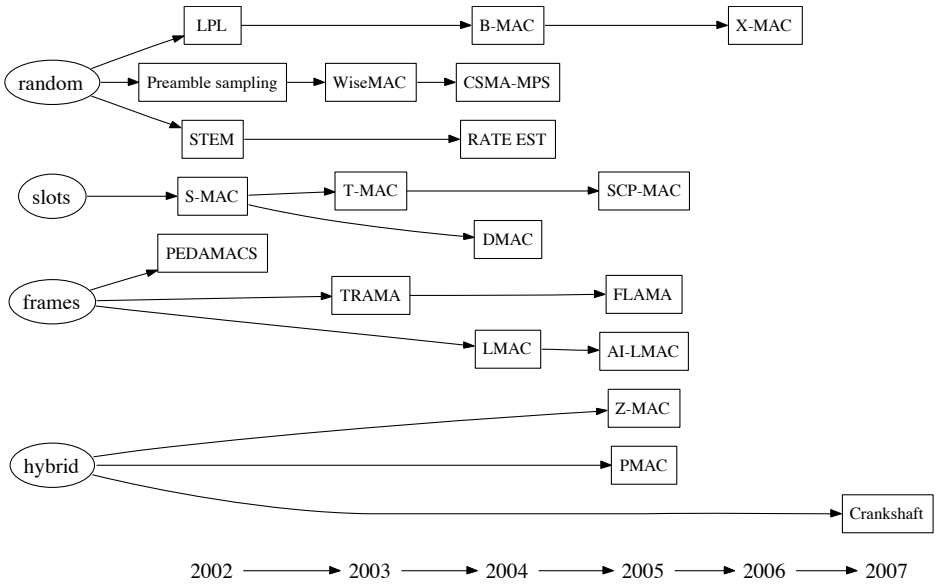

FIG. 3.1. *Taxonomy of MAC protocols according to time organization and historic development.*

Handling collisions has been extensively studied before, both in wired and wireless systems. Unfortunately, the standard solutions cannot be applied 'out of the box' because of the tight WSN constraints detailed in Section 2. *Collision Avoidance* signalling is considered prohibitive because of the short payloads, and contention resolution by means of *random backoff* leads to overprovisioning with nodes listening for long contention windows or to collapse when using short windows. The *binary exponential backoff* (BEB) procedure discussed in Chapter XX addresses the latter concerns, but at the expense of considerable protocol complexity. A much simpler approach was proposed by Jamieson et al. as part of the **Sift** protocol [15].

The key contribution of Sift is that instead of using a uniform random selection within the contention window, a sender competing for channel access uses a skewed distribution giving preference towards the end of the window. This greatly reduces the possibility of collisions (i.e., senders selecting the same "slot" within the contention window) because the low chance of selecting an early slot usually leads to just one lucky winner when many compete, and with few competitors the chance of a collision is already low to begin with. The optimal distribution depends on the number of competing senders, which in practice is unknown. Extensive analysis in [15], however, has shown that using a truncated, increasing geometric distribution with a fixed parameter α shows near optimum performance over a wide range (2-256) of competing senders making it a very practical approach indeed. Note that Sift does not address the hidden terminal problem, and one might consider using collision avoidance to address the residual chance of collision. However, given the overheads involved in RTS/CTS signalling compared to the small WSN payloads, taking the resulting retransmissions for granted is generally accepted as the best approach.

To date, Sift's simple, yet effective contention resolution method has only been adopted by one other MAC protocol (Crankshaft, Section 7.1), but could be applied to all protocols involving random access. Having dealt with the problem of collisions, we now turn our focus on how to reduce the idle listening overhead of basic CSMA.

4.1. Low-Power Listening and Preamble Sampling. Both Hill et al. [13] and El-Hoiydi [7] independently devised a low-level scheme in which nodes can periodically sense the channel (saving energy) and still not lose any messages (due to sleeping most of the time). The idea is to prepend each message with a kind of "busy tone" to alert potential receivers about an upcoming message transfer. Nodes sensing a busy tone would then keep their radios on until the end of the message. By making the busy tone longer than the sleep interval, a sender is guaranteed to wake up the intended receiver irregardless of its phase in the poll cycle. A convenient way to implement the busy tone is to stretch the length of the standard preamble (part of the physical layer header). Figure 4.1 illustrates this periodic sampling scheme. The beauty of it is that the costs shift from the receiver (reduced idle listening) to the sender (long preamble). Since there are many more receivers than senders and the amount of traffic is low, a lot of energy is saved.

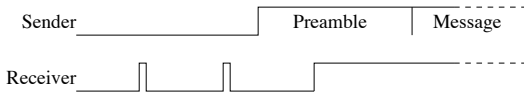

FIG. 4.1. *Low-power listening: a long preamble allows periodic sampling at the receiver.*

Since the periodic sampling effectively occurs at the physical layer, it can be combined with any MAC protocol in the link layer. Hill et al. combined it with CSMA and named it **Low-Power Listening**, LPL for short. El-Hoiydi combined it with ALOHA and named it **Preamble Sampling**. The exact savings of these protocols depend on the ratio of the time it takes to do a carrier sense and the length of the sleep interval. A duty cycle of around 10% is certainly possible without compromising latency too much. For example, on a CC1000 radio a carrier sense takes about 2 ms (cf. Table 2.1) leading to an extended preamble of 20 ms, which is acceptable given the delay-tolerant nature of many WSN applications. A potential drawback is that receiving or overhearing a message has become more expensive, because the time between waking up and receiving the start symbol of the actual message is on average half the length of the stretched preamble (i.e., 10 ms). For low data rate applications, however, this is of little concern.

Note that for every scenario (data rate, node density) an optimal sleep interval exists that balances the costs between receivers and sender. The **B-MAC** protocol [24] allows for runtime configuration of the sleep interval to provide the possibility for application developers to optimize their energy savings. Another contribution of B-MAC is that it includes an optimized carrier sense procedure. Instead of taking a single sample, B-MAC takes five consecutive samples and assesses the channel as being clear if any of those readings falls below a predefined threshold. This effectively eliminates random noise (interference), hurting the original LPL implementation due to too many false alarms (a busy channel, but no preamble).

4.2. WiseMAC and CSMA-MPS. A first refinement to Preamble Sampling was introduced by El-Hoiydi in the **WiseMAC** protocol [8]. With a little bookkeeping, the need for sending out long preambles can be largely avoided. Given that a node typically communicates with just a few nodes, or actually just one (its parent), maintaining the phase offset of when a destination node wakes up becomes feasible. The idea is to start transmitting a message just before the intended receiver wakes up to sample the channel. This saves energy both at the sender, who sends out short

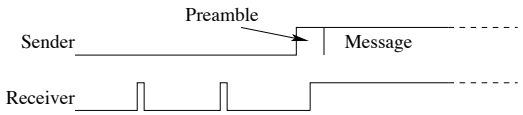

FIG. 4.2. *WiseMAC: a short preamble in sync with the receiver's sampling schedule.*

preambles, as well as at the receiver since the busy-waiting time for the start symbol is reduced to half the length of the short preamble. The savings can be substantial. Continuing the CC1000 example with a 20 ms preamble and 5 ms payload, WiseMAC is able to squeeze out up to 80% (20 out of 25 ms) of the transmission costs and up to 67% (10 out of 15 ms) of the reception costs. As an added benefit, shortening the preambles also reduces overhearing by nodes other than the sender/receiver pair. To enable this optimization, WiseMAC piggybacks the local phase offset on the acknowledgements of the underlying CSMA protocol. It further compensates for clock drift by extending the length of the preamble with a time proportional to the length of the interval since the last message exchange. The end result is that WiseMAC uses short preambles for regular traffic, and falls back to long preambles for infrequent message exchange. This does not, however, hold for broadcast, since the preamble must span the sampling points of all neighbors and is therefore stretched to full length.

A further refinement was proposed by Mahlknecht et al. as part of the **CSMA-MPS** protocol [21]. MPS stands for Minimal Preamble Sampling and introduces an optimization to reduce the need for long preambles in the case of low traffic. With CSMA-MPS a sender sends out a strobed sequence of preambles, allowing the receiver to reply in between, after which the sender can immediately transmit the true message. On average this halves the transmission costs and reduces the reception overhead close to the bare minimum of one carrier sense operation. The idea of the strobed signalling sequence was taken from STEM (discussed below) and its application to CSMA was re-invented two years later by X-MAC [4].

4.3. Wake-up radio. A radically different approach to reducing the idle listening overhead is to equip nodes with a second ultra low-power radio used for simple signalling. By default, nodes sleep with the main radio turned off. They can be awoken by sending a kind of wireless interrupt over the second radio, after which the main radio is switched on to receive the message. Receiving such an interrupt does not require complicated decoding circuitry, so an extremely simple radio consuming very little energy can be used.

The attractive idea of using a low-power wake-up radio was first coined by the **PicoRadio** project [11] detailing a design out of passive components consuming as little as 100 μW. The down side of such extremely simple radios is that they are quite susceptible to noise (generating false alarms), and use broadcast signals (waking up all neighbors) because they cannot even encode a few address bits specifying the identity of a target node. Schurgers et al. proposed to use a slightly more powerful radio and use LPL to keep the energy consumption at negligible levels as part of their **Sparse Topology and Energy Management** (STEM) protocol [28]. The increased latency due to the long preambles is countered by having the target node immediately acknowledge the reception of the wake-up signal instead of waiting for it to finish. To this end a wake-up signal is implemented as a sequence of beacon packets containing the source and destination address, leaving space for the receiver to transmit its ACK message.

The **Rate Estimation MAC** by Miller et al. proposed a software solution to the problem of waking up all neighbors [23]. Essentially many "wake-up all" broadcasts can be prevented by piggybacking an explicit interval on a message telling the receiver when to wake up next. If the sender does a good job of estimating the next time it needs to send/forward a message, no wake-up signal needs to be broadcast. If not, the sender must either wait for the next scheduled wake-up of the receiver, or incur the overhead of waking up all neighbors. In the best case (periodic reporting) an initial wake-up notifies the receiver of the reporting interval, after which no further wake-up signals need to be broadcast. In the worst case (event-based reporting) each message requires a wake-up signal awakening all neighbors, and includes a false estimate of the next event causing superfluous wake ups at the receiver.

Because of the additional hardware costs of a second radio, and the software complexities involved when nodes cannot be awoken individually, the idea of a wake-up radio has not resulted in an actual implementation.

5. Slotted access. The basic idea behind this class of contention-based protocols is to save energy by having nodes agree on a common sleep/active pattern allowing them to operate the radio at arbitrarily low duty cycles. Time is divided into slots, and nodes wake up at the beginning of each slot to handle pending messages waiting for transmission. Channel access is based on contention as with the random access protocols, but the possibility of collision is much higher due to all communication being grouped into the (small) active part of a slot. Therefore effective contention resolution is much more a priority and some slotted protocols even go as far as including collision avoidance signalling (i.e., RTS/CTS handshaking) despite the relatively large protocol overhead. Apart from this issue, slotted protocols mainly differ in their policy on when to switch back from active to sleep mode.

5.1. S-MAC. The main contribution of the **Sensor-MAC** protocol [33] is that its fixed duty-cycle approach is both simple and effective in reducing idle listening overhead. The only complicated part is the synchronization of the nodes on the basic slot structure shown in Figure 5.1. Nodes regularly broadcast SYNC packets including a time stamp at the beginning of a slot, which allows others to adjust their local clocks to compensate for drift. New nodes wanting to join the ad-hoc network start off with listening for an initialization period spanning multiple slots waiting for a SYNC packet to inform them about the common schedule. If no SYNC packet is received a node concludes it is the first one to form a so-called *virtual cluster* and starts broadcasting SYNC packets so others can join in later.

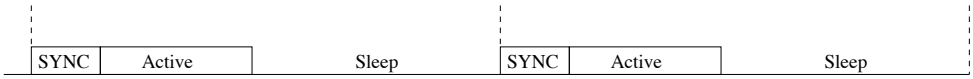

FIG. 5.1. *Slot structure of S-MAC with built-in duty cycle.*

Occasionally two virtual clusters meet, for example due to mobility or bootstrapping a large deployment, in which case either the two schedules must be united or some "bridge" nodes must run both schedules. S-MAC uses the latter option since adding another timer-event (marking the beginning of a slot) is rather easy. More advanced protocols by the same designers, however, have opted for synchronizing all nodes on a single schedule to guarantee that a broadcast message will indeed reach

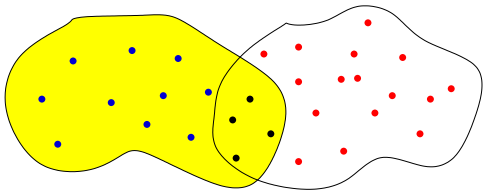

FIG. 5.2. *Overlapping virtual clusters with bridging nodes running both schedules.*

all neighbors and to avoid the problem that bridging nodes will drain their batteries much faster.

Once nodes have joined the (global) slot schedule they will start duty cycling their radio switching it off after every active period. The S-MAC implementation for the Mica2 motes uses a fixed-length active period[2] of 300 ms and a configurable slot length in the order of 1-3 s. Collision avoidance by means of an RTS/CTS handshake is included in the protocol, which features a fixed-length contention window of about 9 ms. The RTS/CTS control packets include information about the length of the DATA packet allowing nodes overhearing these messages to switch off their radios for the remainder of the transfer sequence (up until the ACK packet), much like setting the Network Allocation Vector (NAV) in IEEE 802.11 (see Chapter XX). A surprising effect of S-MAC's overhearing-avoidance mechanism is that energy consumption goes down when the traffic load increases, because in an empty network a node wastes the complete active period on idle listening while overhearing traffic allows it to temporarily switch the radio off.

5.2. T-MAC. The simplicity of S-MAC using a fixed duty cycle has two drawbacks. First, an application developer is left with the burden of selecting the optimal duty cycle before deployment commences. Second, traffic fluctuations can only be dealt with by overprovisioning, that is, by setting the duty cycle to the (anticipated) maximum load at any moment, at any location in the network. In this regard convergecast and event-based reporting leave S-MAC wasting lots of energy. To address these issues, the **Timeout MAC** protocol by van Dam and Langendoen [31] introduced an *adaptive* active period. By default nodes listen only for a short duration at the beginning of a slot (15 ms for T-MAC vs. 300 ms for S-MAC) and go back to sleep when no communication happens. If, on the other hand, a node engages or overhears a message transfer it will schedule another listen period after this transfer to determine if it can then go to sleep. The end result is that a node will stay active until no communication has been observed for the duration of the 15 ms timeout period. Simulations have shown that T-MAC is capable of adapting to traffic fluctuations both in time (event-based reporting) and place (convergecast), and that it outperforms S-MAC running at a fixed duty cycle by as much as a factor of 5 in energy consumption [31].

In principle the timeout mechanism will automatically adapt the duty cycle to the actual traffic in a node's neighborhood. However, T-MAC is a bit too aggressive in shutting down the radio, leaving messages queued for the next slot, which effectively increases latency and reduces throughput. Consider the following scenario in which

[2] A subsequent refinement of S-MAC (called *adaptive listening*) includes a variable-length active part to reduce multi-hop latency [34]. Since the T-MAC protocol behaves similarly and was designed to handle traffic fluctuations as well, we do not discuss adaptive listening further.

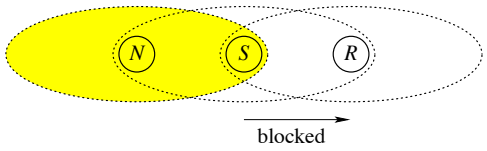

FIG. 5.3. *Hidden-terminal scenario in which node N silences S causing R to go to sleep early.*

a node S wants to send a message to R, but loses contention to a third node N that is not a common neighbor. This forces S to stay silent, so R hears nothing and goes to sleep. When N finishes, S will try to reach R being sound asleep. This is wasting energy and increasing latency because S has to wait until the next slot before it can try to contact R again. T-MAC includes two measures to alleviate this so-called early-sleeping effect (for details refer to [31]). Under low load conditions, however, T-MAC works absolutely fine (2.5 % duty cycle) outperforming S-MAC by quite a margin when traffic is non-uniform and bursty.

5.3. SCP-MAC. Although using an adaptive duty cycle was a major step forward, Ye et al. observed that T-MAC (and their equivalent S-MAC with adaptive listening [34]) still contains ample room for improvement due to the standard approach towards handling/avoiding collisions. In particular, T-MAC's 15 ms timeout period includes a 9.15 ms contention window, which goes by unused most of the time with low data rate applications (event-based reporting). Also the use of the RTS/CTS handshake only adds overhead (2×1.5 ms) in this case. Ideally, all that is required is just a single carrier sense (2 ms) leading to a potential sevenfold (15/2) reduction in energy consumption with duty cycles as low as 0.3 %. What is more, Ye's latest addition to the family of slotted protocols, the **Scheduled Channel Polling MAC** [35], actually realizes that by means of a novel contention resolution mechanism detailed below.

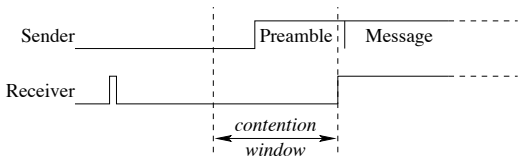

FIG. 5.4. *SCP: sender-only contention resolution by means of a stretched preamble.*

The key insight is to first have the senders contend for the channel and then have all nodes but the winner check if they are the intended receiver (overhearing avoidance). If there is no message to be sent, everybody will observe a clear channel and go to sleep immediately. Figure 5.4 illustrates this process. Each slot starts off with a contention window. A node that wants to send a message chooses a random moment within this window (preferably using the Sift distribution, but SCP-MAC operates with a uniform distribution). At that moment the potential sender switches on its radio, checks the channel and starts sending a preamble if it was clear. The preamble acts as a busy tone and continues until the end of the contention window locking out any other potential senders. Right after the end of a contention window all nodes except the winner, if any, wake up and perform a carrier sense to see if there is a preamble followed by a message. Without any traffic, SCP-MAC thus only needs

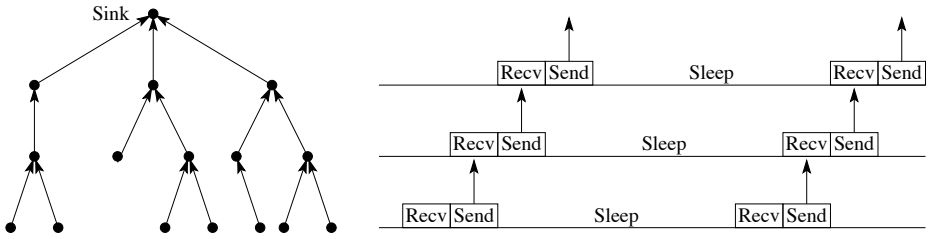

FIG. 5.5. *Convergecast tree with matching, staggered DMAC slots.*

to perform one carrier sense[3] per slot making it the most efficient protocol of its class. Note that since SCP-MAC can only handle one message per slot, it must use shorter slots than S-MAC and T-MAC to warrant reasonable end-to-end latencies and peak throughputs.

It is instructive to observe the similarities with WiseMAC (cf. Figure 4.2). Since SCP-MAC synchronizes all nodes it does not need additional signalling and bookkeeping of clock phases, because it simply "knows" the next wake up time. On the reverse side, WiseMAC can get by without a contention window because traffic is not clustered in a single active period but spread out over the complete slot. Then again, SCP-MAC supports broadcast very well because of this clustering while WiseMAC must use (very) long preambles to reach all neighbors with one message. As always which protocol performs best depends on the application.

5.4. DMAC. As mentioned before, energy efficiency comes at the price of reduced performance. The slotted protocols discussed so far all compromise on latency. When an application injects a message into the network, that message must wait for the next slot to turn up before it can be sent in the first place. Then additional delays may be encountered at each intermediate node. With S-MAC a message may travel multiple hops depending on the length of the active period. With T-MAC the number of hops per slot is limited to three due to the early sleeping effect. With SCP-MAC only one message can be sent per slot.

The **Data gathering MAC** [20] addresses the latency issue for the convergecast communication pattern. DMAC was originally designed to improve S-MAC, but T-MAC and SCP-MAC suffer from long multi-hop latencies too. The basic idea is to stagger the active times according to the level in the spanning tree such that data can quickly flow through from the leaves to the root, see Figure 5.5. Each node first listens to its children, then propagates any messages up to its parent. DMAC uses simple CSMA with acknowledgements. Nodes losing contention need not wait for the next upwards flow, but may try again in an overflow slot scheduled after any occupied Recv/Send pair (not shown in Figure 5.5). To account for interference with traffic higher up in the tree, these overflow pairs are scheduled with a 3 slot gap. The overflow slots essentially increase capacity on demand making DMAC automatically adapt to the traffic load, much like T-MAC's extension of the active period.

The down side of DMAC is that it lacks the flexibility to support communication patterns other than convergecast. In particular local-gossip based on broadcast does not work because neighbors (children, peers, and parents) listen at different times. This could well be the reason that DMAC never passed the simulation stage. Never-

[3]In reality, SCP-MAC uses a second contention window instead of just a carrier sense to counter the residual collisions from the first phase, which is not necessary when using Sift.

theless, staggering active periods is a compelling approach to reduce latency without increasing energy consumption, so it warrants further research.

6. Frame-based access. The third class of MAC protocols that we discuss constrains flexibility even further by grouping slots into frames and scheduling in detail who is to send in each slot. The advantage of a schedule-based (TDMA-like) approach is, of course, that collisions do not occur and that idle listening and overhearing can be drastically reduced. When scheduling communication links, that is, specifying the sender-receiver pair per slot, nodes only need to listen to those slots in which they are the intended receiver eliminating all overhearing. When scheduling senders only, nodes must listen in to all occupied slots, but can still avoid most overhearing by shutting down the radio after the MAC (slot) header has been received. In both variants (link and sender-based scheduling) idle listening can be reduced to a simple check if the slot is used or not. To capitalize on these advantages several MAC protocols have been developed that take classical TDMA solutions using an access point (see Chapter XX) to a WSN setting without any infrastructure.

6.1. PEDAMACS. The first approach to computing and distributing TDMA schedules in a multi-hop sensor network is to leverage the abundant resources available at the sink. In particular, the **Power Efficient and Delay Aware Medium ACcesS** protocol [6] assumes that the sink includes a high-powered radio that can reach all nodes in the network. This allows the sink to synchronize the nodes and to schedule their transmissions and receptions. Since most sensor nodes cannot reach the sink directly to inform it about their communication demands, PEDAMACS includes a special initialization procedure. First the sink sets up a spanning tree and then nodes report back about their local topology (parent, children, and others) and anticipated data rate (periodic reporting). Once all this information has reached the sink, it knows the complete topology (i.e., all links) and can compute a collision-free global schedule, which it broadcasts out to the complete network. Then the data collection phase starts and nodes receive and send messages according to that schedule. Currently PEDAMACS only supports convergecast, but other communication patterns could be handled equally well, see for example the closely related work by Arisha et al. [1]. To handle occasional topology changes, for example due to movement and external interference, PEDAMACS occasionally runs an adjustment procedure where nodes can report differences to the local topology they initially observed.

The assumption that the sink can reach every node in the network is questionable because of obstacles blocking line-of-sight and multi-path reflections making it impossible to decode messages at certain positions within the sink's reach. A second concern is that within the initialization procedure, PEDAMACS takes a CSMA approach to avoid collisions such that all nodes will get their local information to the sink. However, collisions cannot be ruled out completely, so some nodes may not reach the sink, effectively silencing them in the data collection phase. Furthermore by including CSMA as part of the protocol, PEDAMACS as a whole becomes rather complex, which increases its code and memory footprint.

6.2. TRAMA and FLAMA. The approach by Rajendran et al. assumes that all nodes are equal and includes a distributed scheduling component. In their **TRaffic-Adaptive Medium Access** protocol [26] nodes regularly broadcast information about (long-term) traffic that flows through them as well as the identities of their neighbors. By observing these reports a node learns the identities of all its two-hop neighbors, which is used to compute a collision free schedule by means of a distributed

hash function that determines the winner (i.e., sender) of each slot based on the node identities and slot number. The traffic load information of the one-hop neighbors is used to break ties in favor of the busiest node. To reduce overhearing, a sender includes a bitmap in each packet detailing the subsequent receivers it plans to be sending to in the next 100 slots. If the actual traffic is lower than the initial estimate broadcasted to all neighbors, a node releases (some of) its claims by zeroing out the remainder of the bitmap. This allows others to take over and provides limited capabilities to adjust to traffic fluctuations.

Since traffic information needs to be broadcast every 100 slots to keep the scheduling mechanism functioning, the TRAMA protocol entails quite some overhead. The follow-up **FLow-Aware Medium Access** protocol [25] leverages the stability of periodic reporting applications by reverting to a pull-based mechanism. Instead of pushing (broadcasting) traffic information out, nodes now only respond upon explicit requests generated when a new flow is established. This reduces the overhead considerably, but the reported energy consumption data shows that FLAMA still barely outperforms S-MAC [25]. As with PEDAMACS, protocol complexity is increased by a random access component; the global FLAMA schedule includes a few special slots to bootstrap the scheduling process and accommodate new nodes joining the network.

6.3. LMAC. The **Lightweight MAC** protocol by van Hoesel et al. [32] also contains a distributed slot selection mechanism based on two-hop neighborhood information, but unlike TRAMA and FLAMA the schedule does not depend on the slot number making it rather trivial to compute. Besides simplicity, an important design goal was to minimize the number of transitions between receive and transmit mode, because they take time during which the radio cannot be used.

Each node owns a fixed-length time slot, in which it always transmits a header, optionally followed by a payload. The header contains various fields including the destination and length of the data payload. Unlike most other MAC protocols, the correct reception of the data is not acknowledged by the receiver to avoid expensive radio switches; LMAC puts the issue of reliability at the upper layers.

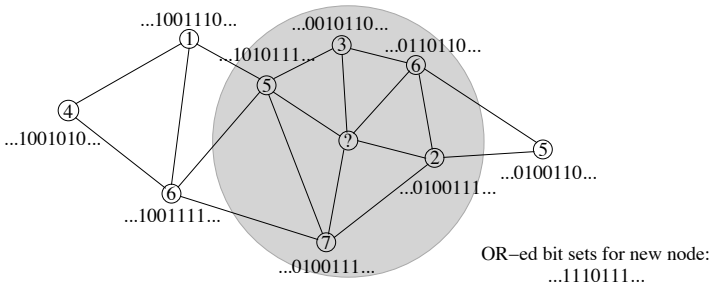

FIG. 6.1. *Slot selection by LMAC; nodes are marked with slot number and occupancy bit set.*

To facilitate new nodes joining the network, each header includes a bitset detailing which slots are occupied by the one-hop neighbors of the sending node (i.e., the slot owner). By OR-ing the occupancy bitsets of all headers in a frame, a new node can easily determine which slots are still available in its two-hop neighborhood. It randomly selects one of those as its own and starts sending out headers to actually claim it. In the unlikely event of two nodes joining at the same time *and* selecting the same "free" slot, a collision is eminent resulting in garbled headers. A neighboring

node observing this (i.e., the header checksum fails) will broadcast the involved slot number as part of its header, signalling the new comers to back off and try again.

LMAC's lightweight slot selection mechanism has one drawback. The number of nodes in any two-hop neighborhood cannot exceed the number of slots in a frame, which needs to be fixed before deployment. Choosing a large number of slots per frame leads to overprovisioning (wasted slots) and protocol overhead (large bitsets); choosing a low number may lock out nodes in (dense) deployments. By allowing nodes to claim multiple slots per frame in the followup **Adaptive Information-centric LMAC** (AI-LMAC) protocol [5], the problem is somewhat alleviated. It also allows the protocol to raise the throughput of the convergecast pattern by allocating more slots to nodes closer to the sink. As for reliability, the decision on how many slots a particular node should claim is left to the layers above. All in all (AI-)LMAC is a neat, simple frame-based protocol that performs well in terms of energy consumption. Refer to [19] for how LMAC beats random and slotted access protocols in many cases.

7. Trends. In the preceding sections we have shown how three different approaches to creating energy-efficient MAC protocols progressed from simple ideas (e.g., active/sleep cycles by S-MAC) to advanced protocols (e.g., channel polling by SCP-MAC). They all addressed idle-listening as the major source of overhead, and the state-of-the-art protocols from each class (random, slotted, and frame-based access) leave little room for further improvement. This has prompted researchers to shift their attention to the reverse side of saving energy, namely reduced performance and narrowed scope. A recent trend is to create hybrid protocols that take advantage of, say, the flexibility of random access and the collision-free nature of framed-based access. Another consideration that has prompted the development of new protocols is the hardware shift towards packet-based radios, which on the one-hand frees the MAC layer from handling individual bytes, but on the other hand deprives it from low-level techniques like stretching preambles. Both trends will be discussed below.

7.1. Hybrid protocols. In a way slotted protocols (e.g., T-MAC and SCP-MAC) can be viewed as already striking a middle ground between random and frame-based access. They impose some structure on when nodes are allowed to communicate, but retain much of the flexibility to adapt to traffic fluctuations and topology changes. The main drawback of slotting, however, is that all communication is grouped into the beginning of a slot raising the chances of collisions. This effectively limits the scope of slotted protocols to low traffic applications.

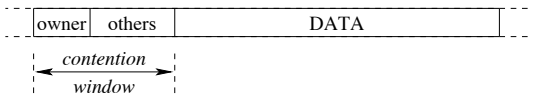

FIG. 7.1. *Z-MAC: slot structure with built-in preference for slot owners.*

The **Zebra MAC** (Z-MAC) protocol [27] takes another approach in combining random access and frame-based scheduling by dynamically switching between the two, which widens the scope of applications that can be supported. Z-MAC starts off by running a distributed slot assignment algorithm that takes the two-hop neighborhood into account to arrive at a conflict-free schedule. Next, nodes must contend for access when wanting to send a message. By default a node may contend for any slot, but an owner gets priority by contending first, see Figure 7.1. When a node observes that it loses too many packets, it broadcasts an explicit notification packet that tells nodes

to switch to high contention mode. In this mode nodes may no longer contend for slots owned by their second-hop neighbors, which prevents them from causing any further collisions as hidden terminals. After a 10 s timeout nodes fall back to normal operation. Z-MAC essentially builds a TDMA-overlay on top of basic CSMA, using B-MAC (low-power listening) to achieve energy-efficiency. An added benefit from layering on top of CSMA is that it makes Z-MAC very tolerant to clock drift unlike many other TDMA schemes.

Another way of combining CSMA with TDMA is followed by **Pattern MAC** (PMAC) [36] and **Crankshaft** [12]. These two protocols share the idea of scheduling *receive* slots. The advantage is that a node only needs to wake up in its own slot(s) to check for incoming traffic, instead of in every slot as with classical, sender-based schemes like LMAC. Crankshaft simply allocates receivers to slots based on node ID (modulo frame length), which avoids the need for maintaining neighbor state. PMAC uses a more elaborate scheme taking traffic load into account and includes a special section at the end of each frame where each node announces its sleeping pattern for the next frame. A pattern is actually a repetition of an n-sleep/1-awake cycle allowing for a representation as a single number. This reduces state per neighbor to an (ID,n) tuple, but does not spread out activity over the entire frame as Crankshaft does. A consequence of scheduling receivers is that neighbors might share the same slot, forcing senders to contend within each slot. Crankshaft uses the efficient contention resolution scheme of SCP-MAC (with Sift, see Section 5.3), while PMAC is based on CSMA/CA including the full RTS/CTS handshake. Crankshaft is the better engineered solution as it consumes less energy, with PMAC having the edge on adaptivity. Crankshaft does, however, include an optimization to reduce the throughput bottleneck around the sink, by having the sink listen in on all slots. Simulation results show that Crankshaft outperforms SCP-MAC especially in high-density scenarios.

7.2. Packet-based radios. MAC design is greatly influenced by the capabilities of the underlying hardware platform. In this respect it is important to observe the trend in cheap, low-power radios to upgrade from byte-level interfaces (e.g., CC1000) to packet-level interfaces (e.g., CC2420). This transition can be largely attributed to the definition of the IEEE 802.15.4 standard for Low-Rate Wireless Personal Area Networks (WPANs) in 2003. This standard specifies a PHYsical and MAC layer for use in consumer electronics, and its commercial potential has brought a new generation radios on the market. The MAC layer provides only limited support for multi-hop networking (see the IEEE 802.15.4 sidebar), but the physical layer fits the WSN community rather well with higher data rates (up to 250 Kbps) at the same energy consumption level as the previous generation.

The switch to packet-based radios is a mixed blessing. On the one hand it frees the micro controller from handling every single byte to/from the radio, which chews up most of the processing resources on simple 8-bit processors like the ATmega128L. On the other hand, it makes life complicated because techniques like low-power listening can no longer be applied due to the lack of control needed to extend the length of the preamble. A crude solution is that, when a long preamble is needed, the message itself can be sent out repeatedly. This works well for small messages and high-speed radios, but reduces the granularity considerably making techniques like scheduled channel polling (Section 5.3) less effective.

The introduction of high-speed radios using more sophisticated coding mechanisms is also a mixed blessing. On the one hand, these radios provide much better energy-per-bit ratios than simpler/slower designs (cf. Table 2.1) and operate at much

The IEEE 802.15.4 standard [14] specifies a physical layer that operates in the unlicensed industrial, scientific and medical (ISM) radio bands; 20 Kbps @ 868 MHz in Europe, 40 Kbps @ 915 MHz in the USA, and 250 Kbps @ 2.4 GHz worldwide. It distinguishes two types of devices; Full Function Devices (FFDs) and Reduced Function Devices (RFDs). An FFD can take the role of a PAN coordinator servicing the traffic of a set of RFDs forming a start topology. FFDs can organize into an (overlay) mesh network.

Being a typical standard, the 802.15.4 MAC layer does specify a single access method, but includes all three modes of organization. *Random access* with the coordinator always listening and RFDs engaging in unslotted CSMA. *Slotted access* with the coordinator sending out beacons detailing a frame layout consisting of a number of contention slots followed by a period of inactivity. *Frame-based access* with the coordinator reserving an additional number of Guaranteed Time Slots (GTS) for specific RFDs with real-time communication requirements. The standard does not detail how multiple coordinators should operate together (e.g., should they tune the length of their beacon intervals?), leaving it up to individual and groups of vendors, such as the Zigbee alliance, to fill this void.

IEEE 802.15.4

lower signal-to-noise ratios (i.e., cover longer ranges). On the other hand, the receive circuitry has become much more complex making idle-listening an even more significant overhead. For example, the CC2420 radio consumes more energy when receiving than when sending (63 vs. 57 mW). Also, because of the higher data rates, the relative cost of switching the radio between send and receive mode has increased forcing MAC designers to pay more attention to this issue.

8. Conclusions. Application scenarios for wireless sensor networks impose strict constraints on energy consumption and system resources, which calls for novel solutions in MAC design. Typical energy-efficient approaches trade off performance in terms of throughput and latency in return for network lifetimes in the order of years with nodes assembled out of commodity components and powered by a set of penlight batteries. Since even a low-power radio is consuming two to three orders of magnitude more energy when switched on than when in sleep mode, the focus of attention is on reducing the so-called idle listening overhead. Without further knowledge nodes must be prepared to handle incoming traffic at any moment, which leads to large energy wastage for typical low-bitrate WSN applications. An additional complication is that traffic is not uniformly distributed, but shows considerable fluctuations in time and space due to the event-based reporting style and convergecast pattern of communication, respectively. A large number of energy-efficient MAC protocols have been proposed, each with its own specific trade-off. We classified these protocols based on how they organize time and access to the radio channel (random, slotted, and frame-based access).

The class of *random access* methods is based on a technique known as Low-Power Listening [13] aka Preamble Sampling [7]. Receivers periodically sample the channel for activity at a duty cycle in the order of 10 %. A sender signals the intended receiver by transmitting a long preamble that spans the receiver's polling interval. This shifts the costs from the receiver (reduced idle listening) to the sender (longer preambles). Optimizations include maintaining clock-phase offsets of neighbors (WiseMAC [8]) and strobing the wake-up signal (CSMA-MPS [21] and X-MAC [4]).

With *slotted access* nodes synchronize on a global schedule with an alternating sequence of active and sleep periods. Prominent protocols from this class are S-MAC [33] featuring a fixed duty cycle, T-MAC [31] with an adaptive duty cycle automatically

accommodating traffic fluctuations, and SCP-MAC [35] with an advanced contention resolution mechanism that only involves the sending nodes, which reduces the basic duty-cycle to well below 1 %. The down side of slotting is that all communication is grouped into active periods, raising the chance of collision.

With *frame-based access* communication is scheduled in great detail with slots being assigned to senders based on information about a two-hop neighborhood making it inherently collision-free. LMAC [32] includes an occupancy bitset in the slot header, which allows a new node joining the network to determine a free slot by simply OR-ing the bitsets of all its neighbors. TRAMA [26] even takes traffic load into account and broadcasts detailed information on traffic flows at every node. This provides adaptivity but at the expense of protocol overhead and complexity.

A recent trend is to combine the flexibility of random, CSMA-style access with the collision-free nature of frame-based, TDMA-style access. Z-MAC [27] implements a TDMA overlay on top of B-MAC and switches dynamically depending on the level of contention. PMAC [36] and Crankshaft [12] schedule *receive* slots with senders contending for access using CSMA and Channel Polling, respectively. This setup minimizes overhearing while still utilizing the complete bandwidth, which makes them suitable for dense networks.

Which MAC protocol achieves the best performance/energy trade-off depends to a large extent on the WSN application at hand as well as the specific hardware platform. In that respect, it is interesting to note that the move of the low-power radio vendors towards high-speed, packet-based radios (supporting IEEE 802.15.4) changes some of the assumptions made by MAC designers. In particular, precise control of the preamble length is no longer possible, making low-power listening and variants less attractive. Hence, some new WSN-specific protocols are expected to emerge in the near future.

Acknowledgments. This work was mainly performed during my sabbatical at ETH Zurich over the summer of 2006, which provided me the unique opportunity to catch up – without being disturbed – with two years of rapid progress in the field of energy-efficient MAC protocols. In addition, I would like to thank Muneeb Ali, Gertjan 'Xfig' Halkes, Andreas Meier, and Tom Parker for proofreading this chapter and for sharing their hands-on expertise with the design, simulation, and implementation of various protocols.

REFERENCES

[1] K. Arisha, M. Youssef, and M. Younis, *Energy-aware TDMA-based MAC for sensor networks*, in IEEE Workshop on Integrated Management of Power Aware Communications, Computing and NeTworking (IMPACCT 2002), New York City, NY, May 2002.

[2] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, *A line in the sand: a wireless sensor network for target detection, classification, and tracking*, Computer Networks, 46 (2004), pp. 605–634.

[3] J. Beutel, *Metrics for sensor network platforms*, in Workshop on Real-World Wireless Sensor Networks (REALWSN), Uppsala, Sweden, June 2006.

[4] M. Buettner, G. Yee, E. Anderson, and R. Han, *X-MAC: A short preamble MAC protocol for duty-cycled wireless networks*, in 4th ACM Conf. on Embedded Networked Sensor Systems (SenSys 2006), Boulder, CO, Nov. 2006, pp. 307–320.

[5] S. Chatterjea, L. van Hoesel, and P. Havinga, *AI-LMAC: An adaptive, information-centric and lightweight MAC protocol for wireless sensor networks*, in 2nd Int. Conf. on

Intelligent Sensors, Sensor Networks and Information Processing, Melbourne, Australia, Dec. 2004.

[6] S. Coleri-Ergen and P. Varaiya, *Pedamacs: Power efficient and delay aware medium access protocol for sensor networks*, IEEE Trans. on Mobile Computing, 5 (2006), pp. 920–930.

[7] A. El-Hoiydi, *Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks*, in IEEE International Conference on Communications (ICC), New York, Apr. 2002.

[8] A. El-Hoiydi and J.-D. Decotignie, *WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks*, in First Int. Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS 2004), Lecture Notes in Computer Science, LNCS 3121, Springer-Verlag, July 2004, pp. 18–31.

[9] O. Gnawali, B. Greenstein, K.-Y. Jang, A. Joki, J. Paek, M. Vieira, D. Estrin, R. Govindan, and E. Kohler, *The tenet architecture for tiered sensor networks*, in 4th ACM Conf. on Embedded Networked Sensor Systems (SenSys 2006), Boulder, CO, Nov. 2006, pp. 153–166.

[10] D. Goense, J. Thelen, and K. Langendoen, *Wireless sensor networks for precise Phytophthora decision support*, in 5th European Conference on Precision Agriculture (5ECPA), Uppsala, Sweden, June 2005.

[11] C. Guo, L. Zhong, and J. Rabaey, *Low power ditributed MAC for ad hoc sensor radio networks*, in IEEE Global Telecommunications Conference (GLOBECOM), vol. 5, San Antonio, TX, Nov. 2001, pp. 2944–2948.

[12] G. Halkes and K. Langendoen, *Crankshaft: An energy-efficient MAC-protocol for dense wireless sensor networks*, in 4th European conference on Wireless Sensor Networks (EWSN'07), Delft, The Netherlands, Jan. 2007.

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

[14] IEEE standard 802.15.4, *Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)*, 2003.

[15] K. Jamieson, H. Balakrishnan, and Y. Tay, *Sift: A MAC protocol for event-driven wireless sensor networks*, in 3rd European Workshop on Wireless Sensor Networks (EWSN'06), Zurich, Switzerland, Feb. 2006, pp. 260–275.

[16] J. Kahn, R. Katz, and K. Pister, *Next Century Challenges: Mobile Networking for "Smart Dust"*, in 5th ACM/IEEE Int. Conf. on Mobile Computing and Networks (MobiCom '99), Seatle, WA, Aug. 1999, pp. 271–278.

[17] S. Kulkarni and M. Arumugam, *TDMA service for sensor networks*, in 24th Int. Conf. on Distributed Computing Systems (ICDCS04), ADSN workshop, Tokyo, Japan, Mar. 2004, pp. 604–609.

[18] K. Langendoen, *The MAC alphabet soup*. `https://apstwo.st.ewi.tudelft.nl/~koen/MACsoup`.

[19] K. Langendoen and G. Halkes, *Energy-efficient medium access control*, in Embedded Systems Handbook, R. Zurawski, ed., CRC press, 2005, pp. 34.1 – 34.29.

[20] G. Lu, B. Krishnamachari, and C. Raghavendra, *An adaptive energy-efficient and low-latency MAC for data gathering in sensor networks*, in Int. Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN), Santa Fe, NM, Apr. 2004.

[21] S. Mahlknecht and M. Boeck, *CSMA-MPS: A minimum preamble sampling MAC protocol for low power wireless sensor networks*, in IEEE Int. Workshop on Factory Communication Systems, Vienna, Austria, Sept. 2004, pp. 73–80.

[22] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, *Wireless sensor networks for habitat monitoring*, in First ACM Int. Workshop on Wireless Sensor Networks and Application (WSNA), Atlanta, GA, Sept. 2002, pp. 88–97.

[23] M. Miller and N. Vaidya, *A MAC protocol to reduce sensor network energy consumption using a wakeup radio*, IEEE Trans. on Mobile Computing, 4 (2005), pp. 228–242.

[24] J. Polastre, J. Hill, and D. Culler, *Versatile low power media access for wireless sensor networks*, in 2nd ACM Conf. on Embedded Networked Sensor Systems (SenSys 2004), Baltimore, MD, Nov. 2004, pp. 95–107.

[25] V. Rajendran, J. Garcia-Luna-Aceves, and K. Obraczka, *Energy-efficient, application-aware medium access for sensor networks*, in 2nd IEEE Conf. on Mobile Ad-hoc and Sensor Systems (MASS 2005), Washington, DC, Nov. 2005.

[26] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, *Energy-efficient, collision-free medium access control for wireless sensor networks*, in 1st ACM Conf. on Embedded Networked Sensor Systems (SenSys 2003), Los Angeles, CA, Nov. 2003, pp. 181–192.

[27] I. Rhee, A. Warrier, M. Aia, and J. Min, *Z-MAC: a hybrid MAC for wireless sensor networks*, in 3rd ACM Conf. on Embedded Networked Sensor Systems (SenSys 2005), San

Diego, CA, Nov. 2005, pp. 90–101.

[28] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, *Optimizing sensor networks in the energy-latency-density design space*, IEEE Transactions on Mobile Computing, 1 (2002), pp. 70–80.

[29] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton, *Sensor network-based countersniper system*, in 2nd ACM Conf. on Embedded Networked Sensor Systems (SenSys 2004), Baltimore, MD, Nov. 2004, pp. 1–12.

[30] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong, *A macroscope in the redwoods*, in 3rd ACM Conf. on Embedded Networked Sensor Systems (SenSys 2005), San Diego, CA, 2005, pp. 51–63.

[31] T. van Dam and K. Langendoen, *An adaptive energy-efficient MAC protocol for wireless sensor networks*, in 1st ACM Conf. on Embedded Networked Sensor Systems (SenSys 2003), Los Angeles, CA, Nov. 2003, pp. 171–180.

[32] L. van Hoesel and P. Havinga, *A lightweight medium access protocol (LMAC) for wireless sensor networks*, in 1st Int. Workshop on Networked Sensing Systems (INSS 2004), Tokyo, Japan, June 2004.

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

[34] ———, *Medium access control with coordinated, adaptive sleeping for wireless sensor networks*, IEEE/ACM Trans. on Networking, 12 (2004), pp. 493–506.

[35] W. Ye, F. Silva, and J. Heidemann, *Ultra-low duty cycle mac with scheduled channel polling*, in 4th ACM Conf. on Embedded Networked Sensor Systems (SenSys 2006), Boulder, CO, Nov. 2006, pp. 321–334.

[36] T. Zheng, S. Radhakrishnan, and V. Sarangan, *Pmac: an adaptive energy-efficient mac protocol for wireless sensor networks*, in 19th IEEE Int. Parallel and Distributed Processing Symposium (IPDPS 2005), Denver, CO, Apr. 2005.