ZeroCal: Automatic MAC Protocol Calibration

Andreas Meier, Matthias Woehrle, Marco Zimmerling, and Lothar Thiele

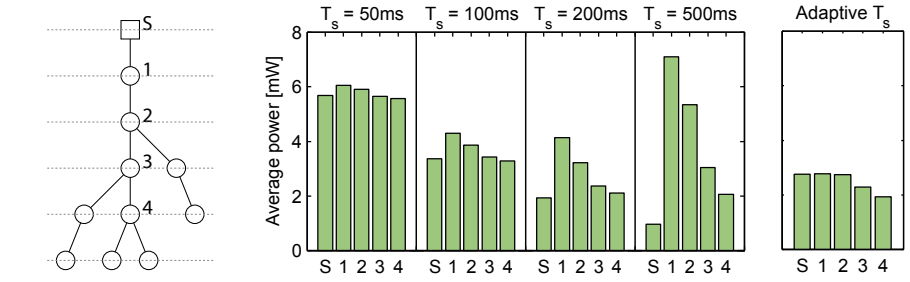
Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland
{a.meier,woehrle,zimmerling,thiele}@tik.ee.ethz.ch

Abstract. Sensor network MAC protocols are typically configured for an intended deployment scenario once and for all at compile time. This approach, however, leads to suboptimal performance if the network conditions deviate from the expectations. We present ZeroCal, a distributed algorithm that allows nodes to dynamically adapt to variations in traffic volume. Using ZeroCal, each node autonomously configures its MAC protocol at runtime, thereby trying to reduce the maximum energy consumption among all nodes. While the algorithm is readily usable for any asynchronous low-power listening or low-power probing protocol, we validate and demonstrate the effectiveness of ZeroCal on X-MAC. Extensive testbed experiments and simulations indicate that ZeroCal quickly adapts to traffic variations. We further show that ZeroCal extends network lifetime by 50% compared to an optimal configuration with identical and static MAC parameters at all nodes.

1 Introduction

The medium access control (MAC) protocol is the core component of the sensor network protocol stack. It is responsible for turning the radio device on and off at regular intervals. This duty-cycling functionality entails a fundamental trade-off between energy consumption and bandwidth: While the radio should be asleep as much as possible to save energy, sufficient bandwidth must be provided to achieve a target delivery rate. Numerous MAC protocols have been devised [5], but up to date it is unclear how to configure these protocols.

One possible approach is to define a network-wide trade-off at compile time; that is, the parameters of the MAC protocol are the same for all nodes and remain unchanged once the sensor network has been deployed. We call this an *identical MAC configuration*. Finding such a configuration is a nontrivial task, as it requires detailed knowledge about the protocol and the network conditions. Moreover, an identical configuration does not account for different traffic volumes at different nodes, *e.g.*, increased routing traffic towards the sink. To illustrate this consider Figure 1(b), showing the average power consumption of four different MAC configurations with identical parameters for the same topology and data rate. While the energy consumption varies considerably for different configurations, we also see that nodes closer to the sink consume more energy as they carry a higher load. With an identical MAC configuration the sink's one-hop neighbors will potentially run out of energy long before the other nodes.



(a) Topology (partial) (b) Identical parameter configurations (c) ZeroCal

Fig. 1. The MAC configuration with identical parameters results either in a very high or imbalanced power consumption. ZeroCal instead adapts the sleep interval T_s to minimize the maximum, which results in a well-balanced average power consumption. Results from testbed experiments with 21 nodes running X-MAC (see Section 5).

These energy bottlenecks greatly reduce the *network lifetime*, which is defined as the time until the first node runs out of energy. Furthermore, sensor networks operate in very dynamic environments. Channel characteristics change due to varying environmental conditions, traffic volume increases or decreases with the frequency of change in the sensed phenomena, and nodes are potentially added or removed. These dynamics are likely to render an identical MAC configuration inefficient over time, suggesting that nodes should continuously adapt their settings to maintain satisfactory performance.

We present ZeroCal, a Zero Configuration algorithm. ZeroCal is a distributed algorithm that automatically configures the MAC protocol at runtime according to the current traffic volume and network conditions (*e.g.*, packet loss, interference, and network topology). Using ZeroCal, each node decides autonomously on its own parameter setting, thereby trying to increase the network lifetime (*i.e.*, reduce the maximum energy consumption among all nodes). The algorithm is based on the observation that a node's MAC configuration does not only influence its own energy consumption but also the one of its children. For example, a node can decide to spend more energy for receiving messages (by waking up more often) in exchange for a reduced transmission energy at its child nodes.

ZeroCal tries to extend network lifetime. Hence, it looks at the maximum energy consumption of a node and its children: The MAC parameters are set such that this maximum is minimized. This results in a well-balanced energy consumption across the network, as shown in Figure 1(c). ZeroCal's configuration process repeats regularly and adapts to changes in traffic volume, independent of whether the change is due to a higher packet loss along an incoming link, an increased sending rate, or a change in the routing topology.

We make the following contributions: *i)* We present ZeroCal, a distributed algorithm that automatically configures the MAC protocol at runtime. Assuming many-to-one data traffic, ZeroCal is readily usable for any asynchronous low-

power listening or low-power probing protocol. *ii)* We validate ZeroCal on a testbed and in simulation, using X-MAC [2] as a case study. *iii)* We demonstrate that ZeroCal quickly adapts to traffic variations and extends network lifetime by 50% compared to an optimal, identical MAC configuration.

The remainder of this paper is organized as follows: Section 2 provides more background on sensor network MAC protocols and reviews related work on MAC protocol adaptation. Section 3 presents the design and underlying models of ZeroCal, which we validate in simulation in Section 4 and demonstrate on a testbed in Section 5. We conclude the paper in Section 6.

2 Background and Related Work

Low-power operation and energy management are of utmost importance. By putting the radio of a sensor node into sleep mode, the energy consumption can be reduced by orders of magnitude from mA to µA. Since the radio's wake-up time is rather short (<2 ms), it is possible to wake up the radio for only a very short time to exchange messages. This duty cycling is the primary task of the MAC protocol. A multitude of duty-cycling MAC protocols for sensor networks exist [5]. In the following, we briefly discuss two of the most prominent classes: *i)* low-power listening (LPL) and *ii)* low-power probing (LPP). Both Contiki and TinyOS feature default MAC protocols based on these two approaches. TDMA-based and slotted protocols are described elsewhere (*e.g.*, in [5]), and are outside the scope of this paper.

Using LPL nodes sleep most of the time and wake up regularly to quickly poll the channel. If a node detects a carrier, it keeps its radio on to receive a message; otherwise, it goes back to sleep. As nodes wake up asynchronously, the sender must transmit a preamble for a period slightly exceeding the sleep interval so that the receiver can detect the carrier. In X-MAC [2] the preamble is a sequence of short advertisement packets that contain the address of the receiver, as shown in Figure 2. After sending an advertisement, the sender listens for an acknowledgment from the receiver. If the sender hears an acknowledgment, it sends the data packet. LPP-based protocols (*e.g.*, RI-MAC [10]) take the inverse approach. Instead of polling the channel, nodes send an announcement when they are awake and subsequently listen for a data packet. The sender must wait for such an announcement from the intended receiver before it can send the data packet. LPP occupies the channel less than LPL because no long preambles are transmitted. The active period, however, is longer with LPP as it must accommodate an announcement and an incoming message.

LPL and LPP have one parameter in common: the sleep interval T_s. This is the time the radio is put to sleep between two active periods. The sleep interval greatly influences the average power consumption of a node, as illustrated in Figure 1(b). A short sleep interval reduces the energy consumption for sending messages but results in an increased energy consumption for polling the channel. Moreover, the sleep interval determines the available bandwidth. In fact, there

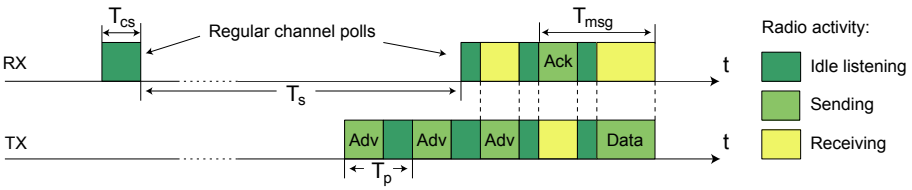


Fig. 2. Basic concept of LPL using the example of X-MAC. The receiver is sleeping most of the time, waking up every sleep interval T_s to poll the channel for T_{cs}. The sender transmits consecutive advertisement packets, waiting after each of them for an acknowledgment from the receiver that allows it to send the data packet.

exists an optimal sleep interval for a given traffic volume that minimizes energy consumption and provides sufficient bandwidth.

Polastre *et al.* [9] discuss the benefit of adapting B-MAC based on varying network conditions. They derive an analytical model of node lifetime and argue that other services could use this model to recompute check interval and preamble length. ZeroCal provides such a service and uses an energy model to optimize and adapt the MAC configuration as traffic volume changes.

Buettner *et al.* [2] adapt the sleep interval of X-MAC for a single sender-receiver pair. Using the estimated probability of receiving a packet, the receiver chooses its sleep interval such that the sum of transmit and receive energy is minimized. In contrast, ZeroCal works on any mesh or tree topology and respects the influence on the sender when choosing a new sleep interval at the receiver. More importantly, ZeroCal ensures that the sender's preamble is long enough so that it can be detected by the receiver. This reliability issue is inherent if nodes adapt their sleep interval autonomously but is not addressed in [2].

Jurdak *et al.* [4] propose a cross-layer framework for network-wide energy optimization and load balancing through greedy local decisions. We show that it is indeed beneficial not to optimize energy in a greedy fashion, but to consider both parent and children in the optimization. Merlin *et al.* [7] present a control-theoretic approach to adapt the duty cycle, which is however only suited for single-hop topologies.

3 ZeroCal

Asynchronous low-power listening and low-power probing MAC protocols feature a trade-off between bandwidth and energy consumption for a specific node and all nodes communicating with it. In data collection, one can either have the parent node spending a lot of energy for idle listening or the child nodes for sending messages. ZeroCal extends network lifetime by minimizing the maximum energy consumption of each parent-children pair in the network. To this end, ZeroCal optimizes the sleep interval of a parent at runtime based on its traffic volume.

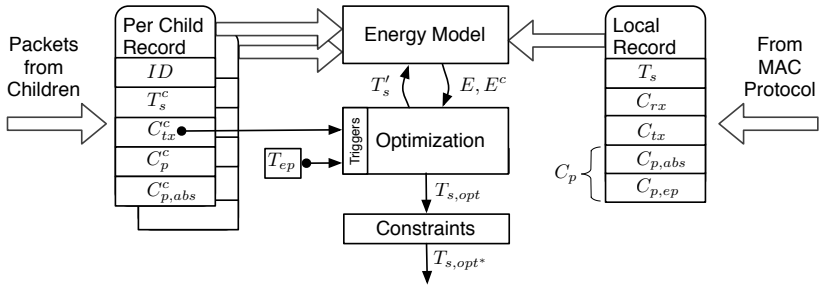


Fig. 3. ZeroCal architecture. ZeroCal keeps records of local and child MAC parameters. Periodically, an optimization process is triggered. Using an energy model, a new optimal sleep interval T_{s,opt^*} is computed respecting bandwidth and protocol constraints.

Since every parent is also a child (except for the sink), this adaptive approach results in a well-balanced energy consumption across the whole network.

ZeroCal adapts the MAC configuration as illustrated in Figure 3. The optimization uses an energy model (see Section 3.3) that is based on records collected from the local MAC and the child nodes (see Section 3.2). The optimization explores the effect of a new sleep interval T_s' on the maximum energy consumption of the node-children pair and returns an optimal sleep interval $T_{s,opt}$. Furthermore, the MAC protocol itself imposes certain constraints on its parameters, which require to adapt $T_{s,opt}$ to $T_{s,opt*}$.

ZeroCal performs the optimization over time windows, called *epochs*. The optimization is triggered when either the transmission count of a child node C_{tx}^c exceeds the threshold C_{eval} or the *epoch time* T_{ep} exceeds the maximum duration of an epoch $T_{ep,max}$. The traffic-dependent trigger is needed to react to a (sudden) increase in traffic volume, whereas the timed trigger ensures that ZeroCal periodically updates the MAC configuration when only few or no messages arrive, *e.g.*, at leaf nodes.

3.1 Parameter Optimization

We now describe how ZeroCal uses the energy estimation to determine an optimal MAC configuration at runtime.

In general, a node can save energy by increasing the length of its sleep interval T_s. Indeed, it could sleep as long as possible to maximize its own lifetime. Such an approach is however very selfish, since it increases the energy consumption for sending messages at child nodes. This raises the question whether it is more important to save energy for oneself or at the child nodes. Since the goal is to prolong the lifetime of both the parent and its children, ZeroCal chooses the sleep interval $T_{s,opt}$ at the parent such that the maximum of the parent's energy

consumption E and of its children E^c is minimized:

$$T_{s,opt} = \underset{T'_s \in [T_{s,min}, T_{s,max}]}{\operatorname{argmin}} \max\left[E, \max_{\forall \text{ children } c}(E^c)\right]. \quad (1)$$

Additionally, ZeroCal has to ensure that the parent is able to detect the preamble of its children. Therefore, the parent's sleep interval $T_{s,opt*}$ must be chosen to be shorter or equal to the longest sleep interval of its child nodes:

$$T_{s,opt*} \leq T_s^c, \quad \forall \text{ children } c\,. \quad (2)$$

Moreover, the sleep interval also limits the available bandwidth at a node. Therefore, it might be necessary to further reduce the sleep interval $T_{s,opt*}$ to increase the bandwidth. For instance, by requiring

$$T_{s,opt*} \leq 1/(C_{tx} + C_{rx})/n\,, \quad (3)$$

we ensure that at most in every n-th sleep interval a packet is being sent or received.

3.2 Collecting MAC Statistics

To compute up-to-date energy estimates during the optimization task, each node keeps a record of counters and sleep intervals of itself and all its children, as shown in Figure 3. Locally the following information is available: The number of sent C_{tx} and received messages C_{rx}, the current sleep interval T_s, and the epoch time T_{ep}. The number of transmitted preamble packets C_p is determined by the absolute preamble count $C_{p,abs}$ in reference to the one at the beginning of the epoch $C_{p,ep}$ via $C_p = C_{p,abs} - C_{p,ep}$.

From the child node the number of sent messages C^c_{tx} is readily available as it corresponds to the number of messages received from the child. Here, we neglect lost data packets along incoming links. We further simplify and set $C^c_{rx} = C^c_{tx}$, approximating that child nodes are pure forwarders and do not generate messages themselves. The sleep interval T^c_s and the absolute preamble count $C^c_{p,abs}$ are piggybacked on data packets. Since we are interested in the number of preamble packets sent during the current epoch, we additionally maintain the preamble count C^c_p that is updated at every packet reception: Assuming a node received the k-th packet in an epoch (*i.e.*, the received absolute value is $C^c_{p,abs}(k)$ and the previous one locally stored is $C^c_{p,abs}(k-1)$), we can determine the difference $\Delta C^c_p = C^c_{p,abs}(k) - C^c_{p,abs}(k-1)$ and subsequently update the preamble count $C^c_p(k) = C^c_p(k-1) + \Delta C^c_p$. After running the optimization, a node resets the epoch time and all counters, except for its local preamble count $C_{p,abs}$ which is stored in $C_{p,ep}$. The overhead for collecting estimation information is low. In our current implementation, a parent reserves 9 bytes per child, and 3 additional bytes are piggybacked on each data packet.

3.3 Energy Model

ZeroCal needs to estimate a node's energy consumption given its MAC configuration. In this work, we use a refined estimation method based on our previous work [6]. We only consider the energy consumption of the radio device, which is a reasonable assumption since the radio is usually the main consumer of energy in the system.

The radio is active while performing regular channel polls T_{cp}, transmitting messages T_{tx}, and receiving messages T_{rx}. We opt to neglect the effect of interference, assuming that the impact on the overall energy budget is minimal compared to T_{cp}, T_{tx}, and T_{rx}. Given these times and the corresponding average power consumptions, we can estimate a node's energy consumption using

$$E = T_{tx} \cdot P_{tx} + T_{rx} \cdot P_{rx} + T_{cp} \cdot P_{cp} \,. \tag{4}$$

We note that P_{tx}, P_{rx}, and P_{cp} are not the power levels of the radio device in transmit, receive, and idle mode. Instead, they correspond to the average power consumption of the logical states of the MAC protocol: sending, receiving, and channel polling. For instance, while sending a message (preamble stream, acknowledgment reception, plus data packet transmission), the radio switches several times between transmit and receive mode. The average power levels depend on hardware platform and protocol implementation, and are measured offline.

We estimate the residence times in the different protocol states using the sleep interval T_s, the number of received C_{rx} and sent messages C_{tx}, and the number of transmitted preamble packets C_p:

$$\begin{aligned} T_{tx} &= C_p \cdot T_p + C_{tx} \cdot T_{msg} \,, \\ T_{rx} &= C_{rx} \cdot T_{msg} \,, \\ T_{cp} &= (T_{ep} - T_{tx} - T_{rx}) \cdot T_{cs} / (T_s + T_{cs}) \,, \end{aligned} \tag{5}$$

where T_p, T_{cs}, and T_{msg} are constants specific to the radio device and the MAC protocol as illustrated in Figure 2. The time spent for sending messages depends on the number of transmitted preamble packets and the number of sent data packets. Sending a data packet takes T_{msg}, which includes the actual data packet, its acknowledgment, and the radio switching times. For every received message, the radio is active for T_{msg}. If there is no traffic, nodes wake up regularly every T_s and go back to sleep after T_{cs}. The length of T_{cs} also includes the time for switching the radio between sleep and receive mode.

The energy estimation according to Equation 5 depends on the current sleep interval T_s; choosing a new sleep interval T_s' affects the times spent in the different logical states of the MAC protocol at both the parent and its child nodes. Assuming a linear relation between the number of preamble packets and the parent's sleep interval, ZeroCal estimates the new times as follows:

– At the parent, the times for receiving T_{rx} and transmitting T_{tx} do not change, whereas the new channel-polling time T_{cp} is given by

$$T_{cp} = (T_{ep} - T_{tx} - T_{rx}) \cdot T_{cs} / (T_s' + T_{cs}) \,. \tag{6}$$

– At the child nodes, only the time for transmitting messages T_{tx}^c is affected:

$$T_{tx}^c = C_p^c \cdot T_p^c \cdot T_s'/T_s + C_{tx}^c \cdot T_{msg}\,. \tag{7}$$

To validate the accuracy of our modeling, we run simulations in Castalia [8] on a binary tree topology (see Section 4 for details) and compare the estimated energy with the energy measured by the simulator. We observe that for both low and high data rates the energy consumption is well estimated across all simulated MAC configurations: The maximum estimation error ranges between 1.8% (low data rate) and 4.1% (high data rate).

Our energy estimation model is general enough to accommodate other MAC protocols as well. For example, to adapt the model to a LPP-based protocol (*e.g.*, RI-MAC [10]), we have to replace the preamble counter C_p in Equation 5 with a counter that keeps track of the number of time intervals (with length T_p) a node waits for an announcement from the receiver.

3.4 System Integration

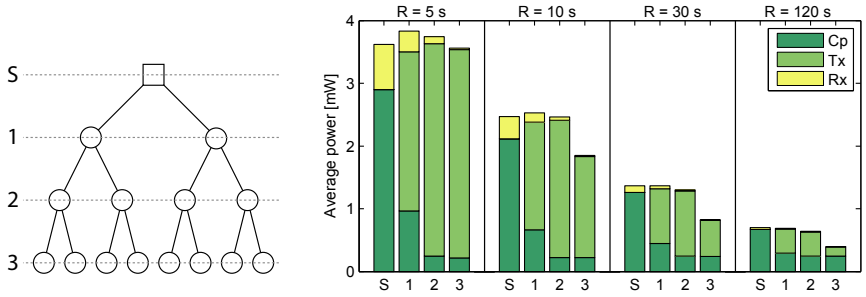
As for the routing, we assume many-to-one data traffic that flows toward a common sink node. On each intermediate node, messages are forwarded to a parent that is closer to the sink with respect to some routing metric (*e.g.*, hop count or ETX [3]). Our approach works on any mesh and tree topology. Furthermore, we do not require a dedicated interface between the MAC and routing layers. However, we assume that unicast transmissions take place only between a child and its parent. ZeroCal optimizes for the current amount of traffic, which typically differs across the network. Hence, ZeroCal's operation is independent of whether traffic originates from local data generation or packet forwarding.

In sensor networks, the sink node typically has unlimited power supply from a base station, which is responsible for additional tasks, such as providing a back channel via GPRS to a central monitoring system [1]. This setup allows for a so-called *always listening sink* that runs a 100% duty cycle. ZeroCal adapts to this scenario: The always listening sink reduces the average transmission time of its one-hop neighbors to a minimum, and the repeated execution of the algorithm propagates this benefit down to the leaf nodes, eventually reducing the energy consumption of all nodes.

4 Simulation

We implement ZeroCal and X-MAC in Castalia [8], a state-of-the-art sensor network simulator. Castalia features a detailed model of the radio device that also accounts for the transition times between the different operational modes and their individual power consumptions. Furthermore, Castalia provides a realistic model of the wireless channel with random packet loss and interference.

For illustration purposes, we use a perfect binary tree of depth 3 in our simulations, as shown in Figure 4(a). In the binary tree, we have one sink with



(a) Simulation topology (b) Average power consumption with ZeroCal

Fig. 4. Simulation with binary tree topology: ZeroCal shows a well balanced average power consumption within the network. For every hop, the node with the highest average power consumption is depicted.

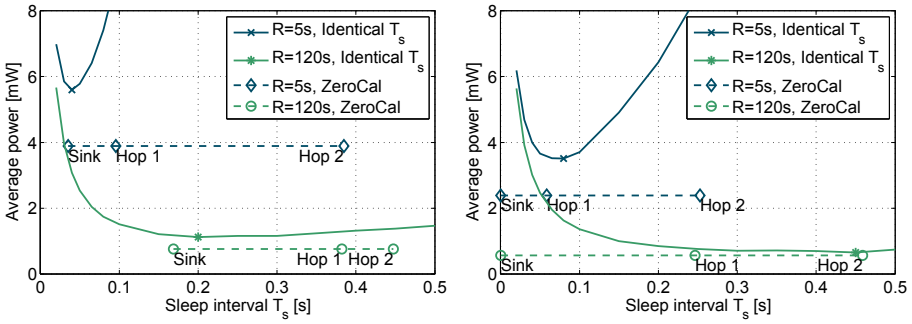
two children (level 1), four grandchildren (level 2), and eight leaf nodes (level 3). Child nodes interfere with each other, and a grandparent is a hidden terminal for a child. We use Castalia's CC2420 radio model and set the packet reception rate of all links to 90% to see whether ZeroCal can cope with packet loss.

Every node but the sink samples and generates data packets with an inter-packet interval R, ranging from 5 s (high data rate) to 120 s (low data rate). To study the long-term behavior of ZeroCal, we run simulations that correspond to one day in real time. We use the following inputs for the parameter optimization: $T_{s,\min} = 20\,\text{ms}$, $T_{s,\max} = 500\,\text{ms}$, $T_{ep,max} = 500\,\text{s}$, $C_{eval} = 50$ packets, and $n = 3$ sleep intervals. Unless otherwise stated, the sink is duty cycled.

4.1 Adaptive Behavior

We first look at the adaptive behavior of ZeroCal for different traffic volumes. Figure 4(b) plots for each level in the binary tree (*i.e.*, hop-distance from the sink) the maximum energy consumption. We see that ZeroCal achieves a well-balanced energy consumption across all levels; the remaining differences are due to the upper bound on the sleep interval ($T_{s,max} = 500\,\text{ms}$). For instance, at a sampling interval of $R = 5\,\text{s}$ the energy consumptions differ only by 7.5% *among all nodes*. This demonstrates ZeroCal's capability to evenly distribute the workload across the entire network.

In fact, ZeroCal gradually aligns the energy consumptions by distributing the *type* of workload. To see this, we take a closer look at the composition of energy consumption at different levels in Figure 4(b): The portion of transmit (Tx) energy increases with hop-distance, whereas the portions of receive (Rx) and channel-polling (Cp) energy decrease. For example, the nodes on level 3 spend relatively more transmit energy than the nodes on level 1, even though they send seven times fewer messages. The rationale behind this behavior is justified by the following reasoning. The sink does not send any messages and can therefore



(a) Duty-cycled sink (b) Always listening sink

Fig. 5. Average power consumption for the network-wide, static (solid lines) and ZeroCal's adaptive (dashed lines) sleep interval. ZeroCal outperforms all static configurations by choosing different sleep intervals at different hop-distances.

invest more energy into frequent wake-ups and receiving ($T_s^S = 35\,\text{ms}$). This in turn reduces the transmission energy for the nodes on level 1, which choose a shorter sleep interval ($T_s^1 = 96\,\text{ms}$) than the nodes on level 2 ($T_s^2 = 385\,\text{ms}$). Finally, the nodes on level 3 select the longest possible sleep interval ($T_s^3 = 500\,\text{ms}$). ZeroCal's periodic parameter optimization results in a step-by-step adaptation toward this optimal point of operation, as changes in the MAC configuration on one level eventually propagate to all other levels in the tree.

4.2 Static versus Adaptive Configuration

We now want to quantify the benefit of ZeroCal. What do we gain from the adaptation as compared to an identical MAC configuration?

With an identical configuration, we face the problem of finding the optimal MAC parameters in advance, which depend on application and deployment characteristics, such as sampling interval, routing topology, and interference. The best we can do is to carry out preliminary experiments on the deployment site to make an educated guess on these ever-changing variables.

Figure 5(a) highlights the importance of choosing the right configuration. The solid lines show the average power consumption as a function of the sleep interval for both a high ($R = 5\,\text{s}$) and a low data rate ($R = 120\,\text{s}$), again on the perfect binary tree. The optimal sleep intervals for an identical configuration correspond to the minima indicated by markers. We see that the curve for the high data rate is very steep; that is, our system would be very inefficient for high traffic even if we are only slightly off the optimal sleep interval. This dependency is more moderate for low traffic. Additionally, we have to consider that the sleep interval limits the available bandwidth. For instance, if we choose a sleep interval longer than 100 ms for the high data rate, packets are potentially lost as nodes are overloaded by incoming traffic. Overall, finding a suitable identical configuration is critical and hard to achieve.

ZeroCal outperforms all identical MAC configurations in our simulations. This can also be seen in Figure 5, where we show ZeroCal's maximum energy consumption with a dashed line. Furthermore we indicate the average sleep intervals chosen by ZeroCal at each hop-distance. For a duty-cycled sink in Figure 5(a), the adaptive sleep interval reduces the maximum energy consumption by 30.6% ($R = 5\,\text{s}$) and 32.7% ($R = 120\,\text{s}$) compared with the corresponding optimal, identical configuration. If we have an always listening sink, we see in Figure 5(b) that ZeroCal adapts well to this situation: the energy savings are 32.0% ($R = 5\,\text{s}$) and 13.3% ($R = 120\,\text{s}$) in comparison to the static configuration. We note again that ZeroCal distributes the workload evenly, avoiding energy hot spots that would otherwise limit the network lifetime.

Looking at the sleep intervals chosen by ZeroCal, we observe that they are shorter for an always listening sink than for a duty-cycled sink (*i.e.*, nodes save energy while polling the channel more frequently). This might surprise at first, but is explained by the fact that the always listening sink allows the sink's one-hop neighbors to save a lot of transmit energy. ZeroCal's periodic adaptation propagates these energy savings eventually to the nodes farther away, which can then reduce their sleep interval to wake up more frequently.

4.3 Large and Irregular Topologies

We also perform simulations on large and irregular topologies to see whether ZeroCal scales. ZeroCal automatically optimizes for the subtree with the highest data load: the corresponding nodes show a well-balanced energy consumption and hence the network lifetime is maximized. The nodes in subtrees with less traffic also balance their energy consumption, yet on a lower energy level.

For instance, with 100 nodes on an area of 160 by 160 meters and up to 8 hops to the sink, we observe the same trend as with the binary tree topology. For high, medium, and low data rates, ZeroCal reduces the maximum energy consumption by 28.7%, 33.7%, and 33.5% compared to an optimal, identical MAC configuration. The energy consumption averaged over all nodes is reduced by 61.7%, 55.3%, and 51.2%, which indicates that ZeroCal optimizes the energy consumption for nodes with little traffic even further.

5 Testbed Experiments

To demonstrate the effectiveness of ZeroCal on real sensor nodes, we implement ZeroCal in Contiki on top of X-MAC. We run experiments on a testbed of 21 Tmote Sky nodes deployed over several offices. We use an irregular tree topology with a maximum hop distance to the sink of 5, as shown in Figure 1(a). We fix the topology to be able to compare the results of different runs. However, there are two types of network dynamics that affect ZeroCal's operation: *i)* varying packet loss due to various sources of interference (*e.g.*, WLAN and moving people), and *ii)* varying traffic volume as nodes change their sampling rate at runtime.

ZeroCal performs an exhaustive parameter search on the sleep interval during the parameter optimization, using the same settings as in our simulations. To speed up the process at nodes with high incoming degree, ZeroCal considers only two of their children during the search: the one with the lowest sleep interval and the one with the highest preamble count. These two nodes are likely to carry the highest energy load among all children and thus matter most when minimizing the maximum energy consumption. Furthermore, ZeroCal uses an adaptive granularity during the search. For a sleep interval close to the minimum $T_{s,min} = 20\,\text{ms}$ ZeroCal makes steps of 4 ms, whereas for a sleep interval close to the maximum $T_{s,max} = 500\,\text{ms}$ it evaluates in steps of 30 ms. We find that the computational speed-up outweighs the slight loss in accuracy.

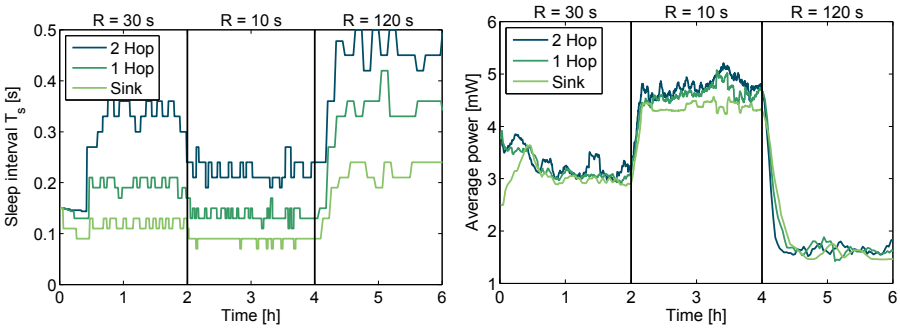
5.1 Static versus Adaptive Configuration

In a first series of experiments, we compare ZeroCal with an identical X-MAC configuration. All nodes generate data packets with a constant inter-packet interval of $R = 30\,\text{s}$. As shown in Figure 1, ZeroCal outperforms the optimal, identical setting ($T_s = 200\,\text{ms}$) by 32.6%, which translates in an increase of 48.4% in network lifetime. We see that ZeroCal achieves a well-balanced energy consumption across all hop-distances. This indicates that ZeroCal configures X-MAC nearly optimal. We also see in in Figure 1(b) that Contiki's default X-MAC configuration ($T_s = 500\,\text{ms}$) can lead to poor performance. If an inexperienced user simply uses the default settings, radio communication requires 60.8% more energy in comparison to ZeroCal's adaptive configuration.

5.2 Adaptation to Network Dynamics

Finally, we analyze how ZeroCal adapts to varying network conditions, which is impossible for an identical MAC configuration. To this end, we let nodes change their sampling interval dynamically every two hours. Starting with a medium rate ($R = 30\,\text{s}$), nodes switch after two hours to a high rate ($R = 10\,\text{s}$), followed by another change to a low rate ($R = 120\,\text{s}$) after four hours. At the same time, sporadic packet losses occur.

Figure 6(a) shows how ZeroCal adapts X-MAC's sleep interval at different hop-distances over time, and Figure 6(b) shows the corresponding trend in the average power consumption. At the beginning, all nodes have the same sleep interval ($T_s = 150\,\text{ms}$). While the sink quickly adapts to a stable T_s, it takes about 25 minutes until the two-hop neighbors adapt their T_s. This is because a node is not allowed to have a longer T_s than its children, as enforced by Equation 2. Hence, the adaptation from the initial (far from optimal) T_s toward a longer T_s starts at the leaf nodes and then propagates step-by-step upward in our tree topology of depth 5 (see Figure 1(a)). Conversely, adapting to the high traffic volume after two hours is very fast. First, the optimization task is performed more often at high data rates and, second, there are no constraints when reducing T_s. Note that the adaption latency mainly depends on the duration of an



(a) Sleep interval (b) Average power over last 10 minutes

Fig. 6. Testbed experiments with changing data rates. ZeroCal adapts especially quickly if the data rate increases and bandwidth is required.

epoch, which we set conservatively ($T_{ep,max} = 500\,\mathrm{s}$) to avoid system instabilities. The variations in T_s after adapting to a new data rate are mainly inflicted by external effects, such as packet loss and interference.

Looking at the evolution of power consumption in Figure 6(b), we note that ZeroCal distributes the load evenly as network conditions change. Only at $R = 10\,\mathrm{s}$, we note a 7% difference in energy consumption between the sink and the other nodes. This is because our energy estimation model ignores effects like overhearing and collisions, which happen more frequently at high data rates. However, as sensor networks typically sample at low data rates, we prefer a simpler model inducing less overhead. When the traffic volume drops, there is a slight adaptation latency. More importantly, however, ZeroCal quickly adapts to a (sudden) increase in traffic volume when additional bandwidth is needed while keeping the energy consumption as low as possible.

6 Conclusions

There has been considerable research on sensor network MAC protocols. Until now, their proper configuration has been mostly neglected. While network dynamics make it hard to define an appropriate configuration at deployment time, we showed that the MAC configuration has indeed a wide impact on energy consumption and available bandwidth. Particularly if expert knowledge is missing, the MAC protocol is often used with its default parameters, which can result in very poor energy efficiency. In the worst case, application-specific bandwidth and lifetime requirements are not satisfied.

To tackle these challenges, we proposed ZeroCal, a distributed algorithm that automatically configures the MAC parameters at runtime in order to increase the network lifetime. The configuration is repeated regularly and ensures that the nodes adapt to variations in traffic volume.

We validated ZeroCal on X-MAC in simulation and testbed experiments. ZeroCal quickly adapts toward a stable operating point, in particular when additional bandwidth is needed. Moreover, ZeroCal outperforms an optimal, identical MAC configuration, extending network lifetime by about 50%.

Acknowledgements

The work presented in this paper was supported by 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. We thank Koen Langendoen and Thiemo Voigt for their feedback on draft versions of this paper.

References

1. J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yuecel. PermaDAQ: A scientific instrument for precision sensing and data recovery in environmental extremes. In *Proc. 8th Int'l Conf. Information Processing Sensor Networks (IPSN '09)*, pages 265–276, San Francisco, CA, USA, Apr. 2009. ACM/IEEE.
2. M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proc. 4th ACM Conf. Embedded Networked Sensor Systems (SenSys 2006)*, pages 307–320, New York, NY, USA, 2006. ACM Press.
3. D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4):419–434, 2005.
4. R. Jurdak, P. Baldi, and C. V. Lopes. Adaptive low power listening for wireless sensor networks. *IEEE Transactions on Mobile Computing*, 6:988–1004, 2007.
5. K. Langendoen. Medium access control in wireless sensor networks. In H. Wu and Y. Pan, editors, *Medium Access Control in Wireless Networks*, pages 535–560. Nova Science Publishers, Inc., may 2008.
6. K. Langendoen and A. Meier. Analyzing MAC protocols for low data-rate applications. In *ACM Transactions on Sensor Networks*, accepted for publication, 2010.
7. C. J. Merlin and W. B. Heinzelman. Duty cycle control for low-power-listening MAC protocols. In *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*, pages 497–502, Sept./Oct. 2008.
8. H. N. Pham, D. Pediaditakis, and A. Boulis. From simulation to real deployments in WSN and back. In *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pages 1–6, June 2007.
9. J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys 2004)*, pages 95–107. ACM Press, New York, 2004.
10. Y. Sun, O. Gurewitz, and D. Johnson. RI-MAC: A receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *Proc. 1st ACM Conf. Embedded Networked Sensor Systems (SenSys 2003)*, pages 1–14, Raleigh, NC, Nov. 2008.