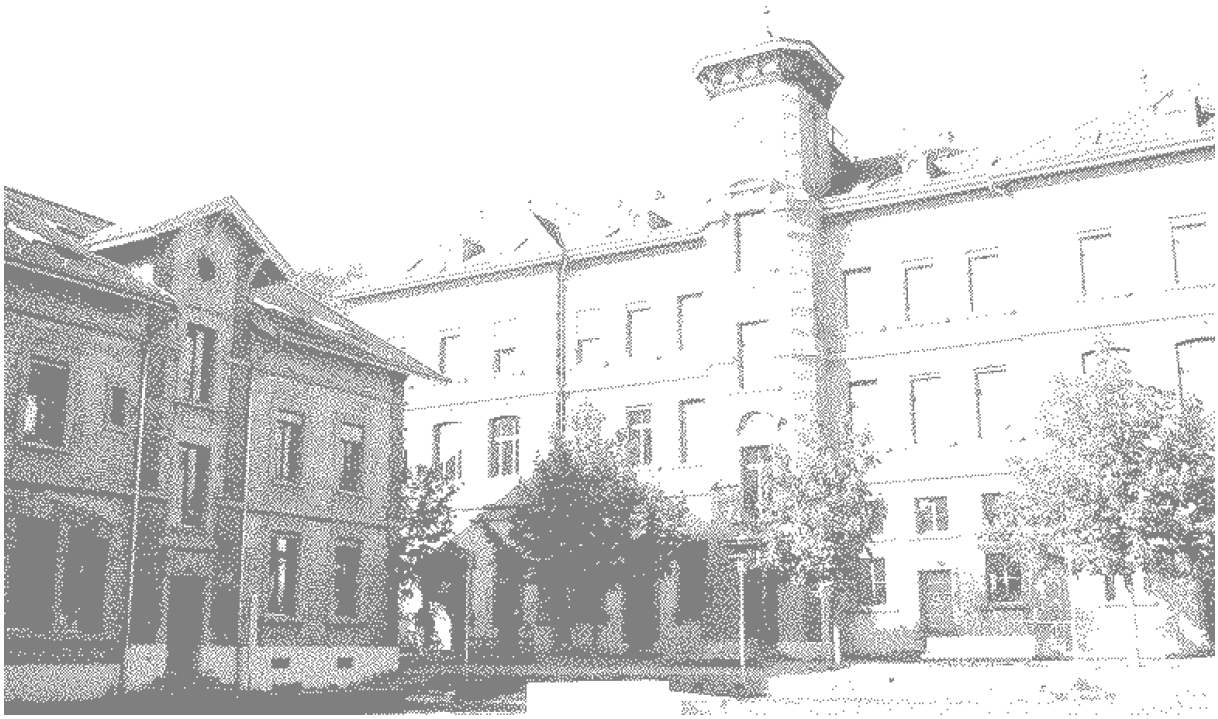
UNIVERSITÄT
BERN

Real-World Experiences with the Maximally Traffic-Adaptive Medium Access Control Protocol

P. Hurni, T. Braun

Technischer Bericht IAM-11-001 vom 29.04.2011

Institut für Informatik und angewandte Mathematik, www.iam.unibe.ch



Real-World Experiences with the Maximally Traffic-Adaptive Medium Access Control Protocol

Philipp Hurni, Torsten Braun
Institute of Computer Science and Applied Mathematics
University of Bern, Neubrueckstrasse 10, CH-3012 Bern, Switzerland
{hurni, braun}@iam.unibe.ch

***Abstract*—Energy-Efficient Medium Access (E^2-MAC) protocols duty cycling the radio transceiver typically trade off energy-efficiency versus classical quality of service (QoS) parameters (throughput, latency, reliability). Today's E^2-MAC protocols are able to deliver little amounts of data with a low energy footprint, but introduce severe restrictions with respect to throughput and latency. Regrettably, they yet fail to adapt to varying traffic load at run-time.**

This paper studies the Maximally Traffic-Adaptive MAC protocol MaxMAC, which targets at achieving maximal adaptability to variable traffic conditions at run-time. By thoroughly analyzing the performance of MaxMAC in a series of distributed experiments, the paper conveys that under variable traffic conditions, MaxMAC effectively combines the advantages of energy unconstrained CSMA (high throughput, high PDR, low latency) with those of classical E^2-MAC protocols (high energy-efficiency).

***Index Terms*—Wireless Sensor Networks, Energy Efficient Medium Access Control, Traffic Adaptivity**

I. Introduction

E^2-MAC protocols are widely adopted in today's sensor networks communities. With E^2-MAC layers turning off the radio transceiver for the major portion of time, the energy consumption can often be decreased by 90% or even more. The hidden cost of the increase in efficiency however comes with deteriorating packet latency, throughput and reliability, which can be seen as a limiting factor for the applicability of sensor networks. Regrettably, most E^2-MAC protocols generally fail to adapt to varying traffic load at run-time.

Numerous event detection and alarming applications require a high energy-efficiency during long periods of inactivity, and reasonable good quality of service during short periods of increased activity. Such scenarios can be found e.g. in monitoring systems for healthcare [1][2], in Disaster-Aid-Systems [3], but also in the broad area of (event-based) environmental monitoring systems. Last but not least, a reasonable QoS also massively facilitates manual real-time interaction with WSNs: with E^2-MAC protocols massively increasing the end-to-end latencies, simple tasks such as disseminating queries to remote nodes or transmitting a large chunk of code/data from or to a particular sensor node can become cumbersome.

In [4], we have presented MaxMAC, an E^2-MAC protocol that achieves maximal adaptivity with respect to throughput and latency at run-time. [4] explores the design space of contention-based E^2-MAC protocols in simulation, and compares MaxMAC against a selection of today's most widely known contention-based E^2-MAC protocols. This paper thoroughly evaluates our prototype implementation of the MaxMAC protocol and compares it against its non-adaptive counterpart WiseMAC [5] and energy-unconstrained IEEE 802.11-like CSMA, using real-world implementations in a controlled indoor WSN testbed. We introduce into related work in the field of traffic-aware MAC in Section II. Section III discusses the basic concept of MaxMAC. Sections IV and V discuss the experiment platform and prototype implementation. In Section VI, we explore MaxMAC's advantages and drawbacks in a series of real-world sensor network experiments, and show that the MaxMAC concept effectively reaches the QoS characteristics of energy-unconstrained CSMA in high-traffic phases, while still exhibiting a high energy-efficiency in periods of sparse traffic. Section VII concludes the paper.

II. Related Work

With most E^2-MAC protocols offering offering little solutions with respect to QoS awareness, researchers have started to work on *traffic-adaptive* E^2-MAC protocol variants in the past few years, in order to achieve a high efficiency during the major part of the time, *but* being able to deliver high QoS (low packet latency, high throughput and reliability) *on demand*. A couple of concepts have yet been applied to reach traffic-adaptive protocol behavior in today's literature on E^2-MAC protocols. However, most approaches are minor variations of existing protocols and still heavily restrain throughput and latency of the MAC layer, a crucial disadvantage which often limits the possibilities in real-world WSN deployments.

T-MAC [6] increases the traffic-adaptivity of S-MAC [7] by prolonging the duty cycles of the nodes when so-called activation events occur. An activation event may be the sensing of any communication in the neighborhood, the end of the own data transmission or acknowledgement, the overhearing of RTS or CTS control messages that may announce further packet exchanges.

X-MAC [8] is an E^2-MAC protocol based on asynchronous listen-intervals. For each packet, X-MAC transmits a strobe of preambles, in between which the receiver can signal reception-readiness with a so-called *EarlyACK*. [8] derives a formula for optimal wake/sleep intervals and given traffic at a certain rate. It outlines a mechanism to let X-MAC adapt the duty cycle and the sleep/wake interval to best accommodate the traffic load in the network. With the basic mechanism of X-MAC still requiring a certain minimal interval between two active intervals and a generally high per-packet overhead, the maximum throughput still remains limited.

ZeroCal [9] is an optimization framework on top of the Contiki [10] X-MAC layer, which chooses the X-MAC parameter settings for the wake-up interval at run-time instead of compile-time. The current settings are periodically recalcu-

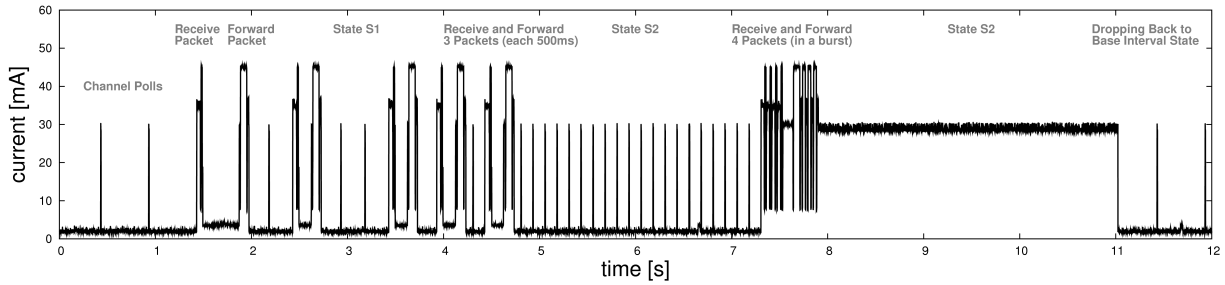


Fig. 1: MaxMAC node forwarding packets at different rates and switching between Base Interval State, S_1, S_2, S_{CSMA}

lated, however not propagated to neighboring nodes.

AMAC [11] relies on the S-MAC active period structure consisting in SYNC, RTS and CTS windows. With low traffic, AMAC neglects the costly RTS/CTS exchange and operates with a large sleep interval between two active periods. With increasing traffic, it multiplies the amount of active periods by a factor of 2^n, thus increasing the net duty cycle by the same factor. Applying this adaptation strategy, the protocol can prevent packet drops to some extent while still conserving energy. However, especially for low traffic, the SMAC active period still wastes a considerable amount of energy, compared to preamble-sampling approaches.

III. The MaxMAC Protocol

With the MaxMAC concept [4], we have recently proposed an E^2-MAC protocol that discovers fluctuations in traffic load and immediately reacts upon them. The protocol is based on established design principles for E^2-MAC protocols developed over the last decade. In its default operation mode, the protocol applies asynchronous contention-based preamble sampling, as introduced in B-MAC [12] and WiseMAC [5]. With increasing traffic load, MaxMAC stepwise shifts its primary objective from the conservation of energy towards provisioning of good quality of service (high throughput, low delay) rather than saving energy.

A. Basic Media Access Mechanism - Preamble Sampling

MaxMAC uses asynchronous preamble sampling for the basic medium access. Nodes keep their radios turned off for most of the time, and only wake up for brief periodic duty cycles to poll the channel for a preamble signal. The sender node transmits a preamble for each frame that signals the upcoming frame transmission to the receiver. MaxMAC learns the wake-up schedules of its neighbors to minimize the length of the preambles in future transmissions - a small preamble only compensates for the maximum clock drift that the two involved nodes' clocks may have developed during the time since the last schedule exchange - a higly effective optimization technique to minimize the transmitted preamble introduced in WiseMAC [5]. The good performance and high efficiency of WiseMAC under low and variable traffic has been independently pointed out in [13] and [4][14]. Among the preamble-sampling-based E^2-MAC protocols, it must be seen as the most efficient approach. The recent survey of nine E^2-MAC protocols [13] points out that "the WiseMAC protocol showed a remarkable consistent behavior across a wide range of operational conditions, always achieving the best or second-best performance". The WiseMAC preamble optimization technique has been recently adopted by ContikiMAC [15], which is replacing X-MAC [8] as the Contiki default MAC layer.

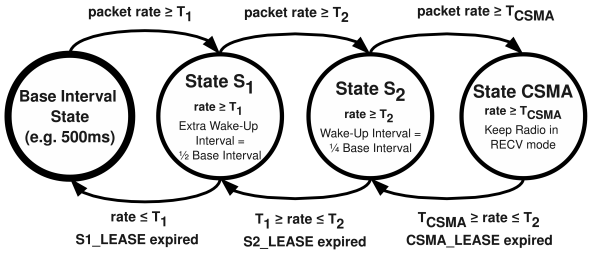


Fig. 2: MaxMAC FSM-based Traffic Adaptivity Concept

B. Run-time Adaptation Mechanisms

MaxMAC introduces adaptation features to instantly react to changing load conditions by altering its behavior at run-time. MaxMAC allocates the energy resources of the sensor node in an *on-demand* manner, letting nodes change their state (and their behavior) and allocate so-called *Extra Wake-Ups* when the rate of incoming packets reaches predefined threshold values, denoted as T_i. The MaxMAC adaptivity concept is depicted in Figure 2 using the three states S_1, S_2 and CSMA, the number of states and thresholds can however be chosen arbitrarily. The threshold values T_1, T_2, T_{CSMA} were chosen to suit for the targeted behavior of our MaxMAC prototype on the MSB430 sensor node platform. As in any MAC protocol, manipulating the threshold values allow the network operator for fine-tuning the MaxMAC protocol and its targeted behavior and performance - e.g., choosing lower values would make sense in delay-sensitive applications. Figure 1 displays an excerpt from the current trace (sampled at 1000 Hz) of a node running MaxMAC and receiving and forwarding packets with increasing rate. Two packets are received and subsequently forwarded at t=1.5s and t=2.5s. Further three packets are received and forwarded at t=3.5s, t=4s, t=4.5s and a burst of 4 packets in a burst is received and forwarded starting at t=7.25s. The node measures the received rate of packets using a sliding window over 1 second.

As illustrated in Figure 1, MaxMAC operates in the *Base Interval* state per default, polling the channel periodically within the Base Interval T. It switches to the states S_1, S_2 when the corresponding load thresholds T_1, T_2 are reached. With the rate of incoming packets reaching the threshold T_1, the node switches to the state S_1 and schedules one additional *Extra Wake-Up* in-between each Base Interval, effectively doubling the amount of duty cycles over time. This state change is well visible in Figure 1 after t=2s, after the node has forwarded the received packet. With exceeding T_2, it switches to the state S_2 and again doubles its wake-up frequency. State changes provoking an

increased wake-up frequency are immediately signaled in the ACK frame. The receiving node thereby *promises* to remain in the new state and keep its increased wake-up frequency for a predefined timespan S1_LEASE. The MaxMAC LEASE timespans (S1_LEASE, S2_LEASE, CSMA_LEASE) define how long a node remains in the new state when signaling the state change in the ACK. LEASE timespans can be *prolonged* with every new ACK transmission. With using the LEASE timespans, fast oscillation between the different states can be mitigated. When the LEASE timespans expire, nodes having received prior state change announcements will assume that the corresponding node has fallen back to its default behavior (polling the channel with the Base Interval T), which prevents them from transmitting at instants when the target is not awake. Increasing the amount of wake-ups is an effective, yet energetically cheap means of increasing network throughput and decreasing end-to-end latency. The decrease in latency is well visible in Figure 1: the time difference between the reception of the first packet and its transmission to the next node is much shorter at higher rates. Packets can be forwarded faster with the receiver being in state S_1, S_2, or even CSMA, as the time gaps between packet receptions and the receiving nodes' next wake-ups decreases. If the additionally scheduled *Extra Wake-Ups* are not used, the waste of energy remains limited, as some few additional channel polls are energetically inexpensive.

Most E^2-MAC protocols have been designed under the assumption of sparse low-rate traffic, and usually severely restrain throughput and introduce significant delays. They have been shown to reach only a fraction of the throughput of that of CSMA in [16] [14]. MaxMAC has been specifically designed to achieve a throughput similar as CSMA in situations of increased network activity, given a certain delay for triggering its run-time traffic adaptation mechanisms. *Extra Wake-Ups* somewhat increase the achievable throughput, but remain much below that of CSMA. MaxMAC thus carries the threshold-based concept a step further: when the rate of incoming packets reaches a further threshold T_{CSMA} (with $T_{CSMA} > T_2 > T_1$), MaxMAC again changes its state and switches to the CSMA state, where it completely abandons any sleep-wake pattern for at least CSMA_LEASE. The node in Figure 1 switches to the CSMA state at t=7.5s after the reception of the packet burst. It propagates this state change to the sender node in the ACK, and hence *promises* to the sender node to remain in the CSMA state for at least CSMA_LEASE, permitting it to transmit packets without waiting for the next wake-ups and without transmitting long preambles. With the LEASE timeouts expiring, MaxMAC falls back to its default behavior, where nodes poll the channel with the Base Interval T. The fallback mechanism is well visible in Figure 1 at t=11s where the examined node leaves the CSMA state and falls back to the *Base Interval* state.

IV. Evaluation Platforms & Techniques

A. The Modular Sensor Boards (MSB430) Platform

We implemented our prototype MAC protocols on the MSB430 [17] sensor node platform using the ScatterWeb^2 Operating System [18]. The MSB430 platform has a CC1020 [19] byte-level radio transceiver operating in the 804-940 MHz ISM frequency band. While the maximum raw bit rate of the CC1020 is 153.6 kbit/s, the utilized ScatterWeb^2 OS currently only supports a data rate of 19.2 kbit/s.

B. Sensor Network Management Devices (SNMD)

We used Sensor Node Management Devices (SNMD) [20] to measure a node's current and voltage and used this data to calculate its energy consumption. SNMDs have been specifically designed to accurately measure current and voltage of sensor nodes with a sampling resolution of up to 20 kHz (up to 500 kHz in buffered mode). SNMDs measure the resistive voltage drop across a 1 Ω shunt. The accuracy of the SNMD has been evaluated using high-precision laboratory equipment for different current ranges. The SNMD firmware corrects each sampled measurement by an error term, which was obtained during evaluative testing in advance, which reduces the measurement error introduced by the measurement circuit below $\pm$ 1% for any current in the range of 0-100 mA in [21]. As the accuracy of the SNMD has been calibrated using highly accurate state-of-the art measurement equipment, we can safely assume that it provides best possible physical hardware-based energy measurements. Throughout the experimental analysis of this paper, we decided to stick to a sampling rate of 1000 Hz, as the accuracy gain with even higher rates proved to be negligible with the chosen node type and bandwidth settings. Figure 3 depicts an MSB430 [17] sensor node attached to an SNMD device, which is connected over mini-USB to a desktop PC.

The energy consumed by the sensor node within any time interval $[t_{start}, t_{end}]$ is the integral of its power consumption $P_S(t)$ over time, e.g., the area below the current draw in Figure 1.

$$E_{[t_{start},t_{end}]} = \int_{t_{start}}^{t_{end}} P_S(t)\, dt$$

Using the power relationship $P_S(t) = V(t) \cdot I(t)$ and the numerical values of $I(t)$ and $V(t)$ obtained by the SNMD, the energy consumed by the node over a time interval $[t_{start}, t_{end}]$ can be numerically calculated as

$$E_{[t_{start},t_{end}]} = \sum_{t_{start}}^{t_{end}} V(t) \cdot I(t)\, \Delta t$$

where Δt equals the inverse of the sampling frequency (e.g., Δt = 1ms for a sampling frequency of 1000 Hz).

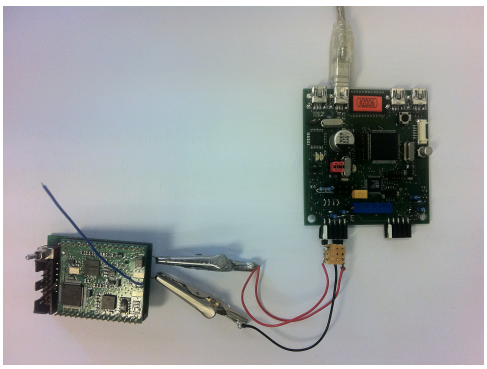


Fig. 3: SNMD with MSB430 node attached

C. Software-based Energy Estimation

Throughout the first two small-scale evaluation experiments in Section VI, we used the SNMD devices to physically measure a node's power consumption. For the later measurements in the distributed indoor MSB430 testbed, we utilized a software-based energy-estimation technique [22] to obtain estimations of the testbed nodes' energy consumptions. In [22], we have shown that the technique consisting in estimating the energy consumption using per-node and per-protocol calibrated parameters achieves an estimation accuracy of less than 1% when measuring the mean absolute estimation error (MAE). We considered this accuracy to be high enough to rely the results of the testbed experiment scenarios on this software-based estimation technique.

D. TARWIS Testbed Management System

We utilized our testbed management system TARWIS [23] to schedule and run the experiments on our distributed indoor testbed. TARWIS is a generic and re-usable Testbed Management System for Wireless Sensor Network Testbeds, developed during the European Union Project WISEBED [24].

V. The MaxMAC Prototype on the MSB430

We evaluated MaxMAC [4] prototype in various scenarios and different topologies, ranging from small benchmarking tabletop experiments to our indoor testbed of 7 nodes distributed across four floors of our institute building. We had chosen WiseMAC [5] and as the starting point and *default behavior* for MaxMAC due to its low per-packet overhead, high efficiency and, even when operating with a fixed static wake-up pattern, already quite high adaptability to variable traffic conditions compared to other E^2-MAC protocols, as pointed out in [13] [14]. Choosing WiseMAC as the major reference E^2-MAC protocol hence seems like a natural choice. Table I lists the packet and header format used with all three evaluated MAC protocols. Table II then lists the main WiseMAC and MaxMAC prototype parameters. The upper half in Table II lists the common WiseMAC and MaxMAC parameters, whereas the lower half lists the MaxMAC-specific parameters, e.g. the state thresholds or the LEASE timeouts. The Base Interval T with which nodes sample the channel was set to 500 ms, the time for a channel poll to 3 ms. We experimentally determined that the MSB430 sensor node with its CC1020 [19] radio driver requires roughly 3 ms to turn the radio on and reliably determine if the preamble byte is being received. The *duty cycle*, calculated as the fraction of the time the radio is kept on (recv/transmit) during each Base Interval T hence amounts to 0.6 %, given the node is idle and neither receives nor transmits any packets. For evaluation and comparison, we compared our WiseMAC and MaxMAC protocol prototypes to a slightly altered version of the default 802.11-like CSMA MAC protocol in ScatterWeb^2 OS [17]. In order to allow for a fair comparison, we implemented a packet burst mode for each MAC protocol, such that nodes can transmit queued packet trains in a burst. Nodes can signal that they have pending packets to the receiver and continue transmitting packets in a burst, receiving an acknowledgment for each frame.

Field	*Bytes*	*Description*
Preamble	variable	predefined bit sequence (0xAA)
Start Delimiter	3	indicates the beginning of the data
Size	1	packet size, including payload
Address Target	1	address of the receiver (0 - 254)
Flags	1	MaxMAC flags (e.g. state info)
Address Source	1	address of the sender (0 - 254)
Number	1	packet sequence number
Type/More Bit	1	packet type, containing more-bit
Millis	1	milliseconds until next wake-up
Payload	28 bytes	payload data
CRC	2	CRC-16 checksum

TABLE I: MaxMAC prototype packet format

Bitrate	19'200 bps
Baudrate	38'400 bps
Packet size	40 byte (incl. header)
Packet Queue size	7 packets
Base Interval T	500 ms
Duty Cycle	0.6 % (3ms)
Threshold T_1	1 packet/s
Threshold T_2	2 packets/s
Threshold T_{CSMA}	3 packets/s
S1_LEASE, S2_LEASE	
CSMA_LEASE	3 s

TABLE II: MaxMAC prototype parameters

VI. Experimental Analysis of MaxMAC

A. Three Nodes Chain Scenario

The first experimental results have been gained in a small-scale experiment setting: three nodes *A, B and C* were aligned on a table with a distance of 50 cm between them, hence with all nodes being in each other's transmission range (cf. Figure 4). We let node *A* generate traffic of variable rate towards node *B* forwarding it to node *C*, c.f. Figure 6. The load alternates between no traffic and load peaks of increasing intensity, ranging from 0.5 packets/s to 6 packets/s. Figure 7 depicts the rate of received packets at the sink node *C*, filtered with a Central Moving Average Filter of 1s and averaged across 20 experiment runs. During the entire experiment duration, node *B* was attached to a SNMD device and the node's current draw was sampled at 1000 Hz.

As one can clearly see comparing the received packets with the offered load, CSMA manages to handle almost all packets from *A* to *C*. It only suffers minor packet loss at the load peaks. The throughput of WiseMAC stalls at a maximum of 3 packets/s, which corresponds to roughly 50% of that of CSMA. Figure 7 clearly shows that MaxMAC with its state-based run-time traffic adaptation mechanism reaches the same throughput as the energy-unconstrained CSMA. As MaxMAC adaptively allocates more duty cycles or even switches to CSMA at a rate of $T_{CSMA} = 3$ packets/s, the protocol manages to handle the load peaks without major packet loss.

Figure 8 depicts the mean current draw of node *B*, averaged over 20 measurements with an SNMD and filtered using a Central Moving Average Filter of 1s. One can clearly see the big gap in the current draw between the E^2-MAC protocols WiseMAC and MaxMAC, and the energy-unconstrained CSMA protocol. With low traffic, CSMA wastes a lot of energy on idle listening. The load peaks are hardly visible at all, as the transceiver does not consume much more power when

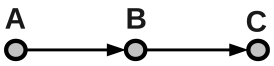


Fig. 4: Three Nodes Chain

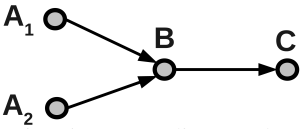


Fig. 5: Contending Nodes

transmitting, compared to idle listening [19]. As MaxMAC switches to the CSMA state with the rate reaching $T_{CSMA} = 3$ packets/s, the power consumption of MaxMAC accordingly jumps to the level of CSMA at this rate, too. Thanks to the run-time traffic-adaptivity mechanisms, MaxMAC reaches the same energy-efficiency in the low-traffic-phases as WiseMAC, but is able to handle the load peaks with much lower packet loss. The *on-demand* scheme of MaxMAC further succeeds remarkably well when the packet rate decreases. With traffic rates decreasing towards after the load peaks, MaxMAC quickly falls back to the Base Interval state, where it again exhibits a very low energy footprint.

Figure 9 depicts the packet delivery rate (PDR), the energy consumption of node *B* and the one-way delay from *A* to *C* in the *Three Nodes Chain* experiment. One can easily see that MaxMAC with a PDR of more than 92.2% achieves a slightly lower PDR than energy-unconstrained CSMA (96.9%), but a far higher PDR than WiseMAC (58.7%). Most of the losses in the MaxMAC experiment runs occurred right at the beginning of the load bursts. As the load exceeds the predefined threshold values, some initial time is necessary to change from the duty-cycling states to the CSMA state, during which most packets were lost. Since the load thresholds have to be exceeded on all the nodes in the chain, congestion effects can occur in the beginning of a load burst, where the nodes close to the sink have not yet changed the state, which finally may result in buffer overflows or collisions. With MaxMAC allocating the radio transceiver *on demand*, and quickly falling back to the *Base Interval* state when the load decreases, the total energy consumed by node *B* amounted to just 33.1 J, as opposed to 98.5 J for CSMA. The evaluation of the one-way delay

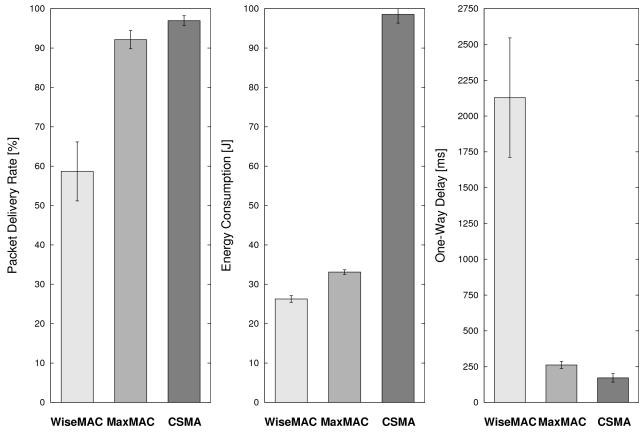


Fig. 9: PDR, Energy Consumption and One-Way Delay in the *Three Nodes Chain* Experiment

revealed that WiseMAC sticking to its strict duty cycling pattern within T=500ms exhibited a far higher mean one-way delay (2.1s) than MaxMAC (261ms) or CSMA (171ms). With WiseMAC, incoming packets are buffered in the transmit queue in every node, and transmitted in a burst as soon as the channel contention has been won, which has a deteriorating impact on the end-to-end latency. CSMA and MaxMAC (at rates $\geq$ 3 packets/s) can rely on the next node constantly being awake and hence immediately transmit any incoming packet, neglecting long preambles, hence generally keeping the latency low.

B. Contending Nodes Scenario

In a second small-scale experiment, we examined the protocol's behavior under variable traffic from two contending nodes A_1,A_2, which is forwarded via node *B* to the sink node *C* (c.f Figure 5). The nodes were again kept on a table with node *B*'s current being measured with an SNMD device. The shape of the offered load generated at nodes A_1,A_2 is depicted in Figure 11. It was chosen to illustrate the behavior of the protocols during phases where neither A_1 nor A_2 is

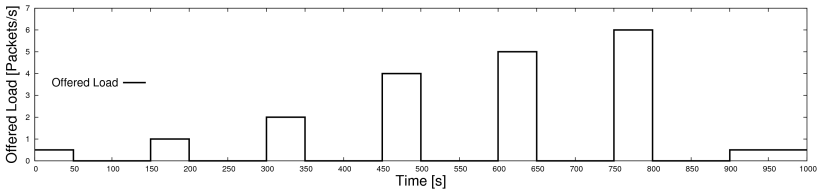


Fig. 6: Offered Load in the *Three Nodes Chain* Experiment

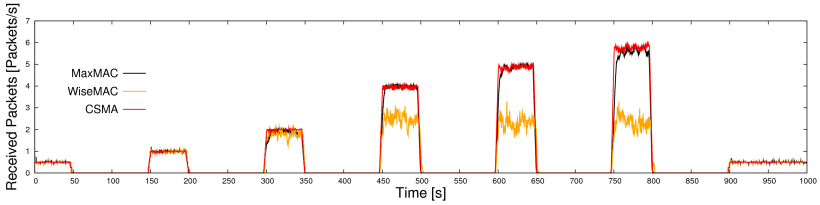


Fig. 7: Packet Reception Rate at Sink Node C in the *Three Nodes Chain* Experiment

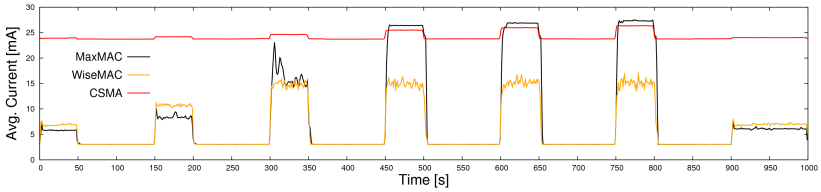


Fig. 8: Average Current of Node B in the *Three Nodes Chain* Experiment

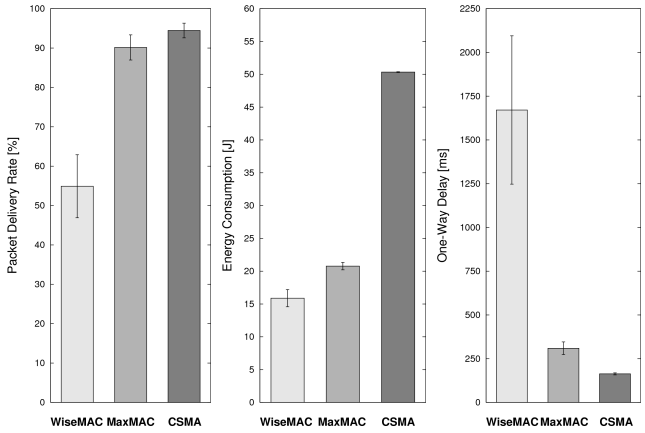


Fig. 10: PDR, Energy Consumption and One-Way Delay in the *Contending Nodes* Experiment

generating load, where *one* of the source nodes is generating load, and when *both* nodes are generating load. Figure 12 depicts the rate of received packets at the sink node *C* over time, filtered with a Central Moving Average Filter of 1s and averaged across 20 experiment runs.

Figure 13 depicts the mean current draw of node *B* (mean of 20 measurements). One can again see the large gap in the average current draw between the E^2-MAC protocols WiseMAC and MaxMAC and the energy-unconstrained CSMA protocol. WiseMAC obviously can not exceed a rate of 1-1.5 packets/s. WiseMAC's performance further degrades in case of high contention at higher traffic rates. In the high load peaks with both nodes sending at increased rate (t=200s and t=375s), the effective throughput degrades, as transmission attempts from the nodes A_1,A_2, but also node *B* forwarding received packets to *C* increasingly cause collisions, since the transmission opportunities are very limited (two brief wake-ups per second).

Figure 10 depicts the packet delivery rate (PDR), the energy consumption of node *B* and the one-way delay from *A* to *C*, in analogy to the results of the former experiment. The

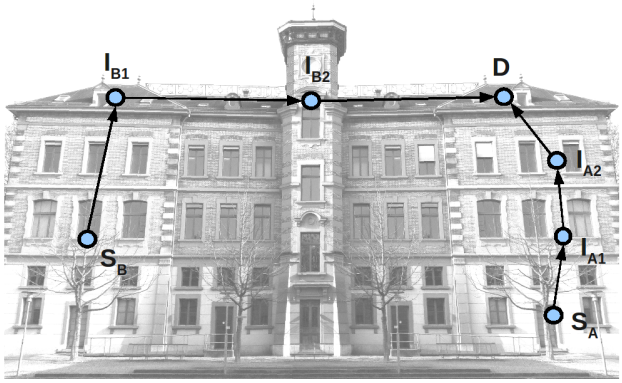


Fig. 14: Indoor Distributed Testbed Scenario

results exhibit a similar picture: MaxMAC with a PDR of roughly 90.3% achieves a slightly lower PDR than energy-unconstrained CSMA (94.5%), but a far higher PDR than WiseMAC (54.9%). Most of the losses within the MaxMAC runs proved to occur at the start of the load bursts *and* during the contention phases where both source nodes A_1,A_2 generated traffic. The evaluation of the energy consumption of node *B* and the one-way delays exhibited similar results as the previous experiment. MaxMAC reaches a similar PDR and latency as CSMA, however at the energy cost of less than 50% of the latter.

C. Distributed Indoor Testbed Experiments

The major advantage of MaxMAC is its ability to switch between energy conservation and the throughput and latency characteristics of energy-unconstrained CSMA depending on the load conditions. In order to examine and verify this property across more than two hops, nodes need to be physically separated, such that transmissions from one node do not impact too heavily on all other nodes, e.g., on nodes exchanging packets on the far other end of the network. In order to achieve a certain *spatial reuse* of the channel, not all nodes should hence be located within each other's transmission

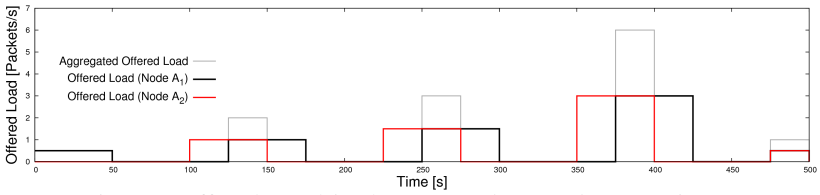


Fig. 11: Offered Load in the *Contending Nodes* Experiment

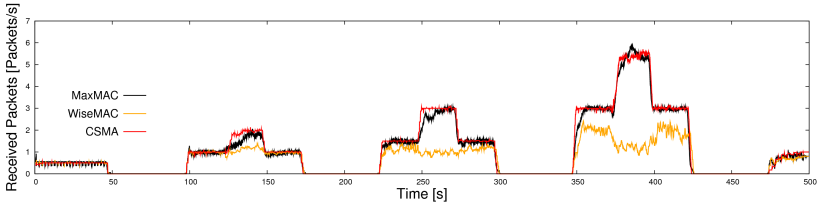


Fig. 12: Packet Reception Rate at Sink Node C in the *Contending Nodes* Experiment

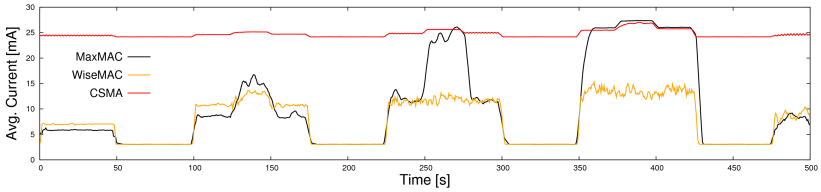


Fig. 13: Average Current of Node B in the *Contending Nodes* Experiment

range. We hence set up an indoor distributed testbed with 7 MSB430 nodes distributed across four floors of our institute building, as schematically depicted in Figure 14. All nodes were placed in different rooms of the indoor testbed to obtain a network where nodes communicate across concrete walls and floors. Figure 14 depicts the links which were used throughout the evaluation, which yielded a high packet delivery rate ($\geq$95%) in case of no other ongoing traffic. It was naturally impossible to perfectly *shield* the links from each other, but a certain *spatial reuse* could nevertheless be achieved.

Distributed Multi-Hop Chain Scenario: We evaluated the maximum achievable throughput across 4 nodes (S_A to D) and 5 nodes (S_A to I_{B2}) in a preliminary experiment. Traffic was generated at different load levels at node S_A during 60s within 20 experiment runs of 260s duration each. The upper graph in Figure 15 depicts the seven different load curves of node S_A for the examined load peaks of 0.5-6 packets/s. Before the load peak at t=90s, the source node S_A sends one packet each 10s towards the destination. After the load peak, the load falls back to zero. Each offered load setting with the different peak load rates was examined with 20 independent experiment runs. We estimated the energy consumption of the nodes during the experiments using our software-based energy estimation framework [22]. Nodes print their energy consumption estimation to the serial interface every 5s. The lower graph in Figure 15 depicts the resulting average current draw of node I_{A1} vs. the experiment duration of the 4-nodes experiment runs with the peak load rate of 3 packets/s. The curves of the three protocols very much resemble the former curves from the small-scale experiments, the step-like shape stems the much lower resolution of only one sample each 5s, as opposed to 1000 Hz with the SNMD. The impact of the load peaks generated during 60s starting at t=90s is clearly visible. MaxMAC is able to deliver the same load as CSMA, and falls back to the default behavior after the load peak. WiseMAC has a lower average current draw during the load peak, it however only delivers a fraction of the offered load of that of CSMA or MaxMAC. With WiseMAC and MaxMAC, the initial full-preamble broadcast along the node chain is well visible in the node's energy consumptions. As nodes do not yet *know* each other's schedule offsets, the first packet traversing the

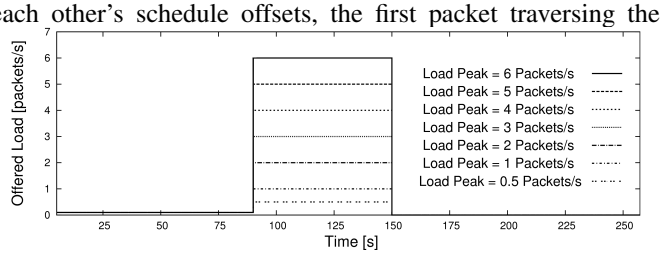


Fig. 15: Traffic Shapes with different Load Rates

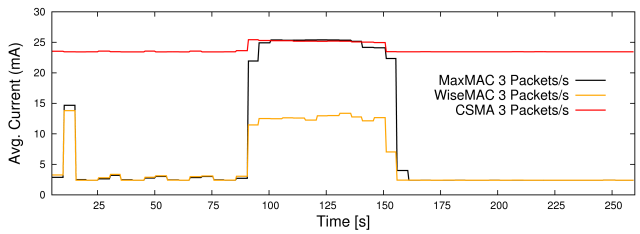


Fig. 16: Software-based Energy Estimation at 3 Packets/s

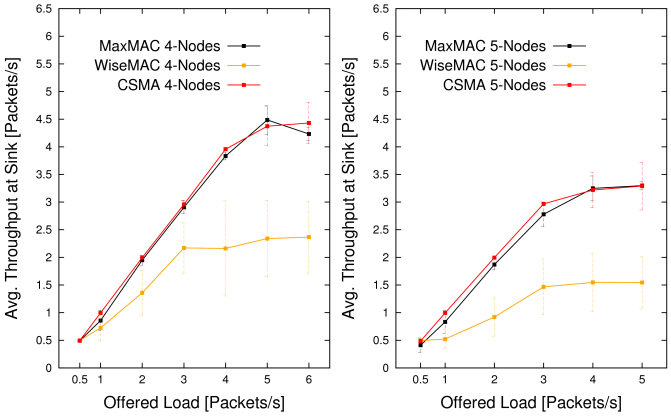


Fig. 17: Throughput across 4 Nodes and 5 Nodes

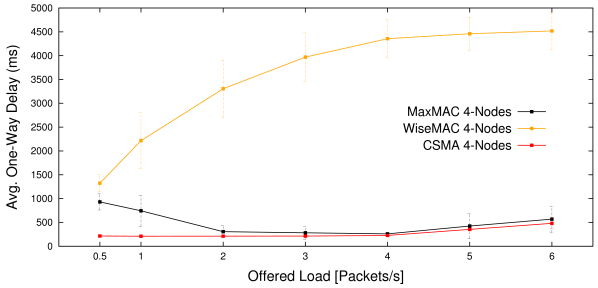


Fig. 18: One-Way Delay dependent on Traffic Rate

chain triggers a broadcast with a preamble spanning over the entire interval duration T in all participating nodes, which are obviously overheard by some of the nodes in the chain, causing the significant spike in the average current at the experiment start.

Figure 16 depicts the obtained throughput during the load phase at the sink nodes. One can clearly see that CSMA and MaxMAC both succeed to deliver almost the full offered load up to a rate of 4.5 packets/s for the 4 nodes experiment, and 3.5 packets/s for the 5 nodes experiment. Due to the fact that all nodes transmit and listen to the same channel, and imperfect spacial reuse in the indoor testbed (e.g. transmissions from I_{B2} *to* D impact to a non-negligible extent also on the nodes I_{A2} and I_{A1}), the maximum throughput across the 60s load peak reaches only 3.5 packets/s.

Figure 18 depicts the latency measured from source to sink vs. traffic rate in the 4 nodes scenario. MaxMAC's latency decreases for rates 0.5 to 4 packets/s. With increasing rate, the protocol allocates Extra wake-ups and switches to CSMA, which pushes the delay into the range of that of CSMA. With rates above 4 packets/s, congestion effects lead to a significant increase of the one-way delay. CSMA suffers from the same congestion effects at high rates as well. Since it does not duty cycle its transceiver, it exhibits a much lower delay at low traffic rates. The one-way delay of WiseMAC significantly increases with the increasing traffic rate. Since WiseMAC is limited to few transmission opportunities (two wake-ups/second), packets are queued in every hop and transmitted in a burst, which significantly increases the end-to-end latency.

Figure 19 depicts the total energy consumed by a node in the experiment, averaged across all nodes participating in the chain scenario. While the energy consumption of CSMA

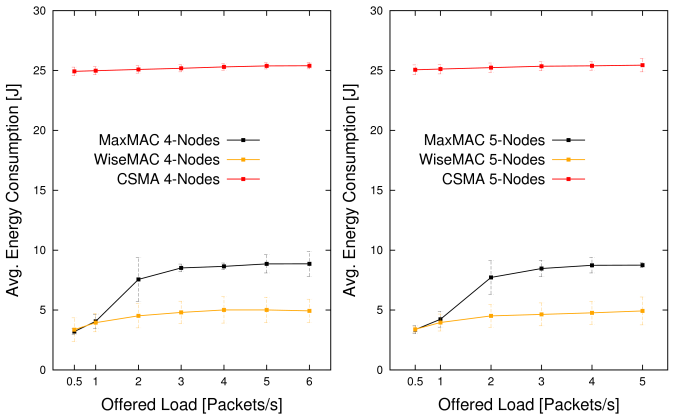


Fig. 19: Average Energy Consumption over 260s

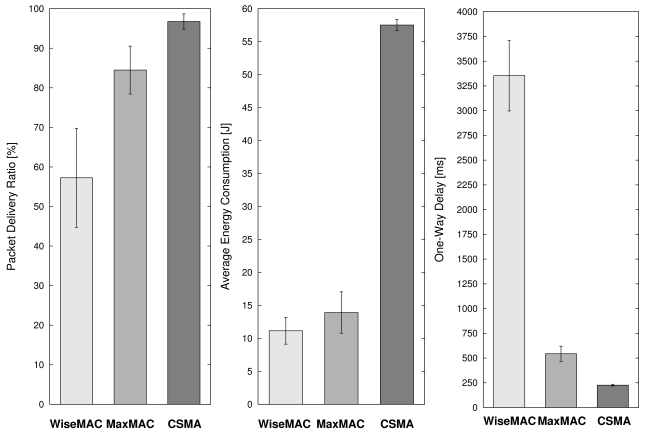


Fig. 22: PDR, Energy Consumption and One-Way Delay in the *Distributed Testbed* Experiment

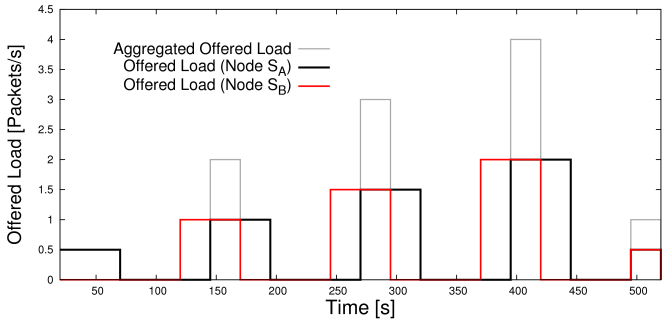


Fig. 20: Offered Load from Nodes S_A and S_B

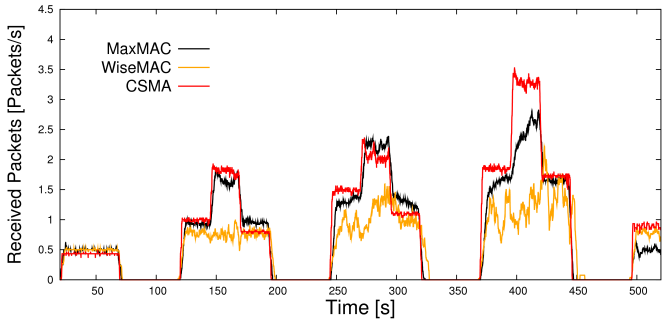


Fig. 21: Packet Reception Rate at Sink Node D

remains constantly high at a level of 25 J, independent of the traffic load, the total energy consumption of MaxMAC increases with the load rate, but peaks at roughly 10 J. This peak is not exceeded, as MaxMAC falls back to the default *Base Interval* state after the traffic phase, where it only samples the channel every T=500ms. MaxMAC hence returns to the low power consumption of WiseMAC and manages to save a major portion of the energy spent in CSMA.

Variable Traffic from Leaf Nodes Scenario: We evaluated the behavior of MaxMAC, WiseMAC and the energy-unconstrained CSMA with variable and contending traffic from different areas of the network. The two leaf nodes S_A and S_B generate variable load across their subtrees towards the sink node D. The shape of the offered load is similar to that of A_1, A_2 in Figure 11, with the minor difference that the rate during the second and third load peaks were reduced to 1.5 and 2 packets/, c.f. Figure 20. Due to interferences of the concurrent transmissions within the building, more generated traffic had a vastly deteriorating impact on the resulting end-to-end throughput. We specifically chose the shape of the offered load to illustrate the behavior of the three examined protocols during phases where neither branch of the tree is generating load, where one leaf node is generating load or when both leaf nodes are generating load. The resulting traffic rate received by node D is depicted in Figure 20. The displayed rates were again calculated using a Central Moving Average filter of 1s and 20 experiment runs. One can clearly see that WiseMAC's maximum throughput stalls at slightly more than 1 packet/s. CSMA reaches a throughput of roughly 3.5 packets/s, and MaxMAC up to 3 packets/s.

The evaluation of the rate of received packets in Figure 20 clearly exhibits that MaxMAC's performance degrades with increasing contention in the distributed testbed. Although more packets are lost in this case, the PDR bars in Figure 22 however still convey a massive improvement concerning the achieved PDR. Applying MaxMAC's run-time traffic adaptation mechanisms still seems to pay off drastically compared to WiseMAC. The packet delivery rate of MaxMAC (84.5%) is significantly higher than that of WiseMAC (56%). MaxMAC's performance however clearly lags behind CSMA (96%), which managed to deliver the major portion of the packets across the busy network. The main reason behind this performance degradation was the phenomenon that MaxMAC LEASE timeouts sometimes expired due to a series of timely-correlated collisions. When intermediate nodes in the chain fall back to the default behavior during a high-traffic phase, some time is necessary to exceed the thresholds and re-establish the fully active chain of nodes, during which most of these losses occurred. Besides the PDR, the bars of the average node's energy consumption in Figure 22 conveys a similar improvement as in the small-scale experiments. The analysis of the average end-to-end latencies in the same figure further indicates that WiseMAC is suffering much more from congestion effects than MaxMAC or CSMA.

Figure 23 depicts the energy consumption estimations of the nodes in one subtree of the distributed testbed network. Across the chain $S \rightarrow B_1 \rightarrow B_2 \rightarrow D$, the energy estimations for the nodes decrease with WiseMAC, but increase with MaxMAC and CSMA. This interesting observation can be explained as follows: WiseMAC and other preamble-sampling based approaches (e.g. X-MAC [8] or B-MAC [12]) generally shift

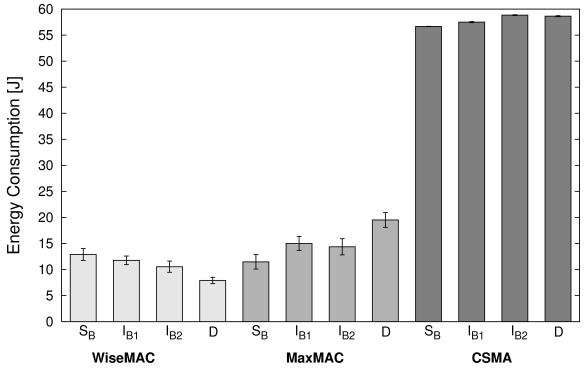


Fig. 23: Distribution of Node Energy Estimations

the cost of the medium access from the receiving node to the transmitting node. A node intending to transmit a packet has to *catch* the receiver in one of its short channel polls, using long preambles to compensate for clock drifts, for medium reservation and for further inaccuracies. The further away a node is from the source node in the WiseMAC evaluation of Figure 23, the lower is hence its energy consumption, as the rate of received packets generally decreases with every hop. The sink node then inherently exhibits the lowest energy consumption, as it does not forward any of the received packets. With MaxMAC, the energy estimations slightly increase across the chain. This increase is most likely explained by the increasing contention of the nodes for the channel in the area of the sink, which necessitate costly retransmission attempts. As the sink node receives packets from both subtrees, it exceeds the MaxMAC thresholds earlier and hence switches to energetically more expensive states (S_1, S_2, S_{CSMA}). With CSMA, the energy consumption estimations exhibit a very slight increase, which we explain with the increasing contention of the nodes for the channel in the area of the sink node.

VII. Conclusions

In this paper we have evaluated the real-world feasibility of the MaxMAC protocol [4], which targets at achieving maximal run-time traffic adaptivity. We demonstrated in a series of experiments that MaxMAC is clearly distinguishable from its non-adaptive counterpart WiseMAC by reaching nearly the same throughput and a similarly low latency as energy-unconstrained CSMA, while still exhibiting the same energy-efficiency during periods of sparse network activity. The MaxMAC protocol hence combines the advantages of energy unconstrained CSMA (high throughput, high PDR, low latency) with those of classical E^2-MAC protocols (high energy-efficiency).

Like most contention-based MAC protocols, MaxMAC is a general-purpose protocol, and does not rely on assumptions which are cumbersome to achieve (e.g, rigid time-synchronization across the entire network). With X-MAC [8] and B-MAC [12]/ContikiMAC [15] being the *default* MAC layers in Contiki and TinyOS, respectively, preamble-sampling MAC protocols are nowadays by far the most widely used protocols in deployment studies. MaxMAC targets at improving the most significant drawbacks of such preamble-sampling MAC protocols, namely their poor performance under variable load. We envision applications of MaxMAC in event-based sensor networks where at certain instants, the provision of high throughput and fast end-to-end response times becomes more important than the conservation of energy, e.g. in healthcare, where nodes attached to patients need to rely on the provision of higher throughput and fast response times when critical values have been sensed, in order to communicate with central entities. The protocol furthermore facilitates real-time human interaction with sensor nodes, e.g. when querying nodes or transmitting large chunks of program code or data across several links.

References

[1] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton, "CodeBlue: An ad hoc Sensor Network Infrastructure for Emergency Medical Care." MobiSys 2004 Workshop on Applications of Mobile Embedded Systems, 2004.

[2] O. Chipara, C. Lu, T. C. Bailey, and G.-C. Roman, "Reliable clinical monitoring using wireless sensor networks: Experiences in a step-down hospital unit." ACM Conference on Embedded Networked Sensor Systems, 2010.

[3] Gao, T. et al, "The Advanced Health and Disaster Aid Network: A Light-weight Wireless Medical System for Triage." IEEE Transactions on Biomedical Circuits and Systems, 2007.

[4] P. Hurni and T. Braun, "MaxMAC: a Maximally Traffic-Adaptive MAC Protocol for Wireless Sensor Networks." European Conference on Wireless Sensor Networks (EWSN), 2010.

[5] A. El-Hoiydi and J.-D. Decotignie, "WiseMAC: An Ultra Low Power MAC Protocol for Multihop Wireless Sensor Networks." ALGOSENSORS, 2004.

[6] T. Van Dam and K. Langendoen, "An Adaptive Energy Efficient MAC Protocol for Wireless Sensor Networks (TMAC)." ACM Conference on Embedded Networked Sensor Systems, 2003.

[7] W. Ye, J. Heidemann, and D. Estrin, "An Energy Efficient MAC Protocol for Wireless Sensor Networks." IEEE International Conference on Computer Communications (INFOCOM), 2002.

[8] Buettner, M., Gary V. Y., Anderson, E. and Han, R., "X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks." ACM Conference on Embedded Networked Sensor Systems, 2006.

[9] Andreas Meier and Matthias Woehrle and Marco Zimmerling and Lothar Thiele, "Zerocal: Automatic MAC Protocol Calibration." Distributed Computing in Sensor Systems, 2010.

[10] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors." IEEE Workshop on Embedded Networked Sensors (EmNets), 2004, http://www.sics.se/contiki/.

[11] S. H. Lee, J. H. Park, and L. Choi., "AMAC: Traffic-Adaptive Sensor Network MAC Protocol through Variable Duty-Cycle Operations." IEEE International Conference on Communications (ICC), 2007.

[12] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks." ACM Conference on Embedded Networked Sensor Systems, 2004.

[13] K. Langendoen and A. Meier, "Analyzing MAC Protocols for Low Data-Rate Applications," *ACM Transactions on Sensor Networks*, vol. 7, no. 2, 2010.

[14] P. Hurni and T. Braun, "On the Adaptivity of today's Energy-Efficient MAC Protocols under varying Traffic Conditions." IEEE Conference on Ultra-Modern Technologies, 2009.

[15] A. Dunkels, L. Mottola, N. Tsiftes, Fredrik Österlind, J. Eriksson, and N. Finne, "The Announcement Layer: Beacon Coordination for the Sensornet Stack." European Conference on Wireless Sensor Networks (EWSN), 2011.

[16] A. El-Hoiydi, "Energy Efficient Medium Access Control for Wireless Sensor Networks." PhD Thesis, EPFL, 2005.

[17] M. Baar, E. Koeppe, A. Liers, and J. Schiller, "The ScatterWeb MSB-430 Platform for Wireless Sensor Networks." SICS Contiki Workshop, 2007.

[18] ScatterWeb2 Operating System - Freie Universität Berlin, http://scatterweb.mi.fu-berlin.de/.

[19] Texas Instruments CC1020, "Single-Chip FSK/OOK CMOS RF Transceiver."

[20] A. Hergenröder, J. Horneber, D. Meier, P. Armbruster, and M. Zitterbart, "Distributed Energy Measurements in Wireless Sensor Networks." ACM SenSys, Demo Session, 2009.

[21] A. Hergenröder, J. Wilke, and D. Meier, "Distributed Energy Measurements in WSN Testbeds with a Sensor Node Management Device (SNMD)," in *International Conference on Architecture of Computing Systems*, 2010.

[22] P. Hurni, B. Nyffenegger, T. Braun, A. Hergenroeder, "On The Accuracy of Software-based Energy Estimation Techniques." European Conference on Wireless Sensor Networks (EWSN), 2011.

[23] P. Hurni, G.Wagenknecht, M. Anwander, and T. Braun, "A Testbed Management System for Wireless Sensor Network Testbeds (TARWIS)." European Conference on Wireless Sensor Networks (EWSN), Poster Session, 2010.

[24] Seventh Framework Programme FP7:, "Wireless Sensor Networks Testbeds (WISEBED)," http://www.wisebed.eu.