A Reliable, Traffic-adaptive and Energy-efficient Link Layer for Wireless Sensor Networks

Markus Anwander and Torsten Braun
Institute of Computer Science and Applied Mathematics
University of Bern, Neubrückstrasse 10, 3012 Bern, Switzerland
Email: braun@iam.unibe.ch

***Abstract*—The paper presents a link layer stack for wireless sensor networks, which consists of the Burst-aware Energy-efficient Adaptive Medium access control (BEAM) and the Hop-to-Hop Reliability (H2HR) protocol. BEAM can operate with short beacons to announce data transmissions or include data within the beacons. Duty cycles can be adapted by a traffic prediction mechanism indicating pending packets destined for a node and by estimating its wake-up times. H2HR takes advantage of information provided by BEAM such as neighbour information and transmission information to perform per-hop congestion control. We justify the design decisions by measurements in a real-world wireless sensor network testbed and compare the performance with other link layer protocols.**

I. Introduction

In this paper, we investigate three major problems of current real world Wireless Sensor Networks (WSNs), namely energy efficiency, reliable data transmission, and interoperability. As sensor nodes are powered by batteries, their lifetime is limited. Reliability mechanisms are required to ensure successful data forwarding in challenging environments. Standard protocol stacks are needed to ensure compatibility among heterogenous nodes. While the IETF mainly addresses routing and application layer protocols [2], [3], this paper focuses on link layer protocols for an IP protocol in WSNs.

The main contribution of this paper is an energy efficient and reliable link layer in a WSN stack. A per-hop congestion detection mechanism is able to identify and correctly handle intra-flow and inter-flow interferences as well as congestion. The link layer protocols work with energy efficient and robust packet oriented radio modules such as CC2420, which is widely used on many sensor nodes such as telosB and MicaZ. While previous work [4], [5] focused on intial design and simulations of the proposed protocols, this paper discusses design choices and evaluations based on implementations running on real sensor nodes. The proposed link layer is divided into two sublayers. According to the naming of the Contiki [1] operating system the lower layer is called *radio duty cycle* (RDC) sublayer. On the RDC sublayer we implemented the *Burst-aware Energy-efficient Adaptive MAC* protocol (BEAM). It uses a traffic prediction mechanism to adapt duty cycles according to current network load. BEAM supports reliability functions of the upper sublayer, which is called *MAC* layer by Contiki. In the following, we call this sublayer *hop-to-hop reliability* (H2H) layer to avoid misunderstandings. On the H2H layer, we implemented the *hop-to-hop reliability protocol* (H2HR) [5], which ensures reliable hop-to-hop forwarding by local retransmissions and provides per-hop congestion control. After discussing related work in Section II and introducing BEAM/H2HR in Sections III - IV, our testbed evaluations motivate several design decisions and compare the performance with other WSN link layers in Section V. Section VI concludes the paper.

II. Related Work

WSN link layer protocols preserve energy by duty cycling the radio module. Synchronous radio duty cycle protocols synchronize wake-up periods among neighboring nodes. Among them are contention based protocols such as Sensor-MAC [11] and time division multiple access (TDMA) based protocols such as the Lightweight Medium Access Protocol [12].

Asynchronous radio duty cycle protocols do not synchronize the wake-up periods among neighboring nodes. Every sensor node can perform its wake-up period independently from the schedule of the neighbor nodes. Low Power Probing (LPP) employs receiver nodes to announce the wake-up periods to a sender by a short *probe* message at the beginning of a wake-up period. After receiving the probe, the sender immediately forwards the data packet to the receiver, e.g., Koala [13]. Low Power Listening (LPL) employs sender nodes to announce the forwarding of a packet to the receiver node A sender transmits a long physical preamble. If a neighbor node detects such a preamble during the wake-up period, then it waits to receive the data. Wireless Sensor MAC (WiseMAC) [14] uses LPL with a long preamble including the receiver address of the data part. XMAC [8] uses beacon strobes instead of a long preamble. Therefore, XMAC can support packet-oriented radio modules such as the CC2420 radio module. XMAC adapts the duty cycle period according to current traffic load. XMAC has been implemented on real sensor nodes using the Contiki operating system [17], but the implementation does not support adaptive duty cycles. ContikiMac [15] uses beacon strobes and applies static duty cycle periods. ContikiMac is able to make use of the energy efficient link layer functions provided by the CC2420 radio module, such as handling acknowledgments or calculating checksums directly by the radio module. If ContikiMAC detects packet loss, then the Contiki CSMA protocol on top of ConitikiMAC may retransmit the packet later. MaxMAC [16] determines the current traffic load, like XMAC, by counting the wake-up periods in which a packet has been received and adapts duty cycles accordingly. MaxMAC does not work with packet oriented radio modules. The long preambles require a bit/byte-oriented radio module. Receiver-initiated MAC protocol approaches and short-preamble bursts in multi-channel WSNs are analysed analytically in [19].

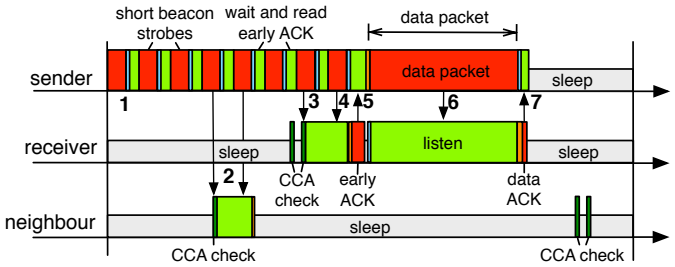


Fig. 1: BEAM using short beacon strobes.

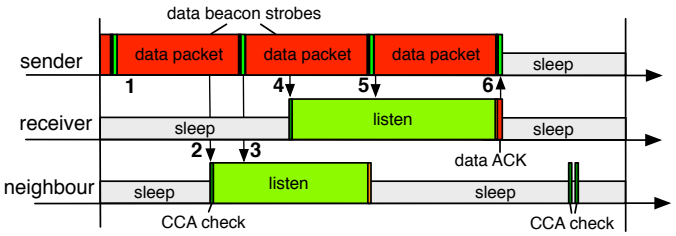


Fig. 2: BEAM with beacon strobes including the payload.

Another issue addressed by this paper is to avoid interferences caused by too many simultaneously ongoing wireless transmissions. The interference aware fair rate control [18] detects congestion at a node by monitoring the average queue length and communicates congestion state to the set of potential interferers. In this paper we rather focus on local coordination and cross-layer information exchange between the MAC and the hop-to-hop reliability layer.

III. Burst Aware Energy Efficient Adaptive MAC Protocol

The main goal of BEAM is to minimize energy consumption by duty cycling the radio module. The term duty cycle is used as the ratio between the time for which the radio module is active and the total time. Duty cycling periodically turns on the radio module to send or receive packets (active period). Between these active periods, the radio module is turned off (sleep period). The time between two activations (wake-ups) is called wake-up interval.

A. Impact of CC2420 Characteristics on BEAM Design

While packet-oriented radio modules reduce flexibility for link layer protocol design, they can efficiently execute common link layer tasks such as channel checking or acknowledgment handling. Shifting those tasks from the microcontroller to the radio module can significantly reduce execution time, preserve energy, and reduce code size. BEAM has been developed to work best with the energy efficient packet-oriented and IEEE 802.15.4 compliant radio module CC2420 [7]. The IEEE 802.15.4 compliant BEAM frame format enables the CC2420 radio module to execute link layer tasks such as sending acknowledgments or CRC calculcations without interaction with the microcontroller.

B. Basic Functionality of BEAM

The design of BEAM is based on LPL. We developed two BEAM modes. One mode is based on XMAC [8] and uses short beacon strobes to announce subsequent data transmissions. The other mode uses beacon strobes including payload. Payload is piggybacked to the beacon strobe.

1) BEAM using Short Beacon Strobes: Figure 1 depicts the BEAM mode based on XMAC using short beacon strobes to announce a pending frame. The numbering of the following list refers to the numbers depicted in Figure 1.

1) A sender with a pending packet repeatedly transmits short beacon strobes, which are IEEE 802.15.4 compliant frames with empty *MAC frame payload*.
2) A neighbour node wakes up to check the channel and receives a short beacon strobe. Since the packet has not been addressed to the node, the radio module is turned off for the remaining wake-up interval.
3) The addressed receiver node wakes up to perform two Clear Channel Assessment (CCA) checks shortly after each other to reliably detect a short beacon strobe. The receiver node detects a transmission at the second CCA check, but the frame start has been missed. The receiver node now waits for the next short beacon strobe.
4) The receiver node receives the next short beacon strobe.
5) After receiving the entire frame, the receiver instantly returns an early acknowledgment. This can be implemented by an automatic acknowledgment (*AUTOACK*) generated by the CC2420 radio module.
6) The sender receives the early acknowledgment, which announces that the receiver is ready. Now, the sender starts the data transmission.
7) After receiving the entire data packet, it is processed and acknowledged by the receiver node.

2) BEAM with Beacon Strobes Including Payload: In this mode, BEAM piggy-backs data and beacon strobe. A sender with a pending packet consistently transmits beacon strobes including payload until an acknowledgment has been received. If a receiver gets corrupted data payload, it can wait for the next beacon frame, which includes the same data, cf. Figure 2.

1) A sender with a pending packet transmits IEEE 802.15.4 compliant beacon strobes including payload.
2) A neighbour node wakes up and detects an ongoing transmission.
3) Since the target address of the next beacon strobe belongs to another node, no acknowledgment has to be sent. After receiving the data, the radio module is turned off for the rest of the wake-up interval.
4) The CCA check of a receiver node detects an ongoing transmission. The radio module stays in listening mode to receive the next beacon strobe.
5) The radio module detects and receives the next beacon strobe.
6) The beacon strobe has been transmitted completely and is acknowledged by the receiver node, possibly by automatic acknowledgement from the radio mod-

ule. After processing the beacon frame, the receiver node turns off the radio module. The transmitter goes back to sleep after the data acknowledgment has been received.

C. Adaptive Duty Cycles

BEAM supports asynchronous adaptive duty cycles to change the frequency of active periods according to current traffic load. Like XMAC and ContikiMAC [9] BEAM is using default wake-up intervals of 125 ms. In addition, BEAM uses three shorter pre-defined wake-up intervals (cf. Table I) to support adaptive duty cycles. Adaptive duty cycle protocols such as XMAC [8] and MaxMAC [16] count the recently forwarded packets to determine the current traffic. Such traffic monitoring mechanisms work quite well in simple network topologies with few internal interferences. Internal interferences are caused by multiple simultaneous packet transmissions, e.g., for multiple connections, inside the WSN. Resulting internal interferences may trigger congestion, which results in less traffic. This causes a smaller number of counted packets and, therefore, traffic monitoring will decrease the duty cycle. Lower duty cycles in case of congestion and high traffic load will further increase packet loss due to buffer overflow.

Per-hop congestion rather requires high duty cycles to increase network bandwidth for high traffic load. Therefore, BEAM uses a forward-looking **traffic prediction** mechanism to determine appropriate duty cycles. This mechanism is based on a two-bit buffer index representing the number of pending packets per neighbor node. The buffer index value 0 indicates an empty buffer, 1 represents a single packet in the buffer, and 2 indicates that there is more than one packet in the buffer, but still less than 50% of the buffer capacity is used. A buffer index value of 3 indicates a used buffer capacity of over 50%. The buffer index is added to every beacon strobe to inform a receiving node about its pending packets. It is written into two of the five reserved bits of the IEEE 802.15.4 *Frame Control Field*. The buffer index is also copied to acknowledgments to inform a sender about total pending packets. This is not possible, if automatic acknowledgements by the radio module are used.

A node stores the received buffer index with the current timestamp and corresponding sender address in its **neighbour table**. This table holds all relevant neighbour node information received by beacon strobes, data frames or acknowledgments. The use of the buffer index depends on the addressing of the received beacon strobes. If the beacon strobe was addressed to this node, the buffer index is stored as *pending* buffer index into the neighbour table. Every time the radio module is turned off, the pending buffer indices are accumulated to calculate the overall expected incoming traffic and resulting duty cycle. In case of an acknowledgment or if the packet has been received by overhearing, the buffer index is stored in the neighbour table as *fuzzy* buffer index. The *fuzzy* buffer index value is not used by BEAM but offered to the H2H sublayer to estimate the overall local traffic load and expected internal interferences. The use of the pending buffer index enables a sender to announce multiple pending packets by a single beacon strobe to a receiver. Abrupt changes of the number of pending packets can be signaled quickly. Table I shows two possible mappings between the accumulated pending buffer indices and the number of wake-ups per second.

TABLE I: Two mappings of the pending buffer indices to the selected duty cycle.

Traffic load	Accumulated buffer indices	Wake-ups per sec. (*implemented*)	Wake-ups per sec. (*low energy profile*)
Low	0	8	1
Moderate	1	16	8
High	2 - 4	64	64
Maximum	>4	256	256

TABLE II: BEAM neighbour table

Value	Description/Purpose
Timestamp	Time of the last successful reception
Address	Address of the neighbour node
Duty cycle index	Currently used duty cycle length
Pending buffer index	Expected packets from this neighbour
Fuzzy buffer index	Pending packets of this neighbour

D. Optimizing Beacon Strobe Transmissions

With a LPL protocol, the exact timing of the next wake-up of a receiver is usually unknown to a sender. Therefore, BEAM starts transmitting beacon strobes as soon as a new packet has been received from the upper layer. The longer the wake-up interval of the receiver is, the more beacon strobe transmissions are required on average. Optimization of beacon strobe transmission estimates the next wake-up time of a receiver. This enables BEAM to delay the start of the beacon strobe transmissions until the receiver wakes up. A two-bit duty cycle index added to the IEEE 802.15.4 *Frame Control* field informs the neighbour node, which of the four pre-defined duty cycles is currently used by the node. It is added to every beacon strobe as well as software acknowledgment and stored in the neighbour table. The duty cycle index and the timestamp for the last detected wake-up are sufficient to estimate the next wake-up time for a neighbour node. The strobe transmission is scheduled 8 ms (= 2 full frame lengths) prior to the estimated wake-up time.

E. Reliability Support

BEAM uses CRC calculations at the sender and the receiver, which can be performed efficiently by the radio module. In this case, automatic acknowledgment by the radio module can also be used. After transmitting a packet, BEAM provides *transmission information* to the upper link sublayer protocol, which can use it for per-hop congestion control, see Section IV-A. BEAM defines four different types of transmission information to be used by the upper link sublayer protocol: Successful Transmission, Missing Acknowledgment, Detected Interference, Busy Channel. BEAM also provides access to its *neighbour table*, cf. Table II.

IV. Hop-to-Hop Reliability Protocol

We introduced the hop-to-hop reliability protocol (H2HR) on the H2H sublayer to ensure reliable hop-to-hop forwarding.

H2HR insistently tries to retransmit every unsuccessfully transmitted packet to avoid energy-costly end-to-end retransmissions. H2HR is able to detect critical network conditions such as congestion and to control the time of each transmission.

A. Per-Hop Congestion Control

The goal of the H2HR per-hop congestion control mechanism is to adapt the traffic flow before it is affected by congestion. H2HR slows down traffic by delaying transmissions if the bandwidth provided by BEAM is not sufficient. In case of too many generated packets, packets should be dropped as soon as possible to not overload the WSN. The most energy-efficient procedure is to drop them already at the generating node. When detecting upcoming congestion, H2HR slows down traffic by determining an appropriate transmission delay using three different information sources:

1) The BEAM **transmission information** enables H2HR to estimate the current channel load and to detect interferences. Considering the frequency and type of the transmission information, we define a counter called *channel load factor* with a value range between 1 and 20. Every *Successful Transmission* reduces this counter by a value of 2. For every *Missing Acknowledgment*, we add a value of 2, for every *Detected Interference* and *Busy Channel* we add 1. Besides the channel load, the receiver load and already executed retransmissions have to be taken into account to determine an appropriate transmission delay.
2) The BEAM **neighbour table** provides information about the current *receiver load* of one-hop neighbour nodes. H2HR uses this information to calculate an appropriate transmission delay for a receiver. If the buffer indices are not available or zero, then no additional delay is used. Otherwise, H2HR delays the next transmission for an appropriate time period as shown in Table III. The delay values have been determined using simulations and real-world experiments.
3) For every retransmission attempt of an individual packet, an additional delay is added. Table IV shows the used relation between **retransmission counts** and delay. The values were again determined by simulation and real world WSN testbed experiments.

Table V shows the different information sources used by the per-hop congestion control mechanism. All information sources are taken into account to determine an appropriate next transmission delay (TX_{delay}). The applied transmission delay (with a maximum value of 1 s) is calculated by:

$$TX_{delay} = Node_{DL} \cdot (f_{CL} \cdot (dp_{RL} + dp_{RC})) \qquad (1)$$

If too many sensor nodes continuously generate and forward packets towards the sink, then the resulting traffic load can be too high to be forwarded in the WSN. The resulting interferences cause congestion and packet loss. The H2HR retransmission delay mechanism works as a hop-by-hop based backpressure mechanism. An increasing value of TX_{delay} reduces the forwarding rate of incoming packets. If the forwarding rate is too low, the number of packets cached in the local packet buffer increases. The per-hop congestion control mechanism of the predecessor node detects this and increases its own TX_{delay} value. In the best case, the retransmission delay mechanism is able to delay the transmission attempts on the entire path up to the nodes generating the packets. This reduces the number of packets injected to the WSN.

TABLE III: Delay by buffer index.

Buffer index for target node	Transmission delay [wake-up intervals]
0	0
1	1 - 3
2	2 - 6
3	4 - 12

TABLE IV: Delay by retransmission count.

Retransmission counts	Transmission delay [wake-up intervals]
1	1 - 3
2, 3	2 - 6
4, 5	4 - 12
6-10	8 - 24

B. Packet Aggregation

A way to reduce internal interferences is to reduce the number of simultaneously forwarded packets within the network. H2HR adds to every aggregated data frame the corresponding IEEE 802.15.4 compliant length and sequence number values. Aggregated packets are announced to the receiver by the *multiple frames* bit, which is one of the five reserved bits of the IEEE 802.15.4 frame.

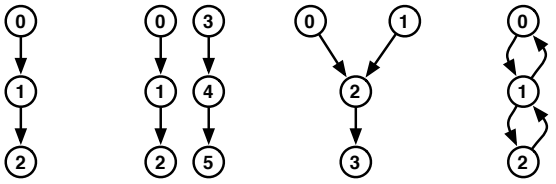
V. Evaluation

A. Evaluation Scenarios and Metrics

Figure 3 depicts four different small-scale network topologies for testbed evaluations, in particular used for protocol optimizations. Figure 4 shows the different traffic patterns and network topologies used in the medium-scale scenarios. We used a real world WSN testbed based on a Wisebed installation [10]. The sensor nodes have been distributed over four floors of an office building with solid walls. In the streaming scenario, four leaf nodes are sending continuously small packets to the sink. It represents a WSN scenario collecting sensor data from the environment. In the event scenario, four nodes detect an event at the same time. These four nodes try to send a message to the sink simultaneously. In the burst scenario the root node sends 800 bytes to all four leave nodes. This may represent a code update. Energy consumption has been measured by a RIGOL DM3052 digital multimeter, which measures electrical current with a resolution of 50'000 samples per second, and a software based energy profiler recording radio state switches.

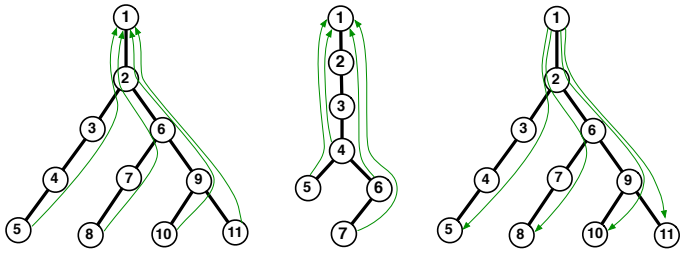
TABLE V: Information sources.

Information source	Information	Unit	Symbol
Transmission information	Channel load	Factor	f_{CL}
Neighbour table	Receiver load	Duty cycles	dp_{RL}
Retransmission count	Retransmission count	Duty cycles	dp_{RC}
Neighbour table	Duty cycle index	Time	$Node_{DL}$



(a) Line (b) Parallel (c) Merging (d) Cross Scenario

Fig. 3: Small-scale network topologies for basic evaluations.



(a) Stream (b) Event (c) Burst Scenario

Fig. 4: Medium-scale network topologies with corresponding traffic pattern.

B. Implemented Protocol Stack

BEAM and H2HR have been integrated into the Contiki network stack, cf. Figure 5. The implemented network stack can be executed on different sensor node platforms. On top of the link layer, we use Contiki's RIME layer to interconnect link and network layer. On the network layer, μIP is used. UDP is used on the transport layer. On the application layer we developed the *UDP end-to-end reliability* (UDP-E2E) protocol to add an end-to-end reliability mechanism. UDP-E2E is based on the sequence number mechanism of RMST [6] and uses negative acknowledgments to request retransmissions.

C. Evaluation of BEAM/H2HR Protocol Optimization Techniques

This subsection evaluates different protocol versions and optimization techniques applied to BEAM/H2HR. Table VI shows the protocol properties used for performance evaluation if not stated otherwise.

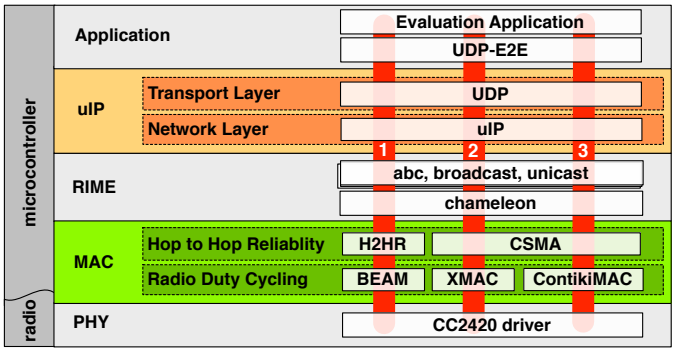


Fig. 5: Reliable Contiki compliant network stacks with UDP.

TABLE VI: Measurement setup for transmission delay optimization evaluations

Strobe mode	Including payload
BEAM-payload	40 bytes
Duty cycle length	125 ms
Transport protocol	UDP
End-to-end reliability	no

TABLE VII: Required energy for different kind of channel checks.

Acknowledgments type	periodic CCA checking	Energy [μJ]	Standard deviation [μJ]
software acknowledgment	no	322.5	0.44
software acknowledgment	yes	116.1	0.0040
hardware acknowledgment	no	82.79	0.012
hardware acknowledgment	yes	**38.92**	0.0038

1) Acknowledgment Mechanism: We compare energy consumption of hardware and software acknowledgments with or without periodic CCA checks. Periodic CCA checks are used to save energy between data transmissions and receiving acknowledgments. Otherwise, the sender will be active after a data transmission until an acknowledgment has been received. The results in Table VII show the required energy for the different CCA check and acknowledgement options. Hardware acknowledgments, e.g., using the *AUTOACK* function of the CC2420, are clearly more energy efficient than software acknowledgments. Periodic CCA checks are also more efficient than remaining active until receiving the acknowledgment. Based on these results, we use hardware acknowledgments and periodic CCA checking for the final BEAM version.

2) Beacon Strobe Transmission Delay Optimizations: Originally, BEAM immediately starts sending beacon strobes after having received a single frame from the upper layer H2HR. The beacon strobe transmission delay optimization estimates the next wake-up of the receiver. The measured energy costs at two different traffic load values using the small-scale scenarios in the real world WSN testbed are shown in Figure 6. The transmission delay optimization (denoted by SYNC in Figure 6) works perfectly at higher traffic load. Because the transmission delay optimization significantly reduces energy consumption and interferences, the final BEAM version uses the transmission delay optimization.

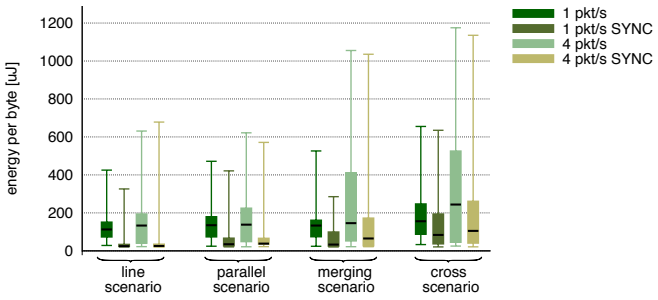


Fig. 6: Energy per byte with and without transmission delay.

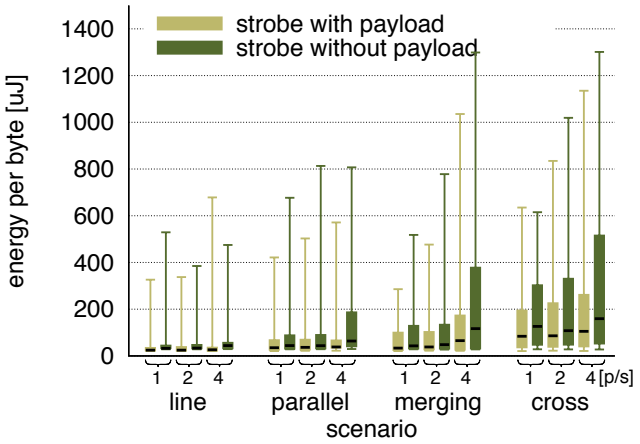


Fig. 7: Energy consumption with 40 bytes payload for involved nodes.

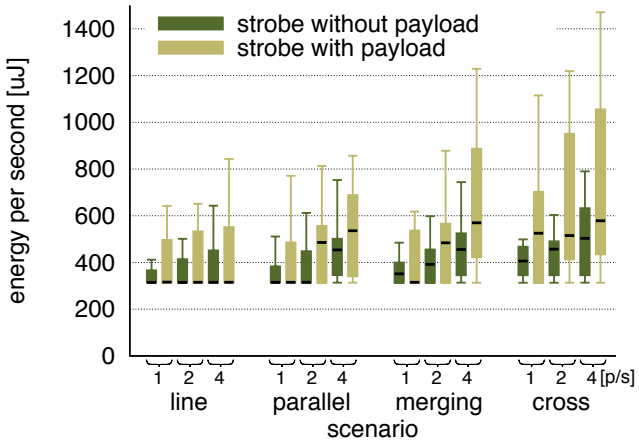


Fig. 8: Energy consumption with 40 bytes payload for non-involved nodes.

3) Beacon Strobe Modes: Both BEAM modes use hardware acknowledgments and transmission delay optimization. We evaluate the two versions in the real world WSN testbed with small-scale network topologies. We add six more nodes to each scenario to evaluate the impact of the beacon strobe mode onto non-involved neighbour nodes. Non-involved neighbour nodes do not forward any traffic, because they are never addressed. In each scenario we sent 10'000 packets. Figure 7 shows the corresponding measured energy for both beacon strobe modes and forwarding 40 bytes. In all scenarios, the variant using beacon strobes with payload requires less energy, especially at high traffic rates with more challenging interferences. The reasons for this observation are:

1) With short beacon strobes, the receiver node has to receive at least two beacon strobes. This requires more energy than receiving a single beacon strobe including payload.
2) With short beacon strobes, additional energy may be required to send the data part after receiving the early acknowledgment.
3) The short beacon strobe mode performed more frequently a retransmission as the receiver only has one chance to receive the data after the early acknowledgment.

According to Figure 8, all non-involved nodes require more energy when using the mode with beacon strobes including payload. Nevertheless, the overall energy required by beacon strobes including payload is lower. Therefore, for the final version of BEAM we use beacon strobes including payload mode. Moreover, beacon strobes including payload are more robust against interferences. They require only two instead of four successful transmissions. In case of bit errors in the data frame, the receiver can just wait for the next beacon strobe.

4) Duty Cycles: We evaluate the highest useful duty cycle or shortest wake-up interval respectively to maximize throughput at high traffic load. The lowest duty cycle is used by all idle nodes and in case of low traffic load. We use the line and parallel scenarios in the *small-scale testbed* for these evaluations. The application on the sender node sends

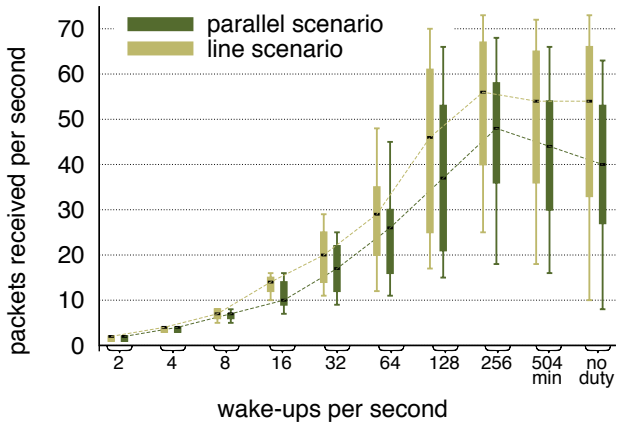


Fig. 9: Throughput of different duty cycles.

64 packets per second towards the network stack. A lot of them are dropped due to insufficient bandwidth. Figure 9 shows the successfully transmitted packets for different duty cycles. In both scenarios, throughput continuously increases up to 256 wake-ups per second. Thus, in the final BEAM version we use a maximum of 256 wake-ups per second. To define a meaningful default wake-up interval, we used the line scenario in the real world WSN testbed. We performed energy measurements for different duty cycles for one hour during day time and for one hour after midnight. Figure 10 shows the measured energy consumption for different default duty cycles. For our further evaluations, we used a default of 8 wake-ups per second for BEAM, because the required energy is reasonably low and ContikiMAC as well as XMAC are also using this value per default.

5) Traffic Prediction: BEAM's traffic prediction adapts the duty cycle according to the expected traffic load. We used the real world WSN testbed small-scale network topologies to compare BEAM with traffic prediction (Figure 12) and traffic monitoring (Figure 11). In contrast to traffic monitoring, traffic prediction allows the receiver to know that packets are pending. Figure 13 shows the associated total packet loss. The

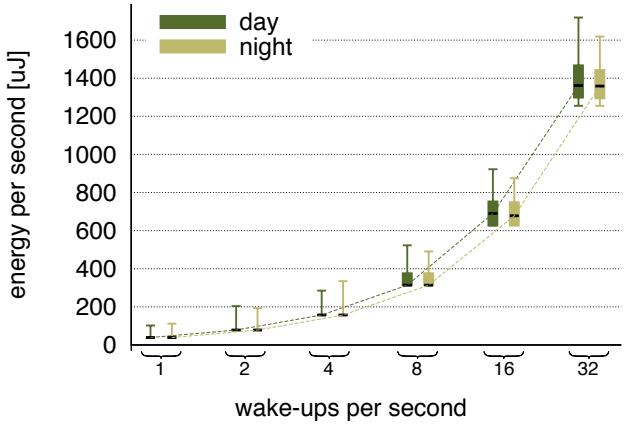


Fig. 10: Energy consumption of different duty cycles.

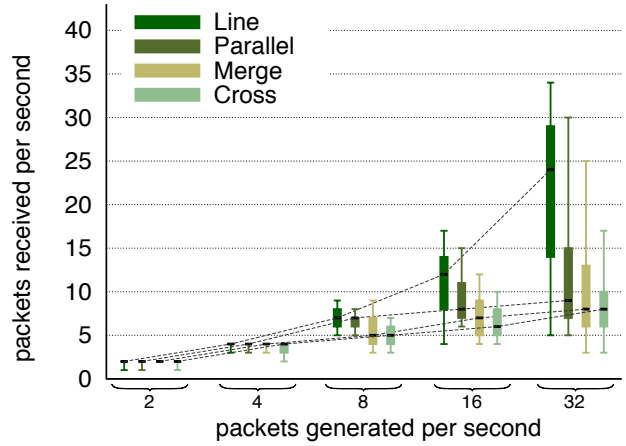


Fig. 11: Throughput with traffic monitoring.

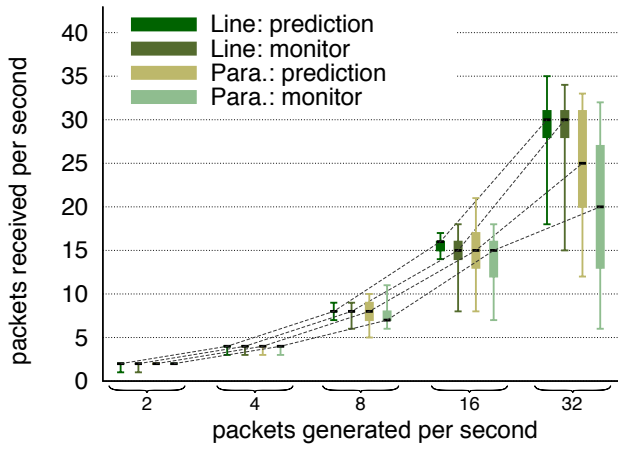


Fig. 12: Throughput with traffic prediction.

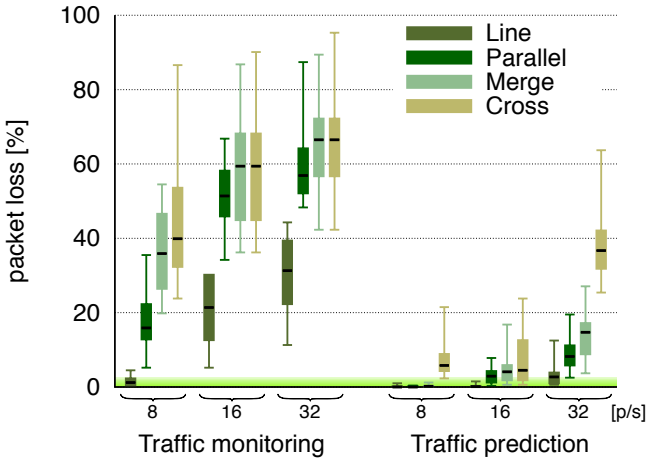


Fig. 13: Packet loss for traffic monitoring and prediction.

two lowest traffic rates (2 and 4 packets/s) show a packet loss ratio below 4% and are skipped in Figure 13 to gain a better overview. The shaded area on the bottom of Figure 13 depicts the highest end-to-end packet loss that can be recovered by our end-to-end retransmissions. Higher packet loss can not be recovered for the given rate of the packet stream. Additional traffic caused by end-to-end reliability mechanisms would only increase packet loss instead of reducing it. The results clearly show that using traffic prediction performs significantly better than traffic monitoring. The final version of BEAM, therefore, uses traffic prediction for duty cycle adaptation.

6) Packet Aggregation: Packet aggregation requires the cooperation of BEAM and H2HR. Up to three UDP/μIP packets with very short application payload can be aggregated into one BEAM frame. For the evaluation we used 37 bytes payload to enable aggregation of several packets. We used the real world testbed small-scale network topologies. Each network topology is tested with two different traffic load values. 10'000 packets are generated per UDP flow. Figure 14 shows the measured energy consumption. For lower traffic load, packet aggregation shows neither an impact on energy consumption nor on reliability. As soon as BEAM must adapt the duty cycle, traffic load is high enough to aggregate packets. For high traffic load, e.g., 4/8 packets/s, packet aggregation can prevent or at least decrease congestion by reducing the number of concurrently forwarded packets. The reduction of forwarded packets decreases the total number of required beacon strobe transmissions, which reduces interferences. Therefore, packet aggregation increases throughput. The final versions of BEAM and H2HR support packet aggregation to optimize performance during high traffic load.

D. Comparing BEAM/H2HR to Real World WSN Protocols

This section compares our final protocol versions of BEAM/H2HR with already existing link layers protocols, namely XMAC/CSMA and ContikiMAC/CSMA, for packet oriented radio modules. The presented measurements are all made in the real world WSN testbed using the three medium-scale scenarios. In this subsection BEAM represents the BEAM/H2HR protocol stack, ContikiMAC represents ContikiMAC/CSMA and XMAC represents XMAC/CSMA.

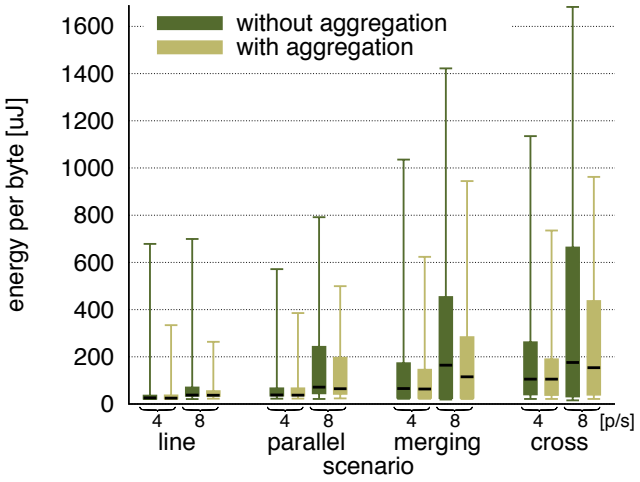


Fig. 14: Energy consumption for packet aggregation.

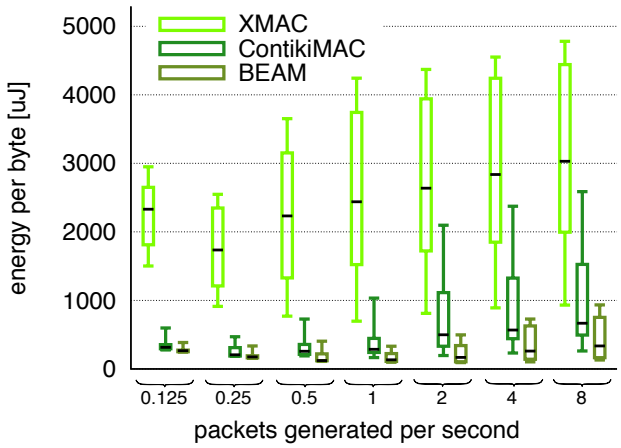


Fig. 15: Energy consumption of the UDP streaming scenario for involved nodes.

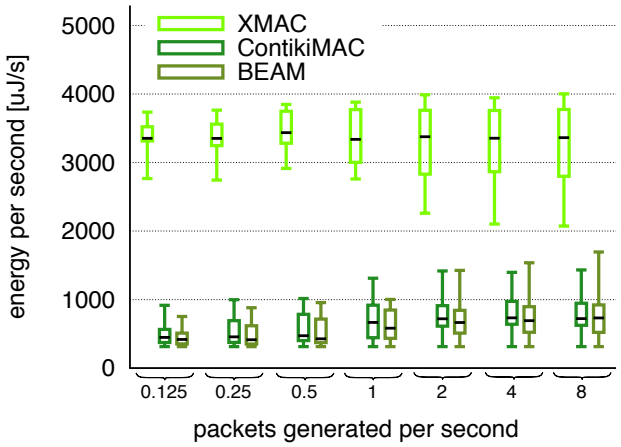


Fig. 16: Energy consumption of the UDP streaming scenario for non-involved nodes.

1) Energy Consumption: First, we compare the energy required to forward traffic in the stream scenario. We analyze different generated traffic rates from 0.125 to 2 packets per second on each of the four end-to-end flows. We use UDP packets with a link layer payload of 40 bytes. For each configuration, we sent 50'000 packets. Figure 15 shows the measured energy required to forward one byte. Below 0.25 packets per second, only one single packet is usually forwarded in the entire network at the same time. This results in quite predictable energy consumption for low traffic load. The total energy consumed by the radio module increases with additional traffic load, while the required energy per byte decreases with additional traffic load. Above 0.25 packets per second, there is an increasing probability of multiple packets at the same time in the WSN. This causes inter-flow interferences, which have to be handled by the reliability mechanisms, i.e., by longer beacon strobe transmissions and additional hop-by-hop retransmissions. Energy costs of ContikiMAC and BEAM clearly increase for above 0.25 packets per second. Energy costs of XMAC only increase slightly. The reason is that the packet error rate of XMAC is too high to be handled by the H2H protocol (CSMA). This means that the number of packets to be forwarded decreases, if they are lost early. Most packets are dropped by CSMA on the first hops.

XMAC requires clearly more energy than ContikiMAC and BEAM, because it requires significantly longer listen intervals to execute periodic channel checks. Moreover, XMAC uses the radio channel capacity less efficiently than ContikiMAC or BEAM, mainly because of the lack of optimized beacon transmission delays. Thus, XMAC requires clearly more beacon strobes to forward a packet. The beacon strobe mechanism of XMAC is more vulnerable to interferences than the beacon strobe mechanisms of ContikiMAC and BEAM. This results in additional retransmissions by CSMA. ContikiMAC and BEAM require nearly the same energy for periodic channel checks at low traffic load. BEAM requires less beacon strobe transmissions at low traffic than Contiki, due to the more precise transmission delay mechanism and less congestion. BEAM achieves a clearly more efficient channel usage than ContikiMAC. Non-involved neighbour nodes periodically perform CCA checks but never have to forward any packet. Figure 16 shows that XMAC requires clearly more energy than the other protocols due to the longer listen intervals to execute periodic channel checks. The non-involved nodes of ContikiMAC and BEAM show similar energy consumption.

2) Reliability and Maximum Throughput: Reliability strongly depends on traffic load, which has strong impact on internal interferences. Figure 17 shows that XMAC/CSMA experiences the highest packet loss. UDP-E2E is not able to repair all lost packets in this case. ContikiMAC with UDP-E2E is able to forward over 99.8% of the packets up to a traffic load of 0.25 packets per second along each of the four paths of the medium-scale stream scenario. BEAM is able to forward up to 2 packets per second without errors. End-to-end reliability mechanisms are only able to retransmit packets successfully, if the hop-to-hop reliable mechanism does not drop too many packets. Every packet dropped on link layer generates many additional interferences by end-to-end retransmissions. The more reliable hop-by-hop reliability mechanisms are, the better

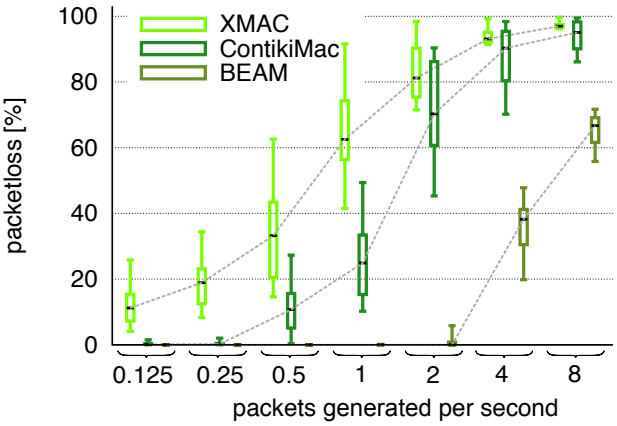


Fig. 17: Packet loss rate.

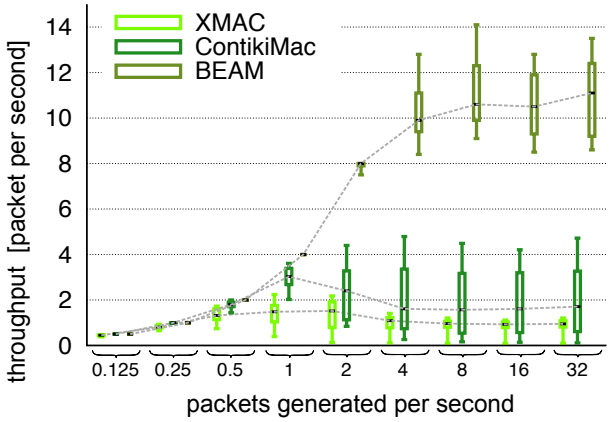


Fig. 18: Maximum throughput.

is the overall reliability performance of the network stack.

Throughput is closely related to packet loss. We generated traffic load, which is much higher than the maximum throughput the underlying protocols can handle. In Figure 18 XMAC and ContikiMAC show significantly lower maximum throughput. The data rate of XMAC and ContikiMAC decreases, if the maximum throughput has been reached, while BEAM is able to keep the data rate, if the maximum throughput has been reached.

VI. Conclusions

In this paper we compared the performance of our contributed link layer stack to existing real world WSN link layer stacks for energy efficient packet oriented radio modules. For evaluations we used a real world WSN testbed deployed over three floors of an office building. Our link layer stack, containing BEAM and H2HR, was the most energy efficient and reliable of the evaluated link layer stacks. Moreover, our contributed link layer protocols show significantly higher throughput than other evaluated protocols. This is due to the fact that our contributed link layer protocols are able to handle intra-flow and inter-flow interferences.

References

[1] A. Dunkels, B. Groenvall, T. Voigt: Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. IEEE Workshop on Embedded Networked Sensors, pp. 455–462. Tampa, USA (2004)

[2] IETF working group Routing Over Low power and Lossy networks (roll), http://datatracker.ietf.org/wg/roll/

[3] IETF working group Constrained RESTful Environments (core), http://datatracker.ietf.org/wg/core/

[4] M. Anwander, G. Wagenknecht, T. Braun, K. Dolfus: BEAM: A Burst-Aware Energy-Efficient Adaptive MAC Protocol for Wireless Sensor Networks. International Conference on Networked Sensing Systems (INSS), pp. 61–72, Kassel, Germany (2009).

[5] G. Wagenkecht, M. Anwander, T. Braun: Hop-to-Hop Reliability in IP-based Wireless Sensor Networks - a Cross-Layer Approach. International Conference on Wired/Wireless Internet Communications 2009 (WWIC'09), pp. 195–202, Enschede, The Netherlands (2010)

[6] F. Stann, J. Heidemann: RMST: Reliable data transport in sensor networks. Proceedings of the First International Workshop on Sensor Net Protocols and Applications, pp. 102–112, Anchorage, USA (2003).

[7] Texas Instruments CC2420: 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver, http://www.ti.com/product/cc2420

[8] M. Buettner, G. Yee, E. Anderson, and R. Han.: X-MAC: a short preamble MAC protocol for duty cycled wireless sensor network. International Conference on Embedded Networked Sensor Systems (ACM SenSys), Boulder, USA, (2006).

[9] A. Dunkels: The ContikiMAC Radio Duty Cycling Protocol. Techreport T2011:13 Swedish Institute of Computer Science, Kista, Sweden, (2011).

[10] G. Coulson, B. Porter, I. Chatzigiannakis, C. Koninis, S. Fischer, D. Pfisterer, D. Bimschas, D. Braun, P. Hurni, M. Anwander, G. Wagenknecht, S. Fekete, A. Krller, T. Baumgartner: Flexible experimentation in wireless sensor networks. Communications of the ACM 55, 1, pp. 82–90, (2012).

[11] W. Ye, J. Heidemann, D. Estrin: An Energy Efficient MAC Protocol for Wireless Sensor Networks. IEEE International Conference on Computer Communications (INFOCOM), pp. 1567–1576, New York, USA (2002)

[12] L. van Hoese, P. Havinga: A Lightweight Medium Access Protocol (LMAC) for Wireless Sensor Networks: Reducing Preamble Transmissions and Transceiver State Switches. 1st International Workshop on Networked Sensing Systems, pp. 205–208, Tokyo, Japan (2004)

[13] R. Musaloiu, C.J.M. Liang, A. Terzis: Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), pp. 421–432, St. Louis, USA (2008)

[14] A. El-Hoiydi, J. D. Decotignie: WiseMAC: An Ultra Low Power MAC Protocol for Multihop Wireless Sensor Networks. International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS), pp. 18-31, Turku, Finland (2004)

[15] A. Dunkels, L. Mottola, N. Tsiftes, F. Osterlind, J. Eriksson, N. Finne: The Announcement Layer: Beacon Coordination for the Sensornet Stack. European Conference on Wireless Sensor Networks (EWSN), pp. 211–226, Bonn, Germany (2011)

[16] P. Hurni and T. Braun: MaxMAC: a Maximally Traffic-Adaptive MAC Protocol for Wireless Sensor Networks. European Conference on Wireless Sensor Networks (EWSN), pp. 289–305, Coimbra, Portugal (2010)

[17] P. Suarez, C.G. Renmarker, T. Voigt, A. Dunkels: Increasing ZigBee network lifetime with X-MAC ACM Workshop on Real-World Wireless Sensor Network (REALWSN), pp. 13–18, Glasgow, Scotland (2008)

[18] S. Rangwala, R. Gummadi, R. Govindan, K. Psounis: Interference-aware fair rate control in wireless sensor networks ACM SIGCOMM Computer Communications Review, pp. 63–74, Vol. 24, No. 4 (2006)

[19] C. Cano, B. Bellalta, M. Oliver: Receiver-Initiated vs. Short-Preamble Burst MAC Approaches for Multi-channel Wireless Sensor Networks Lecture Notes in Computer Science, Springer, pp. 23-32, Vol. 7479 (2012)