Backbone MAC for Energy-constrained Wireless Sensor Networks[1]

Markus Wälchli, Reto Zurbuchen, Thomas Staub and Torsten Braun
Institute of Computer Science and Applied Mathematics
University of Bern
Neubrückstrasse 10, CH-3012 Bern, Switzerland
Email: {waelchli | zurbuche | staub | braun}@iam.unibe.ch

***Abstract*—In this paper we propose a routing backbone construction mechanism that exploits and uses the synchronization messages exchanged by synchronized contention-based MAC protocols. Due to the usage of synchronization messages no additional control traffic is required to setup the routing backbone. Every node running a synchronized contention-based MAC protocol follows a given listen/sleep cycle. Because routing is supported by the backbone, non-backbone nodes can temporarily turn off their radios for multiple listen/sleep cycles. Thus, additional energy can be saved. Accordingly, non-backbone nodes do not have to wake up in every listen/sleep cycle to synchronize with other nodes, but wake up only if required, i.e., if they have to report some sensor readings to a base station. In this case, they synchronize to the backbone, send their data, and go back to sleep after successful transmission. Our approach is applicable to rather static networks with mainly source-to-sink traffic. Most monitoring applications are of this kind.**

***Index Terms*—Sensor networks, medium access control, backbone construction**

I. Introduction

Synchronized contention-based MAC protocols follow periodic listen/sleep cycles. Within the listen period they synchronize to each other and allocate the medium for subsequent data transmissions. During sleep cycle, they either transmit pending data or they sleep for the whole time. The synchronization procedure is similar for most protocols. They mainly differ in the kind of organizing the subsequent channel allocation and data transmission sequence.

In this paper we propose to use the periodically exchanged synchronization (SYNC) messages to construct a routing backbone on the MAC layer. The SYNC messages intrinsically provide all network nodes with neighborhood information, which allows the construction of the backbone. No additional control traffic is needed. Moreover, nodes not required for routing can turn off their radio and can go to sleep for a long-sleep period that lasts for multiple listen/sleep periods. In this long-sleep period non-backbone nodes only wake up if they have to report some sensor readings. If this is the case, the respective non-backbone node wakes up, synchronizes to a node in the backbone, sends its sensor readings to the backbone node and goes back to sleep again. Henceforward, the backbone node is responsible to route the sensor readings of the non-backbone node to the base station (over the backbone). If a backbone node has to report some sensor readings, it does this directly.

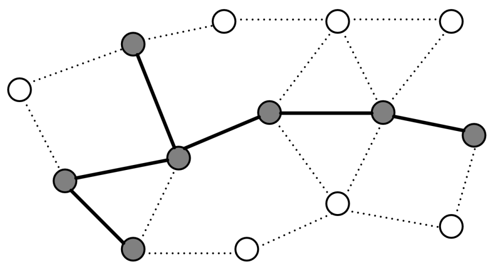


Fig. 1. Routing backbone in a sensor network.

An example of a backbone in a sensor network is depicted in Fig. 1. Backbone nodes are colored gray, while non-backbone nodes are colored white. Communication links in the backbone are illustrated by the bold lines, while other communication links are indicated by dashed lines. Every non-backbone node in Fig. 1 is adjacent, i.e., it possesses a communication link, to some node in the backbone. Hence, the network is connected and routing can be performed over the backbone.

In our work, the backbone construction is based on connected dominating sets (CDS) which are characterized as follows: A dominating set (DS) of a graph $G = (V, E)$ is a subset $V' \subset V$, where each node in $V - V'$ is adjacent to some node in V'. A CDS is a dominating set which builds a connected subgraph of G. To minimize the number of backbone nodes it is desirable to find a minimum connected dominating set (MCDS) of G. Finding an MCDS is however NP-complete [1]. Consequently, heuristics are applied. We propose two fully distributed approaches.

Existing backbone construction mechanisms operate above the MAC layer and do not profit from the information provided by the SYNC messages. Consequently, either additional control packets or a cross-layer approach are required. Furthermore, temporarily turning off the radios of non-backbone nodes is barely considered.

The rest of the paper is organized as follows: In Section II relevant related work considering contention-based medium access control and backbone construction is discussed. Our backbone construction mechanism is presented in Section III.

[1]The work presented in this paper was partly supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

Simulation results are provided in Section IV. The paper ends with conclusions and an outlook on future work in Section V.

II. Related Work

In this section different relevant medium access control (MAC) approaches are presented. MAC protocols can mainly be divided into contention-based protocols and in scheduled protocols which divide the medium access into contention-free time slots that are assigned to the network nodes [2]. Contention-based MAC protocols can be further divided into synchronized and asynchronous MAC protocols [3]. Considering backbone construction and routing, we focus on research in constructing proactive routing backbones.

A. Medium Access Control

Synchronized contention-based MAC protocols for multi-hop wireless sensor networks such as S-MAC [4], T-MAC [2] or DW-MAC [5] are based on low duty-cycles and traffic-adaptive wake-up periods. Running these protocols, every network node maintains a synchronized listen/sleep schedule.

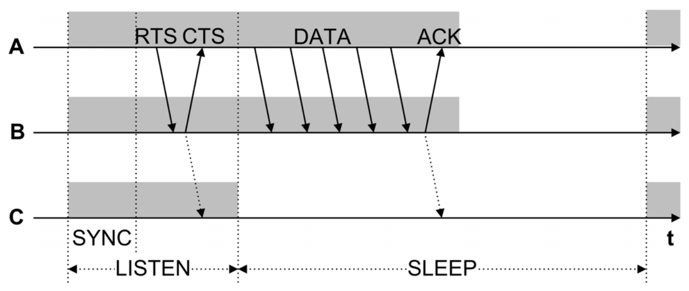


Fig. 2. Synchronized Duty Cycle MAC Protocol.

A typical scenario is shown in Fig. 2. In the contention-based SYNC period network nodes exchange synchronization messages. In the subsequent channel allocation period, node A that has some data pending for transmission allocates the channel with RTS/CTS. During sleep period nodes either transmit their data or sleep if no data is pending for transmission. Node C can go to sleep, because it is not affected by the data transmission between A and B. Synchronized nodes build virtual clusters. Whenever possible, nodes synchronize their listen/sleep schedules to reduce control traffic overhead. In [6] we have proposed a simple local synchronization scheme to achieve a common listen/sleep schedule for all nodes. Thus, a duty cycle for the SYNC period of approximately 1.6% could be achieved. All synchronized contention-based MAC protocols implement a similar SYNC period. They mainly differ in optimizing their traffic-aware behavior, i.e., they optimize the duration for the channel allocation and data exchange sequence [2]. They optionally introduce additional mechanisms to optimize throughput [4], [5].

PMAC [7] exploits neighborhood information to determine local listen/sleep schedules. In PMAC local traffic patterns are learned from the communication observed by a node. These traffic patterns are used to determine the duration of the sleep period of a node. If no traffic is indicated, a node can go to sleep for a long time. PMAC has been tested with constant bit rate communication along a single path. This scenario is tailored to PMAC. A key problem is that all nodes that are not located along a path are in a long-sleep state. Thus, their activation is delayed.

Scheduled MAC protocols such as LMAC [8], TRAMA [9] and A-MAC [10] also require the exchange of periodic synchronization messages. However, unlike contention-based protocols, the operation of scheduled MAC protocols is based on the concept of clusters. In general, they require a cluster leader that allocates slots to its cluster members. Thus, the problem of virtual clustering is not present as the nodes are per se organized into clusters. On the other hand, scheduled MAC protocols require very precise synchronization and scale rather poorly.

Apart from protocols that require synchronization, asynchronous contention-based MAC protocols such as WiseMAC [11], B-MAC [12], X-MAC [13], RI-MAC [3] have been proposed. [11], [12] and [13] are based on preamble-sampling. These protocols send long preambles to reach neighboring nodes that are currently asleep. RI-MAC [3] avoids the transmission of preambles. RI-MAC receiver nodes announce their availability by beacon messages. Based on the reception of such a beacon, a sender node transmits its pending data to the receiver. Asynchronous MAC protocols achieve low duty cycles. However, they require the exchange of preambles or beacons and support broadcast operations poorly. Finally, an integrated routing support on the MAC layer is difficult to implement in asynchronous mode.

B. Backbone Construction and Routing

The selective disconnection of nodes that are temporarily not required for routing has been addressed by numerous protocols on the network layer and above.

SPAN [14] is a distributed algorithm where nodes make local decisions about sleeping or joining a routing backbone. A periodic exchange of up to three-hop neighborhood information is required. GAF [15] nodes form location-dependent virtual clusters. In each cluster at least one active sensor node must be present for routing. GAF requires signaling and a location service (e.g., GPS).

Backbone construction mechanisms based on connected dominating set (CDS) theory have been proposed in [16], [17], [18] and [19]. In [16] a two-tiered approach has been chosen. In a first step the CDS consisting of all nodes with non-adjacent neighbors is constructed. This initial CDS is then reduced by applying two pruning rules. The algorithm requires two-hop neighborhood knowledge. In [18] the pruning rules have been adapted to consider energy levels instead of local connectivity. [17] first constructs a maximal independent set (MIS), which is then connected by choosing appropriate intermediate nodes in a second step. The approach is time-consuming and static. A simple greedy procedure has been proposed in [19]. Backbone nodes broadcast their neighborhood information. Based on this information receivers contend for backbone membership by setting a timer according to their number of neighbors that are not yet covered by the backbone. The algorithm is simple and fast, but minimizes the number of nodes in the CDS rather poorly.

Our approach aims at extending network lifetime rather than minimizing the CDS. Therefore, battery levels are considered in addition to connectivity. Moreover, the CDS is periodically recomputed to account for changes in the energy distribution over the network. Finally, the CDS procedure exploits SYNC messages, which avoids additional control traffic.

III. Routing Backbone on the MAC Layer

Synchronized contention-based MAC protocols support arbitrary communication patterns. On the other hand, many sensor network applications only need source-to-sink communication. Considering this communication pattern and rather static networks, synchronized contention-based MAC protocols can be enhanced to extend network lifetime by exploiting the neighborhood information provided by SYNC messages. Based on this information, a backbone rooted at the base station can be set up. In addition, non-backbone nodes can go to sleep for multiple listen/sleep cycles to save extra energy.

A. Introduction

The operations of a backbone node B and a non-backbone node R are illustrated in Fig. 3. B periodically synchronizes to neighboring backbone nodes, i.e., it periodically sends SYNC messages. R only wakes up if it has some data to be reported. It synchronizes to the backbone, allocates the channel with RTS/CTS and transmits the data (indicated by the burst of arrows). Having received a confirmation (ACK) from B, R goes back to sleep again.

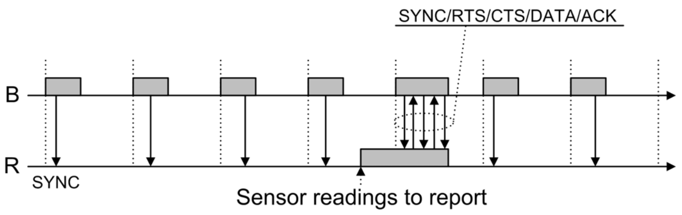


Fig. 3. Backbone node (B) following a listen/sleep cycle and non-backbone node (R) in long-sleep state.

In the current evaluations in Section IV the learning period lasts for 10 minutes. The CDS is maintained for 50 minutes. During this time, non-backbone nodes remain asleep unless they have to report some data. Thus, non-backbone nodes save additional energy. Backbone nodes perform the normal listen/sleep cycle and do not waste more energy than running the unaltered MAC protocol. They have some information piggybacked, but compensate for this by overhearing fewer SYNC messages due to the temporarily decreased network density. Non-backbone nodes do not send SYNC messages in long-sleep state. Accordingly, our protocols save the transmission of SYNC messages compared to the standalone MAC protocol.

The batteries of nodes in the backbone have higher charge levels than the batteries of non-backbone nodes. Accordingly, the backbone is recomputed from time to time (currently every hour) taking the current battery levels of the nodes into account. In every CDS setup phase nodes with high battery levels are favored. In the following, two CDS setup algorithms based on the information collected from SYNC messages are proposed. The first algorithm needs two-hop neighborhood information, whereas the second approach requires only knowledge of the immediate neighborhood. The first algorithm is simpler and terminates faster, but requires more information.

B. CDS based on Multipoint Relaying

To determine the subset of next hops in the CDS, an algorithm similar to the Multipoint Relaying protocol (MPR) [20] has been used. Any backbone node (dominator) elects its next-hop dominators such that they connect the two-hop neighbors most effectively.

1) Neighborhood Discovery: The SYNC messages are slightly modified to discover the two-hop neighborhood. In addition to the current sender s_1, also the sender s_2 of the last SYNC message received by s_1 is added to the SYNC message. Thus, each receiver of a SYNC message learns both, the direct neighbors (s_1) and the two-hop neighbors (s_2) over time. Because the network is assumed to be static, only minor changes occur once the two-hop neighborhood has been learned. The reliability of the neighborhood discovery period and the needed accuracy in neighborhood knowledge are discussed in Section III-D.

In addition to neighbor node s_2, the battery level of each sender is further added to the SYNC message. The battery level is needed to account for changes in the energy distribution in the network. Otherwise, always the nodes with best connectivity would be chosen into the backbone, leading to fast battery depletions of these nodes.

2) Backbone Construction: Having exchanged SYNC messages for a period long enough to allow network nodes to learn their one- and two-hop neighborhood, the CDS process is initiated by the base station. The next-hop dominator set of a dominator node x is calculated as follows:

```
MPR-based dominator election

input: Two-hop neighbor list $L_2$ of node $x$;
       One-hop neighbor list $L_1$ of node $x$;
ouput: MPR set of next-hop dominators of node $x$;

begin
repeat
       If $\exists$ a $y \in L_1$ with exactly one link to a $z$ in $L_2$
          add $y$ to MPR;
          Remove all $z$ in $L_2$ which are now covered;
       else
          For each $y$ in $L_1$ do
                Compute number $\delta(y)$ of nodes in $L_2$ connected
                via $y$ that are not covered by an MPR node;
          Select the $y$ with highest product of $\delta(y)$ and
          remaining battery level into the MPR set;
          Remove all $z$ in $L_2$ which are now covered;
until all two-hop neighbors are covered;
end
```

The CDS setup process is started by the base station (S). Each dominator elected by the algorithm is informed about its state by an extended SYNC message. The procedure continues until all network nodes are covered. To force successful signaling, the extended SYNC messages are retransmitted either until passively confirmed by overheard SYNC messages from successors or until the maximum number of retransmissions is reached.

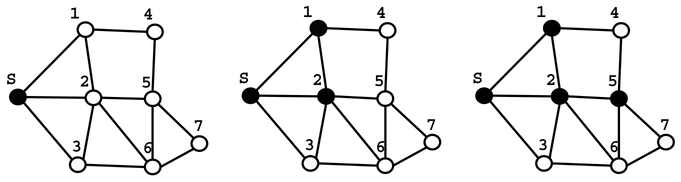


Fig. 4. Example of dominator election with MPR-based CDS.

An example is shown in Fig. 4. The base station S is the starting dominator (black). The set of next-hop dominators of S is $\{1, 2\}$, because 1 is the only node to reach two-hop neighbor 4 and 2 is the only node to reach two-hop neighbor 5. Nodes 1 and 2 cover all two-hop neighbors of S. S informs them by piggy-backing the dominator list $\{1, 2\}$ to its next SYNC messages. Upon reception, both nodes become black and compute their own set of next-hop dominators. Node 1 immediately terminates its election process because no uncovered two-hop neighbors remain (nodes 3, 5 and 6 are covered by nodes S and 2). Node 2 determines node 7 as last uncovered two-hop neighbor. Because node 7 can be reached via nodes 5 and 6, step 2 of the algorithm is applied. Both nodes have the same number of uncovered neighbors, i.e., δ is 1 for both of them. Accordingly, the node with higher remaining battery level (node 5) is chosen into the CDS. The CDS setup ends when node 5 is informed about its state. The algorithm achieves good MCDS approximation, though the result is suboptimal. The MCDS would only consist of nodes S, 2 and 5.

C. *Negotiation-based CDS*

The negotiation-based CDS (N-CDS) computes the CDS without two-hop neighborhood knowledge. One-hop neighborhood information is sufficient.

1) Neighborhood Discovery: The N-CDS again learns the neighborhood information from SYNC messages. No extension of normal SYNC messages is needed, though. Only the ID of the SYNC sender is required, which is transmitted per default. We again assume that after a given initialization period each node knows its one-hop neighbors.

2) Backbone Construction: Base station S is again defined as starting node. The dominator status is determined based on a negotiation-based CDS building process that consists of four steps which are described in the following:

N-CDS based dominator election

1. Every dominator node broadcasts an extended SYNC message called DOMINATOR. This message contains the neighborhood list of the dominator.
2. Each receiver becomes dominated and computes a priority according to its product of battery level and number of remaining uncovered two-hop neighbors, i.e., of nodes that are not yet dominator or dominated.
3. The dominated nodes locally exchange their calculated priorities in a special SYNC message called DOMINATED.
4. The node with highest priority is elected into the backbone.

Because the DOMINATOR message contains the neighborhood list of the dominator, every receiver is informed about neighboring dominated nodes. Consider node 1 in the example in Fig. 5. Node 1 is a neighbor of node S. Having received the DOMINATOR message from S, which contains nodes 1, 2 and 3 in the neighbor list, node 1 can determine its neighbor node 2 as dominated. The same applies to every other receiver of a DOMINATOR message. In step 3, all dominated nodes exchange DOMINATED messages containing the priority of the sender. Because each dominated node knows the dominated nodes in its one-hop neighborhood, the dominated node knows from which nodes it has to receive DOMINATED messages in order to be able to determine the significance of its own priority (step 4). Accordingly, when a dominated node has learned all priorities from neighboring dominated nodes, it becomes a dominator if it has the highest priority among the group. If the priority of a node becomes 0, i.e., it has no uncovered neighbors left, it enters non-backbone state after having broadcast its status in a DOMINATED message.

Due to contention or packet loss, it might happen that a dominated node cannot receive a DOMINATOR or DOMINATED message from a neighbor node in this mutual negotiation process. Accordingly, the node would keep waiting for this message. To prevent this deadlock, each node sets a challenge timer. If the node did not receive any message from a specific neighbor during this period, it assigns priority 0 to the respective node. To summarize, the following terminations of the algorithm per node are possible:

1. A node determines that all its neighbors are covered. In this case the node enters non-backbone state.
2. At the moment a node determines that it has the highest priority, it enters the backbone.
3. A dominated node has still some uncovered neighbor nodes when the challenge timer expires. In this case the node enters the backbone too.

The third item is motivated as follows. Consider an uncovered node y that is only connected over a dominated node x. In order to guarantee network connectivity, node x must become a dominator. Hence, node x becomes dominator when the challenge timer expires. The challenge timer of x will expire since it has an uncovered neighbor y. An example of the N-CDS algorithm is shown in Fig. 5. For simplicity all nodes have the same energy level. Thus, only the node degree determines the priority of a node.

Dominator nodes are colored black. Dominated nodes color themselves gray and uncovered nodes are white. Gateway node S starts the N-CDS algorithm by broadcasting a DOMINATOR message. Having received this message, neighbors 1, 2 and 3 color themselves gray. Node 2 computes the highest priority,

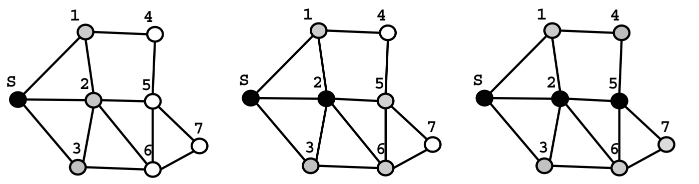


Fig. 5. Example of dominator election with N-CDS.

because it has two neighbors (5 and 6) which are not yet covered. Nodes 1 and 3 both have only one uncovered neighbor. After having exchanged the priorities via SYNC messages, node 2 determines the highest priority (becomes black) and broadcasts the next DOMINATOR message. Receivers 5 and 6 become dominated, i.e., they color themselves gray, and exchange their priorities. Node 5 has the higher priority (neighbors 4 and 7 are not yet covered) and becomes a dominator. Nodes 3, 6 and 7 go to sleep as soon as they have overheard the required DOMINATOR messages from nodes S, 2 and 5. Node 4 goes to sleep if it has overheard the DOMINATED message from node 1 and the DOMINATOR message from node 5. Node 1 finally goes to sleep if it overhears the DOMINATED message from node 4. If communication among some of the nodes was not successful, the challenge timer would expire and some of nodes 1, 3, 4, 6 or 7 might become dominators too.

Due to its negotiation character N-CDS requires a longer CDS setup time than MPR-based CDS. On the other hand, no two-hop neighborhood information is needed. The computed CDS is optimal in the example. In general, MPR-based CDS is expected to perform slightly better due to its two-hop neighborhood knowledge, though. We have developed a CDS setup mechanism that combines both approaches and provides mobility support on the networking layer [21]. It approximates the MPR-based CDS by piggy-backing the neighbor list of the dominator on DOMINATED messages. As N-CDS, the mechanism requires negotiation, leading to the same drawbacks.

D. Reliability and Backbone Reconstruction

Neighborhood discovery and backbone construction are reinitialized by the base station after every long-sleep period. Thus, the current connectivity and energy distributions in the network are considered after every long-sleep period. Local path update strategies could be an alternative. Such approaches have not further been considered due to their complexity.

The accuracy of the one-hop neighborhood information is more critical than the accuracy of the two-hop neighborhood information. Accurate two-hop neighborhood knowledge is only needed to optimize performance of the MPR-based approach. However, the two-hop neighborhood information must be complete enough to ensure that the MPR-based algorithm does not terminate without covering all network nodes. Complete coverage has been achieved in every simulation, though. Collecting neighborhood information by SYNC messages, in particular collecting two-hop neighborhood information, is a time-consuming task. Therefore, we have determined when neighborhood information must be updated. If a node depletes, the according neighborhood information of this node becomes invalid. This has impact on the backbone construction mechanisms. Two possibilities need to be distinguished:

- **Invalid one-hop neighborhood information**: Unavailable dominators could be elected. This must not happen. Hence, the one-hop neighborhood is relearned in every neighbor discovery period, i.e., before every backbone setup.
- **Invalid two-hop neighborhood information**: In the worst case a redundant dominator is elected. This is not desirable but neither critical. Hence, two-hop neighborhood information is only deleted if the according node has not been overheard for a certain amount of time.

In all simulations the complete one-hop neighborhood information and most two-hop neighbors were relearned in every neighbor discovery phase. Two-hop neighbors that have not been overheard were still known due to the timeout criteria.

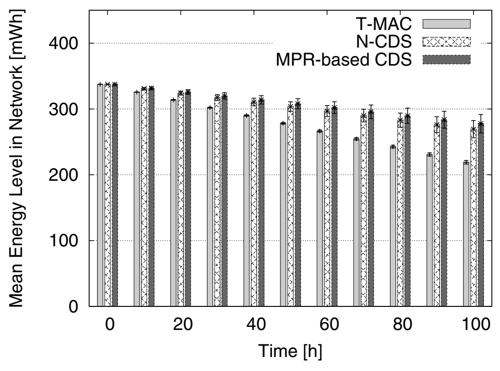
IV. Evaluation

Both approaches have been implemented on top of T-MAC in the OMNeT++ [22] network simulator. For compatibility and comparison reasons T-MAC and an optimal pre-configured shortest-path routing tree have been implemented according to [2]. Any implementation of a routing algorithm on top of T-MAC might have different impact depending on the routing protocol and would thus be difficult to be evaluated. Therefore, we decided to compare our approaches to optimal benchmarks.

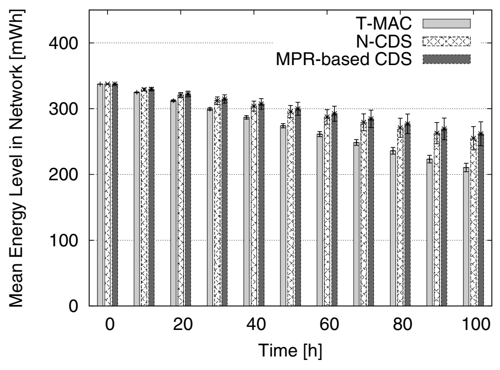
The CDS is rebuilt every hour. The backbone period lasts for 50 minutes and the learning period for 10 minutes. Non-backbone nodes wake up every minute to send their sensor readings to the base station (see also Fig. 3). Having successfully transmitted their data, they go back to sleep again. Unless being in non-backbone state, every sensor node follows a periodic listen/sleep cycle of 610 ms [2]. The minimum duty cycle is 26.5 ms if no data has been scheduled for transmission. Three network sizes of 50, 100 and 200 nodes have been evaluated. The network density is 15 neighbors in average. The network topology has been randomly generated taking network connectivity into account. Any experiment has been repeated 20 times. 95% confidence intervals have been computed. Sensor node properties are set according to values from the Embedded Sensor Board (ESB) platform [23]. The nodes operate in the 868 MHz frequency band. Transmission and interference range are 37 m and 52 m, respectively. The data rate is 115.2 kbps. The energy consumption in transmission mode is 5.2 mA. Idle listening and receiving both require 4.7 mA. In sleep mode the radio consumes 5 μA.

A. Average Energy Consumption in the Network

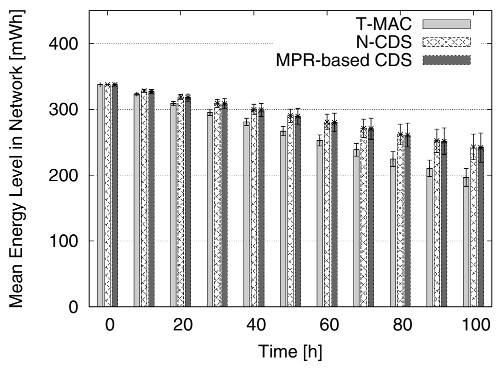
In a first evaluation the average remaining energy level of the batteries in the network has been logged for each protocol. Each node has initially been charged with 335 mWh. 100 hours have been simulated. It has been ensured that all nodes have enough energy so that they do not run out of energy during simulation time. Node failures were not in the scope of the current evaluation.



(a) Network size of 50 nodes.



(b) Network size of 100 nodes.



(c) Network size of 200 nodes.

Fig. 6. Average remaining energy level per node.

Fig. 6 shows the average remaining energy level for all tested protocols and network sizes. Both N-CDS and MPR-based CDS perform better than T-MAC. As expected, MPR-based CDS is slightly more efficient than N-CDS. The results show that the energy savings by non-backbone nodes compensate for the piggy-backing of control data. The network size has some impact on the average consumed energy in the network. This increase in energy consumption is, however, due to the higher data traffic load caused by the increased number of reporting sensor nodes (every node transmits a data message every second). The backbone performance is not affected by the network size. In our evaluation T-MAC has been used. Because SYNC messages are used, T-MAC could be substituted by other synchronized MAC protocols. The benefits of the backbone should be the same. Because T-MAC has been provided with cost-free routing, implementing routing on top of T-MAC would additionally strain energy consumption of T-MAC.

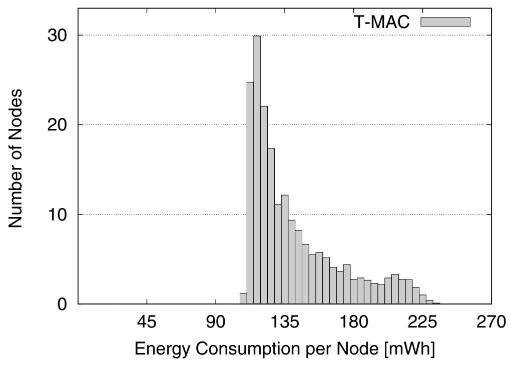
B. Energy Distribution in the Network

The goals of our work have been to provide routing on the MAC layer in order to save energy, and to distribute the energy consumption uniformly over the network. This uniform distribution of the energy consumption is more meaningful to extend network lifetime than the average consumed energy, because the probability that nodes discharge quickly is decreased. The depletion of nodes in particular leads to problems if irreplaceable nodes are affected. Fig 7 shows the energy distribution of the different algorithms in the network consisting of 200 nodes. The average energy consumption for this network size has been provided in Fig. 6(c). The results for networks consisting of 50 and 100 nodes are similar.

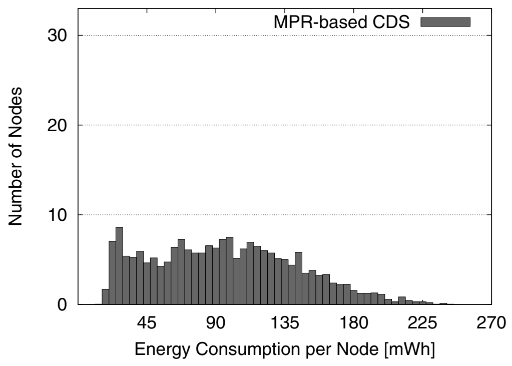
The distribution of nodes with similar energy consumption differs significantly between T-MAC and the CDS-based approaches. In T-MAC all nodes consume more than 100 mWh in 100 hours simulation time and the percentage of nodes with high energy consumption is higher than that of the CDS-based approaches. In T-MAC there are numerous nodes (about 30) with an energy-consumption of about 110 mAh (see Fig. 7(a)). The other nodes consume even more energy. On the other hand, the CDS-based approaches charge many nodes less.

The high fraction of nodes with low energy consumption in CDS-based approaches, i.e., the nodes on the left side in Fig. 7(b) and 7(c), indicates that the periodic setup of the CDS leads to a good energy consumption distribution. However, there are nodes with high energy consumption too. These are the nodes on the right side in Fig. 7(b) and 7(c). Examining the different solutions, neither N-CDS nor MPR-based CDS increase the number of heavily strained nodes compared to T-MAC. Accordingly, the CDS-based algorithms do not increase the number of quickly depleting nodes. The existence of such nodes cannot be avoided, though. The reason is that such nodes are often or always elected into the backbone, because there are no or few alternatives to route information to the base station. To circumvent this drawback, the node distribution in the network would have to be considered during deployment.

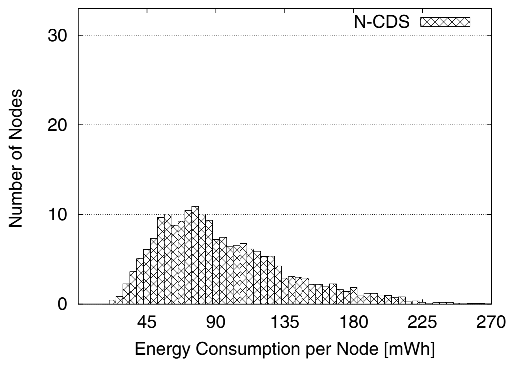
The cumulative distribution function (*cdf*) of nodes with a specific energy consumption is shown in Fig. 8. This function shows the percentage of nodes that consume a certain amount of energy, and it is used to assess the distributions in Fig. 7. MPR-based CDS and N-CDS both outperform T-MAC. Fig. 8 shows that with MPR-based CDS and N-CDS approximately 60% of all nodes consume less than 100 mWh during the simulation time of 100 hours. On the other hand, in T-MAC every node consumes more energy during the same time. It is



(a) T-MAC.



(b) N-CDS.



(c) MPR-based CDS.

Fig. 7. Number of nodes with a specific energy consumption.

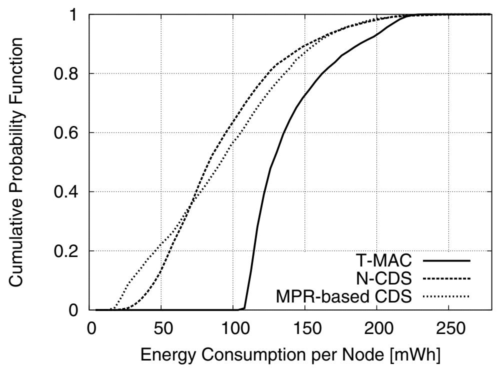


Fig. 8. Cumulative distribution function of energy consumption per node.

not possible to determine whether MPR-based CDS or N-CDS performs better.

To conclude, CDS-based algorithms do not only consume less energy in average (see Fig. 6(c)), but also lead to a better energy consumption distribution. The probability of network partitions is higher for T-MAC than for the CDS-based approaches due to the higher number of heavily strained nodes. Again, since SYNC messages are used, T-MAC could be substituted by other synchronized MAC protocols (e.g., by DW-MAC).

C. Packet Loss

Finally, the average data packet loss of the different protocols has been investigated. The results are depicted in Fig. 9. Only data transmissions have been considered.

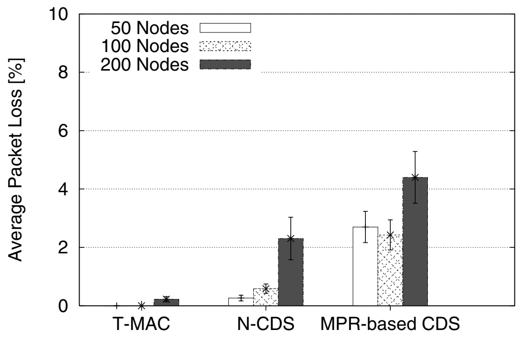


Fig. 9. Average packet loss per simulation.

The optimal routing implemented on top of T-MAC leads to very good performance. T-MAC introduces almost no packet loss in all simulations. On the other hand, both N-CDS and MPR-based CDS lead to packet loss of up to 4%, which is still reasonable. Implementing a routing protocol on top of T-MAC might increase packet loss too due to additional signaling and suboptimal routing decisions. Nevertheless, T-MAC could still perform better than the CDS-based approaches, because the CDS-based mechanisms select many backbone nodes with poor link quality.

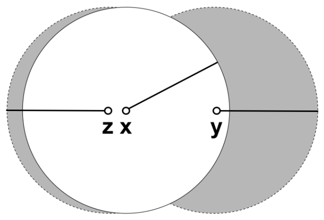


Fig. 10. Area of additional coverage of neighbor nodes of x.

The reason is depicted in Fig. 10. Since a remote neighbor y of x has a larger area of additional coverage (colored gray) than a nearby neighbor z, y in general connects more two-hop neighbors of x than z. On the other hand, remote neighbors y have a comparatively low Received Signal Strength Indicator

(RSSI) and are more exposed to interferences. Accordingly, these nodes have worse link quality than close neighbors.

The election procedures of both CDS-based approaches prefer neighbors with a high probability to reach additional nodes, because both CDS-based algorithms aim at optimizing additional coverage. However, exactly these nodes are the nodes with high probability of poor link quality, because they are located at the border of the parent dominator node. The impact could be lowered by monitoring the RSSI of incoming messages at the nodes. Thus, links with poor quality could be avoided in the backbone election procedures. In addition, symmetric links would thus be reinforced too.

V. Conclusions

In this paper the information provided by periodically exchanged SYNC messages used by synchronized contention-based MAC protocols has been exploited. This information has been used to setup routing backbones, i.e., MPR-based CDS and N-CDS, on the MAC layer. Thus, no additional signaling for routing is required. Non-backbone nodes turn-off their radios for many listen/sleep cycles unless they have some data to be sent to a base station. With this mechanism additional energy can be saved. Non-backbone nodes further avoid the transmission of SYNC messages in the skipped listen/sleep cycles. In order to distribute the energy consumption over all network nodes, the routing backbone has been recomputed from time to time. Thus, early network partitions could be prevented. It has been shown that better energy consumption distributions could be achieved by the backbone approaches than with the standalone MAC protocol. The backbone construction has been completely implemented on the MAC layer based on SYNC messages. Normal data transmissions are not affected by the backbone setup and maintenance procedures.

In the current CDS implementation, data throughput has slightly been affected due to the election of many backbone nodes with poor link quality. This impact has been identified and could be decreased by monitoring and rejecting links with poor link quality.

References

[1] B. N. Clark, C. J. Colbourn, and D. S. Johnson, “Unit disk graphs,” *Discrete Mathematics*, vol. 86, no. 1-3, pp. 165–177, December 1990.

[2] T. van Dam and K. Langendoen, “An adaptive energy-efficient mac protocol for wireless sensor networks,” in *Proc. of the 1st international conference on Embedded networked sensor systems (SenSys ’03)*, Los Angeles, CA, 2003, pp. 171–180.

[3] Y. Sun, O. Gurewitz, and D. B. Johnson, “RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks,” in *Proc. of the 6th ACM conference on Embedded network sensor systems (SenSys ’08)*, Raleigh, NC, USA, 2008, pp. 1–14.

[4] W. Ye, J. Heidemann, and D. Estrin, “Medium access control with coordinated adaptive sleeping for wireless sensor networks,” *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, June 2004.

[5] Y. Sun, S. Du, O. Gurewitz, and D. B. Johnson, “DW-MAC: a low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks,” in *Proc. of the 9th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc ’08)*, Hong Kong, Hong Kong, China, 2008, pp. 53–62.

[6] M. Wälchli, R. Zurbuchen, T. Staub, and T. Braun, “Gravity-based local clock synchronization in wireless sensor networks,” in *Proc. of Networking 2009*, Aachen, Germany, May 2009.

[7] T. Zheng, S. Radhakrishnan, and V. Sarangan, “PMAC: An adaptive energy-efficient mac protocol for wireless sensor networks,” in *Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS’05)*, 2005, pp. 95–107.

[8] L. van Hoesel and P. Havinga, “A lightweight medium access protocol (LMAC) for wireless sensor networks: Reducing preamble transmissions and transceiver state switches,” in *Proc. of the International Conference on Networked Sensing Systems (INSS ’04)*, Tokyo, Japan, June 2004.

[9] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, “Energy-efficient collision-free medium access control for wireless sensor networks,” in *Proc. of the 1st international conference on Embedded networked sensor systems (SenSys ’03)*, Los Angeles, CA, 2003, pp. 181 – 192.

[10] Y. Liu and L. M. Ni, “A new MAC protocol design for long-term applications in wireless sensor networks,” in *Proc. of the 13th International Conference on Parallel and Distributed Systems (ICPADS ’07)*, Hsinchu, Taiwan, 2007, pp. 1–8.

[11] A. El-Hoiydi and J.-D. Decotignie, “WiseMAC: An ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks,” in *Proc. of the 9th IEEE International Symposium on Computers and Communications (ISCC 2004)*, vol. 1, Alexandria, Egypt, June 2004, pp. 244–251.

[12] J. Polastre, J. Hill, and D. Culler, “Versatile low power media access for wireless sensor networks,” in *Proc. of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, Baltimore, MD, USA, November 2004, pp. 95–107.

[13] M. Buettner, G. V. Yee, E. Anderson, and R. Han, “X-MAC: a short preamble mac protocol for duty-cycled wireless sensor networks,” in *Proc. of the 4th international conference on Embedded networked sensor systems (SenSys ’06)*, Boulder, Colorado, USA, 2006, pp. 307–320.

[14] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, “Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks,” *Wireless Networks*, vol. 8, no. 5, pp. 481–494, 2002.

[15] Y. Xu, J. Heidemann, and D. Estrin, “Geography-informed energy conservation for ad-hoc routing,” in *Proc. of the 7th annual international conference on Mobile computing and networking (MobiCom ’01)*, Rome, Italy, July 2001, pp. 70–84.

[16] J. Wu and H. Li, “On calculating connected dominating set for efficient routing in ad hoc wireless networks,” in *Proc. of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications (DIALM ’99)*, Seattle, WA, 1999, pp. 7–14.

[17] P.-J. Wan, K. M. Alzoubi, and O. Frieder, “Distributed construction of connected dominating set in wireless ad hoc networks,” *Mobile Networks and Applications*, vol. 9, no. 2, pp. 141–149, 2004.

[18] J. Wu, F. Dai, M. Gao, and I. Stojmenovic, “On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks,” *IEEE/KICS Journal of Communications and Networks*, vol. 4, no. 1, pp. 59–70, March 2002.

[19] D. Zhou, M.-T. Sun, and T.-H. Lai, “A timer-based protocol for connected dominating set construction in IEEE 802.11 multihop mobile ad hoc networks,” in *Proc. of the 2005 Symposium on Applications and the Internet (SAINT’05)*, 2005, pp. 2–8.

[20] A. Quayyum, L. Viennot, and A. Laouiti, “Multipoint relaying: An efficient technique for flooding in mobile wireless networks,” INRIA, Sophia Antipolis, France, Tech. Rep., 2000.

[21] M. Wälchli, T. Bernoulli, and T. Braun:, “Receiver-based backbone construction and maintenance for wireless sensor or multi-hop networks,” in *Proc. of the Workshop on Mobile Ad-Hoc Networks (WMAN 2007)*, Bern, Switzerland, March 2007, pp. 409–420.

[22] A. Varga, “OMNeT++: a component-based, modular and open-architecture simulation environment,” December 2008. [Online]. Available: http://www.omnetpp.org/

[23] Scatterweb, “The self-organizing wireless communication platform,” October 2007. [Online]. Available: http://www.scatterweb.net