Analyzing Energy Consumption in a Gossiping MAC Protocol[⋆]

Haidi Yue, Henrik Bohnenkamp and Joost-Pieter Katoen

Software Modeling & Verification
Department of Computer Science 2, RWTH Aachen University
D-52056 Aachen, Germany
Fax: +49 241 80 222 17
{haidi.yue|henrik|katoen}@cs.rwth-aachen.de

Abstract. In this paper, we analyze the energy-efficiency of a TDMA protocol (gMAC) for gossiping-based wireless sensor networks. In contrast to most schedule-based TDMA protocols, slot allocation in gMAC is decentralized, allowing adaptation to evolving network configurations. The protocol, modeled in the MoDeST language, is evaluated using the discrete-event simulator of the Möbius tool suite. We investigate the impact of collision-detection mechanisms, initiator positioning, and random silence on the gMAC energy efficiency. As a result, we find the number of active slots that optimize the trade-off between low energy consumption and fast information dissemination.

Keywords: TDMA, energy, wireless sensor networks, formal modeling, simulation

1 Introduction

The Dutch company CHESS develops wireless sensor networks (WSN) comprising battery-powered mobile sensors that exchange data via gossiping-based communication. The sensors are mobile, act in a fully decentralized manner—there is, e.g., no leader—and battery recharging is not possible. CHESS WSNs are, for instance, used in the Dutch flower auction market in Aalsmeer where thousands of trolleys carrying flowers are equipped with autonomous routing capabilities.

To realize an energy-efficient communication mechanism supporting sensor mobility, CHESS developed a TDMA-variant, called gossip-based MAC [15] (gMAC, for short) to control medium access. In TDMA, the time is divided into frames which are subdivided into slots in which nodes

[⋆] This research has been funded by the EU under grant number FP7-ICT-2007-1 (QUASIMODO) and the DFG Excellence Cluster UMIC.

send or receive or idle. Whereas in most TDMA protocols a central access node decides which slot is to be used by which node, in a setting with mobile nodes, a fixed schedule can no longer be maintained: ever changing neighborhood relations between nodes invalidate defined schedules and cause collisions in communication. Therefore, gMAC exploits a fully decentralized slot allocation—each node decides on its own when to send and when not. Moreover, the sensors communicate with each other in an epidemic broadcast-like manner. This forms the basis of gossiping applications in which nodes continuously exchange data [10]. This all-to-all communication prevents the usage of simple (and frequently adopted) energy-conserving strategies like switching off a radio when no communication with the central access node takes place. In our setting, nodes have to listen to messages sent by all their neighbors and only idle during the non-active slots. To enable an implementation with simple (and cheap) microprocessors, CHESS designed gMAC as simple as possible. Therefore, gMAC does not incorporate techniques such as dynamic frame lengths as in EC-TDMA [16], transmission length indications as in A-MAC [12], or organizing neighbor information in a spanning tree as in TreeMAC [14].

gMAC is designed to work with a rather simple radio working in the 2.4 GHz band. Like any TDMA protocol, time is divided into frames. A frame consists of an idle and an active period. The active period is divided into equal-length slots, in which nodes send (once per frame and node) or receive. The beginning of new frames is synchronized among all neighboring nodes up to a certain precision. As nodes decide autonomously in which slot they send, collisions may occur. gMAC supports an indirect collision avoidance mechanism: node X keeps track of the slots in which it received something. This list is communicated by X to its neighbor Y as piggy-back information in the payload. If X did not receive an item in Y's send slot, Y infers that it is using the same send slot as another node, and randomly chooses a new, free one. As this mechanism cannot ensure the complete absence of collisions, a node can randomly decide to not use its send slot, and listen instead.

In this paper, we focus on the energy-efficiency of the protocol under the assumption of perfect clock synchronization. In particular, we investigate the effectiveness of the gMAC collision-detection mechanism (which fraction of real collisions is detected?), initiator positioning of gossiping messages (what is the influence of the position of the gossip initiator on latency?), and random silence on the gMAC energy efficiency (how does this protocol aspect impact collision detection?). As a result, we find the number of active slots that optimize the trade-off between low energy consumption and fast information dissemination for various system configurations. We consider first static network configurations to study the basic protocol mechanisms, and then determine the influence of node mobility. The main findings of our study that were also of interest to CHESS are (i) random silence improves collision detection significantly, (ii) the optimal number of active slots is about the number of neighbors plus one, and (iii) mobility lowers the number of failed transmissions.

Although our analysis technique is simulation, we deliberately take a drastically different approach from using standard simulation packages such as NS2, Opnet, OMNET or GloMoSim, to mention a few. Our starting point is a model of the protocol in the MoDeST language [3], a formalism that supports the modular specification of distributed systems in a mathematically rigorous, though user-friendly, manner. As MoDeST has a formal operational semantics in terms of stochastic timed automata, the simulation model obtained from the protocol models is unambiguous. The automata underlying MoDeST models are simulated using Möbius [7, 5], a discrete-event simulator that has been intensively used in dependability analysis. The formality of the modeling language allows not only the integration with other formal analysis tools (such as model checkers), but, more importantly, yields semantically sound simulation runs. Together with the fact that we do not model entire protocol stacks but rather abstract from lower layer effects, this avoids many of the credibility problems of standard simulations [6, 1]. This approach has, amongst others, been applied to analyze the energy consumption of Zigbee and IEEE 802.15 [8], and the analysis of a plug-and-play communication protocol [4]. Main limitation of our approach is that MoDeST models may exhibit nondeterminism, which cannot be simulated. We thus have to check our models prior to simulation on the presence of nondeterminism.

Organization of the paper. Section 2 describes the CHESS gMAC protocol. Section 3 describes the modeling assumptions and the experimental set-up. Section 4 focuses on results concerning collision detection, and Section 5 focuses on energy consumption. Section 6 concludes.

2 The gMAC Protocol

The gMAC protocol divides time in fixed-length *frames*. A frame is divided in an active and idle period, and both periods are subdivided into *slots* of equal length. A node in the network is synchronized with its immediate neighbors at the beginning of a frame. A node randomly chooses an active slot as *send slot* (the *TX slot*). All other active slots are *receive* *slots* (*RX slots*). During the idle period, the radio is put in idle mode to save energy. In an RX slot, a node listens for incoming messages from neighboring nodes, in its TX slot it sends a message. When the active period is over, it switches to idle mode again, and so forth.

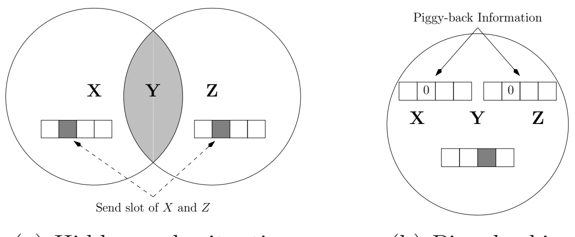
Let S be the number of slots within a frame, and $A \leq S$ the number of active slots. A is a crucial parameter in the protocol design, as the active operation phase costs much more energy than the idle phase. The CHESS network nodes are equipped with an ATMega64 microcontroller and a Nordic nRF24L01 [13] packet radio. The energy demands of the nRF24L01 radio are summarized in Table 1. A is usually much smaller than S. In the gMAC protocol implementation with the aforementioned processor, S=1129, and $A = 8$.

mode	current
transmit	11.3 mA
receive	12.3 mA
idle	0.9 μA

Table 1. Energy demands of the nRF24L01 radio

When a node is powered on, it randomly chooses an active slot as TX slot. In each RX slot, it can receive a message of at most one other node. The well-known *hidden node problem* describes the scenario when more than one node sends messages to the same node in the same slot.

Figure 1(a) depicts a situation where nodes X, Y, Z are positioned such that the middle node Y is within the transmission range (the circles) of both other nodes, and both X and Z are outside each others range. If X and Z select the same TX slot, then their messages will collide in the intersection of their ranges. They cannot sense this themselves, and Y will receive no message at all as it cannot distinguish a collision from the situation where no message was sent.



(a) Hidden node situation (b) Piggybacking

Fig. 1. Hidden node problem and its detection

The gMAC protocol provides a piggy-back technique to make collisions detectable. With each pay-load message, the sender's perspective on the current slot allocation is also transmitted, which we call the *piggy-back* *information.* The piggy-back information is a sequence $(b_0, b_1, ..., b_{A-1})$, where $b_i \in \{0, 1\}$ for $0 \leq i < A$. $b_i = 0$ indicates that nothing has been received in slot i, either because nobody sent something, or due to a collision or message loss. $b_i = 1$ indicates that the sender received something in slot i, or that slot i is the sender's own current send slot. In the example in Figure 1(a), since Y cannot receive anything in the second slot, it writes a 0 in its piggy-back information at the corresponding position and reports this to X and Z on its turn to send, in the third slot (*cf.* Figure 1(b)). Based on this information from Y, nodes X and Z can conclude that there was a collision in their send slot. The gMAC protocol then stipulates that X and Z pick randomly a new send slot among the free active slots, to avoid further collisions. Note that it is possible that no free slot is available when a node needs one. This can happen when the nodes are in a very crowded environment and the number of neighbors exceeds the number of active slots (some of our simulation configurations cover this situation). In this case, the node will keep the old send slot despite the detected collision in that slot.

Although the piggy-back technique helps to detect many collisions, there are still some it cannot find. In Figure 2, node Y has the same send slot as X and Z, *i.e.,* they send and receive at the same time and will therefore never receive anything from each other in this slot, hence the collisions between them will not be detected or resolved. The reason for this is that the piggy-back technique requires at least one common neighbor which is not involved in the conflict, so that it can report the collision. The gMAC protocol provides one more mechanism to break this kind of conflict. When a node reaches its send slot, it can decide with a certain probability p to *not* send, but to listen. This gives a node a chance to overhear what is going on in its own send slot, and an opportunity to pick a new send slot, if necessary.

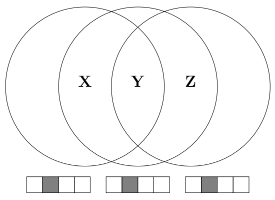


Fig. 2. Problematic scenario for piggy-back technique

3 Experimental Setup

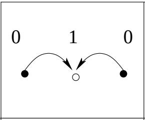
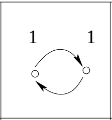
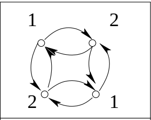
Modeling assumptions. In the real world, the interference range of a node's radio signal is usually larger than its effective transmission range. The magnitude of the interference range is not necessarily equal in every direction. Besides, the ranges can vary from time to time. Depending on the environment, the Nordic nRF24L01 radio has a range between $0.5m$ to $50m$. For the sake of simplicity, we adopt the approach chosen in [2] and use the closed unit disk model in which the interference range equals the transmission range, and is given by a radius r. All network nodes are assumed to have the same transmission range, which means the transmission between nodes is symmetric. We further abstract from other link layer mechanisms, i.e., message losses are assumed to be due to collisions only. gMAC incorporates a mechanism to synchronize clocks of neighboring nodes. Sufficient criteria that ensure the correctness of this clock-synchronization mechanism have recently been mathematically analyzed [9]. As our simulation models satisfy these criteria, we abstract from the clock-synchronization algorithm.

The gMAC protocol accommodates for tolerable de-synchronization by shortening the actual sending period and uses the difference to the slot length as a lead-in to and lead-out from the send period (the so-called guard times). This does of course influence the energy consumption, and therefore we incorporate the guard times in our model.

Set-up. The base model of our experiments is a 15×15 grid network of 225 nodes. Each node has a distance of 1 to its respective horizontal and vertical neighbors (*i.e.,* the distance to the diagonal neighbors is $\sqrt{2}$). A frame consists of 1129 slots. The number A of active slots is a crucial parameter in the protocol, and we analyze the behavior of the gMAC protocol for various A. Since in the real implementation A=8, we choose the transmission range r such that each inner node of the grid has 4 or 8 direct neighbors, respectively. We say a node is *randomly silent*, if it stays silent in its TX slot with some probability. This probability in the current implementation of the CHESS sensor node is $p = \frac{1}{16}$. We adopt this value in our model and use it for all experiments. To get insight into the influence of this parameter, we also performed experiments with other values of p, e.g. $p = \frac{1}{8}$. It turns out that the results pattern is similar to that with $p = \frac{1}{16}$, which that is with random silence, a higher percentage of collisions can be detected. The experiments focus on two major aspects: (i) the effect of the gMAC collision detection mechanism (piggy-backing and random silence), and (ii) the latency of message dissemination versus the required energy consumption. The confidence level of all simulations is set to 0.95 and the relative confidence interval is 0.1.

Collision analysis. We estimate the effectiveness of the collision detection mechanisms by counting both the real number of collisions that occurred in the network (referred to as *Failed Transmissions*, FT for short) and the number of collisions that are detected using the piggy-backing technique (the number of *Detected Collisions*, DC for short) in each frame. Note that although a node can detect collisions, it can neither distinguish with whom it collided nor how many nodes collided. Hence, when considering DC, we can only count the number of nodes that report collisions and not the real number of collisions, i.e. DC represents actually the number of nodes that detect collisions. The values for FT and DC are illustrated for different scenarios in Figure 3 where the number next to a node (small circles) indicates its TX slot. The right-most figure represents an extreme case, where the respective diagonal nodes send and receive at the same time, i.e., while the upper righthand and the lower lefthand nodes are sending, their messages collide at the upper lefthand and lower righthand nodes, and vice versa. Hence communication between all nodes fails, but no node is able to detect it. We vary the transmission range r and the number A of active slots as follows. In networks with at most 4 neighbors, A ranges from 4 to 10, and for at most 8 neighbors, A ranges from 6 to 12. Each of the experiments is run 100 times and lasts at least 1000 frames.

Latency vs. energy consumption. Second, we focus on the latency of message dissemination and the energy consumed by that. We consider the average time required and the total average energy consumed until a message is delivered to all network nodes. We say a node is *infected* if it has received a message. Initially, only one node is infected, the initiator. To get insight into the effect of the position in the network of the message initiator, we consider (cf. Figure 4): a corner node, a middle node at the border, and a center node. Again, the simulations are run for different values of r and A to investigate the influence of these parameters on gMAC's energy consumption. Each experiment is run 600 times.

			
DC	2	0	0
FT	2	2	8

● : Detected Collision ↷ : Failed transmission

Fig. 3. Three collision situations

Different settings. We run all aforementioned experiments for three network settings:

1. A static network without *randomly silent* nodes (for short *grid*),
2. A static network with *randomly silent* nodes (for short *grid+p*),

3. A network with node mobility but no *randomly silent* nodes (for short *grid+m*), so that we can obtain a clear comparison between static and mobile scenarios without influence of *randomly silent* nodes.

Since we want to investigate the influence of local changing of node position on the network, we model the mobility by rotating a fixed row (the fifth row) in the grid one position to the right. The node shifted out is shifted in on the other side. The row is rotated one position every 100 frames for the collision experiment, so that the network has enough time to stabilize after each shift. Since the average time required to deliver a message to all nodes is less than 30 frames, we rotate every 1 or 3 frames to investigate the influence of the moving rate on the latency.

4 Collision analysis

The different variants of gMAC in the different scenarios have been modelled in the MoDeST modelling language [3], and simulated in the Möbius tool set [7]. The models are available from `http://moves.rwth-aachen.de/~henrik/mmb10/`.

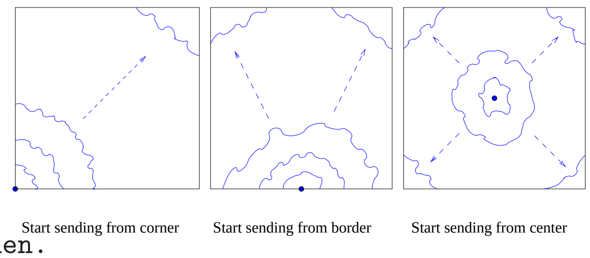
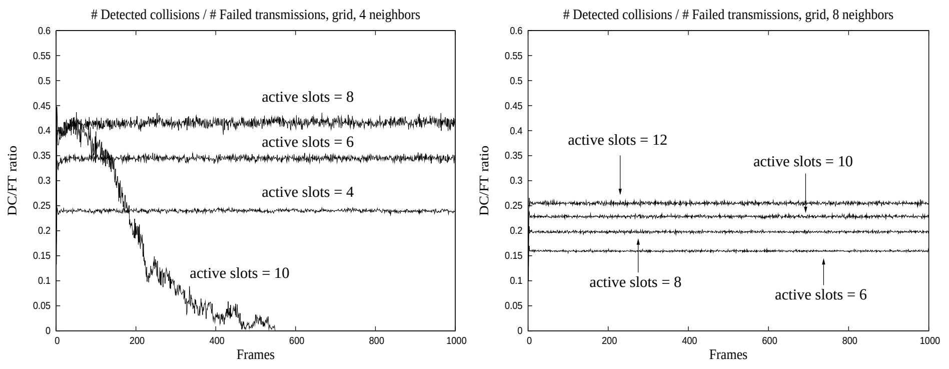

Start sending from corner
Start sending from border
Start sending from center

Fig. 4. Three different initiator positions

Static network. We consider the static grid model *grid* without *randomly silent* nodes. The transmission range is $r = 1.1 < \sqrt{2}$, *i.e.,* each inner node has 4 neighbors. Figure 5(a) shows the fraction $\frac{DC}{FT}$ versus the number of frames for different values of A. A larger percentage means that a larger fraction of collisions is detected by the gMAC piggy-backing method. The graph shows that for increasing A, a larger fraction of collisions is detected. This is confirmed for a network with 8 neighbors, cf. Figure 5(b). The almost horizontal straight lines show that for small A the randomly changing slot allocation does not reduce the number of collisions, but yields a more or less stable number of collisions. If A is large enough, $\frac{DC}{FT}$ goes to zero, as no collisions are detected anymore. This phenomenon occurs, e.g., for A=10 and $r = 1.1$, as can be seen in Figure 5(a). Failed transmissions may still occur, however; for A=10, e.g., on average 7 transmissions in the whole network fail without being detected (not depicted in Figure 5(a)).

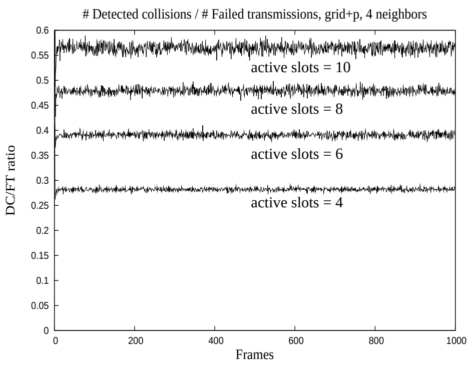
Apart from the fact that some collisions can never be detected, it is generally the case that A needs to be large—in comparison to the number of neighbors—before DC tends to go to 0. Our simulations have shown that, in the case of 4 neighbors, this is the case for $A \geq 9$; for 8 neighbors, this holds for $A \geq 23$.



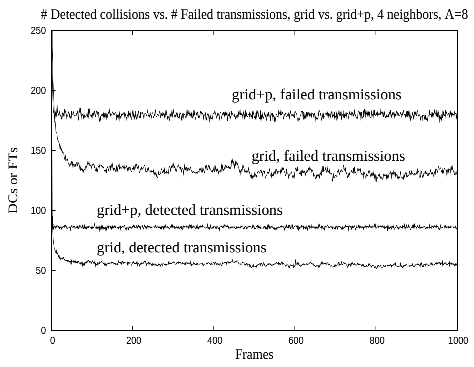
(a) grid, 4 neighbors (b) grid, 8 neighbors

Fig. 5. Collision detection in a static network

Random silence. Figure 6(a) depicts the results of the same experiment run on *grid+p* with 4 neighbors. The comparison with Figure 5(a) reveals that *grid+p* can detect a significantly larger percentage of collisions than *grid*. This percentage increases for larger A. Furthermore, in *grid+p*, even with A=10, the fraction $\frac{DC}{FT}$ does not go to 0 anymore. Our explanation for that is that in *grid+p* nodes are more often receiving than in *grid*, and thus more collisions are detected. Indeed, for A=8, DC in *grid+p* is significantly higher than for *grid*, cf. Figure 6(b). Figure 6(b) also shows that FT increases compared to *grid*. This is unexpected. We believe that the reason for this phenomenon is that, when A is at least the number of neighbors, then *randomly silent* nodes can turn a good slot allocation into a bad one. Consider Figure 7 (left), with 11 nodes (the numbers indicating their TX slots) and only two neighboring nodes in conflict (in slot 2). Let A=5. When the boxed node is silent in slot 2, it will detect a collision, and (randomly) chooses a free send slot, which is slot 3 (all others are in use). As the right figure shows, the boxed node is suddenly in conflict with four nodes (2-hop neighbors) rather than one, causing eight failed transmissions. The new slot allocation is worse than before. Of course, the case illustrated in Figure 7 is quite extreme. In general, when A is larger than the number of direct neighbors it will probably collide only with relatively fewer 2-hop neighbors.



(a) *grid+p* 4 neighbors



(b) DC and FT in *grid+p* and *grid*

Fig. 6. Collision detection under random silence

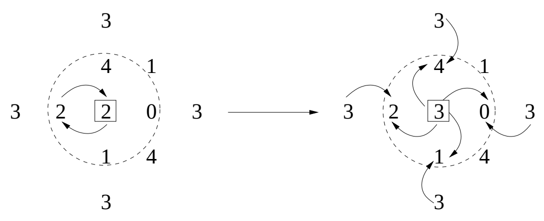
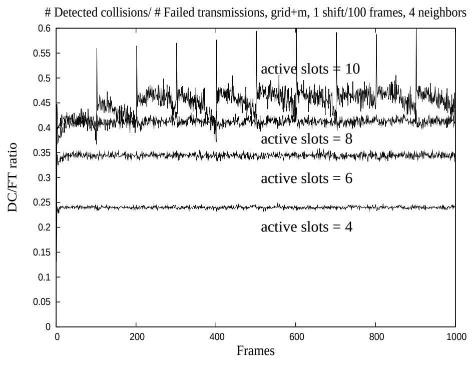


Fig. 7. Changing slot allocation by silent node

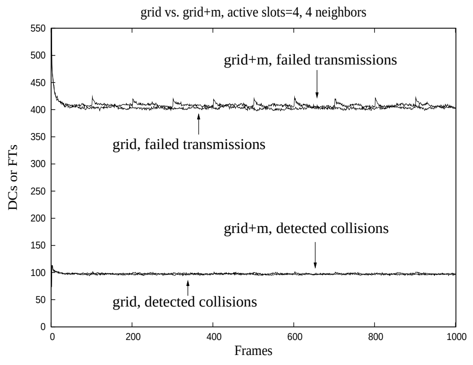
Node mobility. We now consider the influence of node mobility on $\frac{DC}{FT}$. Figure 8(a) shows the results for a network with $r = 1.1$ and a shifting frequency of $\frac{1}{100}$ frames. We can see for $A = 10$ that the amplitudes of the curves shortly after every 100 frames are quite significant, and between each two peaks, the curve tends to go down. For $A \leq 8$, the peaks nearly vanish. This means that for A so small that the static network can not reach a collision-free state, the influence of mobility on collision detection diminishes. In fact, when A is small, $\frac{DC}{FT}$, as well as DC and FT, especially DC, are almost unchanged under node mobility (*cf.* Figure 8(b)). For a higher rate of node shifting, *i.e.,* shifting every 30 frames, the result pattern is similar.

5 Latency vs. energy consumption

Static network. The second type of simulation is concerned with the energy efficiency of message propagation. By latency we mean the average time required to deliver a message to all nodes. Again, we first consider a



(a) *grid+m*, 4 neighbors, shifting every 100 frames



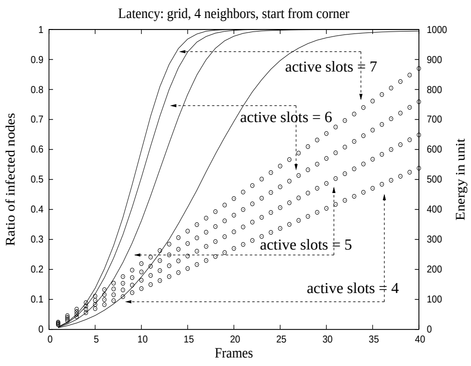
(b) DC and FT in *grid* and *grid+m*

Fig. 8. Collision detection under node mobility

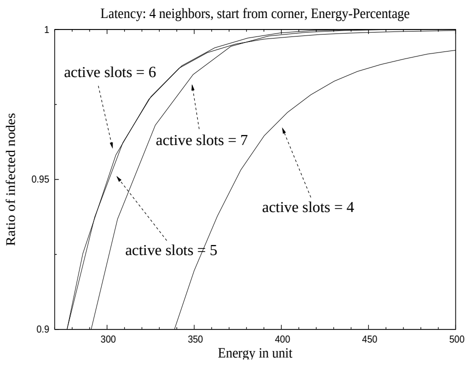
static network. Figure 9(a) shows the experimental results for transmission range r=1.1, and A ranging from 4 to 7. The message initiator is positioned in the corner. The *circle-lines* show the energy consumption (right y-axis) versus the number of frames, and the black, curved lines (left y-axis) show the ratio of infected nodes (i.e., nodes that have received a message) versus the number of frames.

The results confirm that for fixed A, there is a linear dependency between the energy consumption and the number of frames, which is characteristic for TDMA protocols. The slope depends on A; the larger A, the steeper the energy curves. For the message dissemination, it can be observed that after a short warm-up phase, the fraction of infected nodes drastically grows, after which this slowly progresses to one. For increasing A, the percentage of infected nodes converges to more quickly to one, i.e., message dissemination is faster.

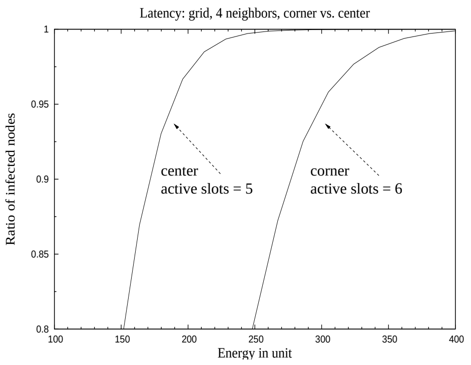
Obviously, the larger the A is, the lower the message latency becomes, but as a pay-off, the energy consumptions increases with larger A. In order to get insight into the trade-off between message dissemination and energy consumption, Figure 9(b) depicts an energy-percentage diagram, which shows the percentage of infected nodes versus the total energy needed to infect all nodes. One clearly sees that A=4 and A=7 are not economical and in the considered scenario, a network with 5 or 6 active slots provides the best result in terms of energy efficiency. Performing the experiments for $A = 8, 9$ and 10 reveals that these settings are less energy-efficient than for $A = 7$. For $A = 10$, e.g., the network tends to be collision-free, but requires twice as much energy as for $A = 5$ without offering a doubled propagation speed. We performed the experiment for three different initial



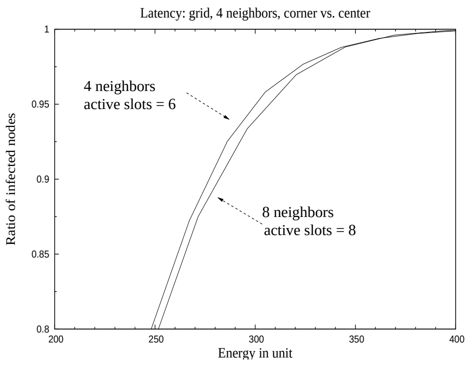
(a) grid, 4 neighbors, start sending from corner



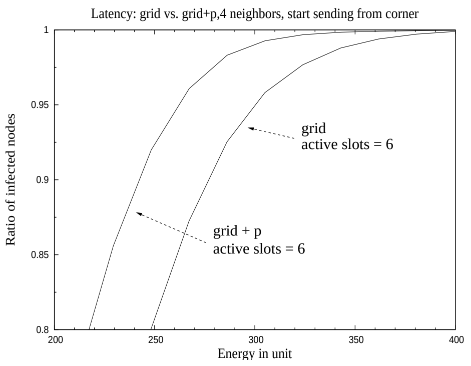
(b) Energy-percentage: grid, 4 neighbors, start sending from corner



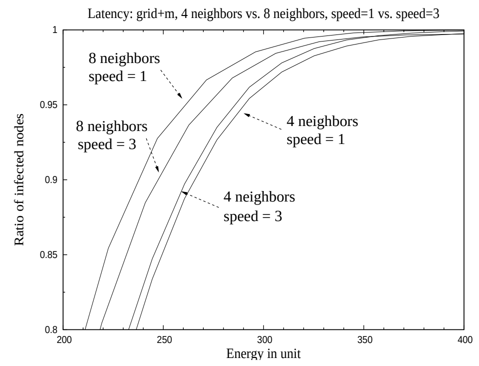
(c) corner vs. center, grid, both 4 neighbors



(d) 4 neighbors vs. 8 neighbors: grid, both start sending from corner



(e) *grid* vs. *grid+p*, both have 4 neighbors and start sending from corner



(f) different transmission ranges and shifting rates

Fig. 9. Latency vs. energy consumption

sending positions and different transmission ranges. All of them exhibit a pattern similar to Figure 9(b). The optimal values of A are summarized in Table 2.

Range \ Position	corner	border	center
4 neighbors	6	5	5
8 neighbors	8	9	8

Table 2. Optimal A values

Figure 9(c) shows another effect of changing the initial sending position. We put the most energy-efficient results from a network with 4 neighbors and initial sending position at the corner or the center in one graph. Obviously, starting from the center needs only two third energy of that starting from the corner. It does not come as a surprise that message dissemination from the center is more efficient than from a corner. However, when we consider the influence of network density on latency, we can see that with a fixed initiator, a network with 4 neighbors or 8 neighbors exhibits almost the same performance (*cf.* Figure 9(d)). This means, although a denser network can propagate messages faster (a result which we have not shown here), it takes still as much energy as in a less dense network to deliver a message to the whole network.

Random silence. The results for *grid+p* show a similar behavior, hence we will not present them here. Interesting is however the comparison between *grid* and *grid+p*. In Figure 9(e), we see the most economical results of *grid* and *grid+p*, both with 4 neighbors and the same initial sending position. The superiority of *grid+p* is quite clear, since roughly 15% energy can be saved if nodes are *randomly silent*. This is not self-evident, since for the used radio, receiving costs actually more energy than sending. We believe that the 15% drop in energy consumption is because *grid+p* has in general more opportunities to receive messages, which accelerates information dissemination.

Node mobility. For a network with mobility, we consider first the case of start sending from the corner. There are two options for transmission range (4 neighbors or 8 neighbors) and for the speed of shifting: every frame, or every third frame. As before, we combine the best results from each of these combinations in one graph (Figure 9(f)) to compare them. Recall that in the simple *grid* network, the density does not have a significant influence on the latency (see Figure 9(d)). However, in *grid+m*, if the other parameters are identical, the difference between a network with

4 neighbors and 8 neighbors cannot be neglected (compare the left-most curve with the right-most one, or the two middle curves). The influence of the speed of shifting is not very significant (compare the left-most two curves or the right-most two curves), and it is difficult to judge which speed overcome the others, for instance, speed=3 performs better than speed=1 for neighbors=4 while the trend is reversed for neighbors=8. This is due to the way we modeled mobility. In our mobility scenario, it takes circa 15 frames to deliver messages to the whole network, and a shifting of every 1 frame or every 3 frames cannot have much influence on the result. Under other mobility models, different results will be obtained.

6 Conclusions and Future work

We reported on the simulative analysis of the CHESS gMAC protocol, aimed for gossiping-based applications in sensor networks. Our analysis reveals that randomly deciding to refrain from using send slots significantly increases the effectiveness of gMAC's collision detection mechanism, and reduces energy consumption by about 15%. Node mobility does not affect the number of detected collisions. We determined the number of active slots that optimize the trade-off between latency and energy consumption. In the setting with 8 neighbor nodes, our experimental results confirm the optimality of CHESS's current node implementation (i.e., A=8).

The presented results are the first quantitative evaluation of the gMAC protocol. More analysis is needed. Future work will focus on considering more realistic radio models based on [11], and to find mathematical explanations for the optimal values. Moreover, a comparison of the given results with the gMAC variant described in [2] is planned.

All simulation models can be downloaded from:

`http://moves.rwth-aachen.de/~henrik/mmb10/`

Acknowledgments

We thank Bert Bos, Frits van der Wateren, Marcel Verhoef (all of CHESS) for discussions on the gMAC protocol and suggestions on the modeling. Thanks to Daniel Klink (RWTH Aachen University), who suggested to compare latency and energy consumption.

References

1. T.R. Andel and A. Yasinac. On the credibility of MANET simulations. *IEEE Computer*, 39(7):48–54, 2006.
2. P. A. M. Anemaet. Distributed G-MAC. Master's thesis, Delft University of Technology, 2008.
3. H. Bohnenkamp, P.R. D'Argenio, H. Hermanns, and J.-P. Katoen. Modest: A compositional modeling formalism for real-time and stochastic systems. *IEEE Trans. on Software Engineering*, 32(10):812–830, 2006.
4. H. Bohnenkamp, J. Gorter, J. Guidi, and J.-P. Katoen. Are you still there? — A lightweight algorithm to monitor node presence in self-configuring networks. In *Dependable Systems & Networks (DSN)*, pages 704–709, 2005.
5. H. Bohnenkamp, H. Hermanns, and J.-P. Katoen. Motor: The MoDeST tool environment. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 4424 of *LNCS*, pages 500–504. Springer-Verlag, 2007.
6. D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of MANET simulators. In *ACM Workshop On Principles Of Mobile Computing (POMC)*, pages 38 – 43, 2002.
7. D. D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. G. Webster. The Möbius framework and its implementation. *IEEE Trans. on Software Engineering*, 28(10):956–969, 2002.
8. C. Gross, H. Hermanns, and R. Pulungan. Does clock precision influence ZigBee's energy consumptions? In *Principles Of Distributed Systems (OPODIS)*, volume 4878 of *LNCS*, pages 174–188, 2007.
9. F. Heidarian, J. Schmaltz, and F.W. Vaandrager. Analysis of a clock synchronization protocol for wireless sensor networks. In *Int. Symp. on Formal Methods (FM)*, 2009. To appear.
10. A.-M. Kermarrec and M. van Steen. Gossiping in distributed systems. *ACM SIGOPS Operating System Review*, 41(5):2–7, 2007.
11. A. Meier, T. Rein, J. Beutel, and L. Thiele. Coping with unreliable channels: Efficient link estimation for low-power wireless sensor networks. In *Int. Conf. on Networked Sensing Systems (INSS)*, pages 19–26, June 2008.
12. R. A. Rashid, W. M. A. E. W. Embong, A. Zaharim, and N. Fisal. Development of energy aware tdma-based MAC protocol for wireless sensor network system. In *European J. of Scientific Research*, pages 571–578, 2009.
13. Nordic Semiconductors. *nRF2401 Single-chip 2.4GHz Transceiver Data Sheet*, 2002.
14. W. Song, R. Huang, and B. Shirazi. TreeMAC: Localized TDMA MAC protocol for real-time high-data-rate sensor networks. In *IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*, pages 1–10, 2009.
15. I. van Vessem. WSN gMac protocol specification. Technical report, CHESS B. V., Haarlem, NL, 2008. Version 1.1. Patent pending US 12 / 215,040.
16. M. Xie and X. Wang. An energy-efficient TDMA protocol for clustered wireless sensor networks. In *Computing, Communication, Control, and Management (CCCM)*, volume 02, pages 547–551, 2007.