Receiver-based Backbone Construction and Maintenance for Wireless Sensor or Multi-Hop Networks*

Markus Waelchli, Thomas Bernoulli, and Torsten Braun

Institute of Computer Science and Applied Mathematics
University of Bern, Neubrückstrasse 12, CH-3012 Bern
{waelchli,bernoull,braun}@iam.unibe.ch

Abstract. Energy savings and topology control are needed tasks of many sensor network applications. Sensors are assumed to be randomly deployed and shall organize themselves independently after deployment. Moreover, sensor networks shall operate as long as possible warranting network connectivity. To support these tasks we propose the maintenance of a virtual backbone. Nodes not participating in the backbone shutdown their radios and go to sleep for a certain time. The backbone nodes are adaptively altered according to the current network conditions. The backbone is thus able to deal with node failures and/or movements. The non-backbone nodes follow long sleep cycles during which they frequently wake up to check the network conditions. After a long sleep period the backbone is reestablished by the base station. The algorithm shows good approximation of the minimum number of nodes in the backbone after backbone setup, as well as the ability to repair link breaks on demand with short delays and low message overhead.

1 Introduction

Applications like tracking frameworks intrinsically perform application specific tasks such as distributed localization, observer determination, in-network processing, etc. Additionally, such applications need some kind of topology control and routing to supply a communication infrastructure, on-demand network configuration, and code distribution. The application should thereby run as energy efficient as possible. To support this needs we propose the maintenance of a virtual backbone built by a connected dominating set (CDS). The CDS adapts itself to local energy distributions in the network and supplies backbone repair mechanisms. Nodes not participating in the backbone shutdown their radios and go to sleep for a predefined long sleep period, thus conserving energy.

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

CDS approaches need knowledge about the local one-hop or two-hop neighborhood. Periodically exchanged hello messages are used to learn that information. The one-hop CDS approaches have the natural advantage of little communication and memory overhead, but on the other hand suffer from bad approximation factors leading to a high percentage of redundant nodes in the backbone. The approximation factor of a backbone, or set of nodes, is defined as the ratio of the number of nodes in the backbone/set to the possible optimum. In our approach we conserve the advantages of both approaches. Only the one-hop neighborhood information is periodically exchanged. Two-hop neighborhood information is only exchanged on-demand when the CDS is established.

The tracking and monitoring application [1] we intend to support bases on static or slow moving sensor networks that focus on long-term lifetime and source to base station communication patterns. In the monitoring state the sensor nodes report their data periodically, but in larger intervals. In the tracking state the sensor nodes in a specific area generate data in a burst, but only for a short period. Both scenarios do not provide constant high network load. Temporally redundant nodes can be determined and shutdown their radios for a certain time to enlarge network lifetime.

The remainder of this paper is organized as follows: The related work is discussed in section 2. Connected dominating set basics and our approach are presented in section 3. The simulation setups and results are discussed in section 4. The conclusions and future work can be found in section 5.

2 Related work

Protocols dealing with periodic sleep-wake cycles are often implemented on the MAC layer. T-MAC [3] and S-MAC [4] are two representatives of contention-based MAC protocols basing on low duty-cycles. The network nodes follow specific listen/sleep schedules. Periodically exchanged SYNC-messages are used to synchronize the network nodes as well as to avoid long-term clock drifts. Other MAC approaches perform energy preservation by time division multiplexing [5]. A third group of MAC protocols operate fully asynchronous and establish communication by sending long preambles that are guaranteed to be sensed when the receiver awakes [6]. MAC level approaches do not provide routing information making these protocols not best suited for our purpose. As SYNC messages are some kind of hello messages our algorithm could maybe be integrated with SYNC-based MAC protocols increasing performance in terms of energy.

On the network layer there exist a number of approaches that enable the selective disconnection of redundant nodes. ASCENT [7] and SPAN [8] both are distributed and randomized algorithms where nodes make local decisions on whether to sleep, or to join a forwarding backbone. SPAN demands periodic information exchange among the network nodes to identify the redundant nodes. Furthermore, two-hop information is needed leading to high complexity. ASCENT is a reactive protocol where nodes build a backbone on demand. The drawback is a bad approximation factor and high delays. GAF [9] nodes form vir-

tual clusters, where redundant nodes are timely disconnected from the network. The temporal state of the nodes depend on their global virtual position. The main drawback of GAF is that it relies on the knowledge of position information and that it does not support routing intrinsically.

Topology control and the construction of virtual backbones have been largely investigated in ad-hoc and wireless sensor networks. The main focus of connected dominating set approaches is on minimizing the number of nodes in the backbone, referenced in literature as the ability of the algorithm to approximate the minimum connected dominating set (MCDS). The MCDS is the optimal backbone size achievable, however known to be NP-hard [2]. The algorithm proposed in [10] first determines a CDS that consists of all nodes which have at least two non-adjacent neighbors. Two pruning rules are introduced to reduce the initial CDS. The need of two-hop neighborhood information and bad performance in certain network topologies make the algorithm not appropriate for our propose. An enhancement based on the remaining energy levels of the nodes instead of their link degrees is presented in [12]. [11] builds a CDS by computing a maximal independent set (MIS) that is connected in a second phase. Again two hop neighborhood knowledge is needed. [13] proposes an algorithm that converges in only one algorithmic step. Each node receiving a dominator message determines a timer based on the number of not yet covered downstream nodes. The nodes compete for dominatorship according to their timer. Nodes that do not reach additional nodes cancel their timer and become dominated. The algorithm has a bad approximation factor. All approaches do not consider sleep cycles. In contrast to the discussed algorithms the main goal of our approach is to extend the network lifetime while supporting node failures and mobility. Furthermore, local decisions about joining the backbone or not are done.

3 Backbone Construction and Maintenance

To be able to set up a virtual backbone, nodes have to learn their neighborhoods by periodical exchange of hello messages. In our approach these hello messages are also used to exchange the control traffic needed to construct and maintain the backbone. The drawback of additional control messages increasing network load and collision probability is thus avoided. Furthermore, the control info is included in the hello messages a configurable number of times replacing additional RTS/CTS and DATA/ACK mechanisms.

To make backbone construction or maintenance decisions the hello message sending periodicity is interrupted and the release of the next hello message prioritized according to the node's context. Whenever no backbone construction or repair mechanism is in action, all hello messages follow their normal intervals just informing the neighbor nodes about the existence of oneself, without carrying additional control information.

3.1 Preliminaries

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 connected dominating set (CDS) is a dominating set which builds a connected subgraph of G. As mentioned before, finding a MCDS is NP-hard. Furthermore, the MCDS does neither focus on energy distributions in the network nor on node mobility and topology change. Therefore, approximating the MCDS is not our prior goal. We propose a distributed algorithm aiming on efficiency and network adaptivity while nevertheless providing a good approximation factor.

The decision whether a node enters the backbone is done in a distributed manner. In the current implementation a node's link degree (CDS-LD) or its energy level (CDS-E) is used for the decision. A further version using a fuzzy logic control engine (FLC) dealing with energy distributions, node movement patterns, etc. is envisioned and may be analyzed in future work.

3.2 Construction of the backbone

The CDS setup is considered as a graph coloring problem. Initially, all nodes are white. The base station (BS) starts the CDS algorithm, coloring itself green and broadcasting a DOMINATOR message. This message contains the node's ID and a list of its neighbors. Thus, each node receiving the DOMINATOR message is able to check if it covers additional nodes by comparing its own neighbor table to the list of neighbors transmitted in the DOMINATOR message. Additionally, all receivers of a DOMINATOR message (i.e. neighbors of the dominator) color themselves gray and broadcast a DOMINATEE message again containing the list of neighbors of the dominator. With this mechanism the neighborhood information of the dominator is disseminated on-demand two hops into the network. All nodes two hops away from a dominator overhearing a DOMINATEE message again compare their own neighborhood information with the neighborhood list of the dominator. Based on that information they determine one of their upstream neighbors (intermediate nodes) as dominator candidate, prioritize it, and schedule a DOMINATOR_CHOICE message for it. This message is delayed according to the priority of the candidate. The priority is determined either by the link degree or the remaining energy of the candidate. In both versions, nodes with only one path to the last dominator are favored and therefore get a higher priority. This is necessary to guarantee network connectivity.

The message flow diagram for the backbone construction phase is depicted in Figure 1. The base station (BS) broadcasts a DOMINATOR message. Nodes 1 and 2 compare their neighbor tables with the neighbor list received from the BS. Node 1 goes to sleep (red state) immediately as it covers no additional nodes. Node 2 has a neighbor (3) that is not yet covered. It schedules a prioritized hello message containing the neighbor list of the BS (DOMINATEE). This message is received by 3. Node 3 schedules a DOMINATOR_CHOICE message according to the number of paths it has to the BS. Elected by node 3, node

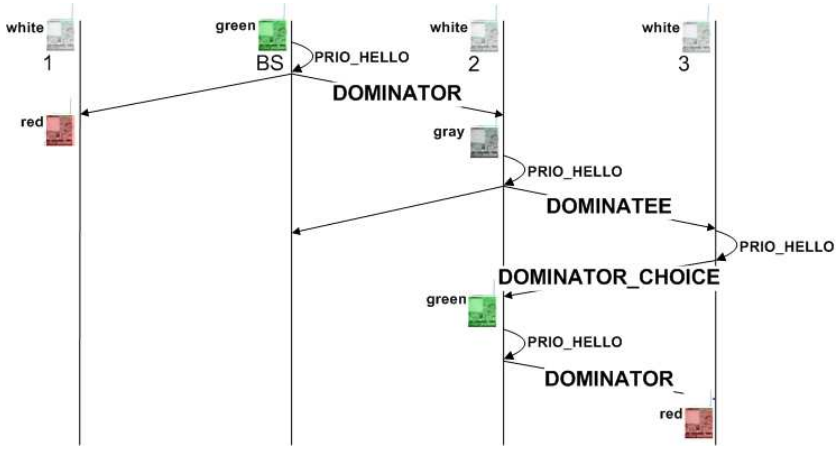


Fig. 1: Receiver-based CDS construction: Colors indicate different node states. Green nodes are dominators, red nodes dominated.

2 becomes dominator after the reception of the DOMINATOR_CHOICE message. In this simple example there are no multiple downstream nodes from the BS with links to nodes two hops away. Accordingly, there is no real contention for the dominator election in this example. However, if there were multiple two hop neighbors they would compete for the dominator election by delaying their DOMINATOR_CHOICE message.

The algorithm terminates as soon as no white nodes remain what happens in every connected network. Obviously, dominators are not white.

Proof. We distinguish two cases:*(i)* Nodes adjacent to a dominator color themselves gray.*(ii)* Nodes two hops away from a dominator always chose an up-link node as dominator which will be elected in case no other dominator message was overheard in time, i.e the node is ensured to have a dominator and color itself accordingly gray *(i)*.

As the down-link nodes determine the state of the up-link nodes we call the algorithm receiver-based. The algorithm is an approximation of the MPR protocol [14] which is used for an efficient broadcast in the OLSR routing protocol [15]. In contrast to MPR our CDS algorithm does not depend on the proactive establishment of two-hop neighborhood information.

As indicated above the control messages used for the backbone construction are included in the hello messages. As soon as a node enters the backbone construction state it sets a setup timer to three times the hello interval and includes its current control info in its hello messages during that time. To determine the release timers of the control messages two intervals are defined. A short interval of 100ms and a long interval of 1s. The intervals are chosen according to typical sensor network properties. Considering the Scatterweb [18] platform we aim at, 14ms to switch to send mode, send a packet, and switch back to idle mode are

needed. Therefore, a contention period of 100ms for the short interval seems reasonable. The timers for the DOMINATOR and the DOMINATEE messages are both randomly chosen from the short interval. The timer for the DOMINATOR_CHOICE message is chosen from both intervals depending on the priority of the backbone candidate. As soon as a control message timer is set, the periodic hello sending mechanism is interrupted and the next hello is scheduled according to the control message timer. In sensor networks any algorithm is confronted with high packet loss due to collisions or bit error rates and small bandwidth. An integration of the control messages and the hello messages is therefore helpful as it scales down the control traffic load and increases the probability of message delivery, while avoiding additional RTS/CTS like mechanisms.

The backbone is maintained for a predefined backbone time. After this time, the backbone is reestablished adapting itself to the new network conditions. During the backbone time the dominated nodes follow a listen/sleep schedule. The dominated nodes wake up periodically and listen the medium before going to sleep again. If a dominated node detects a link break in its vicinity or did not sense any dominator at all during its listen period, the node stays awake. The details of the backbone repair mechanisms are explained in the next section.

3.3 Local Path adaptation and repair

There are two different kinds of link breaks handled: *(i)* a backbone node determines an up-stream link break. *(ii)* a dominated node detects its isolation from the backbone. In both cases a link break is detected when a node did not overhear any hello message for a certain time (three times the hello interval *(i)*, the listen period *(ii)*).

In the latter case the dominated node remains awake and tries to connect to a dominator. In both cases the link break detecting node enters a link break state and starts to broadcast link break notifications to inform its neighborhood about the link break. Each node overhearing a link break notification adapts the link break state, saves the address of the link break node, and starts to propagate the link break information further. As soon as a backbone node with a valid route to the BS, i.e. the node did neither detect a link break itself or was informed about a link break in its path to the BS, receives a link break message, it enters path update state and broadcasts its own path to the BS within its next hello message.

Each node in link break state overhearing such a path update message, adapts the path and rebroadcasts the updated path (with its own ID). In case the rebroadcasting node is a dominated node it enters the backbone and becomes dominator. To minimize the number of resulting dominators, i.e in general not all dominated nodes are needed to repair the path, the path update distribution is done under contention. As mentioned above all nodes build a list of the nodes they received link break messages from. Accordingly, all nodes overhearing a path update message from a node in their link break nodes list cancel their own path update procedure.

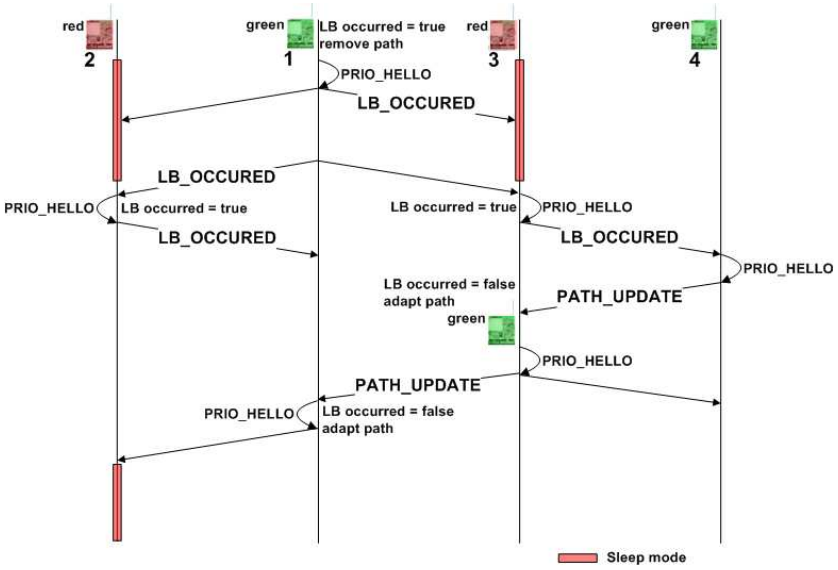


Fig. 2: Message flow if a dominator detects an link break to an up-link dominator.

An example describing the message flow in case of a link break occurrence is depicted in Figure 2. Dominator 1 detects a link break to its up-link node (not depicted) and broadcasts a link break message. As soon as there are some neighbors awake (nodes 2 and 3) they adapt the link-break state and propagate the link break further. As soon as dominator 4 with a valid route to the BS overhears a link break message it responds with a path update message. Dominated node 3 overhears this message, adapts the path, competes for dominatorship (it wins anyway in this case), and finally broadcasts the path update message. After having overheard the path update from dominator 1, node 2 knows the path is repaired and goes back to sleep.

4 Evaluation

The algorithm is implemented in the OMNeT++ network simulator [16]. Both the link degree (CDS-LD) and the energy based (CDS-E) strategy have been implemented and are evaluated in terms of average remaining energy of the nodes, energy variations amongst the network nodes, time to repair link breaks, and approximation factor of the backbone before and after a long sleep period.

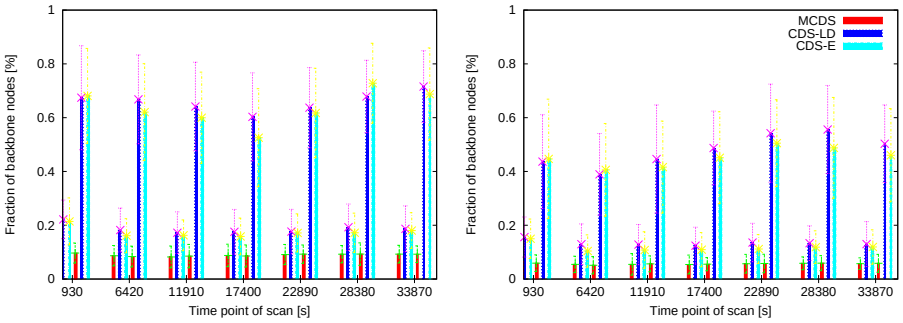
In addition to the standalone OMNeT++ network simulator we used the Mobility Framework [17] to model mobile sensor nodes. The current implementation of the Mobility Framework does only provide a free-space radio propagation model. We implemented the two-ray radio propagation model as it is more realistic. Furthermore, we changed bandwidth to 115.2kBit, channel frequency to 868MHz, transmission power to 0.75mW, receiver sensitivity to -95dBm, etc.

to match the ESB sensor board specifications from Scatterweb [18]. We thus got an maximal transmission range of 37.5m and an interference rage of 52m. We tested a sparse network of approximating 12 neighbors per node and a dense network of about 20 neighbors per node. All nodes are moving at two different, constant speeds of $0.1\frac{m}{s}$ and $0.5\frac{m}{s}$, respectively. We tested four different network sizes of 50, 100, 200, and 400 nodes. We did not evaluate static networks as in these networks link breaks should occur only rarely, but we want to test our algorithm in strain situations. The simulation time was set to ten hours. The backbone period is thirty minutes in our evaluation. Consequently, there are twenty backbone periods simulated in each run. The short sleep period is altered two, four, or eight times the listen period, resulting in 60, 120, or, 240 seconds. We tested all different network setups with ten seeds and calculated the 95% confidence interval of the results. Due to readablity the conficence intervals are only depicted if they are of relevance and because of space we show only the simulation results with 200 nodes network size.

In the evaluation we show that both CDS-LD and CDS-E provide good approximation factors. We furthermore show that the fraction of link breaks that cannot be repaired is small and that the link breaks are repaired fast. Moreover, we show that CDS-E is at least as good as CDS-LD concerning the average energy of the nodes, and more important that it has smaller energy variations than CDS-LD leading to more balanced network charge and longer lifetime.

4.1 Approximation factor of the backbone

In Figure 3 the backbone size of CDS-LD and CDS-E as well as of the minimum connecting dominating set (MCDS) of the network are depicted. The network was scanned at different time points always shortly after the beginning, respectively before the end of a backbone period. The results show that the MCDS is nearly constant throughout the simulation time. Both CDS-LD and CDS-E



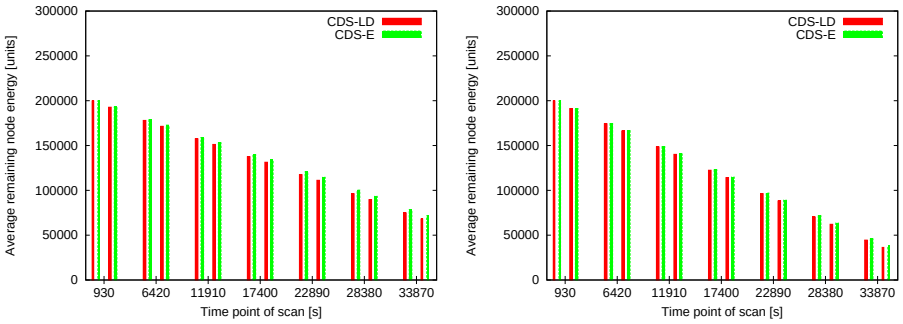
(a) Sparse - slow (b) Dense - slow

Fig. 3: Backbone size at the beginning respectively the end of a backbone period.

approximate the MCDS quite good at the beginning of a backbone period, i.e. shortly after the backbone has been built. The high percentage of nodes in the backbone at the end of a backbone period is linear to the number of link breaks that occurred. With each link break, nodes to repair that link break are looked for. On the other hand, in the current version of the algorithm no mechanism to give up the dominator state is foreseen. Thus, the increase of the number of dominators during the backbone period is obvious. Figure 3 shows the ratio between the backbone sizes at the beginning and the end of a backbone period to be smaller for slow and dense networks where fewer link breaks occur. We argue that the increasing number of dominators is acceptable as link breaks are repaired, the network connectivity guaranteed, and correct paths to the base station supported. In static networks, or networks with slow moving nodes the ratio would obviously become much better. In the current evaluation there are about five link breaks per long sleep period even in the dense and slow scenario. This is good for the evaluation of the impact of link breaks, but may be rather unrealistic for mostly static sensor networks, where much fewer link breaks can be expected. With faster node speeds, all is worse.

4.2 Average energy of the network nodes

In this section we present the results concerning the average energy of the network (see Figure 4). Both CDS-LD and CDS-E show linear charging of the batteries and almost the same energy level in all simulations. Consequently, neither



(a) Sparse - slow

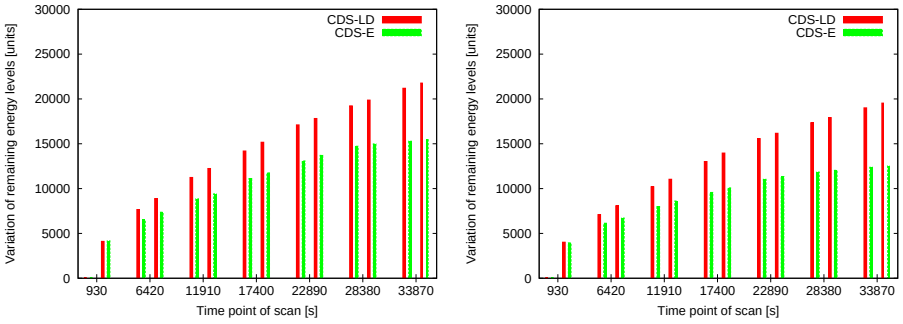
(b) Sparse - fast

Fig. 4: Average remaining energy in the network.

CDS-LD nor CDE-E has any advantages in that respect. In sparse networks with fast moving nodes much more link breaks occur, leading to more dominators, and less dominated nodes that are able to go to sleep. Accordingly, the batteries are charged more under such conditions. In dense networks (not depicted) the average energy is a little higher.

4.3 Energy variation of the network nodes

In contrast to the average energy which is almost the same for both CDS-LD and CDS-E, the remaining node energy variations in the network are considerable smaller for CDS-E (see Figure 5). Balanced energy distribution in the network



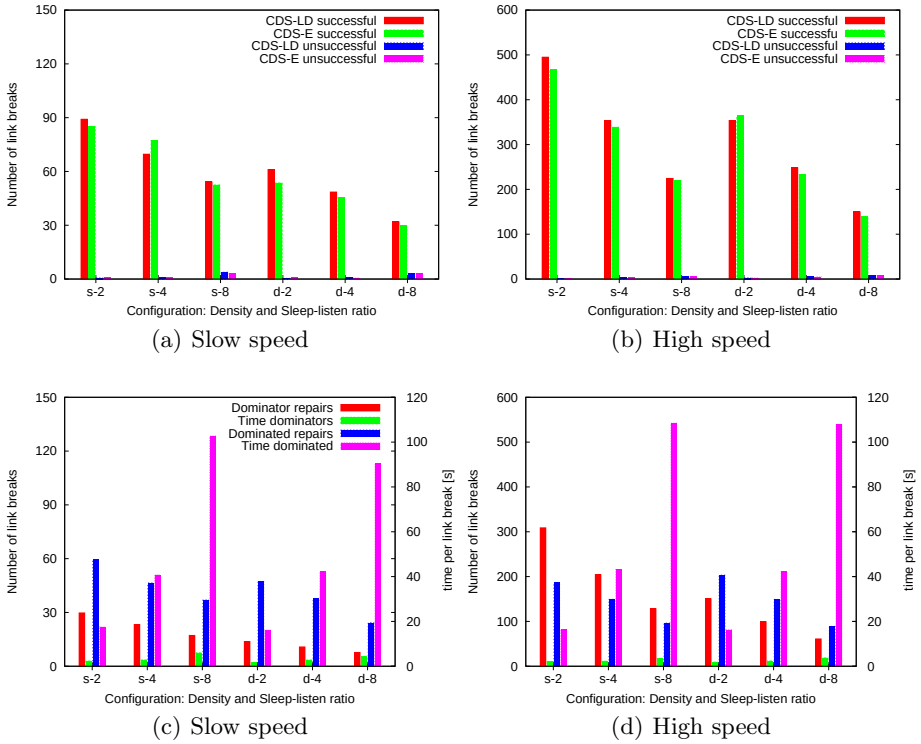
(a) Sparse - slow (b) Dense - slow

Fig. 5: Energy variations amongst the network nodes.

indicate uniform node strains and a longer network lifetime in average. With CDS-LD a certain fraction of nodes (those with the highest node degrees) are more strained than others and will run out of battery sooner. This leads to a smaller node density, what has negative impact on the average energy level (see last section). In our simulations only the first ten hours of a network have been simulated. The ratio between CDS-LD and CDS-E would obviously become larger the longer the network is active. The results of the simulations with higher node speeds look similar.

4.4 Number of link breaks and repair time

Finally, the number of successful and unsuccessful link break repairs as well as the average time to repair a link break have been evaluated. In our evaluation not only the dominated nodes that determine a link break after their listen period trace the time until they receive a path update message, but also the dominated nodes that are only connected to dominators that reported link breaks. This explains the high percentage of dominated nodes with link breaks. Figure 6 shows that the number of unsuccessful repairs is small in all simulations, whereas the number of link breaks increases considerably with a higher speed. The average time to repair a dominator link break is below 5 seconds for all simulations. However, the average time to repair a dominated link break increases much with higher sleep-listen ratios.



(a) Slow speed (b) High speed

(c) Slow speed (d) High speed

Fig. 6: Time for successful and unsuccessful repairs as well as time for dominator and dominated node repairs. (s = sparse, d = dense; 2,4 and 8 are the sleep-listen ratios)

5 Conclusions and future work

One of the reasons to propose a topology control mechanism is the need of having an energy efficient backbone structure that supplies routing, on-demand network configurations, and code distribution for our event detection framework [1]. In the current state, both CDS-LD and CDS-E perform efficiently and approximate the MCDS well after setup. Link breaks are repaired fast by both versions. CDS-E outperforms CDS-LD in terms of energy variations what makes it a better choice to enlarge network lifetime.

To get indications on the performance of our approach we will implement other approaches and compare them in future work. Furthermore, we will implement the algorithm on the ESB sensor board platform from Scatterweb. This will show the performance of our algorithm and its applicability in a real world environment.

References

1. Wälchli, M., Scheidegger, M., Braun, T.: Intensity-based event localization in wireless sensor networks. In: Proceedings of IFIP Third Annual Conference on Wireless On Demand Network Systems and Services (WONS'06), Les ménuires, France (2006)
2. Clark, B.N., Colbourn, C.J., Johnson, D.S.: Unit disk graphs. Discrete Mathematics **86** (1990) 165–177
3. van Dam, T., Langendoen, K.: An adaptive energy-efficient mac protocol for wireless sensor networks. In: Proceedings of the 1st international conference on Embedded networked sensor systems, Los Angeles, California, USA (2003) 171–180
4. Ye, W., Heidemann, J., Estrin, D.: Medium access control with coordinated adaptive sleeping for wireless sensor networks. IEEE/ACM Transactions on Networking **12**(3) (2004) 493–506
5. van Hoesel, L., Havinga, P.: A lightweight medium access protocol (lmac) for wireless sensor networks: Reducing preamble transmissions and transceiver state switches. In: INSS, Japan (2004)
6. El-Hoiydi, A., Decotignie, J.D.: Wisemac: An ultra low power mac protocol for the downlink of infrastructure wireless sensor networks. In: ISCC2004, the 9th IEEE International Symposium on Computers and Communications,, Alexandria, Egypt (2004)
7. Cerpa, A., Estrin, D.: Ascent: Adaptive self-configuring sensor networks topologies. IEEE Transaction on Mobile Computing **3**(3) (2004) 272 – 285
8. Chen, B., Jamieson, K., Balakrishnan, H., Morris, R.: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In: Proceedings of the 7th ACM International Conference on Mobile Computing and Networking, Rome, Italy (2001) 85–96
9. Xu, Y., Heidemann, J., Estrin, D.: Geography-informed energy conservation for ad-hoc routing. In: MobiCom '01, Rome, Italy (2001)
10. Wu, J., Li, H.: On calculating connected dominating set for efficient routing in ad hoc wireless networks. In: DIALM 99, Seattle, WA, USA (1999)
11. Wan, P.J., Alzoubi, K.M., Frieder, O.: Distributed construction of connected dominating set in wireless ad hoc networks. In: Proceedings of Infocom 2002. (2002)
12. Wu, J., Dai, F., Gao, M., Stojmenovic, I.: On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks. In: Journal of Communications and Networks. Volume 4. (2002)
13. Zhou, D., Sun, M.T., Lai, T.H.: A timer-based protocol for connected dominating set construction in ieee 802.11 multihop mobile ad hoc networks. In: Proceedings of the 2005 Symposium on Applications and the Internet (SAINT'05). (2005)
14. Quayyum, A., Viennot, L., Laouiti, A.: Multipoint relaying: An efficient technique for flooding in mobile wireless networks. Technical report, INRIA, Sophia Antipolis, France (2000)
15. Clausen, T., Jacquet, P.: Optimized link state routing protocol. RFC 3626 (2003)
16. Varga, A.: Omnet++ simulator (2006) http://www.omnetpp.org/.
17. Mobility-fw: Framework for omnet++ (2006) http://mobility-fw.sourceforge.net/.
18. Scatterweb: (Sensor platform) http://www.scatterweb.net.