Construction of Connected Dominating Sets in Large-Scale MANETs Exploiting Self-Stabilization

Stefan Unterschütz
Institute of Telematics
Hamburg University of Technology
Hamburg, Germany
stefan.unterschuetz@tu-harburg.de

Volker Turau
Institute of Telematics
Hamburg University of Technology
Hamburg, Germany
turau@tu-harburg.de

Abstract—**Available algorithms for the distributed construction of connected dominating sets in mobile ad hoc networks are inapplicable or suffer from a high complexity. This is mainly due to the resource-restricted nature of wireless devices, such as sensor nodes, and the error-prone character of the communication medium. This work introduces and evaluates a new local, probabilistic self-stabilizing algorithm providing fault-tolerance and scalability for networks of high density.**

I. Introduction

Routing in mobile ad hoc networks (MANETs) is a challenging task due to the lack of a fixed infrastructure. Nodes must frequently perform routing decisions. This is aggravated by the error-prone communication channel and the mobility of the devices. A well-known structure upon the inherently flat hierarchy of nodes is that of a virtual backbone. Connected dominating sets (CDS) have proven to be suitable for such a backbone. Topology-based and location-based routing, broadcast, or even energy conservation algorithms highly benefit from this structure [1], [2].

The construction of CDSs is well studied – so why the need of further research? Different design goals, such as time and message complexity, maintenance, seeking a minimum CDS have been considered so far [1]. In the course of designing a solar power tower plant with thousands of autonomous heliostats connected wirelessly via 802.15.4 compliant hardware[1], we found that none of the existing CDS algorithms is qualified for this particular network. Algorithms based upon trees require $O(D)$ rounds to converge (D being the diameter of the network), so that a usage in networks with huge diameters is inapplicable. Algorithms based upon 3 or even 4-hop neighborhoods are infeasible in dense networks due their high storage and communication needs. Assuming an average node degree of 50, which is reasonable in heliostat fields, storing the three hop neighborhood takes approximately $\pi \cdot 3^2 \cdot 50 \approx 1413$ entries in case the three hop area is completely covered.

In this work we present a localized, probabilistic self-stabilizing algorithm for the CDS construction. Self-stabilizing ensures that starting from an arbitrary system configuration, a valid state is eventually reached in finite time. As a result such algorithms are inherently fault-tolerant. Furthermore, the

goal was to devise an algorithm using a minimal number of messages. The proposed algorithm can be applied in networks of large-scale as well as in networks of a high maximum node degree Δ, because it drastically reduces the number of messages exchanged among neighboring nodes.

This paper is structured as follows. Section II describes current CDS protocols for MANETs. Subsequently, the nomenclature and the mathematical model is given. In Section IV a self-stabilizing algorithm for the construction of CDSs is defined. Our approach is a composition of three algorithms: first a maximal independent set (MIS), afterwards a weakly-connected dominating set (WCDS), and finally a CDS is constructed. Section V describes necessary modifications needed for an efficient implementation. Finally, the protocol is evaluated by simulations using OMNeT++ and by a testbed based on an ATmega128RFA1 module.

II. Related work

Various algorithms for the CDS construction in MANETs have been published. A survey is given by Blum et al. [1]. For distributed algorithms some general techniques are noticeable. Tree-based algorithms build a spanning tree and all non-leaf nodes become dominators. In general, such approaches do not scale well with the diameter and the size of the network. In pruning-based algorithms all nodes are initially dominators and then gradually turned into ordinary nodes. In methods based on multipoint relaying each node selects a set of one hop neighbors, which can cover the two hop neighborhood. By constructing a MIS and adding a multipoint relay a CDS is constructed [3]. In the approach of Alzoubi et al. [4] all pairs of nodes of a MIS, that have a distance of two or three hops, are connected by electing intermediate nodes as dominators.

Despite of this research, there are still open issues concerning the practical realization in MANETs, maintainability and fault-tolerance are rarely considered, or scalability is insufficient. Regarding the latter, if a structure such as a CDS must be employed to avoid the Broadcast Storm Problem in dense networks, then the construction of this CDS should also avoid this problem. In particular current local, distributed CDS algorithms have time, message, or message size complexities of at least $O(\Delta)$ and thus are not applicable in networks of high density (cp. [1]).

---

[1]HelioMesh: http://www.ti5.tu-harburg.de/research/projects/heliomesh/

The concept of self-stabilization is very attractive for MANETs, since it includes fault-tolerance [5]. It ensures that, regardless of the starting configuration, a valid state is reached within a finite time, e.g., even after a transient fault. In the literature two self-stabilizing CDS algorithms can be found, both are based on a tree-construction [6], [7]. Unfortunately, these algorithms suffer from a high communication demand.

III. PRELIMINARY

A sensor network is modeled as a connected graph $G = (V, E)$. V represents the set of wireless nodes and E the set of undirected links between nodes. The i-hop neighborhood of a node v is given by $N^i(v) = \{u \in V \mid dist(v, u) = i\}$. The length of the shortest path between two nodes v_1 and v_2 (measured in number of hops) is denoted by $dist(v_1, v_2)$.

A *dominating set* (DS) is a subset D of V such that all nodes of V are either in D or adjacent to a node in D. A dominating subset D of V is called a *maximal independent set* (MIS) if no nodes of D are adjacent. Two nodes $u, v \in D$ are called *weakly-connected* in D if there exists a path P between u and v such that for every two directly consecutive nodes in P at least one is in D. A dominating set D is called *weakly-connected dominating set* (WCDS) if all pairs of D are weakly-connected. Finally, a dominating set D is called *connected dominating set* (CDS) if the subgraph induced by D is connected. The *2-closure* of $D \subseteq V$ is a subset of V containing D and all nodes of at least one shortest path between any pair of nodes $u, v \in D$ with $dist(u, v) = 2$. The proof of the following Lemma is straightforward and is omitted.

Lemma 1: A MIS is a dominating set. A dominating set D in which each pair of nodes is weakly-connected in D is a WCDS. The 2-closure of a weakly connected dominating set is a CDS.

IV. SELF-STABILIZING ALGORITHM

This section presents the composition of three self-stabilizing protocols to construct a CDS. The first algorithm constructs a MIS. The second algorithm extends the MIS to a WCDS by adding intermediate dominators. Finally, the WCDS is extended to a CDS.

Each node v declares three boolean state variables d_{mis}, d_{weak}, and d_{cds} indicating whether v is member of the corresponding set. The initial description is based on the central daemon and the shared-variable model [8]. Furthermore, it is assumed that each node has immediate access to its 4-hop neighborhood. In the following algorithms each node periodically checks if it is privileged, i.e., a state change is possible. The update procedure of a privileged node is an atomic operation. The relaxation of these assumptions and a practical implementation is discussed in Section V.

A. Maximal Independent Set

Several self-stabilizing algorithms to build a MIS have been proposed. We use the algorithm of Hedetniemi et al. [9] shown below as Algorithm 1. Predicate $P_{\mathrm{mis}}(v)$ becomes *true* if

$D^i_{\mathrm{mis}}(v) = \{u \in N^i(v) \mid u.d_{\mathrm{mis}} = true\}$ is empty. In this case $v.d_{\mathrm{mis}}$ is set to *true* and v becomes a dominator.

Algorithm 1 MIS Construction

$P_{\mathrm{mis}}(v) := D^1_{\mathrm{mis}}(v) = \emptyset$

if $v.d_{\mathrm{mis}} \neq P_{\mathrm{mis}}(v)$ **then** $v.d_{\mathrm{mis}} \leftarrow P_{\mathrm{mis}}(v)$

Lemma 2: If no process is privileged with respect to Algorithm 1 (i.e. each $v \in V$ satisfies $v.d_{mis} = P_{mis}(v)$) the set $D_{\mathrm{mis}} = \{v \in V \mid v.d_{\mathrm{mis}} = true\}$ is a dominating set.

Proof: The result follows from [9] and Lemma 1. ∎

B. Weakly-Connected Dominating Set

The purpose of the second algorithm is to find a weakly-connected set containing D_{mis}. Algorithm 2 achieves this by adding an intermediate dominator between every pair of nodes of D_{mis} with distance three. Algorithm 2 uses the set $D^i_{\mathrm{weak}}(v) = \{u \in N^i(v) \mid u.d_{\mathrm{mis}} = true \vee u.d_{\mathrm{weak}} = true\}$ and $D^{1,2}_{\mathrm{weak}}(v) = D^1_{\mathrm{weak}}(v) \cup D^2_{\mathrm{weak}}(v)$. The variable $v.d_{\mathrm{weak}}$ becomes true if there exist nodes $n_1 \in D^1_{\mathrm{mis}}(v)$ and $n_2 \in D^2_{\mathrm{mis}}(v)$ with a distance of three such that v is the only node with $v.d_{\mathrm{weak}} = true$ that is contained in the one or two hop neighborhood of both n_1 and n_2. If this property is already fulfilled by a another node then n_1 and n_2 would already have been weakly connected in D_{weak}.

Algorithm 2 WCDS Construction

$P_{\mathrm{weak}}(v) := \exists n_1 \in D^1_{\mathrm{mis}}(v), \, n_2 \in D^2_{\mathrm{mis}}(v) \mid dist(n_1, n_2) = 3$
$$\wedge \, (D^{1,2}_{\mathrm{weak}}(n_1) \bigcap D^{1,2}_{\mathrm{weak}}(n_2)) \setminus \{v\} = \emptyset$$

if $v.d_{\mathrm{weak}} \neq P_{\mathrm{weak}}(v)$ **then** $v.d_{\mathrm{weak}} \leftarrow P_{\mathrm{weak}}(v)$

Lemma 3: If no process is privileged with respect to Algorithms 1 and 2 the set $D_{\mathrm{weak}} = \{v \in V \mid v.d_{\mathrm{mis}} = true \vee v.d_{\mathrm{weak}} = true\}$ is a WCDS.

Proof: According to Lemma 2 and 1 the set D_{weak} is a DS. Let $v_1, v_2 \in D_{\mathrm{weak}}$ and P be a path between v_1 and v_2. Assume that P contains a subpath $P_1 = x_1, x_2$ such that $x_1, x_2 \notin D_{\mathrm{weak}}$. Since x_1, x_2 are not privileged regarding Algorithm 1, there exists a node $n \in D_{\mathrm{mis}}$ being adjacent to x_1, x_2 or two nodes $n_1, n_2 \in D_{\mathrm{mis}}$ being adjacent to x_1 and x_2 respectively. In the latter case there exists a node $z \in D_{\mathrm{weak}}$ weakly-connecting n_1 and n_2, since no process is privileged with respect to Algorithm 2. From this it follows that there exists a path P_2 connecting x_1 and x_2 such that for all directly consecutive nodes at least one is in D_{weak}. Now replace P_1 by P_2 in P and denote the resulting path by P'. If there exists a sequence of two consecutive nodes in P' that aren't in D_{weak} the same construction is repeated. After finitely many steps a path weakly-connecting v_1 and v_2 is constructed. Hence, D_{weak} is a WCDS. ∎

C. Connected Dominating Set

To get a CDS it suffices by Lemma 1 to compute the 2-closure of D_{weak}. This can be done by adding intermediate
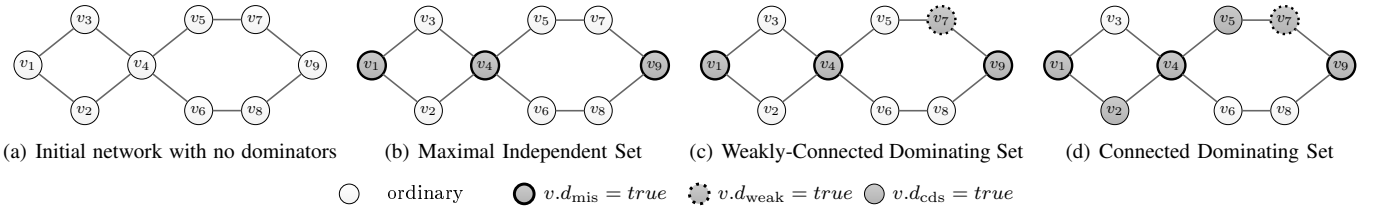
Fig. 1. Example of executing algorithm $\mathcal{A}$.

nodes between every pair of nodes of D_{weak} with distance two; as in Algorithm 2. Unfortunately, this can result in a high number of redundant dominators. In order to keep the number of dominators low Algorithm 3 uses an alternative dominator selection. This is done by adding intermediate nodes to dominators of D_{mis} with a distance of two and dominators with a distance of three which are already weakly-connected. Predicate $P_{\text{cds}}^1(v)$ becomes *true* if v can exclusively connect two one hop dominators of D_{mis}. Predicate $P_{\text{cds}}^2(v)$ becomes true if v has a one hop and two hop neighbor in D_{mis} which are weakly-connected by a node e but not already fully connected by another intermediate node.

Algorithm 3 makes use of the set $D_{\text{cds}}^i(v) = \{u \in N^i(v) \mid u.d_{\text{mis}} = true \vee u.d_{\text{weak}} = true \vee u.d_{\text{cds}} = true\}$.

Algorithm 3 CDS Construction

$S(a,b,v) := D_{\text{weak}}^2(a) \bigcap D_{\text{weak}}^1(b) \bigcap D_{\text{weak}}^1(v)$

$P_{\text{cds}}^1(v) \quad := \exists n_1, n_2 \in D_{\text{mis}}^1(v) \mid dist(n_1, n_2) = 2$
$\qquad\qquad \wedge \ D_{\text{cds}}^1(n_1) \bigcap D_{\text{cds}}^1(n_2)) \setminus \{v\} = \emptyset$

$P_{\text{cds}}^2(v) \quad := \exists n_1 \in D_{\text{mis}}^1(v), \ n_2 \in D_{\text{mis}}^2(v) \mid dist(n_1, n_2) = 3$
$\qquad\qquad \wedge \ S(n_1, n_2, v) \neq \emptyset \ \wedge \ \forall e \in S(n_1, n_2, v) \mid$
$\qquad\qquad D_{\text{cds}}^1(n_1) \bigcap D_{\text{cds}}^1(e)) \setminus \{v\} = \emptyset$

if $v.d_{\text{cds}} \neq P_{\text{cds}}^1(v) \vee P_{\text{cds}}^2(v)$ **then** $v.d_{\text{cds}} \leftarrow P_{\text{cds}}^1(v) \vee P_{\text{cds}}^2(v)$

Lemma 4: If no process is privileged with respect to Algorithms 1, 2, and 3 the set $D_{\text{cds}} = \{v \in V \mid v.d_{\text{mis}} = true \vee v.d_{\text{weak}} = true \vee v.d_{\text{cds}} = true\}$ is a CDS.

Proof: (Sketch) According to Lemma 3 D_{cds} is a weakly-connected dominating set. It suffices to prove that each pair $v_1, v_2 \in D_{\text{mis}}$ with $dist(v_1, v_2) = 2, 3$ is connected. This is true, if no node is privileged regarding Algorithm 3. ∎

D. Complete Algorithm

The proposed algorithm to compute a CDS is the concurrent execution of Algorithms 1, 2, and 3 and is referred to as Algorithm $\mathcal{A}$ in this paper. Figure 1 shows an execution of Algorithm $\mathcal{A}$ for a network with 9 nodes.

The proof of correctness of complete algorithm is left due to the page constraints.

V. EFFICIENT IMPLEMENTATION

Self-stabilization in general offers amazing properties in terms of fault-tolerance. Unfortunately, algorithms designed for a shared-variable model and a central daemon are difficult or even infeasible to realize in an unreliable, distributed

environment. Message-passing, as used in MANETs, is challenging because of the need of cache-coherence, meaning that retrieved state information of neighbors has to be coherent to the actual state. Furthermore, ensuring atomicity of the program execution in a large-scale network introduces a very high controlling overhead. Regarding the proposed CDS algorithm, a node needs to obtain the 4-hop neighborhood which is challenging in dense networks.

This section conducts the transition from the proposed theoretical algorithm to an applicable and implementable approach for constructing a CDS using the concept of probabilistic self-stabilizing. Necessary modification are introduced and a program description is given. It is noticeable that most design decisions are carried out to meet the criteria small footprint, important for memory restricted devices, and scalability. The resulting algorithm runs in an distributed, error-prone environment without the need of atomic operations.

A. Communication and Execution Model

A node v repeatedly executes algorithm $\mathcal{A}$ with a random round time of $[T_{step}, 2 \cdot T_{step}]$. Subsequently the own state, i.e., $v.d_{\text{mis}}, v.d_{\text{weak}}, v.d_{\text{cds}}$, and a list of known 1-hop and 2-hop dominators ($D_{\text{cds}}^1(v), D_{\text{cds}}^2(v)$) is broadcasted. Receivers of this message forward it in order to cover the whole 2-hop neighborhood of node v (a proper explanation why this is sufficient is given in the next section). This update procedure allows nodes to maintain state copies of known neighbors. To avoid collisions a random offset of $[0, T_{fwd}]$ is used for forwarding. Furthermore, let T_{exe} be the maximum time for executing the algorithm with a subsequent state update of the 2-hop neighborhood.

The ratio of T_{exe} to T_{step} is crucial since it determines the probability p for a non-overlapping (i.e., atomic) execution of the algorithm. In general, stabilization in case of a probabilistic distributed execution ($0 < p < 1$) can be proven for algorithms that stabilize under a central daemon [5]. A conflict, e.g., two nodes execute the algorithm concurrently and become redundant dominators, is resolved with a non-zero probability in a subsequent, non-interleaving execution. For the same reason packet loss and collisions, which are violating the cache-coherence, are tolerated if the probability for a valid state update is non-zero. The latter is affected by the ratio between T_{fwd} and the packet propagation delay. Note that the dimensioning of T_{step}, T_{fwd} is a trade-off between communication costs and convergence time. Former is crucial in dense networks, whereas a low convergence time is preferable in case of high mobility.

Particularly in networks of high density the state update becomes a challenge due to the high packet collision probability for small values of T_{step} and T_{fwd}. A countermeasure is to only allow dominators (assume $|D_{cds}| \ll |V|$) to periodically broadcast their state. This is possible, because algorithm $\mathcal{A}$ only depends on the knowledge of existing dominators. Unfortunately, a technique for detecting state changes from dominator to an ordinary node becomes necessary. For this purpose a node, which is recalling its dominator role, is allowed to broadcast this information once. Additionally, to be robust against packet loss, a time to live field (TTL) is used to detect the non-existence of former dominators.

Even with the previous modification the forwarding of a dominator's state scales poorly for dense networks because of a high bandwidth usage, even for a long T_{fwd}. A solution is a 1-counter-based 2-hop broadcasting algorithm [10]: A node receiving a state update from a dominator is only allowed to forward this information if and only if it has not already received more than one copy of the packet by adjacent nodes. This approach avoids the broadcast storm problem. Given that the introduced flooding approach may only partly cover the 2-hop neighborhood, dominators always participate at the forwarding procedure. This guarantees that once a CDS is constructed, it stays valid.

B. Algorithm Realization

For the execution of the self-stabilizing algorithm a node requires knowledge of all 2-hop dominators as well as their 2-hop neighborhood. Unfortunately, this implies the need of storing the 4-hop neighborhood of dominators, which is infeasible for memory-restricted hardware, e.g., sensor nodes.

To overcome this issue we propose a scheme that reduces the vision of a node to its 2-hop neighborhood. Recollect Fig. 1 in which for example v_2 becomes a dominator connecting v_1 and v_4. This decision only depends on the 2-hop neighborhood of v_2 which is shared with v_1 and v_4. The existence of nodes v_5 and v_7 is irrelevant. The restriction of the vision can be mathematically expressed. Let d be any known dominator of v, for the evaluation of the algorithm by v all requests for the above defined sets of neighbors of d are substituted as follows: $\tilde{D}_X^i(d) := D_X^i(d) \bigcap D_{cds}^{1,2}(v)$; where X is either *mis*, *weak* or *cds* and the range of i is (1), (2) or $(1,2)$. Note that this substitution preserves the property of self-stabilization.

Algorithm 4 Extended WCDS Construction

$P_{weak}(v) := \exists n_1 \in D_{mis}^1(v),\ n_2 \in D_{mis}^2(v) \mid$

$\qquad hops(n_1, n_2) = 3 \ \wedge n_1.id > n_2.id$

$\qquad \wedge\ (\ \tilde{D}_{weak}^{1,2}(n_1) \bigcap \tilde{D}_{weak}^{1,2}(n_2)\ ) \setminus \{v\} = \emptyset$

if $v.d_{weak} \neq P_{weak}(v)$ **then** $v.d_{weak} \leftarrow P_{weak}(v)$

The lack of 3-hop, 4-hop dominator information may result in a higher number of dominators depending on the underlying topology. In Fig. 1(d) node v_6 would become a dominator of D_{weak}, because it doesn't know of the existence of v_7. This situation can be avoided by extending Algorithm 2 as shown in Algorithm 4. A dominator that weakly-connects two nodes of D_{mis} has to be a direct neighbor of the node with the highest id.

The restriction of the neighborhood allows an efficient storage of dominator lists. In the following the stored information for node v_4 for the graph shown in Fig. 1(d) is depicted.

$$N^4 = \begin{pmatrix} 4 \\ 1 \\ 2 \\ 7 \\ 5 \end{pmatrix} \quad d_{mis}^4 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad d_{weak}^4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad d_{cds}^4 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

$$A_1^4 = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix} \quad A_2^4 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Vector N^4 contains the own identifier and the identifiers of the known dominators of node v_4. Node v_9 is omitted because it is in the 3-hop range of v_4. The index of the stored identifiers in N^4 is used for interpreting the following vectors: The boolean vectors d_{mis}^4, d_{weak}^4, and d_{cds}^4 hold the state variables of the dominators. Finally, A_1^4 and A_2^4 are the adjacency matrices for the 1-hop and 2-hop neighborhood of nodes listed in N^4. For example, row 2, column 3 of A_1^4 means that v_1 is a 1-hop neighbor of v_2. Using such a vector representation allows a fast evaluation of the rules of the self-stabilizing algorithm by performing vector and bitwise *and* and *or* operations.

As already mentioned, an update message broadcasted by a dominator v contains the own state $v.d_{mis}$, $v.d_{weak}$, $v.d_{cds}$ as well as $D_{cds}^1(v)$ and $D_{cds}^2(v)$. A receiver r first checks if v is already in the list of known dominators and adds it if necessary. Then row 1 of the two adjacent matrices of r, containing 1-hop and 2-hop dominators, is updated. Furthermore, the attached lists $D_{cds}^1(v)$ and $D_{cds}^2(v)$ are used to update the corresponding row in the adjacency matrices. Unknown neighbors of v are ignored.

C. Estimation of complexity

A precise analyses of the complexity is challenging due to the probabilistic nature of algorithm $\mathcal{A}$. For this reason we want to give a rough estimation of the costs. It is assumed that the execution and state broadcasting is atomic and the whole 2-hop neighborhood is covered by state updates. When running the algorithm it takes at most $2 \cdot T_{step}$, i.e., the maximum round time, until a MIS is constructed. Further $2 \cdot T_{step}$ are necessary for connecting 2-hop dominators of the MIS and finally at most $2 \cdot T_{step}$ to connect remaining 3-hop neighbors. The CDS is constructed in at most $6 \cdot T_{step}$, i.e., $O(1)$ rounds. The number of required transmissions for a time interval T_{step} is proportional to the resulting CDS size in case that 1-counter-based 2-hop flooding is used.

Currently, the density of the network is not considered, but in real networks it has an impact on the probability of collisions and interfering executions. For this, further empirical studies become necessary.

VI. Experimental Validation

In this section the two main design criteria of the CDS algorithm, feasibility and scalability, are evaluated. The former is done in a testbed containing 15 sensor nodes. In order to prove scalability, simulations are applied which allow the inspection of networks of large-scale and high density. For this purpose a comprehensive study of dynamic aspects, such as availability and size of the CDS, is done.

Algorithm $\mathcal{A}$ is realized taking all modifications of Section V into account. The actual implementation uses a wrapper framework, developed by the authors, which allows a compilation and accordingly code reuse for the OMNeT++ network simulator [11] and an ATmegaRFA1 microcontroller with embedded 802.15.4 compliant transceiver.
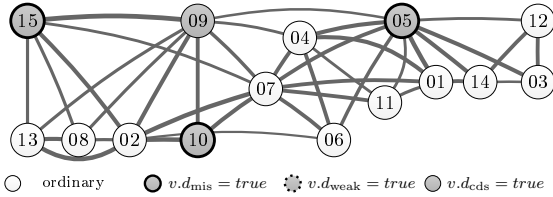
Fig. 2. Resulting CDS of testbed evaluation

A. Testbed

The CDS algorithm is executed in an indoor network deployment consisting of 15 deRFmega128-22A00 modules manufactured by *dresden elektronik*. The data rate is 250 kbit/s. The interval for executing the algorithm is set to $T_{step} = 10\,\text{s}$. The first offset after start-up is given by $[0, T_{step}]$. The random forwarding offset of the 1-counter-based 2-hop broadcasting is set to $T_{fwd} = 100\,\text{ms}$. The time to live value for dominators is set to $TTL = 5$, meaning that after TTL executions, a dominator entry is discarded if no sign of life is received.

Due to the specification of 802.15.4 the packet size of at most 127 Byte limits the size of a node's dominator list. We set this value to 40 leading to a size of 80 Byte by using 16 Bit node addresses. If more than 40 dominators are perceptible by a node, remaining dominators would be ignored (important for dense networks). The state vectors are of size 5 Byte (40 Bit) each, and the two adjacency lists are of size 200 Bytes ($40 \cdot 40$ Bit) each. The overall memory usage of the algorithm is 495 Bytes, which is a small share on the ATmega128RFA1 with 16 KByte of SRAM.

For the visualization of the underlying topology a neighborhood exploration protocol is run at the beginning of the experiment: Each node broadcasted 100 packets with random offsets. A counter array of received neighboring packets is read out at the end of the experiment. Afterwards the CDS algorithm run for a time of 50 s. Figure 2 shows the topology including link states (thickness of edges correlates to quality).

The execution of the algorithm, itemized for each node, is depicted in Fig. 3. At the beginning frequent role changes are observable until the final CDS was constructed after 26 s. It is noticeable that the outcome of the neighborhood exploration protocol may differ to the actual seen link state

of the CDS algorithm. In general, the testbed shows the algorithm's viability, however, further studies are necessary to attest scalability.
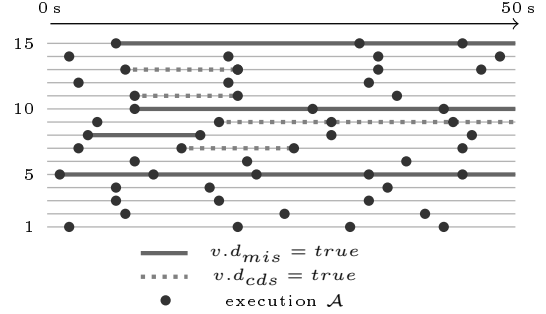
Fig. 3. Algorithm execution in testbed

B. Simulation

For the simulative evaluation of the proposed algorithm, relevant metrics are defined. The stabilization time after an initial deployment is of interest. For this, the predicates coverage (i.e. resulting set is dominating set) as well as connectivity of the induced subgraph are continuously evaluated during runtime. Furthermore, the number of dominators selected by the algorithm is counted, i.e. the numbers $|D_{\text{mis}}|$, $|D_{\text{weak}}|$, and $|D_{\text{cds}}|$. Regarding scalability the size of the CDS should not grow with the density of the network. Due to the limited bandwidth of the wireless channel, the number of messages sent by the algorithm is of interest. The algorithm runs continuously, so the required data rate has to be determined. In this context we distinguish between packets sent by the dominators and packets forwarded by neighboring nodes.

For the simulation we randomly dispersed between 200 and 2000 nodes into a field of size 500×500 units. The communication range is set to 50 units. The average degree is approximately 5.4 and 57.1 for the network of 200 and 2000 nodes, respectively, whereas only connected topologies are processed. For the simulation we increased density instead of network scale since the CDS algorithm is based only upon local knowledge. To provide a high confidence each depicted result is the average of 100 experiments.

The settings for the algorithm are adopted from the testbed evaluation except of T_{step} which is set to $T_{step} = 300\,\text{s}$ to provide non-interleaving operations with a high probability. Since we have used 802.15.4 compliant hardware for the testbed, a comparable MAC-Layer is used for the simulation (e.g., timings, CCA, inter-frame spacing). Note that all topologies are unit-disk graphs and thus no path loss model is applied. However, packet collisions are part of the simulation.

The size of the resulting CDS is shown in Fig. 4 in which a separation is done for dominators exclusively belonging to the sets MIS, WCDS, and CSD. The values stay constant after reaching a size of approximately 160 nodes for the networks of high density. Furthermore, the number of nodes connecting the WCDS to a CDS is marginal for higher densities. Here, we observed that for growing densities the probability of dominators of distance three decreases.
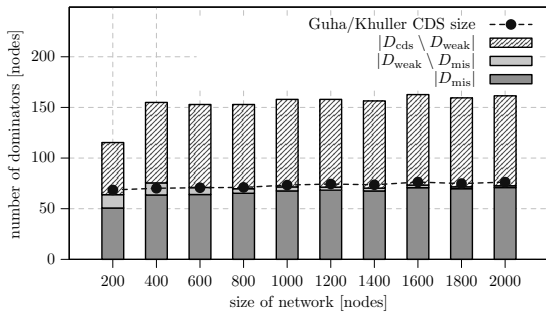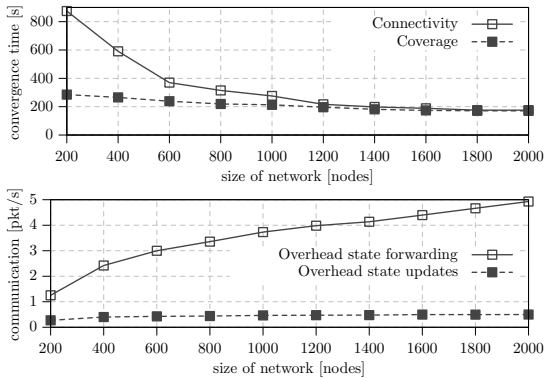
Fig. 4. Classification and number of dominators

Fig. 5. Convergence time and communication overhead

To classify the results we additionally applied Guha and Khuller's centralized CDS algorithm [12]. First, all nodes are marked white. The node with the highest id is then marked black and the whole neighborhood is marked gray. In the following a gray node is marked black (and adjacent white nodes become gray), if it has the highest count of adjacent white nodes. The algorithm terminates when all nodes are marked gray or black. We applied this algorithm for all used topologies. The size of the CDS (black marked nodes) is additionally depicted in Fig. 4. The centralized algorithm outperforms the self-stabilizing algorithm by a constant factor of approximately 2. Nevertheless, the result demonstrates the competitiveness of our distributed algorithm to centralized algorithms, which have global knowledge. Additionally, a higher CDS size, i.e. a higher degree of redundancy, is often preferable to increase the reliableness of the virtual backbone based upon the CDS.

The upper graph of Fig. 5 depicts the convergence time for covering the total network and for achieving connectivity. Obviously, the former is reached faster, because it is the outcome of the MIS algorithm. Due to the increased density the probability for executing the algorithm is increased and thus the convergence time for both properties is decreased to a value below $200\,s$.

Finally, the bottom graph of Fig. 5 shows the communication overhead measured in packets per second for the whole network. The rate for state updates rises from $0.26\,pkt/s$ for 200 nodes up to $0.50\,pkt/s$. The results are comprehensible by calculating the ratio between $|D_{cds}|$ (cf. Fig. 4) and $T_{step} = 300\,s$. The state forwarding creates an overhead of $1.1\,pkt/s$

up to $4.9\,pkt/s$ for large topologies. This utilizes only a small fraction of the possible data rate of the 802.15.4 standard. For both curves the slope is decreasing, so a realization of even larger topologies becomes viable. Note that a spatial reuse of the wireless channel is possible. Thus, the channel utilization is locally smaller than the given values.

VII. CONCLUSION

Connected dominating sets turn out as serviceable construct for efficient network protocols. This paper introduces a probabilistic self-stabilizing algorithm composed of three sub-algorithms for constructing CDSs. The issues of fault-tolerance, scalability and implementability are addressed. The achievement of the design criteria is successfully shown in simulations of large-scale, dense networks consisting of 2000 nodes and in a testbed evaluation with 15 nodes. Compared to recent local approaches for the CDS construction with complexities of $O(\Delta)$, our algorithm stands out for an applicableness in MANETs of high densities.

In the next steps, further studies of the influence of the parameters are required. The modularity of the algorithm also allows an optimization or exchange of the sub-algorithms. In addition an experiment in a heliostat field containing over 100 nodes is scheduled. For this, the link-quality between dominators has to be considered in order to provide stability of the resulting CDS. This can be done taking the link-quality indicator into account, which is obtainable for each received packet. In the context of a large-scale deployment the efficiency of applications based upon CDS can be examined.

REFERENCES

[1] J. Blum, M. Ding, A. Thaeler, and X. Cheng, "Connected dominating set in sensor networks and manets," *Handbook of Combinatorial Optimization*, pp. 329–369, 2004.
[2] F. Ingelrest, D. Simplot, and I. Stojmenovic, "Energy-Efficient Broadcasting in Wireless Mobile Ad Hoc Networks," in *Resource Management in Wireless Networking*, 2004, pp. 543–582.
[3] J. Wu, W. Lou, and F. Dai, "Extended Multipoint Relays to Determine Connected Dominating Sets in MANETs," *IEEE Transactions on Computers*, vol. 55, no. 3, pp. 334–347, 2006.
[4] K. M. Alzoubi, P.-J. Wan, and O. Frieder, "Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks," in *MobiHoc '02: Proc. 3rd ACM Int. Symp. on Mobile ad hoc networking & computing.* New York, USA: ACM, 2002, pp. 157–164.
[5] V. Turau and C. Weyer, "Fault tolerance in wireless sensor networks through self-stabilization," *International Journal of Communication Networks and Distributed Systems*, vol. 2, no. 1, pp. 78–98, 2009.
[6] S. Kamei and H. Kakugawa, "A Self-Stabilizing Distributed Approximation Algorithm for the Minimum Connected Dominating Set," *Int. J. Found. Comput. Sci.*, vol. 21, no. 3, pp. 459–476, 2010.
[7] W. Goddard and P. Srimani, "Anonymous Self-Stabilizing Distributed Algorithms for Connected Dominating Set in a Network Graph," in *IMCIC '10: Proc. Int. Multi-Conf. on Compl. Inform. & Cybern.*, 2010.
[8] T. Herman, "Models of Self-Stabilization and Sensor Networks," in *IWDC*, 2003, pp. 205–214.
[9] S. Hedetniemi, S. Hedetniemi, D. Jacobs, and P. Srimani, "Self-stabilizing algorithms for minimal dominating sets and maximal independent sets," *Comp., Math. & Appl.*, vol. 46, no. 5-6, pp. 805–811, 2003.
[10] B. Williams and T. Camp, "Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks," in *MOBIHOC 2002*, 2002, pp. 194–205.
[11] A. Varga, "The OMNeT++ Discrete Event Simulation System," in *ESM '2001: Proc. 15th Europ. Simul. Multiconf.*, Czech Republic, 2001.
[12] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, no. 4, pp. 374–387, 1998.