Topology Control for Fault-Tolerant Communication in Wireless Ad Hoc Networks *

Bernd Thallner (thallner@ecs.tuwien.ac.at), Heinrich Moser (moser@ecs.tuwien.ac.at) and Ulrich Schmid (s@ecs.tuwien.ac.at)
Technische Universität Wien, Embedded Computing Systems Group E182/2, Treitlstraße 3, A-1040 Vienna, Austria

Abstract. Fault-tolerant communication and energy efficiency are important requirements for future-generation wireless ad hoc networks, which are increasingly being considered also for critical application domains like embedded systems in automotive and aerospace. Topology control, which enables multi-hop communication between any two network nodes via a suitably constructed overlay network, is the primary target for increasing connectivity and saving energy here. In this paper, we present a fault-tolerant distributed topology control algorithm that constructs and continuously maintains a k-regular and k-node-connected overlay for energy-efficient multi-hop communication. As a by-product, it also builds a hierarchy of clusters that reflects the node density in the network, with guaranteed and localized fault-tolerant communication between any pair of cluster members. The construction algorithm automatically adapts to a dynamically changing environment, is guaranteed to converge, and exhibits good performance as well.

Keywords: Topology Control, Wireless Networks, Clustering, Redundant Paths, Fault-Tolerant Communication.

1. Introduction

During recent years, both the number of deployments and the properties/requirements of applications of wireless networks have steadily evolved. The spectrum ranges from cellular networks, which employ a fixed infrastructure of base stations supporting a large number of mobile devices, to wireless sensor networks, where a huge number of small devices create their own communications infrastructure in a fully distributed ("ad hoc") manner.

Increasingly, wireless ad hoc networks are also being considered as an alterative to fixed-infrastructure networks in critical application domains like aerospace and automotive [12]. For example, given that high-end cars are equipped with something like 50+ embedded microcontrollers nowadays, replacing the classic wireline interconnect by a wireless network is attractive in terms of costs and flexibility. Ad hoc

* This research has been supported by the FWF-project P17757-N04 (Theta) and P18264-N04 (SPAWN).

networks are particularly suitable here, since they also simplify systems integration — a notoriously difficult task in a multiple supplier business like automotive.

Whereas the relatively low number of devices and mobility requirements in such applications do not pose new challenges to wireless ad hoc networks, adding guaranteed real-time communication in the presence of failures does: A distributed control system cannot afford late delivery of a time-critical message because of, say, loss of a wireless link and subsequent establishment of an alternative path, but has to disseminate such messages along multiple disjoint paths. In addition, for cost reasons, network nodes have very limited resources and must comply to demanding robustness and power constraints imposed by harsh environmental conditions.

A particularly critical component with respect to energy-efficiency and fault-tolerant communication in wireless networks is topology control [13]: By selecting the particular neighbors a node may communicate directly with, topology control maintains a (sparse) *overlay graph* that is used for multi-hop communication between any two nodes in the network. Since many wireless networks require some dedicated communication hardware[1] for every active connection, the number of selected neighbors per node must be bounded by a (small) constant: For example, CDMA needs a correlator on both sender and receiver side for every wireless link. Consequently, even though wireless communication is a priori broadcast-based, only those receivers will actually receive a message from a specific sender that have set up a suitable receive channel. As usual chipsets provide a fixed number of such devices only, the overlay graph should be regular in order to optimally exploit the available resources and power.

The problem of constructing/maintaining such an overlay graph is further exacerbated by the fact that link performance can degrade, that both links and nodes can go down and recover, and that nodes can move. In order to guarantee unimpaired communication under such circumstances, the overlay network must be robust: It should provide fault-tolerance and adapt quickly to persistent changes in the environment. Suitable overlay graphs must hence be k-connected, for some $k > 1$, and should be maintained in a cost-optimal and self-healing way: If connections or nodes become expensive or go down for some time, the overlay graph must be adapted in order not to include them further.

[1] The same is true if link-level peer-to-peer hardware encryption/decryption is used.

Still, traditional topology control solutions (see Section 2) cannot cope with those requirements, since fault-tolerance is usually sacrificed for power efficiency: In order to reduce the total transmit power, topology control algorithms try to reduce the number of links in the overlay and thereby reduce the redundancy available for tolerating node and link failures. In fact, most topology control algorithms just construct a spanning tree, which obviously minimizes resource usage but must be reconstructed whenever a single wireless link in the tree goes down.

The topology construction approach utilized in this paper avoids this problem, by means of a separation of concerns: It guarantees some specified fault-tolerance, namely, a k-vertex-connected network, by choosing a suitable (but provably minimal) number of links to be included in the overlay graph. Power efficiency is maintained by selecting the most efficient links among the set of available ones. Every (potential) link in the network has associated an arbitrary *weight* for this purpose, that is, we assume that every node can probe or estimate how expensive or difficult it is to communicate with a specific peer. Distance, required transmission power, interference level or any combination of such quantities are legitimate weights here. Note that the assigned weights need not satisfy the triangle inequality and that we do not require homogeneous nodes or uniform transmission ranges. In addition, weights may be time-dependent: For example, a communication peer could be a moving node, and the required transmit power might not only depend on the distance but also on the instantaneous level of the internal (thermal) noise, multiuser interference, signal attenuation, multi-path fading, and many more. Likewise, a link to a receiver that temporarily suffers from excessive temperature or heavy processor load may be considered more costly than usual.

Given such a general weighted communication graph, the fault-tolerant distributed algorithm introduced and analyzed in this paper constructs and continuously maintains a k-regular and k-connected overlay graph. The construction is based on a suitable clustering scheme, which recursively forms groups consisting of k nodes that are treated like single nodes subsequently.

According to the separation of concerns mentioned above, the algorithm actually consists of two reasonably independent parts (that allow to easily adopt our scheme to different wireless networks):

1. The generic construction algorithm (see Section 5), which builds up and continuously maintains the k-regular and k-connected overlay graph. It does so by processing proposals for links to be added to the overlay graph supplied by the specific propose module (see next item).

2. The propose module (see Section 5.2), which tries to find minimal-weight links to be added to the overlay graph. The propose module is network-specific and allows to trade construction complexity for minimality of the weight-sum of the overlay graph (and hence overall power efficiency).

Our analysis reveals that the generic construction algorithm always converges, for any reasonable propose module. The eventually constructed overlay graph is k-connected, which is optimal, and thus ensures that k node-disjoint paths exist between any pair of nodes. It has low total weight and inherently provides failure-locality as well: Even excessively many failures in some part of the system do not impair fault-tolerant communication in other parts. As a by-product, the algorithm produces a hierarchy of clusters represented by a k-ary tree that reflects the nodes' spatial density. This property can be used in higher level services, like multi-hop routing, naming and geo/multicasting.

Organization of our paper: After a short survey of related work in Section 2 and some definitions in Section 3, we introduce our basic method for constructing a fault-tolerant communication topology in Section 4. Section 5 presents our fault-tolerant algorithm, which implements this method in a fully distributed way. Section 6 provides the proof of convergence, and Section 7 introduces an extension of our construction method. In Section 8, we present results of our comprehensive simulation studies. Some conclusions are provided in Section 9.

2. Related Work

Several non fault-tolerant topology control algorithms have been proposed in literature (refer to [13] for an overview). Most of them rely on the homogeneous network assumption with equal transmission range, which may not hold in practice [9].

A few fault-tolerant topology control algorithms ([5], [1] and [8]) have also been presented in recent years (see [5] and [8] for a summary): [5] presented approximation algorithms for minimum weight k-connected subgraphs based on the minimum spanning tree. However, [8] contains a counter-example which shows that the topology does not assure k-connectivity. [1] provides an extension of the CBTC algorithm to construct a fault-tolerant topology, which is proved to be k-connected but rests on the homogeneous network assumption. [8] presented a fault-tolerant extension for the greedy algorithm of [9], and derived a localized algorithm from it. [3] added k-connectivity to XTC, originally developed in [18], which is a strictly local topology control algorithm that can be used on general weighted graphs. Still, unlike our

algorithm, none of these solutions ensures a bounded node degree in general graphs and cannot therefore guarantee a small, a priori bounded number of connections per node.

3. Definitions

As our network model, we consider a simple undirected weighted communication graph $G = (\Pi, \Lambda)$ consisting of a set of n *nodes* $\Pi = \{1, ..., n\}$ and a set of weighted *edges* $\Lambda \subset \Pi \times \Pi \times \mathbb{R}$. It is assumed to be potentially fully connected, in the sense that $w < \infty$ for any edge $(p, q, w) \in \Lambda$.[2]

Note carefully that this seemingly very limiting assumption is not unrealistic in practice, in particular, for the envisioned applications mentioned in Section 1. The requirement is that any node *could* communicate with any peer, although it will actually communicate only with the nodes selected by our algorithm. In case of wireless network technology with power control, for example, this means that every peer must be reachable when maximum transmit power is used, but that (much less) transmit power can actually be used to communicate with the selected neighbor nodes. Moreover, in Section 7 we will introduce an extension of our topology construction scheme that allows to drop the finite weight assumption to some degree.

Our algorithm constructs a low weight *overlay graph* G' that is k-regular and k-connected, for some given k. Recall that a graph is *κ-connected* (also referred to as κ-node-connected or κ-vertex-connected) if the removal of any subset of $\kappa - 1$ nodes leaves the graph connected while there exists a subset of κ nodes whose removal disconnects the graph. A graph is *regular* of degree r if all nodes have degree r. In order to easily distinguish the communication graph G and the overlay graph G', the edges of the latter will be called *connections*.

We consider a two-tier[3] network consisting of *regular nodes* and *gateway nodes*. In addition to the wireless communication links to/from regular nodes, which have to be set up by our algorithm, all gateway nodes are assumed to be fully interconnected with all other gateway nodes via a dedicated backbone network. Formally, the set of nodes Π hence consists of n' *regular nodes* Π' and n'' *gateway nodes* Π'' with $n = n' + n''$ and $\Pi = \Pi' \cup \Pi''$. Note that $n'' \geq 2k - 2$ gateway nodes are sufficient, cf. Theorem 1. In order to ensure that gateway nodes are chosen as peers by our algorithm only after all regular nodes have been

[2] Since the edges are undirected, we actually use (p, q, w) as an abbreviation for $(\{q, p\}, w)$.

[3] Consult [14] for the treatment of a single tier topology without gateway nodes.

exhausted, it suffices to assume that the edge weight between regular and gateway nodes and between gateway and gateway nodes is k^2K and $2k^2K$ respectively, i.e., $\forall p \in \Pi', q \in \Pi'' \Rightarrow (p, q, k^2K) \in \Lambda$ and $\forall p \in \Pi'', q \in \Pi'' \Rightarrow (p, q, 2k^2K) \in \Lambda$, where K is the maximum edge weight between regular nodes. Our algorithm will then automatically construct a low weight overlay graph $G' = (\Pi' \cup B, C)$ with $B \subseteq \Pi''$ and $C \subseteq \Lambda$. Note that possibly not all gateway nodes are employed in the overlay graph here, and that gateway nodes in G' may have degree $k - 1$ due to the additional backbone interconnection. Regular nodes are always used up and always have degree k, however.

4. Topology Construction Method

In this section, we adapt and extend[4] the clustering scheme introduced in [14] to the above two-tier setting. The basic idea is to build groups of nodes that appear like single nodes when viewed from the outside, such that they can be treated like nodes subsequently. Applying this principle recursively eventually induces an overlay graph G' with the desired properties.

Figure 1 shows an example, where 1(a) is the fully connected communication graph G, 1(b) depicts the constructed overlay graph G' for $k = 3$, and 1(c) provides the tree representation corresponding to the constructed topology. From 1(b) it is apparent that the regular nodes $(1, 2, 3)$, $(4, 5, 6)$ and $(8, 9, 10)$ are combined into groups with id A, B, and D, respectively. Such a group is formed if all members agree on the fact that the sum of the weights of their internal connections (e.g. $4 - 5$, $4 - 6$, $5 - 6$) is minimal over all alternative group constructions. Each of the k members of a group is connected to all of the $k - 1$ other members (internal connections) and has exactly one connection left (external connection). Since there are k members in a group, any group has k external connections left, which are available in higher level groups. From the point of view of higher-level groups, groups hence look like nodes. We call the k nodes with one connection left the *terminal nodes* of a group.

For example, group C again consists of three members: A single node 7 and two groups A and B, which are connected via their external connections. Again, all members of C agree on minimality of the sum of their internal connections' weights. Nodes 1, 6 and 7, which are available for further external connections, are the terminal nodes of group C.

[4] More specifically, the results provided in this section differ from [14] in that we dropped the concept of the root group in favor of gateway nodes.

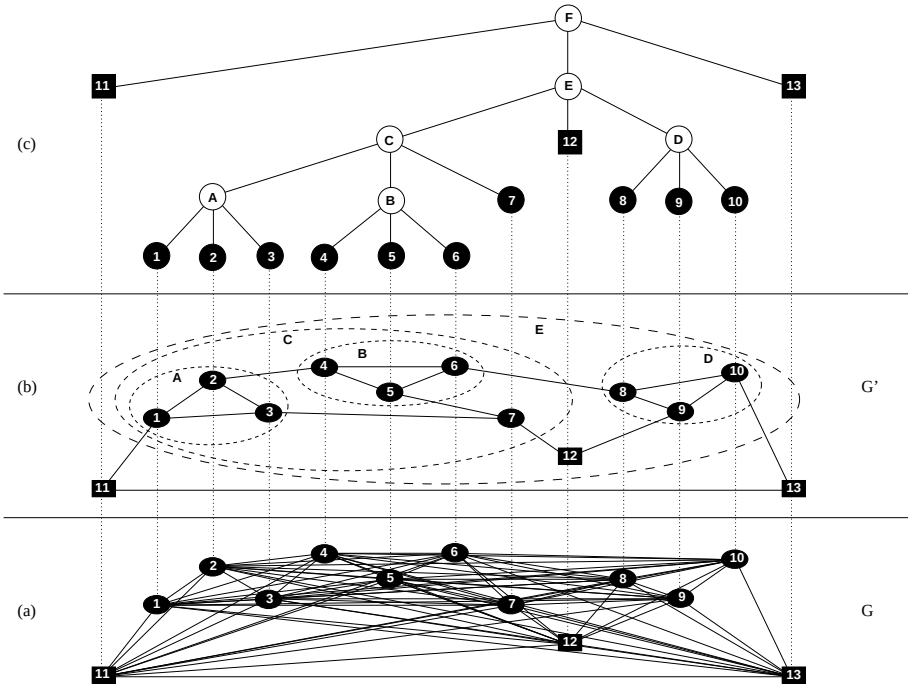

Figure 1. (a) Communication Graph, (b) Overlay graph, (c) Topology

Groups E and F finish the topology and include gateway nodes (11, 12 and 13), which may have degree $k-1$ due to the additional backbone connectivity. Figure 1(c) reveals that the resulting group structure is a k-ary tree. The edges of the tree represent the membership relation among the groups.

Formally, a *group* g is a pair consisting of $members(g)$, a set of k nodes and groups, and $int.con.(g)$ (*internal connections*), a set of $\binom{k}{2}$ edges.

For some given communication graph $G = (\Pi, \Lambda)$, let $\mathbb{G}$ denote the set of possible groups. For example, $\mathbb{G}$ for the graph depicted in Figure 1(a) would contain all groups A-F but also, for example, some group $A' = (\{2,3,4\}, \{(2,3,w),(3,4,w'),(2,4,w'')\})$ and some group $B' = (\{A',5,6\},\ldots)$, etc.

Formally, $\mathbb{G}$ can be constructed recursively: On the lowest level of the topology tree, $\mathbb{G}_0$ contains all groups that can be formed by grouping k nodes, i.e., for all sets $K \subseteq \Pi$ of k distinct nodes, some group g with $members(g) = K$ and $int.con.(g) = \{(p,q,w) \in \Lambda : p \in members(g) \wedge q \in members(g)\}$ is in $\mathbb{G}$. For higher-level groups, we first need some additional definitions:

For every node $p \in \Pi$ we define $nodes(p) = \{p\}$, and, recursively, for every group g, $nodes(g) = \bigcup_{g' \in members(g)} nodes(g')$, i.e., $nodes(g)$ contains all nodes of group g. For example, group C in Figure 1 has $members = \{\mathrm{A}, \mathrm{B}, 7\}$ and $nodes = \{1, \ldots, 7\}$. Likewise, $groups(p) = \{\}$ for a node p, and, for a group g, $groups(g) = \{g\} \cup \bigcup_{g' \in members(g)} groups(g')$.

With respect to terminal nodes, let $T_p = \{p\}$ for all $p \in \Pi$. Going upwards in the topology tree, we define $\mathbb{G}_i$ to contain all groups that can be built atop of $\Pi \cup \mathbb{G}_{i-1}$. Formally, some group g is in $\mathbb{G}_i$ if

1. $members(g) \subseteq (\Pi \cup \mathbb{G}_{i-1})$,
2. the nodes of all members are disjoint, i.e., $\forall g_i, g_j \in members(g) : g_i \neq g_j \Rightarrow nodes(g_i) \cap nodes(g_j) = \emptyset$,
3. the members of g are fully connected with each other via their terminal nodes, i.e., $\forall g_i, g_j \in members(g) : g_i \neq g_j \Rightarrow (p, q, w) \in int.con.(g) \wedge p \in T_{g_i} \wedge q \in T_{g_j}$, with w refering to the weight of the edge (p, q, w) in Λ,
4. if a member of g is a group (rather than a node), each of its terminal nodes is only used once for g's internal connections, i.e., $\forall g_i \in (members(g) \cap \mathbb{G}_{i-1}), \forall p \in T_{g_i} : |\{(p, q, w) \in int.con.(g)\}| \leq 1$.

These conditions, together with the requirement that $|int.con.(g)| = \binom{k}{2}$, imply that each member of g that is a group has exactly one terminal node which was not used in $int.con.(g)$. These nodes, together with the members of g that are nodes, form T_g, the k terminal nodes of g. Note that $\mathbb{G}_{i-1} \subseteq \mathbb{G}_i$, and that there can be many groups with the same members in $\mathbb{G}_i$ (using different internal connections). Since the total number of nodes is finite, eventually, no more groups can be added, i.e., there is some j such that $\mathbb{G}_j = \mathbb{G}_{j+1}$. We define $\mathbb{G} := \mathbb{G}_j$.

Let $G_g = (nodes(g), edges(g))$ denote the graph corresponding to group g, with $edges(g) = int.con.(g) \cup \bigcup_{g' \in members(g) \cap \mathbb{G}} edges(g')$. Observe that in G_g every node has exactly k connections, except for the nodes in T_g, which have $k - 1$ connections (cf. Figure 1(b), where the dashed regions show G_g for the group g whose label appears inside the region).

Finally, $B(G')$ of some overlay graph G' corresponds to G' extended by fully-connecting all gateway nodes contained in the node set of G'. This models the "dedicated backbone network" through which all gateway nodes are connected. Formally, if $G' = (N, E)$, $B(G') := (N, E \cup \{(p, q, w) \in \Lambda : p \in (\Pi'' \cap N) \wedge q \in (\Pi'' \cap N)\})$.

DEFINITION 1. *An overlay graph G' is called* admissible *if its corresponding topology consists of (a)* one *single top-level group g_{final} that*

incorporates all regular nodes and has only gateway nodes as terminal nodes, and (b) at most $k-2$ unused nodes (all of which are gateway nodes).

Formally, $G'$is admissible $\Leftrightarrow \exists g_{final} \in \mathbb{G} : (G' = G_{g_{final}}) \wedge (\forall p \in T_{g_{final}} : p \in \Pi'') \wedge (\Pi' \subset nodes(g)) \wedge (|\Pi - nodes(g)| \leq k-2)$.

The following Theorem 1 shows that no more than $2k-2$ gateway nodes are necessary to construct an admissible overlay graph.

THEOREM 1. *For every graph G with $n' \geq 1$ and $n'' \geq 2k-2$, there exists an admissible overlay graph G'.*

PROOF 1. *Consider the topology corresponding to an overlay graph G', where all regular nodes are used up for constructing a possibly incomplete sub-group X', like group E in Figure 1(b). If X' is incomplete, $x \in \{1, \ldots, k-1\}$ gateway nodes are required to complete X', effectively yielding some group X with at most $k-1$ regular terminal nodes. If X' is complete, one can form a group X with only one regular terminal node by joining X' and $k-1$ gateway nodes. In both cases, a group g where all terminal nodes are gateway nodes, i.e., no regular node with an external connection is left, can be constructed by joining the group X and $k-1$ additional gateway nodes. Note that this construction requires at most $2k-2$ gateway nodes.*

If all gateway nodes except for at most $k-2$ are used, $g = g_{final}$ and the construction is finished. Otherwise, we set $X = g$ and repeat the construction of g above until no more than $k-2$ unused gateway nodes are left.

The following theorems establish the most important properties of admissible overlay graphs.

THEOREM 2. *In every admissible overlay graph G' the node degree is bounded by k and all regular nodes have degree k.*

PROOF 2. *Obvious from the topology construction and Definition 1.*

THEOREM 3. *Each admissible overlay graph G' with $n' \geq 1$ and $n'' \geq 2k-2$ has $\lfloor \frac{n-1}{k-1} \rfloor$ groups.*

PROOF 3. *An admissible graph G' consists of n' regular nodes and as many gateway nodes as possible, therefore the k-ary topology tree of the overlay graph has at least $n-(k-2)$ and at most n leaf nodes, since no more than $k-2$ gateway nodes are left over by the group construction. A k-ary tree with $n^* = n-m, 0 \leq m \leq k-2$, leaf-nodes*

has $\frac{n^*-1}{k-1}$ *internal-nodes according to [7, Ex. 2.3.4.5.6]. We prove that* $X = \lfloor \frac{n-1}{k-1} \rfloor$ *is the only integer number* $X \in \mathbb{N}$ *for* $0 \leq m \leq k-2$ *that equals* $\frac{n^*-1}{k-1}$*: Since obviously* $\lfloor k + x - 1 \rfloor < \lfloor k \rfloor = k$ *for any integer* $k \geq 0$ *and* $0 \leq x < 1$*, choosing* $k = \frac{n^*-1}{k-1}$ *and* $x = \frac{m}{k-1}$ *reveals* $X - 1 = \lfloor \frac{n-1}{k-1} - 1 \rfloor = \lfloor \frac{n^*-1}{k-1} + \frac{m}{k-1} - 1 \rfloor < \lfloor \frac{n^*-1}{k-1} \rfloor = \frac{n^*-1}{k-1}$ *as required.*

THEOREM 4. *$B(G')$ of each admissible overlay graph G' with $n'' \geq 2k-2$ is k-connected.*

PROOF 4. *See Appendix A, Lemma 1 and Theorem 9.*

Up to now, we considered only properties of admissible overlay graphs that are independent of the edge weights. We will now introduce a suitable group weight and a specific joint-minimal criterion to be employed in our construction algorithm, which imply the existence of a unique joint-minimal admissible overlay graph.

Definition 2 introduces the weight of a group, which is primarily the sum of the weights of all internal connections, i.e., the weights of all the edges between its (direct) members. It also ensures, however, that the weight of a parent group is always higher than the weight of any of its members. In Section 6, we will show that this property is essential for ensuring that the joint-minimal admissible overlay graph introduced in Definition 3 below is well defined and that our distributed algorithm is guaranteed to converge.

DEFINITION 2. *The* numeric weight $\omega_{num}(p)$ *of a node p is 0. The* numeric weight $\omega_{num}(g)$ *of a group g is the maximum of the sum of all its internal connection weights and the maximum of its members' weights plus an arbitrary small constant ϵ, formally:*

$$\omega_{num}(g) = \max\left(\Big(\sum_{(p,q,w)\in int.con.(g)} w\Big), \Big(\max_{g'\in members(g)} \omega_{num}(g')\Big) + \epsilon\right)$$

The weight $\omega(g)$ *of a group g is the triple $(\omega_{num}(g), T_g, int.con.(g))$. A group g_a has smaller weight than g_b, formally $\omega(g_a) < \omega(g_b)$, if $\omega_{num}(g_a) < \omega_{num}(g_b)$ or, if equal, some arbitrary, well-defined tie-breaking rule based on the other two components (terminal nodes, actual internal connections) of ω is used.*

Note that the tie-breaking rule is necessary to enforce a total order among all possible groups, since different groups can be formed out of the same set of members, by using different terminal nodes or by connecting the same set of terminal nodes differently.

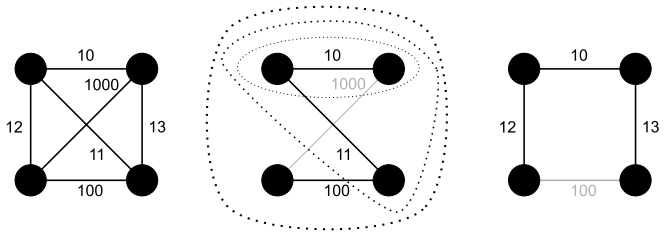

Figure 2. Comparison between a (weighted, non-Euclidean) communication graph, its *joint-minimal* overlay graph and its optimal overlay graph (k=2). For simplicity, no difference is made between regular and gateway nodes in this example. The backbone gateway connection is shown in grey.

The joint-minimum criterion given in Definition 3 requires that, for any group member x of a group, the weight-sum of all internal connections is minimal over all alternative group constructions involving x. Every member of a minimal group must hence arrive at the conclusion that there is no better choice, i.e., they must agree on minimality. It could hence be the case that a lower-weight alternative group existed for some member, but this choice would only be acceptable if all other members of the alternative group also considered this a better choice than their current parent groups.

DEFINITION 3. *An admissible overlay graph G' is* joint-minimal *if no alternative group g', i.e., a group that incorporates some member of an existing group $g \in G'$, can be built that has a lower weight than the current parent $g_y \in G'$ of every member y of g'. Formally, G' is joint-minimal if*

$$
\begin{aligned}
&G' \text{ is admissible} \wedge \exists g_{final} \in \mathbb{G} : G' = G_{g_{final}} \wedge \\
&\quad \nexists (g, g') \in groups(g_{final}) \times (\mathbb{G} - groups(g_{final})) : \\
&\qquad (\forall m \in members(g') : m \in (\Pi \cup groups(g_{final}))) \wedge \\
&\qquad (\exists x \in members(g) : x \in members(g')) \wedge \\
&\qquad (\forall g_y \in groups(g_{final}) : \\
&\qquad\quad (\exists y \in members(g_y) : y \in members(g')) \Rightarrow \omega(g') < \omega(g_y)).
\end{aligned}
$$

Note carefully that the range of group choices (and hence of admissible overlay graphs possibly generated by our construction algorithm) is of course restricted by the joint-minimal criterion: The joint-minimal admissible overlay graph is not necessarily the admissible overlay graph with the smallest total weight, but only the one where all groups satisfy some "local" minimality criterion. Figure 2 gives a simple example for

$k = 2$.[5] Since we will primarily mean joint-minimal when we are talking about minimality in this paper, however, the term "minimal" should be read as "joint-minimal" subsequently unless noted otherwise.

The joint-minimal admissible overlay graph is the admissible overlay graph constructed by our algorithm. This graph is well-defined: Choosing a new group according to this criterion does not lead to a violation of the joint-minimality of any already existing group in the final topology, since Definition 2 ensures that a higher-level group has a higher weight than any of its members. Hence, a newly built group cannot be more attractive (as an alternative member) for any already existing group. In addition, Theorem 1 holds also for the joint-minimal admissible overlay graph, since the edge weights for gateway nodes (k^2K resp. $2k^2K$ with K being the maximum edge weight between regular nodes) have been chosen in a way that guarantees that regular nodes are always preferred by the joint-minimum criterion. The following Theorem 5 proves that the joint-minimal admissible overlay graph indeed exists and is unique.

THEOREM 5. *For every graph G with $n' \geq 1$ and $n'' \geq 2k - 2$, there is exactly one joint-minimal admissible overlay graph G'.*

PROOF 5. *See Appendix A, Theorem 8. The proof inductively constructs joint-minimal groups, with increasing weights, one after the other, and shows that these groups are stable, i.e., not destroyed during later construction steps.*

We will use the term "joint-minimal group" to refer to a group that is contained in the unique admissible overlay graph.

The above theorems reveal several interesting features of our approach. First of all, connection weights may be arbitrary; in particular, they need not to satisfy the triangle inequality. Moreover, by adding additional constraints to Definition 3, overlay graphs with specific additional properties can be built.

If the weights in the communication graph reflect physical distance, our joint-minimal topology construction clusters nodes according to their spatial density. Nodes that are close to each other will be connected near the leafs of the topology tree, which leads to a nice failure-locality property of our overlay graph: Catastrophic failures in a spatially localized area will affect communication only in the immediate

[5] In case of $k = 2$, it is not hard to see that the minimum-weight admissible overlay graph is the tour provided by a solution to the traveling salesman problem (TSP), whereas our joint-minimal overlay graph is equivalent to the result of the greedy (multi-fragment) TSP heuristic [6]. Its competitive factor C is known to satisfy $\log n/(3 \log \log n) \leq C \leq (\lceil \log_2 n \rceil + 1)/2$ in Euclidean graphs; Section 8 will provide some additional data from our simulation studies.

neighborhood. In fact, failures that hit some part of the tree do not severely—if at all—impair fault-tolerant communication in other parts: The proof of Theorem 4 reveals that even failures that completely wipe out all members of some group g have only small impact on the connectivity of the remaining tree that results from purging the whole subtree rooted at g: Since at most one terminal node of g could also be terminal node of a higher-level group, all remaining nodes remain at least $k-1$-connected. In addition, all nodes within some intact subtree rooted at some member (or sub-member) of g remain also $k-1$-connected with each other, since only the single external connection possibly routed via g would be cut by the failures in g.

5. The Distributed Construction Algorithm

In this section, we introduce our distributed algorithm that builds up and continuously maintains the joint-minimal admissible overlay graph in dynamic environments. The complete correctness proof is presented in [10, 15].

5.1. Basic Algorithm

In Figure 3, we first give a high-level pseudo-code description of our algorithm. We assume an asynchronous system with reliable links, enriched with eventually perfect failure detectors [2]. Nodes may crash and connection weights are possibly time-variant. Late joining of nodes is allowed. In [11, 10], implementability issues have also been analyzed for the crash-failure model in asynchronous systems augmented with unreliable failure detectors, as well as for the crash-recovery model.

Our algorithm requires a non-blocking weak atomic commitment service like the one of [4], which is invoked by the candidate members of a to-be-formed group. We parameterize this service with the functions *vote*(), determining the local decision, *decision*(), executing the global decision, and *finalize*(), which will be executed after all participants have executed *decision*(), see [15] for details. In addition, we "piggyback" data on COMMIT votes that is used to recalculate group weights.

In our algorithm every existing group (that is, its terminal nodes) concurrently searches for the minimal-weight next-level group to join. For this purpose every node repeatedly generates group proposals P, consisting of the group members, the group weight, the group internal connections and the group's terminal nodes, which are sent to (the terminal nodes of) the proposed group members for confirming minimality. Generating a new proposal is typically triggered periodically, to

```
1   periodically
2     check neighborhood and generate proposal
3
4   when a new proposal is provided by the local propose module
5       broadcast proposal to all participants
6       if all participants agree that the proposal is better than their current parent group
7           all participants join the proposed new group
8
9   periodically
10      check for group consistency
11      if group is consistent
12          recalculate group weight
13      else
14          all participants leave the group
```

Figure 3. Basic Algorithm

facilitate adaption to changed connection weights, and on detection of a node crash or join. To simplify description and analysis, we encapsulate the functionality of generating proposals in a *propose module* that must satisfy the following specification:

DEFINITION 4. *A* propose module *generates proposals for groups consisting of k members, k terminal nodes, group internal connections and the corresponding group weight. A propose module is* perfect *if it eventually generates proposals corresponding to groups in the final (unique) joint-minimal overlay graph G' infinitely often.*

Note carefully that our basic construction algorithm does not require the propose module to be perfect, see Section 6 for details.

The pseudo-code of our distributed construction algorithm is given in Figure 4. ID is the unique id of a node; the id of a group *gid* is the set of its terminal node ids. The variable $Group[gid]$ stores the group-relevant information: The group members ($Members$), the group weight ($Weight$), the parent group id ($ParentID$), the group internal connections ($Connections$), the terminal nodes ($Terminals$) and the information, whether the group's construction has finished yet ($LockedBy$). This data structure is the same as used for proposals P generated by the propose module (except for $ParentID$ and $LockedBy$, which are initialized to $\perp$ by the propose module).

```
1   var Group[]
2
3   Group[{ID}].Members←{}
4   Group[{ID}].Weight←0
5   Group[{ID}].Connections←{}
6   Group[{ID}].Terminals←{ID}
7   Group[{ID}].ParentID←⊥
8   Group[{ID}].LockedBy←⊥
9
10  loop
11    wait for incoming message
12    if received proposal P from local propose module
13      initiate atomic commit PROPOSE_GROUP:
14        part. = all terminal nodes of all P.Members
15        data = P
16    if received signal to check for broken groups
17      var gid←Group[{ID}].ParentID
18      while gid ≠⊥ ∧Group[gid].LockedBy =⊥
19        initiate atomic commit CHECK_GROUP:
20          part. = all terminal nodes of all Group[gid].Members
21          data = Group[gid]
22        gid←Group[gid].ParentID
23    if received atomic commitment message
24      execute atomic commitment phase
25
26  function vote_PROPOSE_GROUP(group)
27    var gid←element gid of group.Members with ID ∈ gid
28    if want_to_join(gid, group)
29      return VOTE_COMMIT
30    else
31      return VOTE_ABORT
32
33  procedure decision_PROPOSE_GROUP(result, group)
34    var gid←element gid of group.Members with ID ∈ gid
35    if result = COMMIT
36      if want_to_join(gid, group)
37        join_group(gid, group)
38        Group[group.Terminals].LockedBy←nbac_id
39
40  procedure finalize_ PROPOSE_GROUP(result, ..., group)
41    var gid←element gid of group.Members with ID ∈ gid
42    if is_locally_consistent (group) ∧ \
43        Group[group.Terminals].LockedBy = nbac_id ∧ \
44        result = COMMIT
45      if finalize_result = ABORT
46        leave_group(gid)
47      if finalize_result = COMMIT
48        Group[group.Terminals].LockedBy←⊥
49
50  function want_to_join(gid, group)
51    return Group[gid] ≠⊥ ∧ \
52        (Group[gid].ParentID =⊥ ∨group.Weight < \
53          Group[Group[gid].ParentID].Weight) ∧ \
54        (Group[group.Terminals] =⊥ ∨Group[gid.Weight] < \
55          Group[group.Terminals].Weight) ∧ \
56        group.Weight > Group[gid].Weight
57  function is_locally_consistent (group)
58    return Group[group.Terminals] ≠⊥ ∧ \
59      Group[group.Terminals].Members = group.Members ∧ \
60      Group[group.Terminals].Connections = \
61        group.Connections
62
63  function vote_CHECK_GROUP(group)
64    var gid←element gid of group.Members with ID ∈ gid
65    if is_locally_consistent (group)
66      return (VOTE_COMMIT, \
67          (Group[gid].Weight, connection weights))
68    else
69      return VOTE_ABORT
70
71  procedure decision_CHECK_GROUP(result, group, commit_data)
72    var gid←element gid of group.Members with ID ∈ gid
73    if not is_locally_consistent (group)
74      return
75    if result = COMMIT
76      Group[group.Terminals].Weight← \
77        calculate_weight (commit_data, group)
78      if Group[gid].Weight ≥Group[group.Terminals].Weight
79        leave_group(gid)
80      else if Group[group.Terminals].ParentID ≠⊥ ∧ \
81          Group[group.Terminals].Weight ≥\
82          Group[Group[group.Terminals].ParentID].Weight
83        leave_group(group.Terminals)
84    if result = ABORT
85      if Group[gid] ≠⊥ ∧ \
86          Group[gid].ParentID = group.Terminals
87        leave_group(gid)
88
89  function calculate_weight (data, group)
90    var group_weights←create set (∀d ∈ data : d[0])
91    var connection_weight_sums←create set (∀d ∈ data : d[1])
92    return (max(∑ connection_weight_sums, \
93        (max group_weights) + ϵ), group.members, \
94          group.Connections)
95
96  procedure join_group(gid, group)
97    if Group[gid].ParentID ≠⊥
98      leave_group(gid)
99    Group[group.Terminals]←group
100   Group[gid].ParentID←group.Terminals
101   for all c in group.Connections
102     if ID is endpoint in c
103       make connection c
104
105 procedure leave_group(gid)
106   if Group[gid].ParentID ≠⊥
107     for all c in Group[Group[gid].ParentID].Connections
108       if ID is endpoint in c
109         cancel connection c
110     leave_group(Group[gid].ParentID)
111     Group[Group[gid].ParentID]←⊥
112     Group[gid].ParentID←⊥
```

Figure 4. Topology construction algorithm for node ID

The group structure is initialized only with the single-node group of the node itself. New proposals created by the propose module of the node are distributed to the terminal nodes of all members of the proposed group (PROPOSE_GROUP, line 13). An atomic commit procedure is used to ensure that, even in the case of node failure, either all members join the proposed group or none does. Nodes can crash and, thus, periodically, each node must check that the locally stored groups are still valid (CHECK_GROUP, line 19).

After a PROPOSE_GROUP request has been received, the terminal nodes of the member groups only decide on VOTE_COMMIT if the received proposal is better than the current parent group. After a COMMIT group decision has been reached, all participants call *join_group* to update their internal data, provided that the member group has not been destroyed or joined a better proposal in the meantime (which can happen because atomic commit group decision messages need not arrive in the correct order). *join_group*() leaves the current group first, if necessary. Then, the group structure is modified, and connections in the underlying infrastructure are being created.

When CHECK_GROUP is performed periodically, VOTE_COMMIT is only returned if the group is consistent with the group entry of the atomic commit initiator. If the node decides to ABORT, all members that still exist and have not left the group already leave it. *leave_group*() removes the connections built by the group and removes unnecessary data structures. If the participants decide to commit, the group weight is recalculated based on Definition 2 (implemented as function *calculate_weight*()).

5.2. Propose Modules

The propose module is an independent and highly network-specific part of our topology construction. Its purpose is to determine proposals for newly to-be-built groups, which are submitted to the basic construction algorithm. The latter determines whether a proposed group is indeed admissible and, if the case, builds it. Propose modules can be optimized w.r.t. various characteristics of the underlying wireless network, e.g., memory consumption, number of messages versus message size, size of neighborhood or the reaction time to network changes. Furthermore, the propose modules determine the quality (e.g. energy efficiency) of the generated overlay network and the complexity of its construction. The propose module is in fact crucial for liveness and performance, but not for convergence, correctness and fault-tolerance.

Typically, propose modules use simple search strategies to explore the search space of possible group constructions. The particular al-

gorithms are typically complex, however: To be practical, a propose module can only assume that a node knows the weights of its own edges, but no other ones. Hence, the search has to be done in a distributed and message-driven way. In this subsection, we provide a brief overview of a few simple propose modules and their worst case performance. Consult [15] for all the details, as well as for more elaborate propose module implementations.

In general, propose modules can be classified as follows:

(1) Search space exploration:

- Perfect: We call a propose module *perfect*, if it eventually generates proposals corresponding to every group in the joint-minimal admissible overlay graph infinitely often (until the corresponding group is eventually constructed). Hence, in some scenarios, it potentially searches the whole search space $\mathbb{G}$ for groups. Using perfect propose modules, it can be guaranteed that the unique joint-minimal admissible overlay graph is constructed within finite time after the communication graph becomes stable, i.e., neither changes its topology nor its edge weights.

- Non-perfect: We call a propose module *non-perfect*, if it eventually generates proposals corresponding to every group in some admissible overlay graph G' infinitely often (until the corresponding group is eventually constructed). G' must be joint-minimal with respect to the set of groups P proposed system-wide (this will be made precise in Definition 6 in Section 6). Non-perfect propose modules admit a restricted search space $P \subseteq \mathbb{G}$ (e.g., to achieve low message complexity), hence do not necessarily lead to the construction of the unique joint-minimal admissible overlay graph (but see Theorem 6).

(2) Distribution

- Global: A propose module is called *global*, if it uses some kind of central coordinator. Group proposals are proposed in a fully serialized order (e.g., with increasing group weight) here, such that formerly built groups are never destroyed through later group constructions.

- Local: A propose module is called *local*, if the group construction is fully distributed. Suboptimal groups may be built here, which have to be destroyed later on again.

- Locally agreed: A propose module is called *locally agreed*, if the construction is as concurrent as possible but ensures that no group is ever destroyed by later group constructions.

Note that local, global, and locally agreed propose modules can be either perfect or non-perfect.

Our implementation of a local perfect propose module assumes that every node p knows the weights of all the edges to its peers $q \neq p$ locally. It is based on the following idea: For making a proposal, the propose module of any node/group leader[6] p adds itself to a list of proposed members and calculates its potential members set pms_p based on the incompletely built overlay graph $(G')_p^e$ at time e as viewed locally by node p. This set contains all nodes/groups that are candidates for forming a group with less weight than their current parent groups. Then, p sends a SEARCH message to all nodes/leaders in the potential member set. Every node/leader q receiving such a SEARCH message does the same as p, except that it bases the calculation of its pms_q on pms_p instead of $(G')_p^e$. This process of forwarding is repeated $k-1$ times, unless the potential member set becomes empty prematurely. In the former case, a RESULT message containing the proposed members set is sent back to the originator p. Otherwise, an extinction notification for this particular forwarding "thread" is sent. The group comprising the members of the RESULT message with the lowest group weight is eventually proposed by p.

A local non-perfect propose module can be implemented by using a simple depth-first search. Rather than to all nodes/leaders in its potential member set, p sends a SEARCH message only to the node or group adjacent to the not-yet-tried lowest-weight edge here. Due to the DFS search, there is no need for the costly collection of all the group information for building up the initial pms_i of the local perfect propose module before starting the search here.

Both the local perfect and the local non-perfect propose module share a major disadvantage due to being purely local, however: Since proposals are generated concurrently on every node, it could happen that the groups making up the eventually generated admissible overlay graph could be identified only late, after many suboptimal groups (that have to be eventually destroyed) have been formed. A detailed analysis in [15] reveals that this leads to an exponential worst case message complexity $O(n^{2n})$ system-wide in failure-free executions.

This problem is avoided by global propose modules. A straightforward distributed implementation of a global propose module serializes the execution of the above local propose modules: A single node (e.g. the one with the lowest id) is used to initiate and collect the proposals

[6] For propose module implementation efficiency, it makes sense to (statically) elect one of the terminal nodes to be the only one to generate new proposals on behalf of the group.

Table I. Overview of the worst case complexities.

Complexity	Messages per proposal	Time per proposal	Messages total	Time total
Local Perfect	$O(n^{k-1})$	$O(n)$	$O(n^{2 \cdot n})$	$O(n^2)$
Local Non-Perfect	$O(n^2)$	$O(n^2)$	$O(n^{2 \cdot n})$	$O(n^3)$
Global Perfect	$O(n^k)$	$O(n)$	$O(n^{k+1})$	$O(n^2)$
Global Non-Perfect	$O(n^2)$	$O(n^2)$	$O(n^3)$	$O(n^3)$

from the local propose modules of all nodes and group leaders. Only the minimal proposal is proposed to the basic construction algorithm and hence put into the global topology tree $(G')^e$. Using the resulting global perfect resp. global non-perfect propose modules in conjunction with the basic topology construction algorithm yields the same time complexity[7] as their local counterparts. In sharp contrast to local propose modules, however, the message complexity is only polynomial, namely, $O(n^{k+1})$ resp. $O(n^3)$.

Different propose modules hence vary considerably in their worst case complexities, see Table I for a summary and Appendix B for (some of) the proofs provided in [15]. We should mention, however, that the exponential worst case scenarios are quite exotic and occur very rarely in practice. In fact, the simulation results in [15] reveal that the average message complexity is polynomial for all propose modules, see Section 8 for details.

In [15], there are also more elaborate propose module implementations like the *locally agreed* and the *probabilistic* propose modules, which try to optimize certain performance characteristics. Our best-performing propose module implementations have a linear worst case message and time complexity of $O(n)$ only. Note that there are also very efficient possibilities for constructing the topology tree in case of extended groups (see Section 7).

6. Convergence

Due to space limitations, we cannot provide the complete correctness proof and analysis of our algorithm provided in [15, 10] here. We will

[7] We consider the common asynchronous time complexity measure here, where the longest end-to-end message delay in an execution is normalized to the unit time 1. The worst case time complexity is the worst case running time of an algorithm in any such failure-free execution.

hence address the pivotal question of convergence only: Given that the algorithm is executed concurrently at all nodes, it is not obvious that its execution converges to some admissible overlay graph when the communication graph G becomes stable. This is not even obvious when all nodes employ a perfect propose module, in which case the joint-minimal admissible overlay graph should be constructed.

We will demonstrate one potential source of problems by means of the example shown in Figure 5: Rather than using the standard group weights $\omega(.)$ according to Definition 2, we consider a simpler group weight $\omega_{simple}(.)$ in conjunction with our topology construction algorithm here. According to Definition 5, the latter incorporates the sum of the weights of the group internal connections only.

DEFINITION 5. *The* simple weight *of a group* $\omega_{simple}(g)$ *is the sum of the internal connection weights of group* g*.*

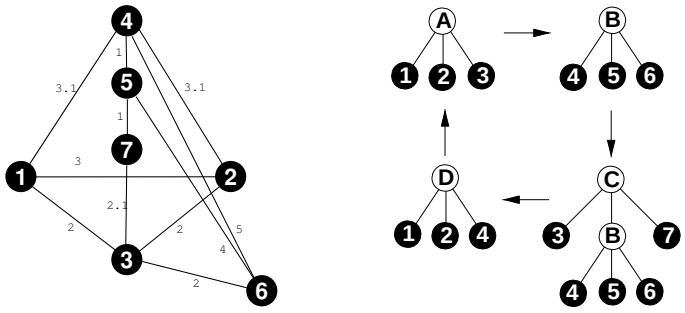

Figure 5. (a) Example of a communication graph leading to a non-convergent group construction under $\omega_{simple}(.)$, (b) resulting cycle $A, B, C, D, A, B, C, D, A, \ldots$

For $k = 3$, consider the communication graph G shown in Figure 5(a), where the weights of all drawn edges correspond to their Euclidean distance, whereas not drawn edges have some (much) higher weight. Even if the (perfect) propose modules always suggest the minium-weight group that can be formed from the already built groups at the moment, our algorithm enters an infinite cycle: The same sequence of groups $A, B, C, D, A, B, C, D, A, \ldots$ is repeatedly constructed/destroyed forever.

This can happen, since the connection weights are such that $\omega_{simple}(A) = 7 > \omega_{simple}(C) = 5.1$, $\omega_{simple}(B) = 10 > \omega_{simple}(D) = 9.2$ and $\omega_{simple}(D) = 9.2 > \omega_{simple}(A) = 7$. This effectively enables two completely independent group building (rather: destruction) sequences, namely A, C, D and B, D, which are not "connected" via any group weight relation between each other. Hence, as depicted in Figure 5(b),

they can [in case of our perfect propose module, will] occur in the cyclic sequence $A, B, C, D, A, B, C, D, A, \ldots$ forever. In fact, the joint-minimal admissible overlay graph does not exist for G if this simple weight function is used.

In order to avoid such cycles, some relation between independent group building sequences has to be enforced. Our original Definition 2 requires that a higher-level group has a higher weight than any of its members. In the above example, this ensures $\omega(C) > \omega(B)$ and hence breaks the cycle.

We now prove that our algorithm always converges, even without a perfect propose module. First, we generalize Definition 3 of the joint-minimal admissible overlay graph to a restricted set of groups $P \subseteq \mathbb{G}$ to choose from:

DEFINITION 6. *An admissible overlay graph G' is* joint-minimal *w.r.t. a given set $P \subseteq \mathbb{G}$ of groups if no alternative group g', i.e., a group that incorporates some member of an existing group $g \in G'$, can be built that has a lower weight than the current parent $g_y \in G'$ of every member y of g'. Formally, G' is joint-minimal w.r.t. P if*

$$
\begin{aligned}
&G' \text{ is admissible} \wedge \exists g_{final} \in P : G' = G_{g_{final}} \wedge \\
&\quad \nexists (g, g') \in groups(g_{final}) \times (P - groups(g_{final})) : \\
&\qquad (\forall m \in members(g') : m \in (\Pi \cup groups(g_{final}))) \wedge \\
&\qquad (\exists x \in members(g) : x \in members(g')) \wedge \\
&\qquad (\forall g_y \in groups(g_{final}) : \\
&\qquad\quad (\exists y \in members(g_y) : y \in members(g')) \Rightarrow \omega(g') < \omega(g_y)).
\end{aligned}
$$

Note that joint-minimal according to Definition 3 is equivalent to joint-minimal w.r.t. $\mathbb{G}$. It is easy to see that Theorem 5 also generalizes to the joint-minimal overlay graph generated w.r.t. a suitable set P of groups: The same inductive construction of "stable" groups, with increasing weights (restricted to members of P here) can be used in its proof here. Of course, the set P must be such that it allows the inductive construction to terminate with an admissible overlay graph, i.e., not to run out of groups prematurely.

THEOREM 6. *For every graph G with $n'' \geq 2k - 2$, there is exactly one joint-minimal admissible overlay graph G' w.r.t. the set P of groups ever constructed by our algorithm.*

Now consider our algorithm in conjunction with a non-perfect or perfect propose module, and let t_{GST} be the time where the communication graph G becomes stable, i.e., neither changes its topology nor

its weights. Since our algorithm periodically checks all existing groups, there is a finite time t'_{GST} after which all groups built before t_{GST} that are inconsistent w.r.t. G after t_{GST} have been destroyed. If P denotes the set of groups which exist in G' at time t'_{GST} or are built later on, then it follows immediately from the definition of non-perfect propose modules that our algorithm will construct the joint-minimal admissible overlay graph w.r.t. P if it never enters an infinite cycle like the one in Figure 5: Since every local propose module must propose the groups in the unique G' infinitely often, our algorithm will eventually build G'.

Theorem 7 finally proves that our algorithm indeed always converges: It shows that every group building action improves the current overlay graph towards the unique joint-minimal admissible overlay graph G', i.e., cannot enter a cycle.

THEOREM 7. *Our topology construction algorithm converges and eventually constructs some admissible overlay graph G' for every communication graph G with $n'' \geq 2k-2$, provided that G remains stable during a sufficiently long period of time. If perfect propose modules are used, then G' is the joint-minimal admissible overlay graph.*

PROOF 6. *By contradiction. Let us assume that, in some execution, there is a cycle $(G')^1, (G')^2, \ldots (G')^j, (G')^{j+1} = (G')^1, \ldots$ for some $j \geq 2$, where any $(G')^e$ is an incompletely built overlay graph that differs from $(G')^{e+1}$ in that a single additional group $g_i^{e+1} \in (G')^{e+1}$ has been built (and the former groups of the new members have been destroyed, if any). By the algorithm, every new group must have lower weight than the current one. For an arbitrary $1 \leq a \leq j$, consider some member x of a group $g \in (G')^a$ with weight $\omega(g)$. We claim that, in $(G')^{a+1}$, there is at least one group g' with weight $\omega(g') < \omega(g)$ (except in case (1), where $\omega(g') = \omega(g)$).*
Actually, there are 5 exhaustive cases:

(1) x remains in the same group $g \in (G')^{a+1}$, hence $g' = g \in (G')^{a+1}$.

(2) x is moved into the new group g_i^{a+1}, hence $g' = g_i^{a+1}$ since the joint-minimum criterion in Definition 3 requires $\omega(g_i^{a+1}) < \omega(g)$ for moving x.

(3) x is freed by moving some other member y of g into the new group g_i^{a+1}, hence $g' = g_i^{a+1}$ by the same argument applied to y.

(4) x is freed by destroying some other member y of g, by moving some of y's members (y must be a group here) into the new group g_i^{a+1}. Hence, $g' = g_i^{a+1}$ and $\omega(g') < \omega(g)$ by Definition 2.

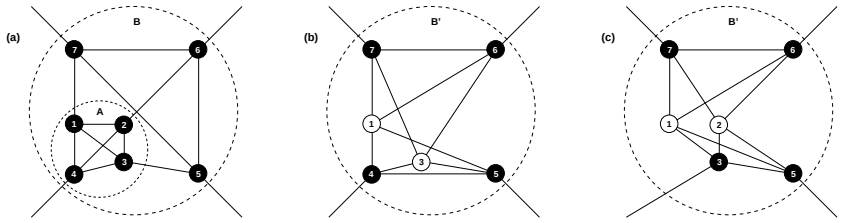

Figure 6. Regular and extended group structure

(5) x is destroyed by moving some of x's members (x must of course be a group here) into the new group g_i^{a+1}. By the same argument as in case (4), $\omega(g_i^{a+1}) < \omega(g)$.

Let x be the last member that drops into a lowest-weight group g' during the entire cycle, by one of the cases (2)–(5), so $\omega(g) > \omega(g')$. This member cannot escape from g', even at the end of the cycle $(G')^1$, hence cannot join the higher-weight group g again. This provides the required contradiction.

Since the algorithm applies the joint-minimum criterion of Definition 6, each successful group construction yields an overlay graph that better approximates the unique joint-minimal overlay graph G' secured by Theorem 6 (resp. Theorem 5 if the propose modules are perfect). Since all propose modules eventually propose the groups of G' infinitely often, G' will be constructed within finite time as asserted.

7. Extended Groups

A drawback of the topology construction method of Section 4 is that it is not particularly efficient for very dynamic environments. A node that joins or leaves the network could trigger a complete restructuring of the topology. Consider the case where a node $p \in T_{g_i}$ with $g_i \in members(g_{final})$ leaves the network, for example: All groups g with $p \in T_g$ have to be rebuilt on that occasion. In this section, we introduce an extension of our method that allows nodes to join and leave a group without reconstructing the entire topology, by keeping changes local. The extended topology construction algorithm also induces a low weight—although not the joint-minimal one—overlay graph $G^* = (\Pi' \cup B, C)$ with $B \subseteq \Pi''$ and $C \subseteq \Lambda$ that is k-regular for regular nodes and k-connected if $n'' \geq 2k-2$ and k is even, or $k-1$-connected if k is odd.

Figure 6(a) shows an example of a group and its parent group with $k = 4$. Figure 6(b) and 6(c) depict the restructured group after removing nodes 2 and 4, respectively. The groups A and B are merged together to form a new group B' which has more than k members. The topology outside B and inside the members of A and B (except A) is not changed. Note that the number of terminal nodes of B' does not change, and that nodes 1 and 3 in 6(b) as well as 1 and 2 in 6(c) have no external connections. While space limitations do not allow us to address this extension in detail, we invite interested reviewers to consult Appendix C or [16] for a formal description.

Introducing extended groups provides a number of additional benefits. For example, a node that is k or $k-1$-node-connected to the network preserves this property even when topology reconstruction is in progress. Most importantly, extended groups permit us to weaken the all-finite-weight assumption: Non-existing edges $(p, q, \infty) \in \Lambda$ were disallowed in Section 3, since x and y cannot become members of a common group—even if it is the only choice for them—if they cannot communicate with each other. With extended groups, this situation can be handled by just allowing x and y to join some nearby group as internal nodes. Particularly promising in this respect is a hybrid approach, which allows extended groups only above some particular level in the topology tree. The precise number of edges allowed to have infinite weight is a question still open for research.

8. Simulation Results

Our theoretical analysis in Section 6 revealed that the algorithm of Figure 3 always constructs an admissible overlay graph with the desired properties. It also provided worst-case bounds for the algorithm's message and time complexity in the standard asynchronous computing model, which, however, abstracts away many system parameters and implementation details: Efficiency-improving mechanisms in the propose modules, the nodes' actual computing & communication speeds, and, of course, the "average" communication graph and its weights the algorithm is typically working on.

In order to assess the algorithm's performance in realistic settings, two comprehensive simulation studies were conducted. For our first suite of experiments [15], we developed a Matlab-based simulation framework for assessing the typical (= average case) performance of our algorithm for all the different propose modules presented in Section 5. Since the primary purpose of those experiments was to confirm and complement our analytical worst-case performance analysis, im-

plementing a simple lock-step[8] execution model was sufficient here. In order to assess our algorithm's typical performance in a more realistic setting, a comprehensive ns2-based simulation framework was developed and used for a second suite of simulation experiments [17].

Essentially, all these experiments confirmed that the analytic worst-case message and time complexity bounds are also quite representative for the typical performance (except for those propose modules that lead to exponential worst-case complexities, which were also found to be polynomial with low degree in the average case). Interestingly, it also turned out that the quality of the resulting overlay graph, i.e., its total weight, is only marginally improved when more elaborate propose modules are used. Consequently, we will present some Matlab simulation results for the non-perfect but fast Local Probabilistic Propose Module only.

In all our Matlab experiments, we randomly placed a certain number of nodes within a 100×100 square and used their Euclidean distance for the edge weights in the communication graph. Figure 7(a) depicts a snapshot of an overlay graph for an example network with 50 nodes and $k = 3$. Gateway nodes with only $k - 1$ connections are emphasized by a surrounding square; (part of) the constructed group structure is circumscribed by the dotted lines.

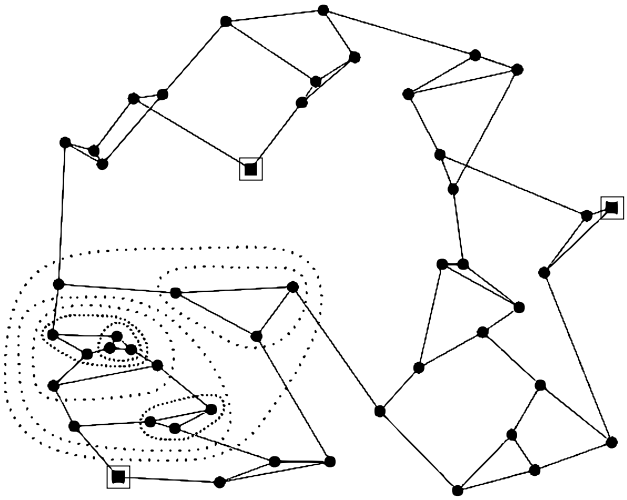

Figure 7. Overlay graph for an example network of $n = 50$ nodes and $k = 3$.

In order to assess the quality of the generated overlay graph, we calculated the sum of the weight of all edges contained in the overlay

[8] In a lock-step execution, every node node executes exactly one computing step—consisting of message reception (if any), some local computation and the optional sending of messages—per round.

Table II. Stretch factors.

Stretch factor	shortest path		second shortest path		longest path	
	avg.	max.	avg.	max.	avg.	max.
Distance stretch factor	2.0	6.3	3.1	14.1	4.9	19.3
Hop stretch factor	5.2	11.0	7.7	14.0	11.4	16.0

and compared it with the optimal solution for $k = 2$. Recall from Section 4 that the optimal solution is the tour provided by the traveling salesman problem (TSP) in this case, whereas the joint minimum criterion of Definition 3 used in our construction algorithm provides a sub-optimal solution only. Our simulation experiments revealed that the total weight computed by our algorithm was only 31% (average) resp. 50% (maximum) larger than the optimal TSP solution, which is very reasonable.

Deeper insights into the quality of the generated overlay graph are provided by the stretch factor for a pair of (random) nodes p and q, that is, the ratio between the weight of a path connecting p and q in the overlay and the weight of the edge (p, q) in the communication graph. Similarly, the hop stretch factor for p, q is just the number of hops between p and q in the overlay (since the number of hops between any pair of nodes is 1 in the communication graph). Table II gives the average and the maximum distance and hop stretch factors for each node-disjoint path from 50 random source and destination pairs in 50 random networks, for $n = 50$ and $k = 3$. We emphasize the very agreeable distance stretch, and the small hop distance between random nodes in the overlay graph, which reveals a small diameter of the overlay graph.

To give a flavor of the typical message and time complexity of our algorithm, Figure 8(a) displays the relation between the number of nodes n and the average number of messages required for the overlay construction divided by n, i.e., amortized over all nodes. Similarly, Figure 8(b) depicts the average number of rounds required for overlay construction over n. Both figures have been obtained by averaging over 25 random graphs with $k = 3$, and reveal a very agreeable average case performance.

Due to the simple execution model used in our Matlab simulation framework, the above message and time complexity is of course just a rough estimate. Those estimates have been confirmed by the wealth of more realistic ns2 simulation experiments documented in [17], how-

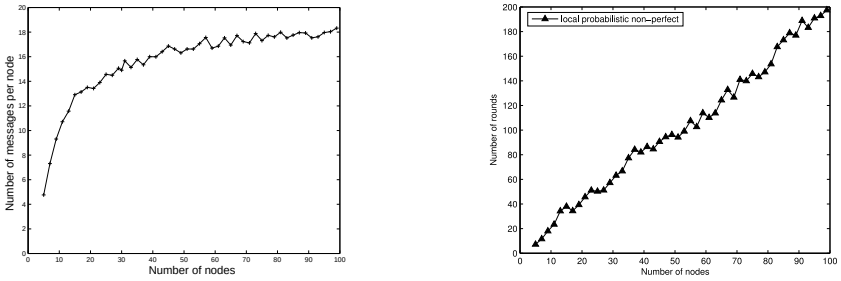

Figure 8. (a) Average number of messages per node. (b) Average number of rounds.

ever, which proved that our algorithm indeed provides low average case message and time complexity, small stretch factors and low total weight.

9. Conclusions

We presented and analyzed a distributed fault-tolerant algorithm for constructing an overlay graph for fault-tolerant communication in wireless ad hoc networks. The constructed overlay graph is k-regular, k-connected, ensures failure locality and has low total weight. The algorithm adapts to a dynamically changing environment, is guaranteed to converge, and exhibits very reasonable performance in systems with up to 100+ nodes. The hierarchy of clusters reflects the spatial density of the nodes and might replace some alternative under- and overlay clustering algorithms used e.g. for multi-hop routing, naming, geo- and multicast, etc. We complemented our analytical worst-case results by means of a glimpse on our comprehensive simulation studies (Matlab, ns2), which reveal a good average case complexity and small stretch factors in realistic settings.

Appendix

These proofs have been added to aid reviewing and will be removed in the final version. They will, however, be available to the journal reader as a downloadable technical report.

A. Topology Construction Method Proofs

THEOREM 8. *For every graph G with $n' \geq 1$ and $n'' \geq 2k-2$, there is exactly one joint-minimal admissible overlay graph G'.*

PROOF 7. *We first show that at least one joint-minimal admissible overlay graph exists, by inductively constructing joint-minimal groups (with increasing weights) one after the other and showing that these groups are stable, i.e., not destroyed during later construction steps. For $i \geq 1$, let g_i be the group added in the i-th step of this construction and G_{i-1} be the set of groups constructed in steps $1, \ldots, i-1$.*

For $i = 1$, g_1 is the (unique) group with minimal $\omega(g_1)$ chosen among all nodes in Π. Clearly, g_1 is joint-minimal and trivially stable. For g_i, $i > 1$, we choose the group with minimal weight $\omega(g_i)$ formed from groups and nodes in $G_{i-1} \cup \Pi$. Clearly, all members of g_i agree on this as the minimal choice. Moreover, it holds that $\omega(g_i) > \omega(g_{i-1})$: For, if g_i does not incorporate g_{i-1}, then it can incorporate only groups and nodes from $G_{i-2} \cup \Pi$, where g_{i-1} is the minimum-weight choice. If g_i incorporates g_{i-1}, then $\omega(g_i) > \omega(g_{i-1})$ according to the definition of $\omega(.)$.

It remains to be shown, however, that the choice to include some member $x \in G_{i-1} \cup \Pi$ in g_i does not violate the joint-minimum criterion for some earlier built group $g_j \in G_{i-1}$, $j < i$, which might already have x as a member. Since $\omega(g_j) < \omega(g_i)$, however, this cannot happen.

Finally, since it is ensured by our weight assumptions that all regular nodes are used up before gateway nodes are considered, Theorem 1 holds also for this inductive construction. Hence, the final group can be built since at least $2k-2$ gateway nodes are available.

It thus only remains to be shown that the joint-minimal overlay graph is unique. So let us assume that there exist two admissible overlay graphs G'_1 and G'_2, which are both joint-minimal. Going up the topology tree of G'_1 and G'_2, at some depth the group structure must be different. More specifically, there must be a group member x in some group g_i in G'_1 which is member in some other (= not corresponding) group g_j in G'_2. Recall that a node or a group can only be member of at most one group. However, either g_i of G'_1 or g_j of G'_2 has lower weight, which implies that the admissible overlay graphs G'_1 and G'_2 cannot both be joint-minimal according to Definition 3.

To show that our admissible overlay graph is k-connected, we first need an additional definition and a few preliminary lemmas.

DEFINITION 7. *A node p is* nd-connected *("node-disjointly connected") to a set $S \subseteq \Pi$, if p is connected to every $s \in S$ via node-disjoint paths. Formally: p is nd-connected to S, if there exists a set P of paths, containing a path from p to s for every $s \in S$, and p is the only node shared between any two paths in P.*

LEMMA 1. *Let $g \in (\mathbb{G} \cup \Pi)$ be a group or node. In G_g, every $p \in nodes(g)$ is nd-connected to T_g, the set of terminal nodes of g. Only nodes from $nodes(g)$ are required for these paths.*

PROOF 8. *Induction on depth d of g's topology tree. For $d = 0$, g is a node. Thus, $T_g = nodes(g) = \{g\}$ and the claim follows trivially.*

For some $d > 0$, suppose that the claim holds for every sub-group of g. Clearly, $p \in nodes(g_a)$ for some specific (direct) sub-group g_a of g and is hence nd-connected to T_{g_a}, resulting in a (multi-)set of k node-disjoint paths $(p, \ldots, t_a)$, for every $t_a \in T_{g_a}$. (If g_a is a node, all k paths are (p); if g_a is a group and $p \in T_{g_a}$, one of the k paths is (p).)

If g_a is a group, then, by construction, T_{g_a} contains one node t'_a that is also element of T_g, and $k-1$ other nodes t_a, each of which is connected to one terminal node t_b of another sub-group g_b of g. (If g_a is a node, it is both element of T_g and connected to all *other sub-groups of g). Thus, we can extend $k-1$ paths by following $k-1$ different $t_a \to t_b$ edges (one for every "neighbor" g_b of g_a). By applying the induction hypothesis, each of these t_bs is connected to all other terminal nodes of its group g_b, including the one which is also element of T_g, and, thus, each path can be continued to a terminal node of g. It is easy to see that all of these $k-1$ paths are node-disjoint, since each one was extended only with nodes from "its" unique g_b.*

The one remaining path does not need to be extended, since $t'_a \in T_g$.

THEOREM 9. *$B(G')$ of each admissible overlay graph $G' = G_{g_{final}}$ with $n'' \geq 2k-2$ is k-connected.*

PROOF 9. *Let p and q be two arbitrary nodes and $g \in groups(g_{final})$ be the lowest-level group such that p and $q \in nodes(g)$. We show that p, q are connected via k node-disjoint paths, which implies the statement of our theorem via Menger's Theorem [19].*

Let g be a group, $p \in nodes(g_a)$ and $q \in nodes(g_b)$ with $g_a, g_b \in members(g)$ and $g_a \neq g_b$. We claim that there are k node-disjoint paths between the two sets of terminals T_{g_a} and T_{g_b}. By the construction of a group, $k-1$ pairs of nodes of $T_{g_a} \times T_{g_b}$ are connected by disjoint paths routed over at most one sub-group. According to Lemma 1, there is always a (group-internal) path between any pair of terminals of this sub-group. Thus, we have one path $(p, \ldots, t_a, t_b, \ldots, q)$ and $k-2$ paths $(p, \ldots, t_a, t_{c1}, \ldots, t_{c2}, t_b, \ldots, q)$ (with different $t_a \in T_{g_a}$ and $t_b \in T_{g_b}$ each). By Lemma 1, we know that all $k-1$ $(p, \ldots, t_a)$ and $(t_b, \ldots, q)$ paths are node-disjoint (except for p and q, of course). Likewise, we have $k-2$ different sub-groups $g_c \in members(g)$, where a g_c-internal path $(t_{c1}, \ldots, t_{c2})$ exists for all $t_{c1}, t_{c2} \in T_{g_c}$.

Hence, we only have to show that the external connections of T_{g_a} and T_{g_b} are connected outside of g as well. Consider the parent group g' of g.

1. *If the terminal nodes $p' \in T_{g_a}$ and $q' \in T_{g_b}$ of group g are non-terminals in g', they are connected in g' via some path $(p', t_{c1}, \ldots, t_{c2}, t_{d1}, \ldots, t_{d2}, q')$, with g_c (terminals: t_{cx}) and g_d (terminals: t_{dx}) being other sub-groups of g'. Lemma 1 shows that the paths $(t_{c1}, \ldots, t_{c2})$ and $(t_{d1}, \ldots, t_{d_2})$ must exist and that they only require nodes from g_c and g_d, respectively.*

2. *If w.l.o.g. $p' \in T_{g'}$ but $q' \notin T_{g'}$, then q' is used for some internal connection in g', i.e., it is directly connected to some terminal t_{c1} of $g_c \in members(g')$. By Lemma 1, there must be some g_c-internal path $(t_{c1}, \ldots, t_{c2})$ to some terminal t_{c2} of g_c, which is also a terminal node of g'. Let $q^* = t_{c2}$ and continue with (3).*

3. *If both p' and q' or q^* are terminals in g', go up to the parent group of g' until either (1) or (2) applies or the final group is reached. If the final group $g' = g_{final}$ is reached, then p' and q' or q^* are terminals in g_{final} and therefore gateway nodes by Definition 1. Since all gateway nodes are fully connected in $B(G')$, the external connection follows trivially.*

Applying Menger's Theorem, the claimed connectivity follows.

B. Propose Modules Proofs

LEMMA 2. *The worst case message complexity of our local perfect propose module for generating a single proposal is $O(n^{k-1})$.*

PROOF 10. *Let $(G')^e_p$ be the incompletely built overlay graph at time e as viewed by node p. The maximum number of groups in $(G')^e$ is $< \lfloor \frac{n-1}{k-1} \rfloor = O(n)$ by Theorem 3. Since, in the worst case, p does not know about those groups (it need not be a member of any of those), $O(n)$ messages are needed for initially collecting the required group information.*

The maximum number of candidates in any initial potential member set is obviously $n + \lfloor \frac{n-1}{k-1} \rfloor$. Assuming that SEARCH *and* RESULT *messages are sent to every node and leader in the worst case, we obtain a message complexity of at most $2 \cdot \sum_{i=0}^{k-1} (n + \lfloor \frac{n-1}{k-1} \rfloor)^i = O(n^{k-1})$.*

LEMMA 3. *The worst case time complexity of the local perfect propose module for generating a single proposal is $O(n)$.*

PROOF 11. *According to the proof of Lemma 2, node p has to establish information about all groups in* $(G')_p^e$. *Since group information is distributed over the whole system, this collection process may take up to* $O(n)$ *time.* pms_p *may incorporate* $O(n)$ *nodes and groups initially. In the search phase, each* SEARCH *and* RESULT *message is forwarded* $k-1$ *times in the worst case, concurrently for all nodes and groups in* pms_p, *resulting in a time complexity of* $2 \cdot (k-1) = O(1)$. *Hence the claimed time complexity of* $O(n)$ *follows.*

LEMMA 4. *The worst case message complexity of our local non-perfect propose module for generating a single proposal is* $O(n^2)$.

PROOF 12. *Let again* $(G')_p^e$ *be the incompletely built overlay graph at time e as viewed by node p. Due to the DFS search, at most a single* SEARCH *and* RESULT *message is sent over any link connecting the at most* $n + \lfloor \frac{n-1}{k-1} \rfloor = O(n)$ *nodes/leaders in* $(G')_p^e$. *The worst case message complexity for generating a single proposal is hence at most* $O(n^2)$.

LEMMA 5. *The worst case time complexity of the local non-perfect propose module for generating a single proposal is* $O(n^2)$.

PROOF 13. *Since, by Lemma 4,* $O(n^2)$ *messages are sent in the worst case, the worst case time complexity cannot be larger*[9] *than* $O(n^2)$.

Combining the perfect resp. the non-perfect propose modules with the basic construction algorithm leads to the worst case time complexities given by Theorem 10.

THEOREM 10. *The worst case time complexity for generating a joint-minimal admissible overlay network* G' *with the basic topology construction algorithm using local perfect resp. local non-perfect propose modules is* $O(n^2)$ *resp.* $O(n^3)$.

PROOF 14. *The time complexity of generating a single perfect proposal is* $O(n)$ *by Lemma 3. Since all nodes generate proposals concurrently, after* $O(n)$ *time, the first minimal proposal, which is always accepted by the construction algorithm and never destroyed, is released (among other proposals). Since there are* $O(n)$ *groups to construct according to Theorem 3, the claimed time complexity of* $O(n^2)$ *follows.*

The result for the local non-perfect propose module is derived analogously, starting from the $O(n^2)$ *time complexity for a single proposal according to Lemma 5.*

[9] We note that there are simple variants of the DFS search algorithm that reduce the time complexity to $O(n)$.

C. Extended Groups

This section provides a formal description of our extended topology. For the remainder of this section, let $\mathbb{G}$ be the set of all regular and extended groups, as defined below.

The set of $members(g_i^*) \subseteq (\mathbb{G} \cup \Pi)$ of an *extended group* g_i^* consists of $k+1 \leq |members(g_i^*)| \leq 2k-2$ members. The set of terminal nodes of an extended group g_i^* consists of exactly k nodes. Since $|T_{g_i^*}| = k$, there is hence no difference between an extended and a regular group for an external node or group. The set of members of group g_i^* consists of terminal members $g \in members(g_i^*)$ with $\exists p \in nodes(g) : p \in T_{g_i^*}$ and internal members g with $\nexists p \in nodes(g) : p \in T_{g_i^*}$. We denote the number of internal members for an extended group g_i^* by $I_{g_i^*} = |members(g_i^*)| - k \quad (1 \leq I_{g_i^*} \leq k-2)$. Note that a regular group g_i can be seen as an extended group with $I_{g_i} = 0$.

An extended group g_i^* is constructed as follows:

1. Each internal member of g_i^* has a connection to each terminal member. Since there are k terminal members, each internal member has k connections.

2. Each terminal member of g_i^* has one external connection, $I_{g_i^*}$ connections to internal members and $k-1-I_{g_i^*}$ connections to other terminal members if k is even. If k is odd, at most one terminal member has $k-2-I_{g_i^*}$ connections instead of $k-1-I_{g_i^*}$ to the other terminal members.

The group weight of an extended group $\omega(g_i^*)$ is defined analogously to Definition 2, except that the sum of the g_i^* group internal connection weights is normalized in order to be comparable with the group internal connection weight of regular groups. For example, if k is even, we can normalize the weight of the extended group with $k \cdot (k-1) + I_{g_i^*} \cdot k$ connections to the weight of the regular group with $k \cdot (k-1)$ connections, hence the group internal connection weight of an extended group is $\frac{k \cdot (k-1) \cdot \text{sum of int. con.}}{k \cdot (k-1) + I_{g_i^*} \cdot k}$.

Accommodating extended groups in the distributed algorithm of Figure 4 requires only a few adaptations related to node joins and leaves:

- If a new node appears, it is integrated into some nearby group $g_i \neq g_{final}$ as internal member.

- If a node $p \in \Pi$ with $p \in members(g_i)$, $g_i \in members(g_j)$ and $g_j \neq g_{final}$ leaves the group, an extended group g_j^* is built with

$members(g_j^*) = (members(g_j) \setminus \{g_i\}) \cup (members(g_i) \setminus \{p\})$. The group g_i is removed. If $p \notin T_{g_j}$ then $T_{g_j^*} = T_{g_j}$; if $p \in T_{g_j}$ then $T_{g_j^*} = (T_{g_j} \cup \{q\}) \setminus \{p\}$ with $q \in (T_{g_i} \setminus \{p\})$. In the latter case, a higher level group g_a where $p \notin T_{g_a}$ but $p \in T_{g_b}$ with $g_b \in members(g_a)$ has to build a new connection to the new terminal node q, cf. Figure 6(c).

– If extended groups, regular groups and nodes are merged and the number of members of the new extended group would become $|members(g_j^*)| \geq k + k - 1$, a new regular group with k members and minimal weight is built and integrated as a single member into the new (extended) group. Note that restructuring proceeds in the opposite direction, from the root to the leaves of the tree, in this case.

We show that the overlay graph G^* with extended groups is k-connected if k is even. We start with some preparatory lemmas. Lemma 6 below shows that each node of an extended group has k connections to the terminal nodes of the extended group.

LEMMA 6. *Let g^* be an extended group and $T_{g^*} \subseteq nodes(g^*)$ with $|T_{g^*}| = k$ be its set of terminal nodes. Then, every $p \in nodes(g^*), g^* \in \mathbb{G}$ has k node-disjoint paths to nodes in T_{g^*} if k is even.*

PROOF 15. *Induction on depth d in the topology tree. For $d = 0$, g^* is a node; the claim hence follows trivially. For some $d > 0$, suppose that the claim holds for every sub-group $g_i \in members(g^*)$. Clearly, $p \in nodes(g_i)$ for some specific i and has hence k node-disjoint paths to T_{g_i}. There are two cases:*

1. *g_i is an internal member of g^*: By construction, every $q \in T_{g_i}$ is connected to one terminal node $q_j \in T_{g_j}$ of group g_j being a terminal member of g^*.*

2. *g_i is a terminal member of g^*: For $q' \in (T_{g_i} \cap T_{g^*})$ the connection is obvious, because a node is trivially connected to itself. We now show that every $q'' \in (T_{g_i} \setminus T_{g^*})$ also has a connection to one terminal node q_j in every T_{g_j} with g_j being one of the $k-1$ remaining terminal members of g^*. By the construction $k - 1 - I_{g^*}$ terminal nodes of T_{g_i} are directly connected to one terminal node $q_j \in T_{g_j}$ of the terminal member g_j. The remaining I_{g^*} terminal nodes are each one indirectly connected with one terminal node $q_j \in T_{g_j}$ of the remaining terminal members g_j via a separate internal member g'_j. By the induction hypothesis each terminal node in $T_{g'_j}$ has a*

connection to all terminal nodes in $T_{g'_j}$ and by the construction any terminal node $\in T_{g'_j}$ of an internal member g'_j has a connection to some terminal node q_j in T_{g_j} of every terminal member g_j.

Applying the induction hypothesis again, q_j is connected to every node in T_{g_j} and hence also to the one in $T_{g_j} \cap T_g$.

Lemma 7 shows that two groups which are both members in an extended group have α direct connections between each other, and both have β connections to terminal members of the parent group, where $\alpha + \beta = k$.

LEMMA 7. *Let T_{g_a} and T_{g_b} be the two terminal node sets of the groups g_a and g_b which are both members of the extended group g_i^*. There are α node-disjoint paths between T_{g_a} and T_{g_b}, and there are $2 \cdot \beta$ terminal nodes in $T_{g_i^*}$ each one connected (node-disjoint to α paths) with a dedicated terminal node in $X_a \cup X_b$, where $X_a \subseteq T_{g_a}$ respectively $X_b \subseteq T_{g_b}$ and $|X_a| = |X_b| = \beta \leq \frac{k}{2}$, with $\alpha + \beta = k$ if k is even.*

PROOF 16. *We must distinguish three cases for g_a and g_b:*

1. *If g_a and g_b are internal members of g_i^*, by the construction there are $\alpha = k$ node-disjoint paths between T_{g_a} and T_{g_b} routed over k terminal members. The claim follows trivially.*

2. *If g_a and g_b are terminal members of g_i^* then we show that there are α' direct connections between the terminal sets T_{g_a} and T_{g_b}, α'' node-disjoint paths routed over internal members, and α''' node-disjoint paths (and node-disjoint to α'' paths) routed over terminal members. Furthermore there are β terminal nodes of T_{g_a} respectively β of T_{g_b} each connected (node-disjoint to the α paths) with a terminal member of g_i^* and therefore with a terminal node of $T_{g_i^*}$. We show that $\alpha' + \alpha'' + \alpha''' + \beta = k$.*

   Each extended group has $I_{g_i^*}$ internal members therefore $\alpha'' = I_{g_i^*}$ paths routed over internal members exist. For each terminal member $|T_{g_i^*} \cap T_{g_a}| = 1$ respectively $|T_{g_i^*} \cap T_{g_b}| = 1$, therefore $\beta = 1 + \beta'$. If $I_{g_i^*} = k - 2$ each terminal member has only a single connection to some other terminal member, therefore $\beta' = 1$, $\alpha' = \alpha''' = 0$ or $\beta' = 0$, $\alpha' = 1$ and $\alpha''' = 0$; the claim follows. For $I_{g_i^*} = k - 3$ the terminal members are connected in one or more rings with $\alpha''' = 2$ and $\alpha' = \beta' = 0$ or $\alpha' = 1$, $\alpha''' = 1$ and $\beta' = 0$; the claim follows. If $0 \geq I_{g_i^*} \geq k - 4$ we assume two sets of terminal members A and B each connected to g_a respectively g_b. By the construction $|A| = |B| = k - 1 - I_{g_i^*}$. Therefore $\alpha' = |A \cap \{g_b\}| = |B \cap \{g_a\}|$,

$\alpha''' = |A \cap B|$ *and* $\beta' = |A \setminus (B \cup \{g_a\})| = |B \setminus (A \cup \{g_b\})|$*, from where the claim follows.*

3. *If w.l.o.g.,* g_a *is an internal member and* g_b *is a terminal member of* g_i^**, we show that there are* α' *direct connections,* α'' *node-disjoint paths routed over terminal members,* α''' *node-disjoint paths (and node-disjoint to* α'' *paths) routed over terminal and internal members (2-member-hops). Furthermore there are* β *terminal nodes of* T_{g_a} *respectively* β *of* T_{g_b} *each connected (node-disjoint to the* α *paths) to a terminal member of* g_i^* *and therefore with a terminal node of* $T_{g_i^*}$*. We show that* $\alpha'+\alpha''+\alpha'''+\beta = k$*. By the construction each internal member has a connection to each terminal member so* $\alpha' = 1$*. Each terminal member has* $k - I_{g_i^*} - 1$ *connections to other terminal members and each internal member has a connection to every terminal member, therefore* $\alpha'' = k - I_{g_i^*} - 1$*. There are* $I_{g_i^*} - 1$ *internal members of* g_i^* *connected to each terminal member of* $T_{g_i^*}$ *and hence* $k - (I_{g_i^*} - 1) - \alpha' - \alpha''$ *connections from* g_a *to terminal members of* $T_{g_i^*}$ *which are not used by* α' *or* α'' *paths already, therefore* $I_{g_i^*} - 1$ *node-disjoint paths routed via a terminal member of* $T_{g_i^*}$ *and a internal members between* T_{g_a} *and* T_{g_b} *exist and* $\alpha''' = I_{g_i^*} - 1$*. Since there are* k *terminal members of* $T_{g_i^*}$*, of which the* α'' *paths use* $k - I_{g_i^*} - 1$*, and the* α''' *paths use* $I_{g_i^*} - 1$*, there remain* 2 *terminal members unused by* α *paths. So* $|T_{g_i^*} \cap T_{g_b}| = 1$ *follows trivially and* g_a *has one connection to a single terminal member of* $T_{g_i^*}$ *not used by any* α *path and therefore* $\beta = 1$*. The claim follows.*

We are now ready for our final theorem:

THEOREM 11. *$B(G')$ of each admissible overlay graph G' with $n'' \geq 2k - 2$ and extended groups is k-connected if k is even.*

PROOF 17. *Let* p, q *be two arbitrary nodes and* $g^* \in \mathbb{G}$ *be the lowest-level group such that* $p, q \in nodes(g^*)$*. We show that* p, q *are connected via* k *node-disjoint paths, which implies the statement of our theorem via Menger's theorem. For* $p \in nodes(g_a)$ *and* $q \in nodes(g_b)$ *with* $g_a, g_b \in members(g^*)$ *and* $g_a \neq g_b$*, we claim that there are* k *node-disjoint paths between the two sets of terminals* T_{g_a} *and* T_{g_b}*. By Lemma 7 there are already* α *node-disjoint paths between* T_{g_a} *and* T_{g_b}*. Furthermore there are two subsets (node disjoint to nodes included in* α *paths)* $T'_{g^*}, T''_{g^*} \subseteq T_{g^*}$ *with* $|T'_{g^*}| = |T''_{g^*}| = \beta$*,* $|T'_{g^*} \cap T''_{g^*}| = 0$ *and* $\beta \leq \frac{k}{2}$ *so that* $\alpha + \beta = k$ *from Lemma 7. Hence, we only have to show that there are* β *external node-disjoint connections between* T'_{g^*} *and* T''_{g^*} *outside of* g^* *as well. By the topology construction each terminal*

node in T_{g^*} —hence also the ones in T'_{g^*}, T''_{g^*} —is connected to a separate group. So by Lemma 6 we can always route up the connection to the parent group until g_{final} is reached. Since in g_{final} all terminal nodes are gateway nodes, which are fully connected by convention, the claim follows. Applying Menger's theorem, the claimed k-connectivity follows.

The lemmas, theorems and proof for k is odd (showing $k-1$-connectivity) are similar and omitted for brevity.

References

1. Bahramgiri, M., M. Hajiaghayi, and V. Mirrokni: 2002, 'Fault-tolerant and 3-Dimensional Distributed Topology Control Algorithms in Wireless Multi-hop Networks'. In: *Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN02)*. pp. 392–397.
2. Chandra, T. D. and S. Toueg: 1996, 'Unreliable Failure Detectors for Reliable Distributed Systems'. *Journal of the ACM* **43**(2), 225–267.
3. Ghosh, S., K. Lillis, S. Pandit, and S. Pemmaraju: 2004, 'Robust Topology Control Protocols'. In: *Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS 2004)*.
4. Guerraoui, R.: 1995, 'Revisiting the relationship between Non Blocking Atomic Commitment and Consensus problems'. In: *Distributed Algorithms (WDAG-9)*.
5. Hajiaghayi, M. T., N. Immorlica, and V. S. Mirrokni: 2003, 'Power Optimization in Fault-Tolerant Topology Control Algorithms for Wireless Multi-hop Networks'. In: *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*. pp. 300–312.
6. Johnson, D. S. and L. A. McGeoch: 1997, 'The traveling salesman problem: a case study in local optimization'. In: E. Aarts and J. K. Lenstra (eds.): *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., pp. 215–310.
7. Knuth, D. E.: 1997, *The Art of Computer Programming*. Addison Wesley.
8. Li, N. and J. C. Hou: 2004a, 'FLSS: A fault-tolerant topology control algorithm for wireless networks'. In: *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*. New York, NY, USA, pp. 275–286.
9. Li, N. and J. C. Hou: 2004b, 'Topology Control in Heterogeneous Wireless Networks: Problems and Solutions'. In: *Proceedings of the IEEE INFOCOM'04*.
10. Moser, H.: 2005, 'Distributed Construction of a Fault-Tolerant Wireless Communication Topology for Networked Embedded Systems'. Master's thesis, Embedded Computing Systems Group, Vienna University of Technology. http://www.ecs.tuwien.ac.at/W2F/papers/thesis_moser.pdf.
11. Moser, H. and B. Thallner: 2006, 'Construction of a fault-tolerant wireless communication topology using distributed agreement'. In: *DIWANS '06: Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*. New York, NY, USA, pp. 35–44.
12. Nolte, T., H. Hansson, and L. L. Bello: 2005, 'Wireless Automotive Networks'. In: J. Kaiser (ed.): *Proceedings of the 4th International Workshop on Real-Time Networks (RTN'05) in conjunction with the 17th Euromicro International Conference on Real-Time Systems (ECRTS'05)*. Palma de Mallorca, Balearic Islands, Spain, pp. 35–38.

13. Santi, P.: 2005, 'Topology control in wireless ad hoc and sensor networks'. *ACM Comput. Surv.* **37**(2), 164–194.
14. Thallner, B.: 2004, 'Fault Tolerant Communication Topologies for Wireless Ad Hoc Networks'. In: *1st Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks (DIWANS'04)*. Florence, Italy. http://www.ecs.tuwien.ac.at/W2F/documents/diwans04.pdf.
15. Thallner, B.: 2005, 'Topology Control for Fault-Tolerant Communication in Wireless Ad Hoc Networks'. Ph.D. thesis, Embedded Computing Systems Group, Vienna University of Technology. http://www.ecs.tuwien.ac.at/W2F/documents/diss_thallner.pdf.
16. Thallner, B. and H. Moser: 2005, 'Topology Control for Fault-Tolerant Communication in Highly Dynamic Wireless Networks'. In: *Proceedings of the third International Workshop on Intelligent Solutions in Embedded Systems (WISES 2005)*.
17. Walter, C.: 2007, 'Simulation and Performance Evaluation of a Topology Control Algorithm in NS2'. Master's thesis, Embedded Computing Systems Group, Vienna University of Technology. http://www.vmars.tuwien.ac.at/documents/extern/2331/thesis_walter.pdf.
18. Wattenhofer, R. and A. Zollinger: 2004, 'XTC: A Practical Topology Control Algorithm for Ad-Hoc Networks'. In: *Proceedings of the 4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*.
19. Whitney, H.: 1932, 'Congruent Graphs and the Connectivity of Graphs'. *Journal of Mathematics* **54**, 150–168.