Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Computer Networks 43 (2003) 459–480
COMPUTER NETWORKS
www.elsevier.com/locate/comnet

GS^3: scalable self-configuration and self-healing in wireless sensor networks ☆

Hongwei Zhang *, Anish Arora

Department of Computer and Information Science, The Ohio State University, 2015 Neil Avenue, Dreese Lab 395, Columbus, OH 43210, USA

Abstract

We present GS^3, a distributed algorithm for scalable self-configuration and self-healing in multi-hop wireless sensor networks. The algorithm enables network nodes in a 2D plane to configure themselves into a cellular hexagonal structure where cells have tightly bounded geographic radius and the overlap between neighboring cells is low. The structure is self-healing under various perturbations, such as node joins, leaves, deaths, movements, and state corruptions. For instance, the structure slides as a whole if nodes in many cells die at the same rate. Moreover, its configuration and healing are scalable in three respects: first, local knowledge enables each node to maintain only limited information with respect to a constant number of nearby nodes; second, local self-healing guarantees that all perturbations are contained within a tightly bounded region with respect to the perturbed area and dealt with in the time taken to diffuse a message across the region; third, only local coordination is needed in both configuration and self-healing.

Keywords: Multi-hop wireless sensor network; Geography-aware self-configuration; Local self-healing; Locality; Cellular hexagon

1. Introduction

As increasingly small network nodes become available, many "sense-compute-actuate" networks are being realized. Several of these networks use unattended wireless sensor nodes [7,11,12], which communicate with one another via intermediate node relays due to limited transmission range or in order to save energy [15,20]. The number of nodes is potentially large (thousands and millions of nodes are considered in earthquake relief and unmanned space vehicle scenarios, for instance) [7]. Thus, scalability is a key issue for large-scale multi-hop wireless sensor networks.

One way to achieve scalability is by "divide and conquer", or hierarchical control. Network nodes are first grouped into a set of clusters by some clustering criterion. A leader is elected in each cluster to represent the cluster at higher levels. The

☆ An extended abstract containing some preliminary results of this paper appeared in 21st ACM Symposium on Principles of Distributed Computing (PODC 2002). This work was partially sponsored by DARPA grant OSU-RF-01-C-1901, NSF grant CCR-9972368, and an Ohio State University Fellowship.

* Corresponding author. Tel.: +1-614-292-1932/1836; fax: +1-614-292-2911.

E-mail addresses: zhangho@cis.ohio-state.edu (H. Zhang), anish@cis.ohio-state.edu (A. Arora).

1389-1286/$ - see front matter
doi:10.1016/S1389-1286(03)00354-2

same clustering scheme may be iteratively applied to the cluster leaders to form a hierarchy. In this hierarchy, local control is applied at each level to achieve certain global objectives.

Most existing work on clustering in wireless networks [3,18] treats a network as a geography-unaware graph. The clustering criteria adopted are, for instance, the number of nodes in a cluster and the logical—as opposed to geographic—diameter (in the number of hops) of clusters. However, geography-unaware clustering can be such that the communication links between a cluster leader and other nodes in its cluster are long, the geographic overlap between neighboring clusters is large, and routing traffic load is unbalanced across different clusters [8]. Therefore, more energy is consumed when a non-leader node communicates with its cluster leader by the only long link between them, simultaneous transmissions at neighboring clusters collide frequently, and energy dissipation is not balanced among clusters. Consequently, the lifetime of a network and the communication quality as well as efficiency in the network are reduced. Therefore, in order to save energy and improve communication quality as well as efficiency, the geographic radius of clusters should be taken into account in clustering algorithms.

Other reasons for considering the geographic radius of clusters (which we simply call *radius*, henceforth) in wireless sensor networks, especially large-scale, resource constrained multi-hop ones, include:

- Many multi-hop wireless sensor network applications, such as environment monitoring and temperature sensing, are inherently geography-aware, and so reflecting geography in the underlying network structure enables optimization of system performance.
- Cluster radius affects the efficiency of such local coordination as data aggregation and load balancing.
- Cluster radius affects the potential degree of frequency reuse in networks. The smaller the cluster radius, the more the frequency reuse.
- Cluster radius affects the scalability as well as availability of networks, since it affects the number of clusters and the number of nodes in each cluster (the more nodes in a cluster, the more available the cluster is).
- Given that nodes are approximately uniformly distributed and such fidelity control mechanisms as that in [4] adapt the number of active nodes in each area of a network, guaranteed radius of a cluster also guarantee the number of nodes in the cluster.

Challenges and opportunities. While accounting for geography in clustering, it is desirable that the radius of clusters be bounded from above as well as from below. The tightness of the bound impacts load balancing as well as the uniformity of energy dissipation in a network (and hence the lifetime of the network). It is also intuitively desirable that the tightness of the bound reflect such intrinsic network properties as node distribution density.

Given that multi-hop wireless sensor networks are expected to be untethered and of large scale, they demand automatic management [5]. Therefore, self-configuration is required in these networks, and it needs to be scalable to large network sizes. Moreover, self-healing is required in wireless sensor networks, because such complex perturbations as node join, node leave, node movement, node crash, and state corruption are expected to occur in these networks. Since even node crash and message loss perturbation can drive a network protocol into arbitrary states [13], self-healing of a network from arbitrary states is desirable. Furthermore, given that wireless sensor networks are of large scale, self-healing that is local is essential for the stability, availability, and scalability of a network [2]. By local self-healing, perturbations are dealt with and their impact contained in the locality near where the perturbations have occurred.

The demand of geography-aware clustering and local self-healing are not readily achievable in general large-scale systems. However, wireless sensor networks offer some distinguishing properties such as node distribution is dense, location information of nodes is effectively available, the degree of node mobility is low, and there exist some gateways between a sensor network and external networks such as the Internet [21,22]. These

properties offer opportunities to solve the problem in efficient ways, and we exploit them in the paper.

Contributions of the paper. In this paper, we design a distributed algorithm (GS^3) for configuring a wireless planar network into clusters (which we henceforth call *cells* due to their geographic nature). More specifically, the network nodes configure themselves into a *cellular hexagonal structure*, in which the network nodes are partitioned into hexagonal cells each with a radius that is tightly bounded with respect to a given value R (an ideal cluster radius) and zero overlap between neighboring cells. One node in each cell is distinguished, as the *head* of the cell, to represent this cell in the network. All heads in a network form a directed graph, called the *head graph*, which is rooted at a "big node" that is the interface between the wireless sensor network and external networks such as the Internet.

Our algorithm yields a local self-healing system. The head graph and cellular hexagonal structure are self-healing in the presence of various perturbations, such as one or more node joins, leaves, dies, moves, and state corruptions. The self-healing capability and the modular design of GS^3 enable different modules to be integrated so as to cater to different network models, in static as well as dynamic networks, in immobile as well as mobile networks, and in networks with one big node as well as multiple big nodes. Moreover, the self-healing is local such that the head graph and the cellular hexagonal structure remain stable upon perturbations in the following ways: (1) unanticipated node leaves within a cell are masked by the cell; (2) in case multiple cells experience node deaths at about the same time (due to energy exhaustion), independent shift of each cell enables the head graph as well as the cellular hexagonal structure to slide as a whole yet maintain consistent relative location among cells and heads; (3) in case the root of the head graph moves d away from its previous location, only the part of the head graph that is within $\sqrt{3}d/2$ radius from the root needs to change accordingly. Thus, a stable communication infrastructure for other services, such as routing, is configured in a dynamic or mobile network.

Our algorithm achieves scalability in three respects: (1) *local knowledge* enables each node to maintain the identities of only a constant number of nearby nodes; (2) *local self-healing* guarantees that all perturbations are dealt within and their impact is confined to a tightly bounded region around the perturbed area; the cellular hexagonal structure self-stabilizes within the time to diffuse a one-way message across the perturbed area; (3) only *local coordination* is needed in both the self-configuration and self-healing processes. (The complexity and convergence properties of GS^3 are summarized in Appendix A.1.)

The rest of the paper is organized as follows. In Section 2, we present the system model and problem statement. We then develop algorithms for static networks, dynamic networks, and mobile dynamic networks in Sections 3–5 respectively. We discuss related work in Section 6. Section 7 concludes the paper and makes further comments on the system model. For reasons of simplicity, we relegate the detailed description of algorithm modules to the Appendix.

2. System model and problem statement

2.1. System model

The system model consists of two parts: models for system nodes and perturbations.

System nodes: A system consists of a set of nodes in a 2D plane, each having a certain wireless transmission range.

> *Node distribution:* There exists R_t (called *radius tolerance*) such that, with high probability, there are multiple nodes in each circular area of radius R_t in the plane. [1]

There are two kinds of nodes: big and small. Intuitively, the big node acts as the initiator as well

[1] More specifically, nodes are distributed uniformly in the plane and the number of nodes in a circular area of certain radius is a Poisson random variable. We discuss this in detail in Section 4.3.4.

as the access point for small nodes. [2] That is, the big node initiates operations (such as clustering) at small nodes, and acts as the interface between small nodes and external systems such as the Internet. For convenience, we assume that the system has one big node, and all the other nodes are small (in Section 7, we discuss the case of multiple big nodes).

Wireless transmission: Nodes can adjust transmission range, and detect relative location with respect to other nodes. Destination-aware message transmission is reliable, but destination-unaware message transmission (such as broadcast) may be unreliable. [3]

Perturbations: We consider two types of perturbations: dynamic and mobile. The former consists of node joins, leaves, deaths, and state corruptions, and the latter consists of node movements.

Perturbation frequency: Node joins, leaves, and state corruptions are unanticipated and thus rare. Node death is predictable (e.g., as a function of its rate of energy dissipation). The probability for a node to move distance d decreases as d increases.

For pedagogical reasons, we classify networks into three: in a *static network*, there are neither dynamic nor mobile nodes; in a *dynamic network*, there can be dynamic nodes, but no mobile nodes; in a *mobile dynamic network*, both mobile nodes and dynamic nodes can exist.

2.2. Problem of self-configuration and self-healing

Informally, the self-healing configuration problem is to partition a system such that the maximum distance between nodes within a partition is bounded, each partition, called *cell*, has a unique distinguished node, called *head*, and the heads are organized into a *head graph* that is self-healing under perturbations. Nodes other than the head in a cell are called *associates*, and they communicate with nodes beyond their cell only through the cell head.

We define:

- *Head graph:* a tree that is rooted at the big node and consists of all cell heads.
- *Cell radius:* the maximum geographic distance between the head of a cell and its associates.

Formally, the problem is to design an algorithm that given R (*ideal cell radius*) where $R \geqslant R_t$, constructs and maintains a set of cells and head graph that meet the following requirements:

(a) Each cell is of radius $R \pm c$, where c is a small value with respect to R, and is a function of node distribution density characterized by R_t. [4]
(b) Each node is in at most one cell. [5]
(c) A node is in a cell if and only if the node is connected to the big node (i.e., there is a path between the node and the big node, and every two neighboring nodes in the path are within transmission range of each other). [6]

[2] Many wireless sensor networks have some central points that control system-wide operations. For example, in a field of disaster recovery, there is usually a commander for a group of rescue workers that is the central point. Sensor networks are also used to sample environment for sensory information (e.g., temperature) and propagate it to some central points [8].

[3] A network node can detect the strength of a received signal, and calculate its distance from the communicating peer [19]. Thus nodes can calculate relative location among themselves just by local information exchange in a dense network, even without GPS support. Moreover, when a node sends a message to some known node(s), the message transmission can be made reliable through such mechanisms as acknowledgement and retransmission.

[4] The primary goal of geography-aware self-configuration is to organize nodes into cells with certain ideal radius R that depends on application scenarios (e.g., data aggregation ratio and node distribution). In practice, a system may not be able to organize itself into cells of exactly the ideal radius R due to discrete node distribution, but the deviation of the actual radius from R still needs to be small enough, and be a function of node distribution density. Since the density of node distribution is characterized by R_t, the deviation should be a function of R_t.

[5] By guaranteeing that each node belongs to only one cell, energy can be saved, and the number of cells as well as control complexity is reduced.

[6] If a node is able (unable) to communicate with the big node before configuration, it should still be able (unable) to do so after configuration.

(d) The number of children for each node in the head graph is bounded. [7]
(e) The set of cells and the head graph are self-healing in the presence of dynamic as well as mobile nodes. By self-healing, a system can recover from a perturbed state to its stable state by itself. [8]

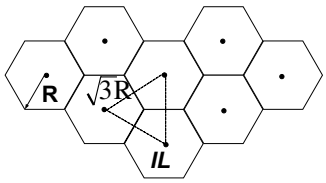


Fig. 1. Cellular hexagonal structure.

3. Static network

3.1. Concepts

Recall that in static networks, nodes are neither dynamic nor mobile. So we solve the problem without considering perturbations (i.e., requirement (*e*) is ignored). Moreover, we assume there is no R_t-gap in static networks, where an R_t-gap is a circular area of radius R_t with no node inside. R_t-gaps are dealt with as a rare perturbation in dynamic networks in Section 4.

Let us first consider an ideal case of the problem: given a plane with a continuous distribution of nodes, we may divide it into cells of equal radius R with minimum overlap between neighboring cells to obtain a cellular hexagonal structure as shown in Fig. 1. In this structure, each cell is a hexagon with the maximum distance between its geometric center and any point in it being R. Let the geometric center of a cell be the "head" of all points in the cell. [9] Then the distance between the heads of any two neighboring cells is $\sqrt{3}R$. And each cell that is not on the boundary of the plane is surrounded by 6 neighboring cells.

Of course, node distribution is not continuous in reality. Thus there may be no node at the geometric center of some cell and it may be impossible to divide the network into exact hexagons as in Fig. 1. But in scenarios where there are multiple nodes in any circular area of radius R_t, we can still approximate this structure by letting some node within R_t distance from the geometric center of a cell be a head, as is allowed in cellular networks [16].

Our solution is achieved in three steps. First, we cover a system with a hexagonal virtual structure as in Fig. 1 such that the big node is located at the geometric center of some cell. Second, for each cell C in the virtual structure, we choose a node k closest to the geometric center of C as a head, and the geometric center of C is called the ideal location (simply denoted as IL hereafter) of k, IL(k); Third, every non-head small node j covered by a cell C becomes an associate and chooses the best (e.g., the closest in a clockwise sense) head as its head, $H(j)$. A head together with its associates form a cell, and the IL of the head is also called the IL of the cell.

We designate the cell where the big node is as the central cell, and each set of cells of equal minimum distance from the central cell in terms of the number of cells in between as a cell band. If cells in a band are of d-cell distance from the central cell, this band is called a d-band, and the central cell alone forms the 0-band.

Next, we discuss a scalable distributed algorithm that implements the above concepts.

3.2. Algorithm

Overview. Generally speaking, there are two kinds of clustering methods: bottom–up and

[7] Since network traffic flows from children to parents along the head graph until reaching the big node, in order to guarantee load balancing and uniform rate of energy dissipation, the number of children for each node in the head graph should be bounded.

[8] In large-scale wireless sensor networks, complex perturbations can drive a network protocol into an arbitrary state, and the network cannot be managed manually. Therefore, self-healing of a network from arbitrary states is required. This goal is achieved by the technique of self-stabilization [1].

[9] The advantage of a cell head being at the center of the cell is that communication as well as energy efficiency is improved, since most communication within a cell is between the cell head and non-head nodes and the distance between the cell head and non-head nodes is minimized if the cell head is at the center of the cell.

top–down. In the wireless sensor network literature, only bottom–up clustering has been considered. But it cannot guarantee the exact placement and tight radius of clusters [10,12], therefore it does not solve the self-configuration problem as described in this paper. In contrast, the existence of big nodes in wireless sensor networks enables the top–down clustering approach, and our algorithm explores this approach with distributed control.

The self-configuration algorithm consists of a one-way diffusing computation across the network. The big node H_0 initiates the computation by acting as the head for the 0-band cell (i.e., the cell whose IL is at H_0), and selecting the heads of its neighboring cells in its *search region*. Then each newly selected head selects the heads of its neighboring cells in its search region, and so on until no new head is selected. Every node that has participated in the computation but not been selected as head becomes an associate and chooses the best head in the system as its head.

In the diffusing computation, the actual location of selected cell heads may deviate from the IL of the cell due to discrete node distribution. In order to prevent the accumulation of such deviation as the diffusing computation propagates far away from the big node, and to guarantee the exact placement as well as tight radius of cells, a unique Global Reference $\overrightarrow{\mathrm{GR}}$ direction [10] is diffused across the network along with the diffusing computation. Moreover, when a head selects its neighboring cell heads, it uses the IL of its cell instead of the actual location of itself (see Fig. 3 for detail).

If head i is selected by head j, we say that j is the *parent* of i, $P(i)$, and i is a *child* of j, $\mathrm{CH}(j)$. $P(H_0)$ is H_0. Then the search region of a head i is defined as the area that is within $\sqrt{3}R + 2R_t$ distance from $\mathrm{IL}(i)$ and between the two directions: L direction (LD) and R direction (RD) with respect to direction $\overrightarrow{\mathrm{IL}(P(i)),\mathrm{IL}(i)}$ (see Fig. 3). In order to guarantee that every node connected to H_0 is covered by the diffusing computation, $\langle LD, RD\rangle$ is chosen as $\langle 0°, 360°\rangle$ and $\langle -60° - \alpha, 60° + \alpha\rangle$ for H_0 and the other heads respectively, where $\alpha = \sin^{-1}(R_t/\sqrt{3}R)$.

In most cases, a $(d+1)$-band cell head is selected by a d-band head ($d \geqslant 0$). But in the case where the speed of the diffusing computation differs at different directions with respect to H_0, it is also possible that a $(d+1)$-band head is selected by a $(d+2)$-band head ($d \geqslant 1$). But this does not affect the correctness of GS^3-S, and it is dealt implicitly in the algorithm in Section 4. For simplicity, we do not discuss this any further here.

Algorithm modules. The algorithm (GS^3-S) consists of two programs [11] (described in Fig. 2): *Big_node* for the big node and *Small_node* for small nodes. Underlying these two programs are modules used for head organization: HEAD_ORG, used to organize heads, and HEAD_ORG_RESP as well as ASSOCIATE_ORG_RESP, used to respond to HEAD_ORG.

In HEAD_ORG, a head i (including the big node) organizes neighboring heads in its search region. It first gets the state (e.g., geographic location) of all the nodes in its search region by local information exchange; then it selects the neighboring heads using the low-level module HEAD_SELECT; last, it broadcasts the selected set of heads to nodes within $\sqrt{3}R + 2R_t$ distance. In HEAD_SELECT (described in Fig. 3), head i first calculates the ILs for the neighboring cells in its search region; then for each IL that is not the IL of an existing head, i selects the best node less than R_t away from the IL as a head.

In HEAD_ORG_RESP, a head sends its state in response to HEAD_ORG at another head at most $\sqrt{3}R + 2R_t$ away. In ASSOCIATE_ORG_RESP, which is executed by a small node i in response to HEAD_ORG at a head j at most $\sqrt{3}R + 2R_t$ away, if i already has a head, i sets j as its head only when j is better than its current head; if i does not have a head, it sends its state to j, and waits for j's decision of whether i is selected as a head, and sets its status accordingly.

[10] The global reference direction $\overrightarrow{\mathrm{GR}}$ can be any one, even though it needs to be consistent across the network.

[11] We use the notation of Guarded Command [9] to write algorithms.

Program *Big_node*

var q: {*bootup, work*}; //node status

q = *bootup* → HEAD_ORG(0^o, 360^o, R, R_t); //Big node organizes the 1-band cells;
//Then transits to status *work*

Program *Small_node*

var q: {*bootup, head, associate, work*}; //node status

q = *bootup* → ASSOCIATE_ORG_RESP; //Small nodes listen to nearby HEAD_ORG;
//Then transits to status *head* or *associate*

[]

q = *head* → HEAD_ORG(-60^o-α, 60^o+α, R, R_t); //Heads organize neighboring heads in their search regions;
//Then transits to status *work*

[]

q = *work* → HEAD_ORG_RESP;

[]

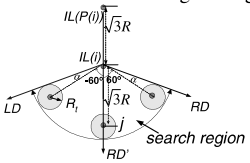
q = *associate* → ASSOCIATE_ORG_RESP; //Associates respond to HEAD_ORG;
//Remain status *associate*

Fig. 2. Self-configuration algorithm for static networks (GS^3-S).

Module *HEAD_SELECT* (SmallNodes, ExistingHeads, LD, RD, R, R_t)

Step 1: Calculate *IL*s of neighboring heads, *NH*, in the search region of *i*.

Use $\overrightarrow{IL(P(i)),IL(i)}$ as reference direction (*RD'*) (if *P(i)* = *i*, *RD'* can be any direction), *IL(i)* as origin, and $\sqrt{3}R$ as radius, go both clockwise and counterclockwise, the points on the arc that are $j\times60^o$ ($\lfloor LD/60 \rfloor \le j \le \lfloor RD/60 \rfloor$) degree from *RD'* are the *IL*s of neighboring heads.

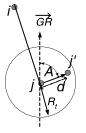


Step 2: Remove the set of *IL*s that are the *IL*s of some existing heads from *NH*. I.e. $NH \leftarrow (NH - EH)$, where $EH = \{j : j \in NH \wedge (\exists\, k \in \text{ExistingHeads} : (\, dist(j, k) \le R_t\,))\}$.

Step 3: For each *IL j* in *NH*, let $CA(j) = \{k : k \in \text{SmallNodes} \wedge dist(k, j) \le R_t\}$. *CA(j)* is the set of small nodes within R_t distance from *j*.

Step 4: For each *IL j* in *NH*, since *CA(j)* is non-empty, select the *highest ranked* node *j'* in *CA(j)* as the cell head corresponding to *j*, and set *CH(i)* as (*CH(i)* ∪ {*j'*}).

Every node *k* in *CA(j)* is lexicographically ordered by $<d, |A|, A>$, where *d* is the distance between *j* and *k*, *A* stands for the angle ($-180^o \le A \le 180^o$) formed by $\overrightarrow{GR}$ and $\overrightarrow{j,k}$ (*A* is negative if $\overrightarrow{j,k}$ goes clockwise with respect to $\overrightarrow{GR}$ and positive if counter-clockwise), and *d* has the highest significance.



❑

Time complexity: θ (|SmallNodes|)

Fig. 3. Module HEAD_SELECT used in HEAD_ORG.

A more detailed description of the individual modules of GS^3-S is given in Appendix A.2.

3.3. Analysis

In this subsection, we discuss the invariant, fixpoint, self-stabilization, and other properties of algorithm GS^3-S.

Notation

- *Physical network:* $G_p = (V_p, E_p)$, where $V_p = \{j : j \text{ is a node in the system}\}$ and $E_p = \{(i,j) : i \in V_p \wedge j \in V_p \wedge (i \text{ and } j \text{ are within transmission range of each other})\}$.
- *Head Graph:* $G_h = (V_h, E_h)$, where $V_h = \{i : i \in V_p \wedge i \text{ is a cell head}\}$ and $E_h = \{(i,j) : i \in V_h, j \in CH(i)\}$.
- *Head level structure:* the set of heads in a system and the geographic relation (such as relative direction and distance) among them.
- *Geographic coverage:* the geographic coverage of a node is the circular area on a plane that is centered at the node and has a radius equal to the current transmission range of the node. The geographic coverage of a system is the union of the geographic coverage of all the nodes in a system.
- *Boundary cell:* a cell that is on the boundary of the geographic coverage of a system or is a neighbor of an R_t-gap perturbed cell (Section 4).
- *Inner cell:* a cell that is not a boundary cell.

3.3.1. Invariant

We show the correctness of algorithm GS^3-S using an invariant, i.e., a state predicate that is always true in every system computation. Note that an invariant depends on the granularity of actions. Here we consider each algorithm module (e.g., HEAD_ORG) as an atomic action. The invariant SI for GS^3-S is $I_1 \wedge I_2 \wedge I_3$, where I_j $(j = 1,2,3)$ is individually closed under algorithm actions. The predicates are as follows:

I_1(Connectivity) $= I_{1.1} \wedge I_{1.2}$, where

- $I_{1.1}$: Every pair of heads that is connected in the head graph G_h is connected in the physical network G_p, and vice versa.
- $I_{1.2}$: The head graph G_h is a tree rooted at the big node H_0.

I_2 (Hexagonal structure) $= I_{2.1} \wedge I_{2.2} \wedge I_{2.3} \wedge I_{2.4}$, where

- $I_{2.1}$: Each inner cell head i has exactly 6 neighboring heads that form a cellular hexagon centered at i and of edge length $\sqrt{3}R$, with vertices' location deviation at most R_t. That is, the distance between neighboring heads is bounded by $[\sqrt{3}R - 2R_t, \sqrt{3}R + 2R_t]$.
- $I_{2.2}$: Each boundary cell head has less than 6 neighboring heads, and the distance between neighboring heads is bounded by $[\sqrt{3}R - 2R_t, \sqrt{3}R + 2R_t]$.
- $I_{2.3}$: Each head, except for the big node H_0, has at most 3 children heads. H_0 has 6 children heads if it is an inner cell head and at most 5 children heads otherwise.
- $I_{2.4}$: Each cell is of radius $(R + R_{\text{random}})$, where R_{random} is bounded by $[-2R_t/\sqrt{3}, 2R_t/\sqrt{3}]$. Each associate is no more than $(R + R_{\text{random}})$ away from its head.

I_3 (Inner cell optimality): Each associate in an inner cell belongs to only one cell and chooses the best (e.g., closest) head as its head.

Theorem 1. *SI is an invariant of algorithm GS^3-S, where SI $= I_1 \wedge I_2 \wedge I_3$.*

Theorem 1 and I_2 imply

Corollary 1. *The distance between neighboring cell heads is bounded by* $[\sqrt{3}R - 2R_t, \sqrt{3}R + 2R_t]$.

Corollary 2. *The heads and their cells form a cellular hexagonal structure* (*shown in Fig.* 4) *with bounded head location deviation* R_t.

3.3.2. Fixpoint

A fixpoint is a set of system states where either no action is enabled or any enabled action does not change any system state we are interested in (e.g., G_h). It therefore characterizes the result of the self-configuration process. The fixpoint SF for GS^3-S is $F_1 \wedge F_2 \wedge F_3 \wedge F_4$ as follows:

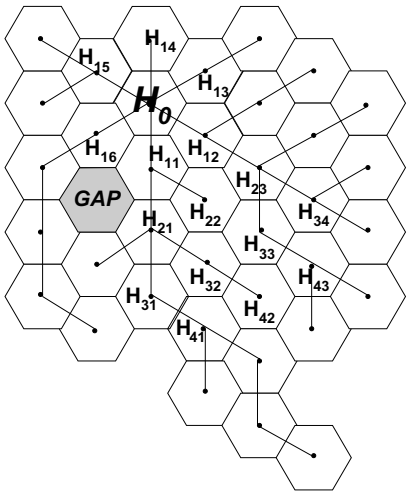


Fig. 4. Self-configured cellular hexagonal structure.

- F_1 (Connectivity) and F_2 (Hexagonal structure) are the same as I_1 and I_2 respectively.
- F_3 (Cell optimality): Each associate belongs to only one cell and chooses the best (e.g., closest) head as its head.
- F_4 (Coverage): The set of heads and cells covers all the nodes that are connected with the big node in the physical network G_p.

Theorem 2. *SF is a fixpoint of algorithm GS³-S, where* $SF = F_1 \wedge F_2 \wedge F_3 \wedge F_4$.

Requirements (a), (b), and (d) in the problem statement are satisfied by Theorems 1 and 2.

Theorem 2, F_1 and F_4 imply

Corollary 3. *At a state in SF, a node is in a cell if and only if it is connected to the big node in the physical network* G_p, *and vice versa.*

Requirement (c) in the problem statement is satisfied by Corollary 3.

3.3.3. Self-stabilization

Theorem 3. *Starting at an arbitrary state, every computation of GS³-S reaches a state in SI within a constant amount of time.*

Theorem 4. *Starting at an arbitrary state in SI, every computation of GS³-S reaches a state in SF within time* $\theta(D_b)$, *where* D_b *is the maximum Cartesian distance between the big node and any small node in the system.*

Theorems 3 and 4 imply

Corollary 4. *Starting at an arbitrary state, every computation of GS³-S reaches a state in SF within time* $\theta(D_b)$.

Termination of the diffusing computation follows from Corollary 4.

3.3.4. Scalability

The self-configuration algorithm GS³-S is scalable in that it only requires *local coordination* among nodes within $\sqrt{3}R + 2R_t$ distance from one another, and each node maintains the identities (e.g., MAC address) of only a *constant* number of nodes, 1 for associates and at most 6 for heads, irrespective of network size.

4. Dynamic network

4.1. Concepts

Recall that, in dynamic networks, nodes can join, leave (e.g., fail-stop), die, and node state can be corrupted. Excluding node death, which is predictable, the other perturbations are unanticipated and therefore rare. There may also be R_t-gaps in node distribution. In this section, we extend GS³-S to GS³-D to deal with these perturbations.

We propose three mechanisms to deal with node leave and death: head shift, cell shift, and cell abandonment. Self-stabilization easily handles the remaining perturbations, i.e., node joins and state corruptions.

Head shift: In dynamic networks, the associates in a cell are divided into two categories: *candidate* and *non-candidate*. Associates within R_t distance from the IL of the cell are head candidates, with the rest being non-candidates. In the case where only unanticipated head leaves occur, a new head

can be found with high probability from the set of candidates, due to the low probability of all candidates in a cell leaving at the same time. Moreover, the extreme case where all candidates leave can be dealt with by *cell shift*.

Cell shift: In case node death occurs, it is possible that the set of candidates of a cell becomes empty due to energy exhaustion after a long enough period of system operation. In this case, the IL of the cell is changed to another point IL′ within the geographic coverage of the cell such that the corresponding candidate set is non-empty, which is enabled by the fact that energy usually dissipates faster at a head than at an associate. In many envisioned large-scale wireless sensor networks, the traffic load across a network is statistically uniform due to in-network processing such as data aggregation [24], which means statistically uniform energy dissipation across the network. Given the fact that statistically there are multiple nodes in any circular area of radius R_t at the beginning of the self-configuration, the lifetime of any two sets of candidates at different cells is statistically the same with low deviation, especially for cells close by. Therefore, if the ILs at different cells change independently but in the same deterministic manner in terms of the relative position between IL and IL′, the head graph as well as head level structure will *slide* as a whole but maintain consistent relative location among cells and heads.

Cell abandonment: It is possible albeit rare that a cell is so heavily perturbed that nodes in an area of radius larger than R_t die at the same time. Even though cell shift may be able to change the IL of the cell to IL′, the distance between IL′ and the ILs of all neighboring cells may deviate beyond the range $[\sqrt{3}R - 2R_t, \sqrt{3}R + 2R_t]$. In this case, we let the cell to be abandoned in the sense that every node in it becomes an associate of one of the neighboring cells. (Note that, because of the sliding of the head level structure resulted from cell shift, a new head can be selected within the abandoned cell later.)

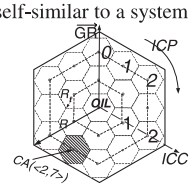
4.2. Algorithm

Overview. In GS^3-D, when a head i tries to select the heads for its neighboring cells in its search region, it is possible that there is an R_t-gap at the IL of a neighboring cell C (in this case, C is called an R_t-gap perturbed cell). Given the low probability of this case, i does not select head for cell C, and every node in C becomes an associate of a neighboring cell of C (this is similar to cell abandonment). Due to node join and the sliding of head-level structure, new nodes may show up in the area of C or the IL for C is changed such that there is a node within R_t distance to the IL of C later. By periodically checking this, head i will select the head for C whenever it shows up later.

When a node j joins an existing system, it tries to find the best existing head as its head if there is any within $\sqrt{3}R + 2R_t$ distance. Otherwise, j tries to find the best associate as its *surrogate head* if there is any associate within its radio transmission range. If both trials fail, j gives up and retries the above process after a certain amount of time. In the above process, if a head k within $\sqrt{3}R + 2R_t$ distance is executing HEAD_ORG, j responds with ASSOCIATE_ORG_RESP and becomes either a child head or an associate of k.

Node leave or deat h is dealt with by intra-cell and inter-cell maintenance. In *intra-cell maintenance*, *head shift* enables the highest ranked candidate to become the new head of a cell when the head of the cell fails or proactively becomes an associate when it is resource scarce or a candidate better serves as head; when the candidate set is weak (e.g., empty), *cell shift* enables the cell head to strengthen the candidate set by selecting a better IL for this cell if any such IL exists (described in Fig. 5); *cell abandonment* enables nodes within a heavily perturbed cell to become an associate in one of its neighboring cells. In *inter-cell maintenance*, a parent head and its children heads monitor one another. If a head h leaves and the intra-cell maintenance in its cell fails, the parent of h, $P(h)$, tries to recover it first. If $P(h)$ fails too, each child of h tries to find a new parent by themselves; also, a head chooses the neighboring head closest to the big node as its parent; an optional action is for a cell to synchronize its IL with that of its neighboring cells, which affects the tightness of cell radius with respect to R locally within its one-hop neighborhood.

We call a cell C formed in the initial phase of self-configuration an *original cell*, and the *IL* of C an *original ideal location (OIL)*. To maximize the lifetime of the hexagonal structure, for any original cell C, the union of its candidate sets of all the *IL*s should cover all nodes in C. Let $CA(IL_k)$ be the R_t-radius circular area centered at an ideal location IL_k. Then a cell can be divided into a set of such *CA*s as shown in the following figure, which is self-similar to a system being divided into a set of cells:



Analogous to "bands", we call each set of *CA*s of equal minimum distance to its *OIL* (in terms of *CA*s in between) an *Intra Cell Cycle (ICC)*. The set of *CA*s on the same *ICC* is numbered, called *Intra Cycle Position* (*ICP*), in an increasing order clockwise with respect to $\overrightarrow{GR}$ (for a certain *ICC*, the range for *ICP* is [0, 6×*ICP*-1]). Then the *IL*s in a cell can be lexicographically ordered by tuple <*ICC, ICP*>, and act as the "current *IL*" of a cell in an increasing order.

Fig. 5. Method to change the IL of a cell.

Node state corruption is dealt with by "sanity checking". Periodically (with low frequency) each head h checks the hexagonal relation with its neighboring heads, according to the system invariant. If the invariant is violated, h asks its neighboring heads to check their state. If all its neighboring heads are valid, the state of h must be corrupted, and h becomes an associate; if some of its neighboring heads are invalid, h cannot decide whether it is valid at this moment, and will check this next time.

Algorithm modules. Compared with GS^3-S, GS^3-D (described in Fig. 6) has modified modules for head organization, new modules for node-join, intra-cell maintenance, inter-cell maintenance, and sanity checking.

Modified modules for head organization are as follows. In HEAD_ORG, executed by a head i, i maintains not only its children heads set, but also its neighboring heads set and candidates set. In HEAD_SELECT executed by a head i, i does not select head for a cell in its search region if there is an R_t-gap at the IL of the cell. In HEAD_ORG_RESP, executed by a head i in response to the HEAD_ORG at a head j, i sets j as its parent if j is better (e.g., closer to the big node) than its current parent.

Node-join consists of three modules: SMALL_NODE_BOOT_UP used by a bootup node trying to find a nearby head or associate; HEAD_JOIN_RESP and ASSOCIATE_JOIN_RESP used by a head or an associate respectively in response to the SMALL_NODE_BOOT_UP at a nearby bootup node, where it sends its state to the bootup node and listens to its decision to join or not.

Intra-cell maintenance consists of four modules: HEAD_INTRA_CELL, CANDIDATE_INTRA_CELL, ASSOCIATE_INTRA_CELL, and BIG_SLIDE.

In HEAD_INTRA_CELL, executed by a head i, it exchanges heartbeats with associates in its cell. [12] Head i becomes an associate when it is resource scarce, a candidate better serves as head, or the big node is in its cell and resumes its role as head. When the candidate set is weak, i strengthens it using the low-level module STRENGTHEN_CELL that implements the concept of *cell shift*. If its cell is heavily perturbed such that the hexagonal property within its neighborhood has deviated too much, i abandons its cell and transits to status *bootup*.

In CANDIDATE_INTRA_CELL, executed by a candidate i, i exchanges heartbeats with its head. When its head fails or becomes an associate, i coordinates with other candidates in its cell to elect a new head. When its head transits to status *bootup*, i transits to status *bootup* too. When a

[12] The frequency of heartbeat exchanges can be tuned to minimize the control overhead and to adapt to such network states as traffic and degree of dynamics in a network.

```
Program Big_node
  GS3-S with modified HEAD_ORG;
  []
  q = work  → HEAD_JOIN_RESP;                                  //Deals with node join; remains status work
  []
  q = work → [HEAD_INTRA_CELL | HEAD_INTER_CELL];  //Deals with node leave:
                                                               //Remains status work or transits to status
                                                               // big_slide
  []
  q = big_slide  → BIG_SLIDE;          //The big node does not act as head; remains status big_slide or transits
                                       //to work

Program Small_node
  GS3-S with modified HEAD_ORG and HEAD_ORG_RESP;
  []
  q = bootup → SMALL_NODE_BOOT_UP;     //Remains status bootup or transits to associate or surrogate
                                       // associate
  []
  /* Head node */
    q = work → HEAD_JOIN_RESP;         //Deals with node join; remains status work
    []
    q = work → [HEAD_INTRA_CELL | HEAD_INTER_CELL];   //Deal with node leave;
                                                      //Remain status work or transits to
                                                      // associate
    []
    q = work --Ts--> SANITY_CHECK;                    //Sanity checking; remains status work or transits to
                                                      //associate
    []
  /* Associate node */
     (q = associate ∨ q = candidate) → ASSOCIATE_JOIN_RESP;   //Deals with node join;
                                                              //Remains status associate or
                                                              //candidate
    []
    q = candidate → CANDIDATE_INTRA_CELL;          //Deal with node leave;
                                                   //Remain status candidate, or transits to head
                                                   //or bootup
    []
    q = associate → ASSOCIATE_INTRA_CELL;          //Deal with node leave;
                                                   //Remain status associate, or transits to head
                                                   //or bootup
```

Fig. 6. Self-configuration algorithm for dynamic networks (GS^3-D).

head j that is better than its current head shows up, i sets j as its new head.

ASSOCIATE_INTRA_CELL executed by a non-candidate i is almost the same as CANDIDATE_INTRA_CELL except that i transits to status *bootup* when its head fails.

In BIG_SLIDE executed by the big node H_0, H_0 keeps the head in the coverage of its original cell as head, and resumes head role when the *OIL* of its cell becomes the current IL.

Inter-cell maintenance is implemented by the module HEAD_INTER_CELL. In HEAD_INTER_CELL, executed by a head i, i exchanges heartbeats with its neighboring cell heads. If a neighboring head j is closer to H_0 than its current parent, i sets j as its new parent. If a child j fails

and the intra-cell maintenance at its cell fails too, i tries to deal with it using HEAD_ORG in the direction of j. If the parent of i (i.e., $P(i)$) fails, and the failure is not recovered by the intra-cell maintenance at $P(i)$'s cell or by $P(i)$'s parent, i tries to find a new parent using low-level module PARENT_SEEK. If i is a boundary cell head, it periodically checks, using HEAD_ORG, whether new nodes show up in the direction where it does not have a child. When a neighboring head, a child, or its parent changes IL, i optionally synchronizes its IL using low-level module SYN_CELL.

Sanity checking is implemented by the module SANITY_CHECK whose time complexity is $\theta(D_c)$, where D_c is the diameter of a contiguous state-corrupted area.

A more detailed description of the individual modules of GS^3-D is given in Appendix A.2.

4.3. Analysis

New notation

- *Head neighboring graph*: $G_{hn} = (V_{hn}, E_{hn})$, where $V_{hn} = V_h$ of the head graph G_h, and $E_{hn} = \{(i,j)$: i and j are neighboring heads$\}$.

4.3.1. Invariant

The invariant of GS^3-D is the same as that of GS^3-S except for the following (formal descriptions are given in Appendix A.3):

- In $I_{2.1}$ and $I_{2.2}$, if the $\langle ICC, ICP\rangle$ value (see Fig. 5) of a head i is different from that of a neighboring head j, the distance between them is bounded by $[d - 2R_t, d + 2R_t]$, where d is the distance between IL(i) and IL(j), and is bounded by $(0, 2\sqrt{3}R)$.
- In $I_{2.3}$, the number of children heads of a head other than the big node is at most 5.
- In $I_{2.4}$, the radius of an inner cell is bounded by $(0, 2R + R_t]$ if its $\langle ICC, ICP\rangle$ value is different from that of any of its neighboring cell; and $|R_{random}|$ is at most $(\sqrt{3} - 1)R + 2R_t + d_p$ for boundary cells, with d_p being the diameter of the R_t-gap perturbed area adjoining the boundary cell (d_p is 0 if there is no R_t-gap perturbed area).

Theorem 5. *Let DI be SI (invariant of GS^3-S) with I_2 relaxed as above, then DI is an invariant of algorithm GS^3-D.*

4.3.2. Fixpoint

The fixpoint of GS^3-D is the same as that of GS^3-S except for the following:

- $F_{1.2}$ is strengthened as: the head graph G_h is a minimum-distance (with respect to the big node H_0) spanning tree of the head neighboring graph G_{hn} rooted at H_0, i.e., the path between H_0 and a head i in G_h is a shortest path between H_0 and i in G_{hn}.
- $F_{2.4}$ is relaxed as: ($F_{2.4}$ of GS^3-S)$\wedge$($|R_{random}|$ is at most $2R_t/\sqrt{3} + d_p$ for boundary cells).

Theorem 6. *Let DF be SF (fixpoint of GS^3-S) with $F_{1.2}$ and $F_{2.4}$ updated as above, then DF is a fixpoint of algorithm GS^3-D.*

F_1, F_2, F_3, and F_4 imply

Corollary 5. *At a state in DF, Corollaries* 1, 2 *and* 3 *hold in dynamic networks.*

4.3.3. Self-stabilization

Theorem 7. *Starting at an arbitrary state, every computation of GS^3-D reaches a state in DI within time* $O(D_c)$, *where D_c is the diameter of a contiguous state-corrupted area.*

Theorem 8. *Starting at an arbitrary state in DI, every computation of GS^3-D reaches a state in DF within time* $O(\max\{D_d/c_1, T_d\})$, *where D_d is the geographic diameter of the network, c_1 is the average speed of message diffusing, and T_d is the maximum difference between the lifetime of the candidate sets of two neighboring cells.*

Theorems 7 and 8 imply

Corollary 6. *Starting at an arbitrary state, every computation of GS^3-D reaches a state in DF within time* $O(\max\{D_d/c_1, T_d\})$.

Requirement (e) in the problem statement is satisfied by Theorems 7 and 8.

4.3.4. Statistically low deviation from ideal hexagonal structure

Of course, R_t-gaps may always exist in networks, and this implies the potential existence of non-ideal cells that are not hexagonal. If the IL of a cell C in the ideal virtual structure (as shown in Fig. 1) lies in an R_t-gap, then every node in the geographic coverage of C joins some neighboring cell C' of C in the self-configured cell structure, which makes C' assume a shape other than the "ideal" hexagon. Moreover, due to the existence of R_t-gaps, the radius of such non-ideal cells as C' depends on the diameter of the R_t-gap perturbed region (i.e., the set of contiguous R_t-gap perturbed cells such as C) adjoining it. However, as shown below, the number of non-ideal cells and the diameter of R_t-gap perturbed regions are small due to dense node distribution in wireless sensor networks.

We assume that nodes are uniformly distributed such that the average number of nodes within any circular area of radius 1 is λ. Let m_0 be the number of nodes within any circular area of radius 1, then m_0 is a Poisson random variable with probability distribution function

$$P_{m_0}(k,\lambda) = e^{-\lambda}\frac{\lambda^k}{k!},$$

where $P_{m_0}(k,\lambda)$ is the probability that $m_0 = k$. Then the number of nodes m_t in any circular area of radius R_t is a Poisson random variable with probability distribution function

$$P_{m_t}(k,\lambda,R_t) = e^{-R_t^2\lambda}\frac{(R_t^2\lambda)^k}{k!},$$

where $P_{m_t}(k,\lambda,R_t)$ is the probability that $m_t = k$. Thus, the probability α that there exists no node in an area of radius R_t is $e^{-R_t^2\lambda}$.

Therefore, if there are n cells in the ideal virtual structure, the expected number of non-ideal cells G_e after configuration is

$$\sum_{k=0}^{n} k\cdot\binom{n}{k}\cdot(\alpha)^k\cdot(1-\alpha)^{n-k} = n\alpha$$

and the expected ratio of non-ideal cells in the system is G_e/n which is α. Moreover, the expected diameter of an R_t-gap perturbed region is

$$2R\sum_{k=0}^{\infty} k\cdot\alpha^k = \frac{2\alpha}{(1-\alpha)^2}R.$$

For example, in a system of radius 1000, if $R = 100$ and $\lambda = 10$, the expected ratio of non-ideal cells as a function of R_t/R is shown in Fig. 7. The ratio is small and converges to 0 quickly as R_t/R increases. The expected diameter of an R_t-gap perturbed region is shown in Fig. 8. The expected diameter is also small and converges to 0 quickly as R_t/R increases. From Figs. 7 and 8, we see that both the expected ratio of non-ideal cells and the expected diameter of an R_t-gap perturbed region are proximately 0 once R_t/R is greater than or equal to 0.02.

4.3.5. Stability and scalability

4.3.5.1. Stable cell structure. In the presence of dynamic nodes, the cell structure is stable in the following senses: (1) In case of *node join*, the cell structure remains unchanged except for the possibility that the head of some cell is replaced by a new node if the new node better serves as head; (2) *Node leave* within a cell is masked within the cell by head shift such that the rest of the structure

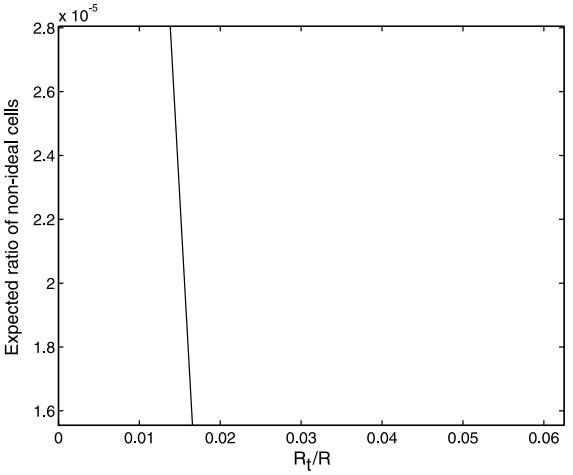


Fig. 7. The expected ratio of non-ideal cells, when $\lambda = 10$.

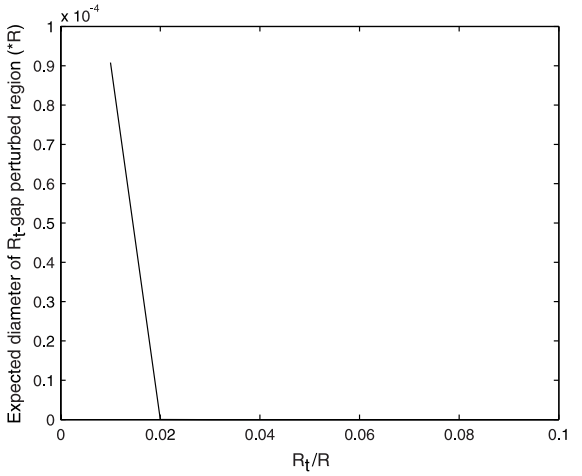


Fig. 8. The expected diameter of an R_t-gap perturbed region, when $\lambda = 10$.

remains unchanged; (3) In case of *node death* such that candidate sets of many cells die, independent cell shift at each cell enables the head level structure to slide as a whole but maintain consistent relative location among cells and heads, which lengthens the lifetime of the structure by a factor of $\Omega(n_c)$, where n_c is the number of nodes in a cell; (4) In case *intra-cell maintenance fail*s, inter-cell maintenance enables a system to stabilize to its stable state within a one-way message diffusing time across the perturbed area; (5) In case of *state corruption*, sanity checking ensures that the erroneous state is corrected by checking the hexagonal properties among heads.

4.3.5.2. Scalable self-healing. The self-healing of the head graph and hexagonal structure is scalable in three senses: (1) *local self-healing* enables the system to stabilize from a perturbed state to its stable state in a one-way message diffusing time across the perturbed area through local coordination among nodes within $\sqrt{3}R + 2R_t$ distance from one another; (2) *local knowledge* enables each node to maintain the identities of only a constant number of nodes within $\sqrt{3}R + 2R_t$ distance, irrespective of network size; (3) the head graph and hexagonal structure can *tolerate multiple simultaneous perturbations* due to the locality property of GS^3-D.

5. Mobile dynamic network

5.1. Concepts

Recall that, in mobile dynamic networks, not only can nodes be dynamic, they can also move. The probability of movement is inversely related to the distance of movement. In this section, we extend GS^3-D to GS^3-M to deal with node mobility.

Conceptually, node mobility is modeled as a correlated node join (at the new location) and leave (from the old location). GS^3-D is easily adapted to deal with the mobility of small nodes (more detailed description is given in Appendix A.2). Thus, we focus on how to deal with big node movements.

In mobile dynamic networks, the head graph needs to be maintained such that, in spite of the movement of the big node H_0, it is connected and the path between H_0 and every head is of minimum distance. To achieve this, the closest head to H_0 in the network acts as the *proxy* for H_0 during the time when H_0 is not a head, and the distance from the proxy to H_0 is set as 0. Then, just by algorithm GS^3-D, the head graph can be maintained as a minimum distance tree to the proxy, and thus every head is of minimum hops to H_0. Moreover, the impact of the movement of H_0 on the head graph is contained within a local range of radius $\sqrt{3}d/2$, where d is the distance that H_0 moves.

5.2. Algorithm

Overview. In mobile dynamic networks, if the big node H_0 moves more than R_t away from the IL of its cell, it retreats from the head role, and transits to status *big_move* where it moves around and maintains a proxy-relationship to its proxy. Whenever H_0 moves within R_t distance to the IL of a cell later, it replaces the existing head of the cell to act as head.

Algorithm modules. Compared with GS^3-D, GS^3-M (described in Fig. 9) has a new module BIG_MOVE, modified modules for big node, intra-cell maintenance, and inter-cell maintenance. (A more detailed description is given in Appendix A.2.)

Program *Big_node*
GS³-D with removed BIG_SLIDE, modified intra-cell as well as inter-cell maintenance modules;
[]
q=*big_move* → BIG_MOVE; //The big node moves; remains status *big_move* or transits to *head*

Program *Small_node*
GS³-D with modified intra-cell as well inter-cell maintenance modules;

Fig. 9. Self-configuration algorithm for dynamic mobile networks (GS³-M).

5.3. Analysis

5.3.1. Invariant and fixpoint

The invariant as well as fixpoint of GS³-D is preserved in GS³-M, except for one more fixpoint predicate F_5 for GS³-M as follows:

F_5 (Proxy optimality): The big node chooses the closest neighboring head as its proxy.

Theorem 9. *Let MI be DI (invariant of GS³-D), then MI is an invariant of algorithm GS³-M.*

Theorem 10. *Let MF be DF (fixpoint of GS³-D)∧F_5, then MF is a fixpoint of algorithm GS³-M.*

5.3.2. Self-stabilization

Theorem 11. *When the big node moves from point A to B on a plane, its impact on the head graph G_h is contained within a circular area centered at point C and of radius $\sqrt{3}d/2$, where C is the midpoint of segment $\overline{AB}$ and d is the cartesian distance between A and B.*

Theorem 12. *Starting at an arbitrary state, every computation of GS³-M reaches a state in MI within time $O(D_c)$, where D_c is the diameter of a contiguous state-corrupted area.*

Theorem 13. *Starting at an arbitrary state in MI, every computation of GS³-M reaches a state in MF within time $O(\max\{D_d/c_1\}, T_d\})$, where D_d is the diameter of the network, c_1 is the average speed of message diffusing, and T_d is the maximum difference between the lifetime of the candidate sets of two neighboring cells.*

Theorems 12 and 13 imply

Corollary 7. *Starting at an arbitrary state, every computation of GS³-M reaches a state in MF within time $O(\max\{D_d/c_1, T_d\})$.*

5.3.3. System stability

In mobile dynamic networks, node mobility is dealt as a special kind of node dynamics. So the stability property of the head level structure and head graph in dynamic networks is preserved in mobile dynamic networks. The invariant and fixpoint of GS³-M only depend on local coordination, which enables them to tolerate a high degree of node mobility because local coordination converges fast.

6. Related work

In [10], a distributed algorithm LEACH is proposed for clustering in wireless sensor networks. But, as mentioned by its authors, LEACH guarantees neither the placement nor the number of clusters in a system, and perturbations are dealt with by globally repeating the clustering operation, which is not scalable. In [3], another algorithm for clustering is designed, but it only considers logical radius of clusters instead of their geographic radius, which can reduce wireless transmission efficiency because of large geographical overlap between clusters [8]. The radius dif-

ference among clusters can be large too. Moreover, its healing procedure is not local, because the healing there depends on multiple rounds of message diffusing across the whole system, instead of a one-way diffusing just within perturbed areas as in our algorithm. And, given a certain density of node distribution, the geographic radius ensured by our algorithm implicitly guarantees a bound on the logical radius of clusters, but not vice versa. In [12], an access-based clustering algorithm is presented that focuses on the stability of clusters, but the algorithm does not consider the size of clusters and it requires GPS at every node.

In [16], a cellular hexagonal structure is described for cellular networks, but it is pre-configured and there is no ability of self-healing. In [6,18], different algorithms for topology control in networks are developed, but they are either centralized or semi-centralized, and thus are not scalable.

In [15,17,20], algorithms for topology control in wireless sensor networks for energy saving are developed. In [22], adaptive fidelity control and routing algorithms are developed for wireless sensor networks. Our self-configuration algorithm provides a stable network infrastructure for tasks such as routing or power control, and thus is orthogonal to these works.

In [14], self-stabilizing algorithms are proposed that mend faults locally in time, but they are not local in space. [1] proposes self-stabilizing algorithms for tree maintenance that is local in space but not local in time. The self-stabilization in GS^3 is local both in time and in space.

7. Conclusion

In this paper, we have presented an algorithm (GS^3) for self-configuring a network into cells of tightly bounded geographic radius and low overlap between cells. GS^3 enables network nodes to organize themselves into a cellular hexagonal structure with a set of proved properties. The structure configured by GS^3 is self-healing, thus GS^3 is applicable to both static networks and networks with dynamic as well as mobile nodes. Moreover, the self-healing is local, which makes GS^3 applicable to networks with a high degree of dynamics and mobility. GS^3 is also scalable due to its properties of local knowledge, local self-healing, and local coordination. GS^3 yields a stable structure even in the presence of dynamic and mobile nodes, which enables a more stable as well as available infrastructure for other network services such as routing, power control, and QoS.

GS^3 is readily extended to the following cases: (1) in a mobile dynamic network where there are multiple big nodes, GS^3 enables each small node to choose the best (e.g., closest) big node to communicate, by letting each small node maintain the current big node it chooses. (2) Due to its locality property, GS^3 is also applicable to the case where nodes are not deployed on an exact 2D plane, but nodes within each neighborhood (e.g., a circular area of radius R) are locally planar. (3) GS^3 is also applicable to the case where the ideal cell radius R is larger than the maximum transmission range of small nodes, because R does not affect the correctness of the algorithm.

In the paper, we have discussed local self-healing in GS^3, but we have not studied in detail how to deal with different degrees of node dynamics and mobility. This is a subject of future work. Moreover, the tightness of the bound on cluster radius in GS^3 reflects the density of node distribution in a network, and we plan to study how to incorporate other properties such as network traffic characteristics in the bound.

GS^3 takes advantage of such model properties of wireless sensor networks as dense node distribution, relative location information among nodes, and the existence of big nodes to solve the problem of scalable self-configuration and self-healing. We believe these model properties can be exploited in a richer class of problems in wireless sensor networks and deserve further exploration.

Appendix A

In the appendix, we present the complexity and convergence properties of GS^3, detailed description of modules in GS^3-S, GS^3-D and GS^3-M, and the invariant as well as fixpoint of GS^3-D.

A.1. Complexity and convergence properties of GS³

Information maintained at each node	$\theta(\log n)$
Factor of *lengthened lifetime* of head level structure by intra-cell & inter-cell maintenance	$\Omega(n_c)$
Convergence time under *perturbations*	$O(D_p)$
Convergence time to the stable state in *static networks*	$\theta(D_b)$
Convergence time from an arbitrary state to the stable state in *dynamic/mobile networks*	$O(D_d)$

n, the number of nodes in a system; n_c, the number of nodes in a cell; D_p, the diameter of a contiguous perturbed area; D_b, $\max\{\text{dist}(H_0, i)$: i is a small node, and $\text{dist}(H_0, i)$ is the cartesian distance between the big node H_0 and $i\}$; D_d, the diameter of the system, i.e., $\max\{\text{dist}(i, j)$: i and j are small nodes, and $\text{dist}(i, j)$ is the cartesian distance between i and $j\}$.

A.2. Description of modules in GS³-S, GS³-D and GS³-M

In this subsection, we give more detailed description of some algorithm modules in GS³-S, GS³-D and GS³-M as follows. The complete program is presented in [23].

A.2.1. Algorithm GS³-S

(a) HEAD_ORG (LD, RD, R, R_t): There are four arguments to HEAD_ORG: (1) L direction (LD) and R direction (RD) with respect to direction $\overrightarrow{P(i), i}$ (see Fig. 3). LD and RD determine the search region of a head in the process of organizing its neighboring cell heads. (2) ideal radius R and radius tolerance R_t.

The function of HEAD_ORG executed by a head i is for head i to organize the neighboring cell heads in its search region. HEAD_ORG executed by head i works as follows: first, head i reserves wireless channel and broadcasts message *org* within $\sqrt{3}R + 2R_t$ distance; second, head i listens to replies (message *org_reply* or *head_org_reply*) from nodes no more than $\sqrt{3}R + 2R_t$ away and within (LD, RD) search region for certain amount of time and calculates the set of small nodes and head nodes (*SmallNodes* and *ExistingHeads* respectively) in the search region; Third, using the low level module HEAD_SELECT (see Fig. 3), head i selects neighboring cell heads *HeadSet*; fourth, head i broadcasts message $\langle HeadSet\rangle$ to nodes within $\sqrt{3}R + 2R_t$ distance, revokes channel reservation, and transits to status *work*.

In HEAD_SELECT executed by head i, head i needs to select neighboring cell heads in its search region. It achieves this in two steps: first, it calculates the ideal locations for those possible neighboring cell heads; second, for each possible neighboring cell, if there is any small node that is in the R_t-radius circular area centered by the ideal location of the cell, select the highest ranked such node as the cell head. The algorithm is described in Fig. 3 and its time complexity is $\theta(|SmallNodes|)$.

(b) HEAD_ORG_RESP: When a head node i (at status *head* or *work*, and not including the big node) receives a message *org* from a head j, it replies with a message *head_org_reply*, and waits until head j's HEAD_ORG process finishes (by overhearing its message $\langle HeadSet\rangle$). No status transition in this module.

(c) ASSOCIATE_ORG_RESP: When a small node i is at status *bootup* or *associate*, it will execute ASSOCIATE_ORG_RESP process upon receiving a message *org* from a head j. If node i is at status *bootup* or status *associate* but head j is better (such as closer, with higher remaining energy) than its current head $H(i)$, node i replies a message *org_reply* to head j. Then waits for head j's message $\langle HeadSet\rangle$ If node i is selected as a cell head, it sets head j as its parent head, and transits to status *head*; otherwise, node i sets head j as its head, and transits to status *associate*. On the other hand, if node i fails to hear the message $\langle HeadSet\rangle$ from head j after a certain amount of time, it transits back to its status at the beginning of the process (i.e., *bootup* or *associate*).

A.2.2. Algorithm GS³-D

A.2.2.1. Intra-cell maintenance

(a) HEAD_INTRA_CELL: In HEAD_INTRA_CELL executed by a head i, head i executes the following actions:

(i) It periodically broadcasts message *head_intra_alive* within its cell, and updates its candidate

as well as associate set according to replies from the associates in its cell.

(ii) If head i receives a message *associate_alive* or *associate_retreat* from an associate, it needs to update candidate as well as associate set properly.

(iii) If i is resource scarce or a candidate better serves as head, i broadcast a message *head_retreat* within its cell and retreats back to be an associate.

(iv) If i receives message replacing_head from the big node H_0 or a head candidate j, it retreats to be an associate, and sets H_0 or j as its head.

(v) If the candidate set of its cell is weak, i calls STRENGTHEN_CELL to strengthen it.

(vi) If the distance IL of its cell that of all its neighboring cells deviates too much from $\sqrt{3}R$, exceeding certain threshold T_d, it abandons the cell by broadcasting a message *cell_abandoned* within its cell and transiting to status *bootup*.

In STRENGTHEN_CELL, head i first finds the next ideal location (IL) of its cell whose corresponding candidate set is not empty, according to the cell's current $\langle ICC, ICP\rangle$ value and the ordering of all ILs in its cell (see Fig. 5). Then it calculates the new candidate set with respect to the new IL. Last, it broadcasts two messages (*head_intra_alive* containing the new candidate set, and *head_retreat*) within its cell, and retreats to be an associate. Time complexity is $O(n_c)$, where n_c is the number of nodes in a cell.

(b) CANDIDATE_INTRA_CELL: In CANDIDATE_INTRA_CELL executed by a candidate i, i executes the following actions:

(i) Upon receiving a message *head_intra_alive* from a head j: if j is its head, i checks whether it is still in j's candidate set, and transits to status *associate* if not; otherwise, replies a *head_intra_ack* message. If j is not its head and is better than its current head, i sends an *associate_retreat* message to its current head and *associate_alive* message to head j.

(ii) If i receives a message *head_retreat* from or detects the failure of its current head, it coordinates with other candidates in this cell to elect the highest ranked candidate as the new head. The head candidates in a cell are ranked in the same way as that in HEAD_SELECT (see Section 3).

(iii) If i receives a message *cell_abandoned*, *head_retreat_corrupted*, *head_disconnected*, or *syn_cell* from its head, it transits back to boot up status.

A.2.2.2. Inter-cell maintenance

(a) HEAD_INTER_CELL: In HEAD_INTER_CELL executed by a head i, head i executes the following actions:

(i) Periodically broadcasts message *head_inter_alive* as heartbeat to its parent as well children heads.

(ii) Upon receiving a message *head_inter_alive* from head j, update children set, and neighboring head set properly. If j is not i's parent head but is better (closer to the big node, for example) than its current parent head, i sets j as parent head, and sends a message *new_child_head* to j.

(iii) If i receives a message *new_child_head* from j, update children heads set as well neighboring heads set accordingly.

(iv) If a neighboring cell C_n (including child as well as parent cell) has a new head due to intra-cell maintenance, i updates neighboring head set, children head set, or parent head accordingly. If C_n has a newer $\langle ICC, ICP\rangle$ value, head i synchronizes its cell to the new $\langle ICC, ICP\rangle$ by calling SYN_CELL process (this is optional).

(v) If i receives a *syn_cell* message from a neighboring cell's head j, it updates (remove j) neighboring head and child head sets accordingly. If j is i's parent head, i executes PARENT_SEEK to find a new parent head. If *syn_cell* message carries a newer $\langle ICC, ICP\rangle$ value, i executes SYN_CELL.

(vi) If i is a boundary head and there is no head at certain neighboring cell area in its search region, it periodically executes HEAD_ORG to check whether new nodes have shown up in this direction.

(vii) If a child head j fails, i executes HEAD_ORG in j's direction, trying to organize a new head.
(viii) If i's parent head $P(i)$ fails, and $P(i)$'s failure has not been recovered by $P(i)$'s parent head, i executes to PARENT_SEEK. If i receives a message *parent_seek* from a head j and they don't have the same parent head, it replies a *parent_seek_ack* message.
(ix) If i receives a message *sanity_check_req* from a neighboring head j, it checks its own status. If its status is valid, i replies a message sanity_check_valid message to j; otherwise, i executes SANITY_CHECK.
(x) If i receives a *head_retreat_corrupted* message from a neighboring cell's head j, it updates (remove j) its neighboring head set and children head sets accordingly. If j is i's parent head, i executes PARENT_SEEK.

In SYN_CELL, head i first calculates the new IL with respect to the new $\langle ICC, ICP\rangle$ value. Then it calculates the candidate set corresponding to this IL. If the candidate set is not empty, i broadcasts a message *head_retreat* within its cell; otherwise, it broadcasts a message *syn_cell* to its neighboring heads that includes the current $\langle ICC, ICP\rangle$ value. Last, i transits to status *big_ slide* if it is the big node or status *associate* otherwise. Time complexity is $O(C)$, where C is a constant.

In PARENT_SEEK, let ST denote the sub-tree of G_h rooted at head i. Head i ranks its neighboring heads in almost the same way as that in HEAD_SELECT, except that $i, P(i)$ instead of $\overrightarrow{GR}$ is used as reference direction. Then i tries to find a neighboring head as parent head in an increasing order. If it succeeds in finding such a head j, i sets j as its parent; otherwise i lets its children heads on the boundary of ST's geographic coverage try to find a new parent head in the same way. If any of its child head j succeeds, i sets j as its parent; otherwise i broadcasts a message *head_disconnected* within its cell, and transits back to boot up status. Its time complexity is $O(|FNH|)$, where *FNH* denotes the set of head in (G_h-ST) that has a neighboring head in ST.

(b) ASSOCIATE_INTER_CELL: If an associate (including both candidate and non-candidate) receives a message *org*, it calls ASSOCIATE_ORG_RESP.

A.2.2.3. Sanity checking. In order to deal with status corruption, every head periodically executes SANITY_CHECK. In SANITY_CHECK executed by head i, it first checks if its $\langle ICC, ICP\rangle$ value is equal to that of all its neighboring cells. If yes, it checks whether its status satisfies the hexagonal relationship of the system invariant. If no, it broadcasts a message *sanity_check_req*, and waits for replies from its neighboring cells' heads. If all its neighboring cells' heads reply a message *sanity_check_valid*, head i broadcasts a message *head_retreat_corrupted* within its cell. If it has not got the message *sanity_ check_valid* from any of its neighboring cells after certain amount of time, head i exits this module without changing its status. Time complexity is $\theta(A)$, where A denotes the size of the contiguously affected area.

A.2.3. Algorithm GS³-M

A.2.3.1. BIG_MOVE. In BIG_MOVE, the big node keeps listening to heartbeats (*head_intra_alive* message) from all nearby heads, and always chooses the best (closest, for example) head as its proxy. When its proxy is replaced by a candidate h_n in the proxy's cell, the big node reset its proxy as h_n. When the big node moves into the R_t-radius circular area of a cell, it replaces the existing head as head, and transits back from status *big_move* to status *work*.

A.2.3.2. Modified intra-cell and inter-cell maintenance. The modification to the intra-cell as well as inter-cell maintenance is to maintain the cell head, candidate set, and big node's proxy relationship in the presence of mobile nodes. As for big node, if it retreats from the head role because of the IL change of any of its neighboring cells, it transits to status *big_move* instead of *big_slide* in dynamic mobile networks.

A.3. Invariant and fixpoint of GS³-D in dynamic networks

Notation

- *Visible node*: a node that is connected to the big node H_0 in V_p.
- *Neighboring_heads*(i) : $\{j : j$ is a head $\wedge$ (head i and j's geographic coverage adjoins)$\}$.
- $\mathrm{Dist}(i,j)$: cartesian distance between nodes i and j.
- $H(i)$: the head of the cell that the associate node i is in.

A.3.1. Invariant

The invariant of GS³-D differs from that of GS³-S at I_2 when a cell and its neighboring cells have different $\langle ICC, ICP\rangle$ values.

- I_1 (*connectivity*)
  Same as in static networks.
- I_2 (*Hexagonal structure*)
  - $I_{2.1}$: (for inner heads)
    $I_{2.1}$ *for static networks* $\wedge$
    ($\forall$ inner_head $i : \forall\, j$
    $\in$ neighboring_heads(i):
    $\langle \mathrm{ICC}(i), \mathrm{ICP}(i)\rangle \neq \langle \mathrm{ICC}(j), \mathrm{ICP}(j)\rangle \Rightarrow$
    $((\mathrm{dist}(\mathrm{IL}(i), \mathrm{IL}(j)) - 2R_t$
    $\leqslant \mathrm{dist}(i,j) \leqslant \mathrm{dist}(\mathrm{IL}(i), \mathrm{IL}(j)) +$
    $2R_t) \wedge (0 < \mathrm{dist}(\mathrm{IL}(i), \mathrm{IL}(j))2\sqrt{3}R))$
)
  - $I_{2.2}$: (for boundary heads)
    $I_{2.2}$ *for static networks* $\wedge$
    ($\forall$ boundary_head
    $i : \forall j \in$ neighboring_heads(i):
    $\langle \mathrm{ICC}(i), \mathrm{ICP}(i)\rangle \neq \langle \mathrm{ICC}(j), \mathrm{ICP}(j)\rangle$
    $\Rightarrow ((\mathrm{dist}(\mathrm{IL}(i), \mathrm{IL}(j)) - 2R_t \leqslant$
    $\mathrm{dist}(i,j) \leqslant \mathrm{dist}(\mathrm{IL}(i), \mathrm{IL}(j)) +$
    $2R_t) \wedge (0 < \mathrm{dist}(\mathrm{IL}(i), \mathrm{IL}(j)) \leqslant 2\sqrt{3}R))$
)
  - $I_{2.3}$: modify $I_{2.3}$ *for static networks* by changing
    ($\forall$head $i : |CH(i)| \leqslant 3$) to
    ($\forall$head $i : |CH(i)| \leqslant 5$)
  - $I_{2.4}$: (cell radius)
    $I_{2.2}$ *for static networks* $\wedge$
    ($\forall$ inner cell $C : (\exists j \in$
    neighboring_heads$(i) : \langle \mathrm{ICC}(i), \mathrm{ICP}(i)\rangle \neq$
    $\langle \mathrm{ICC}(j), \mathrm{ICP}(j)\rangle) \Rightarrow (\forall$ associate i
    $\in C : \mathrm{dist}(i, H(i)) < 2R + R_t)) \wedge$
    ($\forall$ boundary cell C' : associate $i \in C'$:
    $\mathrm{dist}(i, H(i)) \leqslant \sqrt{3}R + 2R_t + d_p$)
- I_3 (*Inner cell optimality*)
  Same as in static networks.

A.3.2. Fix point

The fixpoint of GS³-D differs from that of GS³-S at $F_{1.2}$ that is strengthened in GS³-D.

- F_1 (*connectivity*)
  - $F_{1.1}$: Same as in static networks.
  - $F_{1.2}$: G_h is a minimum-distance (with respect to the big node H_0) spanning tree of G_{hn}, and G_h is rooted at H_0.
    $F_{1.2}$ *for static networks* $\wedge \forall_i \in$
    $(V_h - \{H_0\}) : \mathrm{hops}(H0, v_i) =$
    $MIN(H_0, v_i)$,
    where $MIN(v_1, v_2)$ is the length (by hops) of the shortest path between v_1 and v_2 in G_{hn}.
- F_2 (*hexagonal structure*)
  - $F_{2.1}$, $F_{2.2}$, and $F_{2.3}$ are the same as in static networks.
  - $F_{2.4}$ is relaxed as: ($F_{2.4}$ of GS³-S)$\wedge$($|R_{random}|$ is at most $((\sqrt{3}-1)R + 2R_t + d_p)$ for boundary cells).
- F_3 (*cell optimality*): Same as in static networks.
- F_4 (*coverage*): Same as in static networks.

References

[1] Anish Arora, Mohamed Gouda, Distributed reset, IEEE Transactions on Computers 43 (9) (1994) 1026–1038.

[2] Anish Arora, Hongwei Zhang, LSRP: Local stabilization in shortest path routing, IEEE-IFIP DSN, 2003.

[3] Suman Banerjee, Samir Khuller, A clustering scheme for hierarchical control in multi-hop wireless networks, IEEE INFOCOM 2001, pp. 1028–1037.

[4] Alberto Cerpa, Deborah Estrin, ASCENT: adaptive self-configuring sensor networks topologies, IEEE INFOCOM, 2002.

[5] Computer Science and Telecommunications Board (CSTB), Embedded Everywhere: A Research Agenda for Networked Systems of Embedded Computers, National Academy Press, Washington, DC, 2001.

[6] Shlomi Dolev, Evangelos Kranakis, Danny Krizanc, David Peleg, Bubbles: adaptive routing scheme for high-speed dynamic networks, SIAM Journal on Computing 29 (3) (1999) 804–833.

[7] Deborah Estrin, Ramesh Govindan, John Heidemann, Satish Kumar, Next century challenges: scalable coordination in sensor networks. ACM MobiCom, 1999, pp. 263–270.
[8] Deepak Ganesan, David Culler, Deborah Estrin, et al., An empirical study of epidemic algorithms in large scale multihop wireless networks, IRP-TR-02-003, 2002.
[9] Mohamed Gouda, Elements of Network Protocol Design, Wiley, New York, 1998.
[10] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, An application-specific protocol architecture for wireless microsensor networks, IEEE Transactions on Wireless Networking 1 (4) (2002) 660–670.
[11] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister, System architecture directions for networked sensors, ASPLOS, 2000, pp. 93–104.
[12] Ting-chao Hou, Tzu-Jane Tsai, An access-based clustering protocol for multihop wireless ad hoc networks, IEEE Journal on Selected Areas in Communications 10 (7) (2001) 1201–1210.
[13] Mahesh Jayaram, George Varghese, Crash failures can drive Protocols to Arbitrary States, ACM Principles of Distributed Computing, 1996, pp. 247–256.
[14] Shay Kutten, David Peleg, Fault-local distributed mending, Journal of Algorithms 30 (1) (1999) 144–165.
[15] Li Li, Joseph Y. Halpern, Paramvir Bahl, Yi-Min Wang, Roger Wattenhofer, Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks, ACM Principles of Distributed Computing, 2001, pp. 264–273.
[16] V.H. Mac Donald, Advanced mobile phone service: the cellular concept, The Bell System Technical Journal (1979).
[17] Volkan Rodoplu, Teresa H. Meng, Minimum energy mobile wireless networks, IEEE Journal on Selected Areas in Communications 17 (8) (1999) 1333–1344.
[18] Theodoros Salonidis, Pravin Bhagwat, Leandros Tassiulas, Richard LaMaire, Distributed topology construction of Bluetooth personal area networks, IEEE INFOCOM, 2001, pp. 1577–1586.
[19] S.R. Saunders, Antennas and Propagation for Wireless Communication Systems, Wiley, Chichester, UK, 1999.
[20] Roger Wattenhofer, Li Li, Paramvir Bahl, Yi-Min Wang, Distributed topology control for power efficient operation in multihop wireless ad hoc networks, IEEE INFOCOM, 2001, pp. 1387–1388.
[21] Alec Woo, David E. Culler, A transmission control scheme for media access in sensor networks, ACM Mobicom, 2001, pp. 135–221.
[22] Ya Xu, John Heidemann, Deborah Estrin, Geography-informed energy conservation for ad hoc routing, ACM Mobicom, 2001, pp. 70–84.
[23] Hongwei Zhang, Anish Arora, GS^3: scalable self-configuration and self-healing in wireless networks, OSU-CISRC-4/02-TR08, ftp://ftp.cis.ohio-state.edu/pub/tech-report/2002/TR08.pdf, The Ohio State University, April 2002.
[24] Jerry Zhao, Ramesh Govindan, Deborah Estrin, Residual energy scans for monitoring wireless sensor networks, USC-CSD-TR-01-745, May 2001.

Hongwei Zhang is a Ph.D. student in the Department of Computer and Information Science at The Ohio State University, USA. His research interest lies in computer networking, distributed computing, and fault tolerance. Especially, he is interested in scalable self-configuration, dependability, and stability in such large scale dynamic systems as the Internet, wireless sensor networks, and mobile ad-hoc networks. He received the B.E. and M.S. degrees in Computer Science from Chongqing University, China in 1997 and 2000 respectively. (URL: http://www.cis.ohio-state.edu/~zhangho)

Anish Arora is a Professor of Computer Science at the Ohio State University. His research is on fault tolerance, security, and timelines properties of systems, especially distributed and networked systems of large scale. Recent case studies in his research have centered on sensor networking and home networking, with support from DARPA, NSF, and Microsoft Research. He is a leading expert in self-stabilization, and has chaired or co-chaired seminars and symposia in this area in 1998, 1999, 2000, and 2002. He is program co-chair of the 25th International Conference on Distributed Computer Systems. Arora received the B. Tech. Degree from the Indian Institute of Technology at New Delhi and the Master's and Ph.D. degrees from the University of Texas at Austin, all in Computer Science. From 1989 to 1992, he worked at the Microelectronics and Computer Technology Corporation (MCC) in Austin, TX. (URL: http://www.cis.ohio-state.edu/~anish)