GS^3: Scalable Self-configuration and Self-healing in Wireless Networks*

Hongwei Zhang
Department of Computer and Information Science
The Ohio State University
2015 Neil Avenue, DL 395
Columbus, Ohio 43210 USA
+1-614-292-1932
zhangho@cis.ohio-state.edu

Anish Arora
Department of Computer and Information Science
The Ohio State University
2015 Neil Avenue, DL 395
Columbus, Ohio 43210 USA
+1-614-292-1836
anish@cis.ohio-state.edu

ABSTRACT
We present GS^3, a distributed, scalable, self-configuration and self-healing algorithm for multi-hop wireless networks. The algorithm enables network nodes in a 2D plane to configure themselves into a cellular hexagonal structure such that cells have tightly bounded geographic radius and low overlap between neighboring cells. The structure is self-healing under various perturbations, such as node joins, leaves, deaths, movements, and state corruptions. For instance, it 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 healing guarantees that all perturbations are contained within a tightly bounded region with respect to the perturbed area and dealt with in a one-way message diffusion time across the region; third, only local coordination is needed in both configuration and self-healing.

Keywords
Multi-hop wireless networks, self-configuration, geography-aware, cellular hexagons, self-healing, self-stabilization, locality, scalability, dynamics, mobility

1. INTRODUCTION
As increasingly small network nodes are becoming available, many "sense-compute-actuate" networks are being realized. Several of these networks use unattended wireless nodes [1,2,4], which communicate with one another via intermediate node relays due to limited transmission range or energy [7,8]. The number of nodes is potentially large (thousands and millions of nodes are considered in earthquake relief and unmanned space vehicle scenarios, for instance) [1]. Thus, scalability is a key issue for large-scale multi-hop wireless 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 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 some global objective.

Most previous work on clustering [3,12] treats a network as a geography-unaware graph. The clustering criteria adopted are, for instance, the number of nodes in a cluster and the cluster size. These criteria do not take the geographic radius of clusters (simply called *radius*, henceforth) into account, which we argue is desirable in wireless networks, especially in large-scale, resource constrained multi-hop networks: 1) many multi-hop wireless network applications, such as environment monitoring and temperature sensing, are inherently geography-aware and so reflecting geography in the underlying structure enables optimization of system performance. 2) Cluster radius affects energy dissipated for intra cluster coordination and thus the lifetime of a network. 3) Cluster radius affects the efficiency of local coordination functions such as data aggregation and load balancing. 4) Cluster radius affects the quality of communication over a shared wireless transmission medium; also, the larger the cluster radius, the less the frequency reuse. 5) Cluster radius affects the scalability and availability of a network, since it affects the number of clusters and the number of nodes in each cluster (the more the nodes in a cluster, the more available the cluster is).

Moreover, given that expected multi-hop wireless networks are of large scale, they are subject to node failure, node join and leave, mobility, and state corruption, and they usually cannot be managed manually [5], self-configuration and self-healing is necessary in multi-hop wireless networks.

Contributions of the paper In this paper, we present a distributed algorithm (GS^3) for configuring a wireless locally 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

* This work was partially sponsored by DARPA grant OSU-RF-01-C-1901, NSF grant CCR-9972368, and an Ohio State University Fellowship.

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 *head graph*, that is rooted at a "big" node, which is the interface between the wireless network and external networks such as Internet.

Our algorithm yields a self-healing system. The head graph and cellular hexagon structure are self-healing in the presence of various perturbations, such as one or more node joins, leaves, deaths, movements, and state corruptions. More specifically, the self-healing is such that the head graph and the cellular hexagon structure remain stable in the following senses: 1) unanticipated node leaves within a cell are masked by the cell; 2) in case several cells experience node deaths at about the same time (due to energy exhaustion), an independent shift of each cell enables the head graph as well as the cellular hexagon 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, an originally dynamic or mobile system is turned into a stable infrastructure for other network services such as routing. The self-healing capability and the modular design of algorithm GS^3 enable different modules to be integrated so as to cater to different scenarios, in static as well as dynamic networks, immobile as well as mobile networks, and networks with just one big node or multiple big nodes.

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 the impact is confined to) a tightly bounded region around the perturbed area; the structure self-stabilizes within the time to diffuse an 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 A1.)

The rest of the paper is organized as follows. In Section 2, we present the system model and problem statement. We then develop algorithm for static networks, dynamic networks, and dynamic mobile networks in Section 3, 4, and 5 respectively. We discuss related work in Section 6. Section 7 concludes the paper and makes further comments on system model. For reasons of space, we relegate the detailed description of algorithm modules, and proofs for theorems of the paper to [14].

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 on a 2D plane, each having a certain wireless transmission range.

> **Node distribution assumption** 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.

There are two kinds of nodes: big and small. Intuitively, the big node acts as the initiator as well as the access point for small nodes. 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 Internet. For convenience, we assume that the system has one big node, and all other nodes are small (in Section 7, we discuss the case of multiple big nodes).

Many wireless networks have some central control points that control system wide operations. E.g., sensor networks are used to sample the environment for sensory information (e.g. temperature) and propagate this data to a central point [6]. Also, in disaster recovery or battlefield scenarios, there is usually a commander for a group of rescue workers or soldiers that is the central point.

> **Wireless transmission assumption** Nodes can adjust transmission range, and detect location relative to other nodes. Destination-aware message transmission is reliable, but destination-unaware message transmission (such as broadcast) may be unreliable.

A network node can detect the strength of a received signal, and calculate the distance from its communicating peer [15]. 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 always be made reliable through mechanisms like acknowledgement and retransmission.

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 assumption** 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 consumption). The probability for a node to move distance d is proportional to $1/d$.

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 dynamic nodes and mobile 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 various 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 $R \geq R_t$, constructs 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 bounded value with respect to R, and is a function of R_t.
b) Each node is in at most one cell.
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).
d) 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.

Motivation *a)* 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, but the difference between the actual radius and R still need to be small enough, and is a function of R_t. *b)* 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. *c)* If a node is able (unable) to communicate with the big node before configuration, it should still be able (unable) to do so after it. *d)* In large-scale wireless networks, automatic recovery is important. Moreover, even simple perturbations like node crashes can drive a network protocol into arbitrary state [17]. Thus self-healing from arbitrary states is necessary to deal with complex perturbations resulting from dynamic and mobile behavior of system elements. This goal is achieved by the technique of self-stabilization.

3. STATIC NETWORK

3.1 Concepts

Recall that in static networks, nodes are neither dynamic nor mobile. So we solve our problem without considering perturbations (i.e., requirement *d)* 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 hexagon structure as shown in Figure 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. 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.

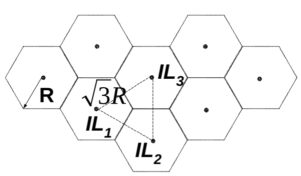


Figure 1: Cellular hexagon structure

Of course, in reality, node distribution is not continuous, 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 Figure 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 [10].

Our solution is achieved in three steps. First, we cover a system with a hexagonal virtual structure as in Figure 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 pc of C as a head, and pc is called the Ideal Location of k, IL(k); Third, for every non-head small node j covered by a cell C, we let j be an associate and chooses the best (e.g. the closest in a clockwise sense) head as its head, H(j); Thus, 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 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. Along with the diffusing computation, a unique Global Reference ($\overline{GR}$) direction is diffused across the network.

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

In most cases, a *(d+1)*-band cell head is selected by a *d*-band head (d $\geq$ 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 $\geq$ 1). But this does not affect the correctness of GS^3-S, and it is dealt implicitly in algorithm in Section 4. For simplicity, we do not discuss this case any further.

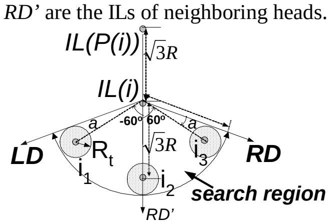
Algorithm modules The algorithm (GS^3-S) consists of two programs (described in Figure 2): *Big_node* at H_0 and *Small_node* at all the small nodes. Underlying these two programs are modules used for head organization: HEAD_ORG, used to organize heads, and HEAD_ORG_RESP as well as ASSOCIATE_ORG_RESP, used to respond to a HEAD_ORG.

Program *Big_node*
var q: {*bootup, work*}; //node status
/* Big node boots up and organizes the 1-band cells */
q = *bootup* → HEAD_ORG(0^o, 360^o, R, R/4) //transit to status *work*
Program *Small_node*
var q: {*bootup, head, work, associate*}; //node status
/* Small nodes boot up, listen to nearby HEAD_ORG */
q = *bootup* → ASSOCIATE_ORG_RESP //transit to status *head or associate*
[]
/* Heads organize neighboring heads in their search regions */
q = *head* → HEAD_ORG($-60^o-\alpha$, $+60^o+\alpha$, R, R/4) //transit to status *work*: $\alpha = \mathrm{Sin}^{-1}(R_t/\sqrt{3}R)$
[]
q = *work* → HEAD_ORG_RESP
[]
/* Associates respond to HEAD_ORG */
q = *associate* → ASSOCIATE_ORG_RESP //remain status *associate*

Figure 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 $j\times 60^o$ ($\lfloor LD/60 \rfloor \le j \le \lfloor RD/60 \rfloor$) degree from *RD'* are the ILs of neighboring heads.

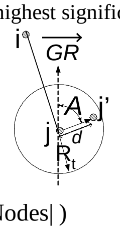


Step 2: Remove the set of IL that is the IL of some existing head from NH. I.e. NH ← (NH – EH), where EH = {j : j ∈ NH ∧ (∃ k ∈ ExistingHeads : (dist(j, k) ≤ R_t))}.

Step 3: For each IL j in NH, let CA(j) = {k : k ∈ SmallNodes ∧ dist(k, j) ≤ R_t}. CA(j) is the set of small nodes within R_t distance from IL 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 IL 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|)

Figure 3: HEAD_SELECT module used in HEAD_ORG

In HEAD_ORG, a head *i* (including H_0) 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 Figure 3), head *i* first calculates the *IL*s for the neighboring cells in its search region; then for each IL *j* that is not the IL of an existing head, *i* selects the best node less than R_t away from *j* as a head.

In HEAD_ORG_RESP, a head sends its state in response to a 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 a 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 if *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.

(A more detailed description of these modules is given in [14])

3.3 Analysis

In this subsection, we discuss the invariant, fixpoint, self-stabilization, and other properties of algorithm GS^3-S (proof for the closure of $I_{2.4}$ in GS^3-S is given in Appendix A2, and the rest proofs are given in [14]).

Notation

Physical network $G_p = (V_p, E_p)$, where V_p = { j: *j* is a node in the system} and E_p = {(i, j): i ∈ V_p ∧ j ∈ V_p ∧ (i and j are within transmission range of each other)}

Head Graph $G_h = (V_h, E_h)$, where V_h={i: i∈V_p ∧ i is a cell head} and E_h ={(i, j): i ∈ V_h, j ∈ CH(i)}

Head level structure: the set of heads in a system and the geographic relation (distance, relative direction) 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

Inner cell: a cell that is not a boundary cell

Visible node: a node that is connected to H_0 in V_p

Neighboring_heads(i): { j: j is a head ∧ (head i and j's geographic coverage adjoins}

Dist(i, j): cartesian distance between nodes i and j

H(i): the head of the cell that the associate node i is in

CH(i): the set of children heads of head i

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 every algorithm module (e.g. HEAD_ORG) as an atomic action. Our invariant **SI** = $I_1 \wedge I_2 \wedge I_3$, where I_j (j = 1, 2, 3) is individually closed under algorithm actions. The predicates are as follows.

I_1 <u>(Connectivity)</u> **=** $I_{1.1} \wedge I_{1.2}$, where

- $I_{1.1}$: Every pair of heads that is connected in G_h is connected in G_p, and vice versa.

  (∀ i, j ∈ V_h: there is a path between i and j in G_h ⇔ there is a path between i and j in G_p)

- $I_{1.2}$: G_h is a tree rooted at the big node H_0.

  $((P(H_0) = H_0) \wedge (hops(H_0) = 0)) \wedge$
  $(\forall i \in (V_h - \{H_0\}): hops(H_0, i) = hops(H_0, P(i))+1) \wedge$

$(\forall i, j \in V_h$: i and j are connected in $G_h) \wedge$
$(\forall i, j \in V_h$: there is a path of length ≥ 2 between i and j $\Rightarrow$
$(P(i) \neq j \wedge P(j) \neq i)$
where hops(i, j) is the path length between i to j in G_h.

<u>I_2 (Hexagonal Structure)</u> $= I_{2.1} \wedge I_{2.2} \wedge I_{2.3} \wedge I_{2.4}$, 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.

  $(\forall$ inner cell head i: (| neighboring_heads(i) | = 6)
  $\wedge(\forall j \in$ neighboring_heads(i): $\sqrt{3}R\text{-}2R_t \leq dist(i, j) \leq \sqrt{3}R+2R_t))$

- $I_{2.2}$: Each boundary cell head has less than 6 neighboring heads, and the distance among them is bounded by [$\sqrt{3}$ R-2R_t, $\sqrt{3}$ R-2R_t].

  $(\forall$ boundary cell head i: | neighboring_heads(i) | < 6)
  $\wedge(\forall j \in$ neighboring_heads(i): $\sqrt{3}R\text{-}2R_t \leq dist(i, j) \leq \sqrt{3}R+2R_t))$

- $I_{2.3}$: Each head, except for 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.

  $(\forall$ head $i \neq H_0$: $| CH(i) | \leq 3) \wedge$
  $(H_0$ is an inner cell head $\Rightarrow$ $(| CH(H_0) | = 6)) \wedge$
  $(H_0$ is a boundary cell head $\Rightarrow$ $(| CH(H_0) | \leq 5))$

- $I_{2.4}$: Each cell is of radius ($R+R_{random}$), where $|R_{random}|$ is at most $(2R_t/\sqrt{3})$. Each associate is no more than ($R+R_{random}$) away from its head.

  $(\forall$ inner cell C: $\forall$associate $i \in C$: $R-(2R_t/\sqrt{3}) \leq dist(i,H(i)) \leq$
  $R+(2R_t/\sqrt{3}))$

<u>I_3 (Inner Cell Optimality)</u>: Each associate in an inner cell belongs to only one cell and chooses the best (e.g. closest) head as its head.

$(\forall$associate i in an inner cell: $\forall$head $j \neq H(i) \Rightarrow H(i)$ <u>better than</u> j)

Theorem 1: SI is an invariant of algorithm GS^3-S.

Theorem 1 and I_2 imply

Corollary 1: The distance among 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 Figure 4) with bounded head location deviation R_t.

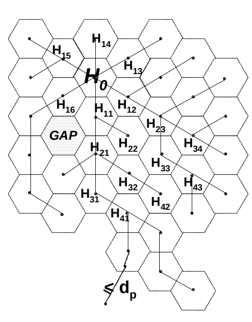


Figure 4: Self-configured cellular hexagon structure

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. Our fixpoint **SF** = $F_1 \wedge F_2 \wedge F_3 \wedge F_4$ as follows.

<u>F_1 (Connectivity)</u> and <u>F_2 (Hexagonal Structure)</u> are the same as I_1 and I_2 respectively.

<u>F_3 (Cell Optimality)</u>: Each associate belongs to only one cell and chooses the best head as its head.

$(\forall$associate i : $\forall$head $j \neq H(i) \Rightarrow H(i)$ <u>better than</u> j)

<u>F_4 (Coverage)</u>: The set of heads and cells covers all the visible nodes in a system.

$(\forall$visible node i : $\exists$head j: $j = H(i)$)

Theorem 2: SF is a fixpoint of algorithm GS^3-S.

Requirement *a)* and *b)* are satisfied by Theorem 1 and 2.

Theorem 2, F_1 and F_4 imply

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

$(\forall$node i : $H(i) \neq NULL \Leftrightarrow$ there is a path between i and H0 in Gp)

Requirement *c)* is satisfied by Corollary 3.

3.3.3 Self-stabilization

Theorem 3: Starting from any state, every computation of GS^3-S reaches a state where SI holds within a constant amount of time.

Theorem 4: Starting from any state where SI holds, every computation of GS^3-S reaches a state where SF holds within time $\theta(D_b)$, where D_b = max{dist(H_0, i): *i* is a small node, and dist(H_0,i) is the cartesian distance between H_0 and i}.

Theorem 3 and 4 imply

Corollary 4: Starting from any state, every computation of GS^3-S reaches a state where SF holds within time $\theta(D_b)$.

Termination of the diffusing computation follows from Corollary 4.

3.3.4 Scalability

The self-configuration algorithm GS^3-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. failure), 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 GS^3-S to GS^3-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 still be dealt with using *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 long enough 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, since energy usually exhausts faster at a head than at an associate. In many envisioned large-scale wireless networks, the traffic load across a network is statistically uniform due to in-network processing such as data aggregation [16], which means statistically uniform energy dissipation across the network. Given the fact that statistically there are multiple nodes in any R_t-radius circular area 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 *IL*s at different cells change (the relative position between IL and IL′) independently but in the same deterministic manner, 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 rarely that a cell is so heavily perturbed that nodes in a larger than R_t-radius area 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 *IL*s of all neighboring cells may deviate beyond $\sqrt{3}$R. 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 an abandoned cell later.)

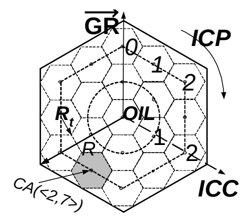
4.2 Algorithm

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*. 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). However, 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 case, 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 death 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 figure 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 H_0 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 ILs in a cell can be lexicographically ordered by tuple <ICC, ICP>, and are considered for becoming the current IL of a cell in an increasing order.

Figure 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, as described in Figure 6, has modified head organization modules, new modules for node join, intra-cell maintenance, inter-cell maintenance, and sanity checking (detailed description of these modules is in given [14]).

<u>Modified head organization modules</u> 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

```
Program Big_node
  GS^3-S with modified HEAD_ORG
  []
  /* Deal with node join */
  q = work → HEAD_JOIN_RESP   //remain status work
  []
  /* Deal with node leave:  remain status work or transit to status big_slide */
  q = work → [HEAD_INTRA_CELL | HEAD_INTER_CELL]
  []
  /* The big node does not act as head */
  q = big_slide  → BIG_SLIDE  //remain status big_slide, or transit to status
                                work
Program Small_node
  GS^3-S with modified HEAD_ORG & HEAD_ORG_RESP
  []
  q = bootup → SMALL_NODE_BOOT_UP //remain status bootup, or
                     transit to status associate or surrogate associate
  []
  /* Head node */
    /* Deal with node join */
    q = work → HEAD_JOIN_RESP      //remain status work
    []
    /*Deal with node leave: remain status work, or transit to status associate*/
    q = work → [HEAD_INTRA_CELL | HEAD_INTER_CELL]
    []
    /* Sanity checking: remain status work, or transit to status associate */
    q = work --Ts--> SANITY_CHECK
    []
  /* Associate node */
    /* Deal with node join: remain status associate/candidate */
    (q = associate ∨ q = candidate) → ASSOCIATE_JOIN_RESP
    []
    /* Deal with node leave: remain status candidate/associate, or transit to
       status head or bootup */
    q = candidate → CANDIDATE_INTRA_CELL
    []
    q = associate → ASSOCIATE_INTRA_CELL
```

Figure 6: Self-configuration algorithm for dynamic networks (GS^3-D)

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. Head *i* becomes an associate when it is resource scarce, a candidate serves better 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* (description of STRENGTHEN_CELL is given in [14]). 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 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*, 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 whether new nodes show up in the direction where it does not have a child, using HEAD_ORG in that direction. When a neighboring head, a child, or its parent changes its IL, *i* optionally synchronizes its IL using low-level module SYN_CELL (the description of PARENT_SEEK and SYN_CELL is given in [14]).

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.

4.3 Analysis

New notation

Head Neighboring Graph $G_{hn} = (V_{hn}, E_{hn})$, where $V_{hn} = V_h$ of 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 three points (formal descriptions are given in [14]):

- In $I_{2.1}$ and $I_{2.2}$, if the <ICC, ICP> value (see figure 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 (instead of 3).
- In $I_{2.4}$, the radius of an inner cell is bounded by $(0, 2R+R_t]$ if its <ICC, ICP> 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 gap-perturbed area adjoining the boundary cell (d_p is 0 if there is no gap-perturbed area).

Theorem 5: DI is an invariant of algorithm GS^3-D, where DI = SI (invariant of GS^3-S) with I_2 relaxed as above.

4.3.2 Fixpoint

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

- $F_{1.2}$ is strengthened as: G_h is a minimum-distance (with respect to the big node H_0) spanning tree of G_{hn} rooted at H_0, i.e. the path between H_0 and a head *i* in G_h is a minimum distance 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: DF is a fixpoint of algorithm GS^3-D, where DF = SF (fixpoint of GS^3-S) with $F_{1.2}$ and $F_{2.4}$ updated as above.

F_1, F_2, F_3, and F_4 imply

Corollary 5: At DF, Corollary 1, 2, and 3 hold in dynamic networks.

4.3.3 Self-stabilization

Theorem 7: Starting from any state, every computation of GS^3-D reaches a state where DI holds within time $O(D_c)$, where D_c is the diameter of a continuous state-corrupted area.

Theorem 8: Starting from any state where DI holds, every computation of GS^3-D reaches a state where DF holds within time $O(\max\{(D_d/c_1), T_d\})$, where c_1 is the average speed of message diffusing and T_d is the maximum difference between the lifetime of the candidate set of two neighboring cells.

Theorem 7 and 8 imply

Corollary 6: Starting from any state, every computation of GS^3-D reaches a state where DF holds within time $O(\max\{(D_d/c_1), T_d\})$.

Requirement *d)* is satisfied by Theorem 7 and 8.

4.3.4 Remarks

4.3.4.1 Scalable self-healing

The self-healing of the head graph and hexagonal structure is scalable in three senses: first, *local self-healing* enables the system to stabilize from a perturbed state to its stable state (fixpoint) 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; second, *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; third, the head graph and hexagonal structure can *tolerate multiple simultaneous perturbations* due to the locality property of GS^3-D.

4.3.4.2 Stable cell structure

In the presence of dynamic nodes, the cell structure is stable in the following senses: *1)* In the 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 remains unchanged; *3)* In the 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.

5. MOBILE DYNAMIC NETWORK

5.1 Concepts

Recall that in mobile dynamic networks not only can nodes be dynamic, but 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 [14]). 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* of H_0 during the time when H_0 itself 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 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 has a new module BIG_MOVE, modified big node, intra-cell maintenance, and inter-cell maintenance, as shown in Figure 7 (a more detailed description is given in [14]).

Program *Big_node*
  GS^3-D with removed BIG_SLIDE, modified intra-cell as well as inter-cell maintenance modules
  []
  /* During status of "big move" */
  q=*big_move*→BIG_MOVE //remain status *big_move*, or transit to status *head*
Program *Small_node*
  GS^3-D with modified intra-cell as well inter-cell maintenance modules

Figure 7: Self-configuration algorithm for dynamic mobile networks (GS^3-M)

5.3 Analysis

5.3.1 Invariant & Fixpoint

The invariant as well as fixpoint of GS^3-D is preserved in GS^3-M, except for one more fixpoint predicate F_5 for GS^3-M as follows.

F_5 (Proxy optimality): The big node H_0 chooses the best neighboring head as its proxy. i.e.

($\forall$ head i : proxy of H_0 *better than* i)

Theorem 9: MI is an invariant of algorithm GS^3-M, where MI = DI (invariant of GS^3-D).

Theorem 10: MF is a fixpoint of algorithm GS^3-M, where MF = DF (fixpoint of GS^3-D) $\wedge$ F_5.

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 entered 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 from any state, every computation of GS^3-M reaches a state where MI holds within time $O(D_c)$, where D_c is the diameter of a continuous state-corrupted area.

Theorem 13: Starting from any state where MI holds, every computation of GS^3-D reaches a state where MF holds within time $O(\max\{(D_d/c_1), T_d\})$, where c_1 is the average speed of message diffusing and T_d is the maximum difference between the lifetime of the candidate set of two neighboring cells.

Theorem 12 and 13 imply

Corollary 7: Starting from any state, every computation of GS^3-M reaches a state where MF holds 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 GS^3-M only depend on local coordination, which enables them to tolerate high degree of node mobility because local coordination converges fast.

6. RELATED WORK

In [18], a distributed algorithm LEACH is proposed for clustering in sensor networks, but it offers no guarantee about placement and the number of clusters in a system. And perturbations are dealt with by globally repeating the clustering operation. In [3], another such distributed algorithm is designed, but it considers only the logical radius of clusters instead of their geographic radius, which can reduce wireless transmission efficiency because of large geographical overlap between cells. Also, its convergence under perturbations depends on multiple rounds of message diffusion, instead of the one-way diffusion within perturbed areas as in our algorithm. Moreover, given a certain level of node distribution density, the geographic radius ensured by our algorithm implicitly guarantees a bound on the logical radius of clusters. In [4], 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 [10], a cellular hexagon structure is described for cellular networks, but it is pre-configured and there is no ability of self-healing. In [11,12], different algorithms for topology control in networks are developed, but they are either centralized or semi-centralized, and thus are not scalable.

In [7–9], algorithms for topology control in wireless networks for energy saving are developed. In [13], 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 is thus orthogonal to these works.

In [19], self-stabilizing algorithms are proposed that mend faults locally in time, but they are not local in space. [20] 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 hexagon structure with a set of proved properties. GS^3 is self-healing, and thus applicable to both static networks and networks with dynamic as well as mobile nodes. GS^3 is also scalable because of its local knowledge, local self-healing, and local coordination properties. GS^3 yields a stable structure even in the presence of dynamic and mobile nodes, which enables a more available infrastructure for other system services such as routing, power control, QoS etc.

Our algorithm is readily extended to the following cases: 1) in a mobile dynamic network where there are multiple big nodes, by letting each small node maintain the current big node it chooses, GS^3-M enables each small node to choose the best (e.g. closest) big node to communicate. 2) Due to its locality property, GS^3 is also applicable to the case where nodes are not deployed on a 2D plane, but where 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.

GS^3 is local and its convergence time is low, thus it is applicable to networks with high degree of dynamics and mobility. More detailed study of dealing with different degrees of node dynamics and mobility is underway.

8. REFERENCES

[1] Deborah Estrin, Ramesh Govindan, John Heidemann and Satish Kumar, "Next century challenges: scalable coordination in sensor networks", *ACM MobiCom* 1999.

[2] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister, "System architecture directions for networked sensors", *ASPLOS* 2000.

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

[4] Ting-chao Hou, Tzu-Jane Tsai, "An access-based clustering protocol for multihop wireless ad hoc networks", *IEEE JSAC*, July 2001.

[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] Alec Woo, David E. Culler, "A transmission control scheme for media access in sensor networks", *ACM SIGMOBILE* 2001.

[7] 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 PODC* 2001.

[8] 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.

[9] Volkan Rodoplu, Teresa H. Meng, "Minimum Energy Mobile Wireless Networks", *IEEE JSAC*, Aug. 1999.

[10] V. H. Mac Donald, "Advanced mobile phone service: the cellular concept", *The Bell System Technical Journal*, 1979.

[11] Shlomi Dolev, Evangelos Kranakis, Danny Krizanc, David Peleg, "Bubbles: adaptive routing scheme for high-speed dynamic networks", *SIAM Journal on Computing*, 1999.

[12] Theodoros Salonidis, Pravin Bhagwat, Leandros Tassiulas, Richard LaMaire, "Distributed topology construction of Bluetooth personal area networks", *IEEE INFOCOM*, 2001.

[13] Ya Xu, John Heidemann, Deborah Estrin, "Geography-informed energy conservation for ad hoc routing", *ACM Mobicom*, July 2001.

[14] Hongwei Zhang, Anish Arora, "GS^3: Scalable Self-configuration and Self-healing in Wireless Networks", OSU-CISRC-4/02-TR08, April 2002.

[15] S. R. Saunders, "Antennas and propagation for wireless communication systems", Wiley (UK), 1999.

[16] Jerry Zhao, Ramesh Govindan, Deborah Estrin, "Residual energy scans for monitoring wireless sensor networks", USC-CSD-TR-01-745, May 2001.

[17] Mahesh Jayaram, George Varghese, "Crash failures can drive Protocols to Arbitrary States", *ACM PODC 1996*.

[18] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks", to appear in *IEEE Transactions on Wireless Networking*.

[19] Shay Kutten, David Peleg, "Fault-Local Distributed Mending", *Journal of Algorithms*, Jan. 1999.

[20] Anish Arora, Mohamed Gouda, "Distributed Reset", *IEEE Transactions on Computers*, 43(9), 1994.

9. APPENDIX

A1. The complexity and convergence properties of GS^3

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 any 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_d: max{dist(i, j): *i* and *j* are small nodes, and dist(i, j) is the cartesian distance between *i* and j}

D_p: the diameter of a contiguous perturbed area

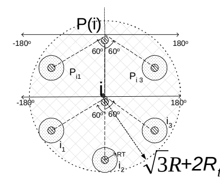
D_b: max{dist(H0, i): i is a small node, and dist(H0,i) is the cartesian distance between the big node H0 and i}

A2. Sample proof

We present the proof for the closure of $I_{2.4}$ in GS^3-S as follows. For the complete set of proofs for theorems in the paper, see [14].

Proof: The modules in GS^3-S that can affect $I_{2.4}$ are HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_ RESP.

Suppose $I_{2.4}$ holds before a round of HEAD_ORG, HEAD_ORG_ RESP, and ASSOCIATE_ORG_RESP execution. We just need to prove that after this round of execution of these modules, initiated by a head i that executes HEAD_ORG, $I_{2.4}$ still holds. This round of head organization can affect head *i*, its possible children heads i_1, i_2, i_3, its parent head P(i), and the two neighboring heads (p_{i1}, p_{i3}) at the same band that are under the care of the same parent head as head *i*, and their covered cells, as shown in the picture below. But we only need to prove that the $I_{2.3}$ holds for the cell C_i covered by head i without loss of generality, because we can prove that $I_{2.3}$ holds for all other related cells in the same way as for cell C_i.



If head i is an inner cell head and thus C_i is an inner cell, then head i is surrounded by six neighboring heads as shown above. Then C_i is also surrounded by six neighboring cells. So, any point in C_i will lie in the triangle formed by head i and two of its immediately neighboring heads (i_1 and i_2, for example). According to the way ASSOCIATE_ORG_RESP works, any point in this triangle chooses the closest head to join. Thus, the maximum distance between a point in head i's cell and head i is $(R+2R_t/\sqrt{3})$, as shown in the figure. Thus, the radius for any inner cell is at most $(R+2R_t/\sqrt{3})$.

Thus $I_{2.4}$ still holds after a round of a round of HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_RESP execution.

❑