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

Hongwei Zhang Anish Arora

Department of Computer and Information Science
The Ohio State University
2015 Neil Avenue, DL 395
Columbus, Ohio 43210 USA
{zhangho, 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

---

* Tel: +1–614–292–1936; Fax: +1–614–292–2911; Web: http://www.cis.ohio-state.edu/ {~zhangho, ~anish}. This work was partially sponsored by DARPA grant OSU-RF-01-C-1901, NSF grant CCR-9972368, and an Ohio State University Fellowship.

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 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 scalablity 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 our algorithm are summarized in Appendix 1.)

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 discussion of complexity as well as convergence properties of our algorithm, description of algorithm modules, and proofs for theorems of the paper 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 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 included a cell if and only if it is connected to the big node.
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, automation is important. Moreover, node crash can drive a crashing network protocol into arbitrary state [17]. Thus self-healing ability of a large-scale system is a must when the system elements are dynamic, mobile, and under perturbations from exterior environments. One way to achieve it is by 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 kind of rare perturbation in dynamic networks discussed 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.

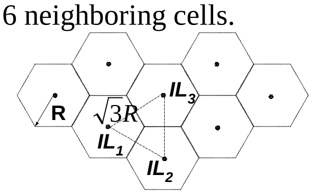


Figure 1: Cellular hexagon structure

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 traditional 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 p_c of *C* as a head, and p_c 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.

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 $\alpha = Sin^{-1}(R_t/\sqrt{3}R)$.

In most cases, a *(d+1)*-band cell head is selected by a *d*-band head (d ≥ 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 ≥ 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.

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 Appendix 2)

3.3 Analysis

In this subsection, we discuss the invariant, fixpoint, self-stabilization, and other properties of algorithm GS^3-S (proofs are given in Appendix 4).

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.

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

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

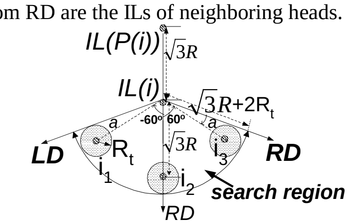
Dist(i, j): cartesian distance between nodes i and j.

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-α, $+60^o$+α, R, R/4) //transit to status *work*: $\alpha = \sin^{-1}(R_t/\sqrt{3}R)$
[]
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×60^o (⌊LD/60⌋ ≤j ≤ ⌊RD/60⌋) degree from RD are the ILs of neighboring heads.

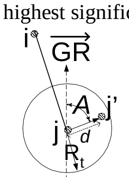


Step 2: Remove the set of IL that is the IL of some existing head from NH. I.e. NH ← (NH − EH), where EH = {j : j ∈ NH ∧ (∃ k ∈ ExistingHeads : (dist(j, k) ≤ R_t))}.

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

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.

<u>**I1** (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.
  ($\forall$ i, j $\in V_h$: there is a path between i and j in $G_h \Leftrightarrow$ 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>**I2** (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 has exactly 6 neighboring heads that form a cellular hexagon centered at head *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): $\sqrt{3}R-2R_t \leq dist(i, j) \leq \sqrt{3}R+2R_t$))
- $I_{2.2}$: 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): $\sqrt{3}R-2R_t \leq dist(i, j) \leq \sqrt{3}R+2R_t$))
- $I_{2.3}$: 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 of ($R+R_{random}$) distance to 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>**I3** (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 $[\sqrt{3}R-2R_t, \sqrt{3}R-2R_t]$.

($\forall$head i: ($\forall$j$\in$ neighboring_heads(i): $\sqrt{3}R-2R_t \leq dist(i, j) \leq \sqrt{3}R+2R_t$)

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

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.

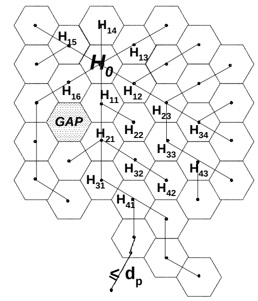


Figure 4: Self-configured cellular hexagon structure

<u>**F1** (Connectivity)</u> and <u>**F2** (Hexagonal Structure)</u> are the same as I_1 and I_2 respectively.

<u>**F3** (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>**F4** (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 H_0 in G_p)

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.

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 Appendix 2).

<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 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.

<u>Node join</u> 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.

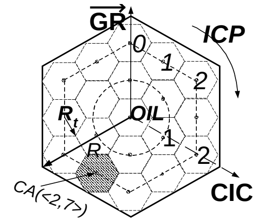
<u>Intra-cell maintenance</u> consists of four modules: HEAD_INTRA_CELL, CANDIDATE_INTRA_CELL, ASSOCIATE_INTRA_CELL, and BIG_SLIDE.

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 Appendix 2). 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

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 Postion* (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

```
Program Big_node
  GS3-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
  GS3-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)

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 Appendix 2).

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 Appendix 3):

- 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 GS3-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

- 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.

- Stable head level structure

In the presence of dynamic nodes, the head level structure is stable in the following senses: *1)* In the case of *node join*, the head level 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 Appendix 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* 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 Appendix 2).

```
Program Big_node
   GS3-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
   GS3-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.

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

($\forall$ head i : proxy of H_0 <u>*better than*</u> 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, but it offers no guarantee about placement and the number of cluster heads in a system. Moreover, the clustering operation is periodically repeated globally in the system over its lifetime. In [3], a distributed algorithm for clustering in wireless networks is designed, but it only considers logical radius (hops) of clusters, instead of their geographic radius, which makes long intra-cluster link possible. 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 certain node density in a network, the geographic radius in our algorithm implicitly guarantees 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 self-healing consideration. In [11,12], different algorithms for topology control in networks are developed, but they are either centralized or semi-centralized, and thus 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.

[19] proposes an algorithm for fault-local mending in time, but it is not local in space. [20] proposes the application of local detection paradigm to self-stabilization, but it is not local in time even though it is local in space. The self-healing 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.

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, *http://www.cis.ohio-state.edu/~zhangho/publications/GS3_prog.pdf.*

[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] Yehuda Afek, Shay Kutten, "The Local Detection Paradigm and its Applications to Self Stabilization", *4th International Workshop on Distributed Algorithms*, Sept. 1990.

APPENDIX

In this appendix, we present the complexity and convergence properties of our algorithm, description of some modules in GS^3-S, GS^3-D and GS^3-M, the invariant as well as fixpoint of GS^3-D, and proofs for various theorems in this technical report.

Appendix 1: Complexity and convergence properties of GS^3-S/D/M

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

Appendix 2: Description of modules in GS^3-S, GS^3-D and GS^3-M

In this subsection, we give more detailed description of some algorithm modules in GS^3-S, GS^3-D and GS^3-M as follows. The complete program is presented in [14].

1) Algorithm GS^3-S

a) HEAD_ORG (LD, RD, R, Rt)

There are four arguments to HEAD_ORG: 1) L direction (LD) and R direction (RD) with respect to direction $\overrightarrow{P(i),i}$ (see Figure 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 Figure 3), head i selects neighboring cell heads *HeadSet*; fourth, head i broadcasts message ⟨*HeadSet*⟩ 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 Figure 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 ⟨*HeadSet*⟩). 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 ⟨*HeadSet*⟩. 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 ⟨*HeadSet*⟩ 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*).

2) Algorithm GS^3-D

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 <ICC, ICP> value and the ordering of all ILs in its cell (see Figure 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 a *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.

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 <ICC, ICP> value, head i synchronizes its cell to the new <ICC, ICP> 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 <ICC, ICP> 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 <ICC, ICP> 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 <ICC, ICP> 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 $\overrightarrow{i,P(i)}$ 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.

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 <ICC, ICP> 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 exit this module without changing its status. Time complexity_is θ(A), where A denotes the size of the contiguously affected area.

3) Algorithm GS^3-M

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*.

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.

Appendix 3: Invariant and fixpoint of GS^3-D (dynamic networks)

1) Invariant

The invariant of GS^3-D differs from that of GS^3-S at I_2 when a cell and its neighboring cells have different <ICC, ICP> 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

∧

(∀ inner_head i: ∀j ∈ neighboring_heads(i):
<ICC(i),ICP(i)> ≠ <ICC(j),ICP(j)> ⇒
((dist(IL(i),IL(j))-2Rt ≤ dist(i, j) ≤ dist(IL(i),IL(j))+2Rt)
∧
(0 < dist(IL(i), IL(j)) ≤ $2\sqrt{3}$ R)
)
)

- **$I_{2.2}$:** (for boundary heads)

$I_{2.2}$ for static networks

∧

(∀ boundary_head i: ∀j ∈ neighboring_heads(i):
<ICC(i),ICP(i)> ≠ <ICC(j),ICP(j)> ⇒
((dist(IL(i),IL(j))-2Rt ≤ dist(i, j) ≤ dist(IL(i),IL(j))+2Rt)
∧

$(0 < dist(IL(i), IL(j)) \le 2\sqrt{3} R)$

)

)

- **$I_{2.3}$**: modify $I_{2.3}$ *for static networks* by changing *($\forall$ head i: | CH(i) | ≤ 3)* to ($\forall$ **head i: | CH(i) | $\le$ 5)**
- **$I_{2.4}$**: (cell radius)

$I_{2.2}$ for static networks

$\wedge$

($\forall$ inner cell C:

($\exists j \in$ neighboring_heads(i):<ICC(i),ICP(i)> $\neq$ <ICC(j),ICP(j)>)$\Rightarrow$

($\forall$ associate i $\in$ C : dist(i, H(i)) < 2R+Rt)

)

$\wedge$

($\forall$ boundary cell *C'*: $\forall$associate i $\in$ *C'*: dist(i, H(i)) $\le \sqrt{3}$ R+2R_t+d_p)

- **I_3** <u>(Inner cell optimality)</u>

Same as in static networks.

2) Fix Point

The fixpoint of GS^3-D differs from that of GS^3-S at $F_{1.2}$ that is strengthened in GS^3-D.

- **F_1** <u>(connectivity)</u>
- **$F_{1.1}$**: 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 $\wedge$

($\forall$ $v_i \in (V_h - \{H_0\})$: 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** <u>(hexagonal structure)</u>

$F_{2.1}$, $F_{2.2}$, and $F_{2.3}$ are the same as in static networks. $F_{2.4}$ is relaxed as:

($F_{2.4}$ of GS^3-S) $\wedge$ (|R_{random}| is at most (($\sqrt{3}$ -1)R+2R_t+d_p) for boundary cells).

- **F_3** <u>(cell optimality)</u>: Same as in static networks.
- **F_4** <u>(coverage)</u>: Same as in static networks.

Appendix 4: Proofs for theorems in the report

We present proofs for the critical theorems in this report. For the complete set of proofs, check [14].

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

- **I_1**: <u>Connectivity</u> (safety property of head level graph)
- **$I_{1.1}$:** Any pair of heads that are connected in G_h are also connected in G_p, and vice versa.

($\forall$ v_{h1}, $v_{h2} \in V_h$: there is a path between v_{h1} and v_{h2} in G_h $\Leftrightarrow$ there is a path between v_{h1} and v_{h2} in G_p)

Proof:

G_p only depends on the nodes in the system and their communication capability, thus has nothing to do with the program actions. G_h only depends on the set of head nodes in the system and the parent-child relationship among them. Thus the set of actions that are related to G_h are those of HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_RESP. At the same time, HEAD_ORG_RESP and ASSOCIATE_ORG_RESP operate under the control of HEAD_ORG, so the critical module is HEAD_ORG.

In order to prove this invariant, we only need to prove it is closed under a round of HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_RESP. The proof is as follows:

1) Suppose $G_h'(V_h', E_h')$ is the G_h before a round of HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_RESP. G_h' and G_p satisfy $I_{1.1}$;
2) After a round of HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_RESP, G_h becomes $G_{h2}(V_{h2}, E_{h2})$.

$\Rightarrow$:

Case one: V_h' is empty

If V_h' is empty, G_{h2} would be such that V_{h2} is the set composed of the big node H_0 and its children heads generated in HEAD_ORG process and E_{h2} is the set of edges that goes from the big node to its children heads. By the process HEAD_ORG, the big node and its children heads are within transmission range of one another and they are at most ($\sqrt{3}$ R+2R_t) away from each other. Thus the big node and its children heads must be directly connected in G_p.

So we only need to prove that any two different heads h1 and h2 are connected in G_p. And this is obvious because both h1 and h2 are connected to H_0 in G_p.

So the claim holds in this case.

Case two: V_h' is not empty

If V_h' is not empty, there must be a head h1 in V_h' such that $V_{h2} = V_h' \cup CH(h1)$ and $E_{h2} = E_h' \cup \{(h1, j): j \in CH(h1)\}$. By the proof of case one, we can easily know that the claim holds for any two nodes that are in the set of $\{h1\} \cup CH(h1)$.

So we only need to prove the claim between a node $h2 \in (V_h' - \{h1\})$ and a node $h3 \in CH(h1)$. If the set $(V_h' - \{h1\})$ is empty, the claim trivially holds. If the set $(V_h' - \{h1\})$ is not empty, then there must be a path p1 between h2 and h1 in G_p and a path p2 (actually just one hope edge) between h3 and h1. So there must be a path p3 between h2 and h3 and p3 is the concatenation of p1 and p2 by head node h1.

So the claim holds in this case.

$\Leftarrow$:

By $I_{1.2}$, G_h is a tree, thus any two heads h1 and h2 are connected in G_h and there would always be a path between then in G_h. So this claim trivially holds.

Thus, after a round of a round of HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_ RESP, G_h and G_p still satisfy $I_{1.1}$.

❑

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

  $(hops(H_0) = 0) \wedge (P(H_0) = H_0) \wedge$

  $(\forall\ v_i \in (V_h - \{H_0\})$: (there is a path between v_i and H_0) $\Rightarrow$ $(hops(H_0, v_i) = hops(H_0, P(v_i))+1) \wedge$

  $(\forall\ v_i, v_j \in V_h$: there is a path between v_i and v_j in $G_h) \wedge$

  $(\forall\ v_i, v_j \in V_h$: there is a path of length no fewer than 2 between v_i and $v_j \Rightarrow (P(v_i) \neq v_j \wedge P(v_j) \neq v_i))$,

  where $hops(v_1, v_2)$ denotes the length of the path from v_1 to v_2 in G_h.

Proof:

Same as the analysis in the proof of $\mathbf{I_{1.1}}$, the modules that can affect this invariant are HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_RESP.

3) Suppose $G_h'(V_h', E_h')$ is the G_h before a round of HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_RESP, and G_h' satisfies $\mathbf{I_{1.2}}$;
4) After a round of HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_ RESP, G_h becomes $G_{h2}(V_{h2}, E_{h2})$.

Case one: V_h' is empty

If V_h' is empty, G_{h2} would be such that V_{h2} is the set composed of the big node H_0 and its children heads $(CH(H_0))$ generated in HEAD_ORG process and E_{h2} is the set of edges that goes from H_0 to nodes in $CH(H_0)$. By the way HEAD_ORG works, for any node $h \in CH(H_0)$, hops(h) would be 1. For any two different heads $h1, h2 \in CH(H_0)$, $(P(h1) \neq h2 \wedge P(h2) \neq h1))$ must hold.

So the claim holds in this case.

Case two: V_h' is not empty

If V_h' is not empty, there must be a head h1 in V_h' such that $V_{h2} = V_h' \cup CH(h1)$ and $E_{h2} = E_h' \cup \{(h1, j): j \in CH(h1)\}$. By the proof of case one, we could easily know that the claim holds for the set of heads of $\{h1\} \cup CH(h1)$. So we only need to prove the claim between a node $h2 \in (V_h' - \{h1\})$ and a node $h3 \in CH(h1)$.

If the set $(V_h' - \{h1\})$ is empty, the claim trivially holds.

If the set $(V_h' - \{h1\})$ is not empty, then there must be a path p1 between h2 and h1 in G_h and a path p2 (actually just one hope edge) between h3 and h1. So there must be a path p3 between h2 and h3 in G_h and p3 is the concatenation of p1 and p2 by head h1. At the same time, h2 must have a parent head $P(h2) \in V_h'$, P(h3) is h1 that is different from h2, and $h3 \notin V_h'$. So $(P(h3) \neq h2 \wedge P(h2) \neq h3))$ must hold.

So the claim holds in this case.

Thus, after a round of a round of HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_ RESP, G_h and G_p still satisfy $\mathbf{I_{1.2}}$.

❑

- **I_2**: <u>Hexagonal map of heads and inner cells</u>

- **$I_{2.1}$**: Each inner cell head i has exactly 6 neighboring heads that form a cellular hexagon centered by head i and of edge length $\sqrt{3}R$, with vertices' location deviation at most R_t.

  $(\forall$ inner cell head i:

  $(|\ neighboring\_heads(i)\ | = 6) \wedge$

  $(\forall j \in neighboring\_heads(i): \sqrt{3}R-2R_t \leq dist(i, j) \leq \sqrt{3}R+2R_t\ )$

)

Proof:

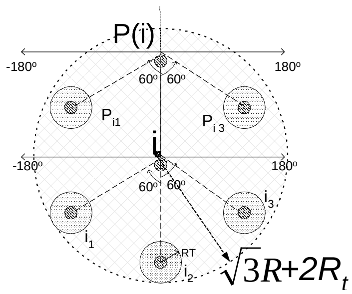
Same as the analysis in the proof of I_1, the modules that can affect $I_{2.1}$ are HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_RESP.

Suppose $I_{2.1}$ holds before a round of HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_ RESP execution. We just need to prove that after the execution of a round of HEAD_ORG, HEAD_ORG_ RESP, and ASSOCIATE_ORG_RESP, initiated by a head i that executes HEAD_ORG, $I_{2.1}$ still holds. Because this round of head organization will only affect head i and its children heads i_1, i_2, i_3, we only need to prove that the $I_{2.1}$ holds for head *i*, i_1, i_2, and i_3. Let's first consider head i.

1) head i is an inner_head ⇒ | neighboring_heads(i) | = 6

a) if head i is the big node, this claim holds obviously just by the way the process HEAD_ORG works;

b) if head i is a small head node, we can get the picture below from the design of HEAD_ORG and the program of Big Node and Small Node.



From the picture above, we can see that head i has 3 next-band heads (i_1, i_2, i_3), 1 parent head (P(i)), and 2 neighboring heads (p_{i1}, p_{i3}) at the same band that are under the care of the same parent head as head *i*, even though they might not be generated by P(i). Thus node i has 6 neighboring heads around within ($\sqrt{3}$ R+2R_t) radius.

Also it is easy to see that head almost centers the hexagon formed by its 6 neighboring heads, with possible deviation at most R_t.

2) head i is an inner_head ⇒ (∀j ∈ neighboring_heads(i): $\sqrt{3}$R-2R_t) ≤ dist(i, j) ≤ $\sqrt{3}$R+2 R_t)

From HEAD_ORG and the picture above, we can see that: for all neighboring head j of node *i*, dist(IL(i), IL(j)) = $\sqrt{3}$R. At the same time "dist(k, IL(k)) ≤ R_t" holds for any head k, thus "$\sqrt{3}$R-2 R_t ≤ dist(i, j) ≤ $\sqrt{3}$R+2 R_t" holds too.

As for head i_1, i_2, and i_3, we can prove, in the same way as above for head *i*, that $I_{2.1}$ also holds for them.

Thus $I_{2.1}$ still holds after a round of a round of HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ ORG_RESP execution.

❑

- $\mathbf{I_{2.2}}$: Each boundary head i has less than 6 neighboring heads, and the distance between i and its neighboring heads is hexagonally bounded. That is,

(∀ boundary_head i: | neighboring_heads(i) | < 6) ∧

(∀ boundary_head i: (∀j ∈ neighboring_heads(i): $\sqrt{3}$ R-2 R_t ≤ dist(i, j) ≤ $\sqrt{3}$ R+2 R_t)

Proof:

Since we have proved that $\mathbf{I_{2.1}}$ is an invariant, we just need to prove that $\mathbf{I}_{2.1}$ ⇒ $\mathbf{I}_{2.2}$ in proving that $\mathbf{I}_{2.2}$ is an invariant. The proof of $\mathbf{I}_{2.1}$ ⇒ $\mathbf{I}_{2.2}$ is as follows:

Boundary heads are generated in the same way as inner heads. The only difference is that their cells are on the boundary of the system's geographic coverage such that there is no neighboring head in certain $(60+2\alpha)^o$ region around itself, where α denotes the angular deviation corresponding to the R_t head's location deviation. Since each inner head has exactly 6 neighboring heads in its ($\sqrt{3}$ R+2R_t) radius, each boundary head should have less than 6 neighboring heads in its ($\sqrt{3}$ R+2R_t) radius. And the distance between boundary head i and its neighboring heads is bounded in the same way as inner head does.

❑

- $\mathbf{I_{2.3}}$: Each head, except for the big node, has no more than 3 children heads. The big node H_0 has 6 children heads if it is not on the system's boundary and it would have 1~5 children heads if it is on the boundary of the system but not disconnected from the small nodes. That is,

(∀ head i: | CH(i) | ≤ 3) ∧

(H_0 is not on the boundary of system coverage ⇒ (|CH(H0)| = 6)) ∧

(H_0 is on the boundary of system coverage but not disconnected ⇒ (1 ≤ | CH(H_0) | ≤ 5))

Proof:

The modules that can affect $\mathbf{I_{2.3}}$ are HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_ RESP.

Suppose $\mathbf{I_{2.3}}$ holds before a round of HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_ RESP execution. We just need to prove that after the execution of a round of HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ORG_RESP, initiated by a head i that executes HEAD_ORG, $\mathbf{I_{2.3}}$ still holds. Because this round of head organization will only affect head i and its children heads i_1, i_2, i_3, and we are only considering a head's children heads, we only need to prove that the $\mathbf{I_{2.3}}$ holds for head i.

If head i is not the big node, from the design of HEAD_ORG, its search region is only 180°. Ideally there would be only one head in 60^o, thus at most there would be no more than three next-band heads (children heads) initiated by head i. Also, if the default value of R_t is R/4, the way HEAD_ORG works also guarantees that no more than three next-band heads initiated by i.

If head i is the big node, its search region is 360^o, thus it would have 6 children heads if the big node is not at the boundary of the system's geographic coverage. If it is at the boundary of the system but not disconnected, the big node H_0 would have 1~5 children heads because there is no neighboring head in certain $(60+2\alpha)^o$ region around the big node, where α denotes the angular deviation corresponding to the R_t head's location deviation.

Thus $\mathbf{I_{2.3}}$ still holds after a round of a round of HEAD_ORG, HEAD_ORG_RESP, and ASSOCIATE_ ORG_RESP execution.

❑

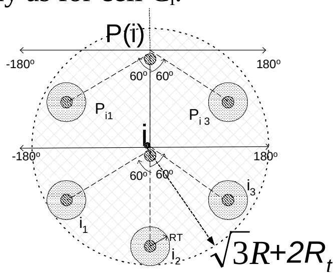
- $\mathbf{I_{2.4}}$: Each cell is of radius ($R+R_{random}$), where $|R_{random}|$ is at most ($2R_t$ / $\sqrt{3}$). And each associate is of ($R+R_{random}$) distance to its head.

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

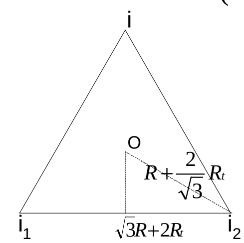
Proof:

The modules 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), as shown in the following figure. 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.

❑

- $\mathbf{I_3}$: <u>Inner Cell Optimality</u> (for associate nodes)
- Each associate node in an inner cell chooses the best (closest, most remaining energy, etc.) neighboring head to join. That is,

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

Proof:

The modules that can affect I_3 are HEAD_ORG and ASSOCIATE_ORG_RESP.

Suppose I_3 holds before a round of HEAD_ORG and ASSOCIATE_ORG_RESP. We just need to prove that after this round of execution of these modules, initiated by a head i that executes HEAD_ORG, I_3 still holds. Because HEAD_ORG only happens at boundary heads at any moment, any round of HEAD_ORG execution could only, from the perspective of I_3, affect those associates that was in a boundary cell before the execution but is in an inner cell after the execution. For any such associate node j, it must have chosen the best head around it as its head because the way ASSOCIATE_ORG_ RESP works. So I3 holds for every such associate j.

Thus I3 holds after any round of HEAD_ORG and ASSOCIATE_ORG_RESP execution.

❑

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

- TRUE *leads to* Invariant ($\mathbf{I_1} \wedge \mathbf{I_2} \wedge \mathbf{I_3}$)

Proof:

In order to prove "TRUE *leads to* Invariant", we just need to prove "$\overline{Invariant}$ *leads to* Invariant" because "Invariant *leads to* Invariant" is obvious.

Because Invariant ($I_1 \wedge I_2$) is closed under all the program actions and there is no state corruption in static networks (according to the definition), the system will not be able to reach any state where $\overline{Invariant}$ would hold. Thus $\overline{Invariant}$ is FALSE all the time. So "$\overline{Invariant}$ *leads to* Invariant" is equal to "FALSE leads to Invariant" that is trivially true. Therefore, "$\overline{Invariant}$ *leads to* Invariant" is true.

According to the analysis above, "TRUE *leads to* Invariant" hold.

❑

3) **Theorem 4**: Starting from any state where SI holds, algorithm GS^3-S reaches a state where SF (SF = $F_1 \wedge F_2 \wedge F_3 \wedge F4$) holds, and the convergence time is $\theta(D_b)$, 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}. That is,

- Invariant ($\mathbf{I_1} \wedge \mathbf{I_2} \wedge \mathbf{I_3}$) *leads to* fix point ($\mathbf{F_1} \wedge \mathbf{F_2} \wedge \mathbf{F_3} \wedge \mathbf{F_4}$)

Proof:

a) Invariant ($\mathbf{I_1} \wedge \mathbf{I_2} \wedge \mathbf{I_3}$) *leads to* $\mathbf{F_3}$

We only need to prove that $I_1 \wedge I_2 \wedge I_3 \wedge \overline{F_3}$ leads to F_3, since F_3 naturally leads to F_3, and I_1 as well as I_2 is invariant.

For any associate node i, the scenario where $\overline{F_3}$ could hold is when some better neighboring head j around it is still at state q_{head} and has not carried out the process HEAD_ORG yet. Because HEAD_ORG and HEAD_ORG_RESP guarantee that two neighboring heads within ($\sqrt{3}$ R+2R_t) range cannot initiate HEAD_ORG in parallel, associate i is able to hear the ORG messages from all its neighboring heads, including head j. The way ASSOCIATE_ORG_RESP works guarantees that associate i will choose the best (such as closest, highest remaining energy, etc.) head to associate with after all such better heads js finish their HEAD_ORG process. So $\overline{F_3}$ will be false and F_3 will be true after all the better neighboring heads around associate i finish their HEAD_ORG processes.

Suppose the number of better heads around associate i is BETTER_HEAD when $\overline{F_3}$ is true. Then BETTER_HEAD is no less than 0. When $\overline{F_3}$ is true, at least one HEAD_ORG process is enabled, and whenever a HEAD_ORG process finishes, the value of BETTER_HEAD will decrease by 1. Thus, it only takes BETTER_HEAD rounds of HEAD_ORG process for associate i to transfer from state $\overline{F_3}$ to state F_3, which is a finite procedure. Thus "$I_1 \wedge I_2 \wedge \overline{F_3}$ *leads to* F_3" holds.

Since it only takes finite time C_{head_org} for a HEAD_ORG process to finish, the state transition from $\overline{F_3}$ to F3 would only take BETTER_HEAD $\times$ (C_{head_org} + C_{gap}) (i.e. θ**(BETTER_HEAD)**) amount of time, where C_{gap} denotes the maximum interval between two neighboring heads' HEAD_ORG process.

b) Invariant ($\mathbf{I_1} \wedge \mathbf{I_2} \wedge \mathbf{I_3}$) *leads to* $\mathbf{F_1} \wedge \mathbf{F_2}$

If we could prove that $I_1 \wedge I_2 \wedge I_3 \Rightarrow F_{2.4}$, then "$I_1 \wedge I_2 \wedge I_3$ *leads to* $F_{2.4}$" holds, which also means "Invariant ($I_1 \wedge I_2 \wedge I_3$) *leads to* $F_1 \wedge F_2$" since I_1 is the same as F_1 and F_2 is equal to $I_2 \wedge F_{2.4}$.

Now let's prove $I_1 \wedge I_2 \wedge I_3 \Rightarrow F_{2.4}$. Because $I_{2.4} \equiv F_{2.4} \wedge$ (There is no R_t -radius gap in the system $\Rightarrow$ R' $\le$ R+$\frac{2}{\sqrt{3}}$ R_t), we only need to prove that $I_1 \wedge I_2 \wedge I_3 \Rightarrow$ (There is no R_t -radius gap in the system $\Rightarrow$ R' $\le$ R+$\frac{2}{\sqrt{3}}$$R_t$). According to the way HEAD_ORG works, the boundary cell would be no bigger than the inner cell, if there is no R_t -radius gap. Otherwise, the HEAD_ORG process will be continuously initiated. Thus the boundary cell's radius is still no more than (R+$\frac{2}{\sqrt{3}}$ R_t) according to $I_{2.4}$ that says any inner cell's radius is no more than (R+$\frac{2}{\sqrt{3}}$ R_t).

c) Invariant ($\mathbf{I_1} \wedge \mathbf{I_2} \wedge \mathbf{I_3}$) *leads to* $\mathbf{F_4}$

We only need to prove that $I_1 \wedge I_2 \wedge I_3 \wedge \neg F_4$ leads to F_4, since it is obvious that $I_1 \wedge I_2 \wedge I_3 \wedge F_4$ leads to F_4.

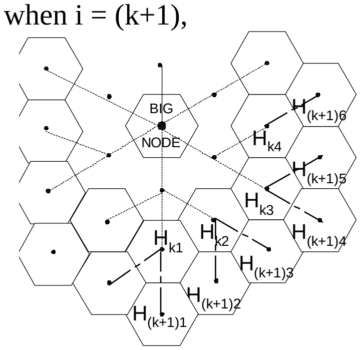
Since the way HEAD_ORG works guarantees that all the visible areas of the system can be covered by the HEAD_ORG process in the end (see the proof of this claim later), there will be a HEAD_ORG process waiting to take place whenever $\neg F_4$ holds. Because the system's coverage is finite and every HEAD_ORG process is able to cover another ($\sqrt{3}R+2R_t$)-radius circular area, the number of possible HEAD_ORG process occurrence is finite. Therefore, "$I_1 \wedge I_2 \wedge I_3 \wedge \neg F_4$ leads to F_4" holds.

Now, let us prove that all the visible areas of the system can be covered by the HEAD_ORG process in the end. We prove it by induction on the area encircled by heads of i-band away from the big node, denoted Round Area *RA(i)* (i.e. area of radius ($\sqrt{3}R \times i + R_t + R$) and i is the number of hexagons away from the big node).

Base: when i = 0, 1, clearly holds

Hypothesis: the claim holds when i = k

Induction: when i = (k+1),



As we can see from the picture, any point that is in RA(k+1) but not in RA(k) will be covered by some (k+1)-band head. And each (k+1)-band head can be taken care of by some k-band head, either directly or indirectly, even though some of them might not be generated directly by a k-band head due to different progress speeds of the self-configuration process at different directions spreading from the big node. Thus the claim holds when i is (k+1).

❑

4) **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.

- **I_1**: <u>Connectivity</u> (safety property of head level graph)

  - **$I_{1.1}$:** Any pair of heads that are connected in G_h are also connected in G_p, and vice versa. That is, ($\forall$ v_{h1}, $v_{h2} \in V_h$: there is a path between v_{h1} and v_{h2} in $G_h \Leftrightarrow$ there is a path between v_{h1} and v_{h2} in G_p)

Proof:

G_p only depends on the nodes in the system and their communication capability, thus has nothing to do with the program actions. G_h only depends on the set of head nodes in the system and the parent-child relationship among them. Thus the set of actions that are related to G_h are those of HEAD_ORG, HEAD_ORG_RESP, ASSOCIATE_ORG_RESP, intra-cell maintenance, inter-cell maintenance, and system state sanity check. New node join does not affect this claim, because it does not affect the head level structure directly.

At the same time, starting from a state where the Invariant holds, system state sanity check will not be enabled. So, in proving $I_{1.1}$, we only need to prove that it is closed under a round of HEAD_ORG, HEAD_ORG_RESP, ASSOCIATE_ORG_RESP, intra-cell maintenance and inter-cell maintenance.

$\Rightarrow$:

In dynamic immobile networks, the only modifications to processes HEAD_ORG, HEAD_ORG_RESP and ASSOCIATE_ORG_RESP that could affect the claim differently from that of the static networks is the case where an existing head i selects a better parent head j. In this case, there is an edge (i, j) added to G_h. According to HEAD_ORG and HEAD_ORG_RESP, head i and j are no more than ($\sqrt{3}R+2R_t$) away from each other and within transmission range of each other. Thus edge (i, j) must exist in G_p too. Therefore the claim is still closed under the modified HEAD_ORG, HEAD_ORG_RESP and ASSOCIATE_ORG_RESP processes in dynamic immobile networks.

In intra-cell maintenance, the sub-modules that could affect G_h is STRENGTHEN_CELL and REPLACING_HEAD. STRENGTHEN_CELL itself does not affect G_h, because it just demotes the current head to associate and promotes another associate to be head. The result of STRENGTHEN_CELL is a REPLACING_HEAD process. So we only need to prove that the claim is closed under REPLACING_HEAD. The way REPLACING_HEAD works guarantees that the newly elected head i is no more than $2\sqrt{3}R$ away from its children head as well as neighboring heads and is within transmission range of one another. Therefore, they must be an edge between head i and each of its children heads in G_p. Thus, the claim is closed under REPLACING_HEAD.

In inter-cell maintenance, the sub-modules that can affect G_h is HEAD_ORG, SYN_CELL and PARENT_ SEEK. Since the claim is closed under HEAD_ORG, we only need to prove that it is closed under SYN_CELL and PARENT_SEEK. SYN_CELL does not affect the claim directly because the head is just removed from G_h. Instead only SYN_CELL's result, REPLACING_ HEAD, could affect the claim. Since the claim is closed under REPLACING_HEAD, it is also closed under SYN_CELL. In PARENT_SEEK initiated by a head j, if head j succeeds in finding a neighboring head as parent, j must be no more than $2\sqrt{3}R$ away from its parent and they are within transmission range of each other. Thus there must be an edge between head j and its new parent in G_p. Therefore, the claim is closed under PARENT_SEEK.

$\Leftarrow$:

By $I_{1.2}$, G_h is a tree, thus any two heads h1 and h2 are connected in G_h and there would always be a path between then in G_h. So this claim trivially holds.

Therefore, after a round of a round of HEAD_ORG, HEAD_ORG_RESP, ASSOCIATE_ ORG_RESP, intra-cell maintenance, inter-cell maintenance and system state sanity check execution, G_h and G_p still satisfy $I_{1.1}$.

❑

- **$I_{1.2}$:** G_h is a tree rooted at the big node H_0. That is,
  $(hops(H_0) = 0) \wedge (P(H_0) = H_0) \wedge$

  $(\forall\ v_i \in (V_h - \{H_0\})$: (there is a path between v_i and H_0) $\Rightarrow (hops(H_0, v_i) = hops(H_0, P(v_i))+1) \wedge$

  $(\forall\ v_i, v_j \in V_h$: there is a path between v_i and v_j in $G_h) \wedge$

  $(\forall\ v_i, v_j \in V_h$: there is a path of length no fewer than 2 between v_i and $v_j \Rightarrow (P(v_i) \neq v_j \wedge P(v_j) \neq v_i))$,

  where $hops(v_1, v_2)$ denotes the length of the path from v_1 to v_2 in G_h.

Proof:

The set of actions that are enabled at an Invariant state and that can affect G_h are those of HEAD_ORG, HEAD_ORG_RESP, ASSOCIATE_ORG_RESP, intra-cell maintenance, and inter-cell maintenance. New node join does not affect this claim, because it does not affect the head level structure directly.

In order to prove $I_{1.2}$, we only need to prove that it is closed under a round of HEAD_ORG, HEAD_ORG_RESP, ASSOCIATE_ORG_RESP, intra-cell maintenance, and inter-cell maintenance.

In dynamic immobile networks, the only modifications to processes HEAD_ORG, HEAD_ ORG_RESP and ASSOCIATE_ORG_RESP that could affect the claim differently from that of the static networks is the case where an existing head j selects a better parent head k. In this case, G_h is still a tree because, even though the edge between j and its previous head in G_h is removed, the added edge (j, k) guarantees that the sub-tree rooted at head j in G_h is still connected to other nodes in G_h through head k.

In intra-cell maintenance, the sub-modules that could affect G_h is STRENGTHEN_CELL and REPLACING_HEAD. The result of STRENGTHEN_ CELL and REPLACING_HEAD is that one head j in G_h is replaced by an associate k in j's cell that is not in G_h previously. The way STRENGTHEN_CELL and REPLACING_HEAD work guarantees that j's relationship with its parent head as well as children heads are transferred to node k. Thus, the structure of G_h is maintained after any STRENGTHEN_CELL or REPLACING_HEAD operation, except that node j is replaced by node k. So, the claim is closed under intra-cell maintenance.

In inter-cell maintenance, the sub-modules that can affect G_h is HEAD_ORG, SYN_CELL and PARENT_ SEEK. Since the claim is closed under HEAD_ORG, we only need to prove that it is closed under SYN_CELL and PARENT_SEEK. The result of SYN_CELL is that one head j in G_h is replaced by an associate k in j's cell that is not in G_h previously, if such k exists. If such k really exists, the way SYN_CELL and its resulting REPLACING_HEAD work guarantees that j's relationship with its parent head as well as children heads are transferred to node k. Thus, the structure of G_h is maintained after the SY_CELL and REPLACING_HEAD operation. If such k does not exist, process PARENT_SEEK will be initiated, which guarantees that the sub-tree previously rooted at head j in G_h will be connected to the remaining part of G_h if it is not completely separated from the system. So $I_{1.2}$ is still closed in this case. As for PARENT_SEEK, it will be initiated at head j only if the sub-tree rooted at j is disconnected from the other part of G_h because of the removal of edge (j, P(j)). The result of PARENT_SEEK is that this sub-tree gets re-connected to the remaining G_h if the sub-tree is not disconnected from the remaining G_h, or this sub-tree disappears (i.e. all heads in this sub-tree go to boot-up state) if it is completely disconnected from the remaining G_h. Thus $I_{1.2}$ is closed under PARENT_SEEK. Therefore, $I_{1.2}$ is closed under inter-cell maintenance.

Therefore, after a round of HEAD_ORG, HEAD_ORG_RESP, ASSOCIATE_ORG_ RESP, intra-cell maintenance, and inter-cell maintenance execution, $I_{1.2}$ still holds if it held before the execution of these modules.

❑

- **I_2**: Hexagonal map of heads and inner cells

- **$I_{2.1}$**: Each inner head i has exactly 6 neighboring heads, and the 6 neighboring heads of head i forms a cellular hexagon that is centered by head i, with ***bounded vertices' location deviation***. That is,

  (∀ inner_head i: | neighboring_heads(i) | = 6) ∧

  (∀ inner_head i (∀j ∈ neighboring_heads(i):

  $(\langle CIC(i),ICP(i)\rangle = \langle CIC(j),ICP(j)\rangle) \Rightarrow (\sqrt{3}R-2R_t \leq dist(i, j) \leq \sqrt{3}R+2R_t) \wedge$

  $(\langle CIC(i),ICP(i)\rangle \neq \langle CIC(j),ICP(j)\rangle) \Rightarrow (dist(IL(i),IL(j))-2R_t \leq dist(i, j) \leq dist(IL(i),IL(j))+2R_t) \wedge (0 < dist(IL(i), IL(j)) \leq 2\sqrt{3}R)$

)

Proof:

The set of actions that are enabled at an Invariant state and that can affect $I_{2.1}$ are those of HEAD_ORG, HEAD_ORG_RESP, ASSOCIATE_ORG_RESP, intra-cell maintenance, and inter-cell maintenance. New node join does not affect this claim, because it does not affect the head level structure directly.

In order to prove $I_{2.1}$, we only need to prove that it is closed under a round of HEAD_ORG, HEAD_ORG_RESP, ASSOCIATE_ORG_RESP, intra-cell maintenance, and inter-cell maintenance.

In dynamic immobile networks, the only modifications to processes HEAD_ORG, HEAD_ ORG_RESP and ASSOCIATE_ORG_RESP that could affect the claim differently from that of the static networks is the case where an existing head j selects a better parent head k. Even in this case, $I_{2.1}$ should still be closed under a round of head organization process, because it affects neither j's nor k's geographical location. Also, in dynamic immobile networks, a HEAD_ORG process will be initiated by a head j only if j and its parent head P(j) are at the same <CIC, ICP> point within their respective cells. Under this premise, the next band head organization process acts essentially the same as that in static networks in terms of head level's hexagonal feature. So, $I_{2.1}$ is closed under a round of HEAD_ORG, HEAD_ORG_RESP and ASSOCIATE_ORG_RESP execution in dynamic immobile networks.

In intra-cell maintenance, the sub-modules that can affect head level structure are REPLACING_HEAD and STRENGTHEN_CELL. The result of REPLACING_HEAD is that one head j in a cell is replaced by a head candidate k in j's cell that is no more than R_t away from the ideal location of this cell. The way REPLACING_HEAD works guarantees that j's relationship with its neighboring heads is transferred to node k. Thus, $I_{2.1}$ is closed under any round of REPLACING_HEAD. The result of STRENGTHEN_ CELL initiated by a head j is that j retreats to be an associate node and another associate node k previously in j's cell become the head of this cell. If the newly elected head k is at the same <CIC, ICP> point as its neighboring cell's head m, they are at the same relative location to their respective OIL, with on more than R_t location deviation. According to the HEAD_ORG, HEAD_ORG_RESP and ASSOCIATE_ORG_RESP, the original ideal location (OIL) of any two neighboring cells is exactly $\sqrt{3}$R. Therefore, the distance between k and m should be $\sqrt{3}$R with no more than $2R_t$ deviation, the same as that in static networks. However, if k and m are at different <CIC, ICP> points, this regularity does not hold any more. But the distance between k and m is still no more than $2\sqrt{3}$R, because the way STRENGTHEN_CELL works guarantees that the maximum distance between any possible ILs of two neighboring cells is no more than $2\sqrt{3}$R. So, the $I_{2.1}$ is closed STRENGTHEN_CELL too. In a word, $I_{2.1}$ is closed under intra-cell maintenance.

In inter-cell maintenance, the sub-modules that can affect head level structure are HEAD_ORG, SYN_CELL and PARENT_ SEEK. Since the claim is closed under HEAD_ORG, we only need to prove that it is closed under SYN_CELL and PARENT_SEEK. The result of SYN_CELL is that one head j is replaced by an associate k in j's cell, if such k exists. If such k really exists, the way SYN_CELL and its resulting REPLACING_HEAD work guarantees that j's relationship with its neighboring heads is transferred to node k. Thus, $I_{2.1}$ closed under SY_CELL and its resulting REPLACING_HEAD operations. If such k does not exist, process PARENT_SEEK will be initiated. As for PARENT_SEEK, it will be initiated at head j only if the sub-tree rooted at j is disconnected from the other part of G_h because of the removal of edge (j, P(j)) in G_h. The result of PARENT_SEEK will not add any new head to the system or change the location of any existing heads, even though it might remove some existing heads from the head level graph. Thus $I_{1.2}$ must be closed under PARENT_SEEK. Therefore, $I_{2.1}$ is closed under inter-cell maintenance.

Based upon the analysis above, after a round of HEAD_ORG, HEAD_ORG_RESP, ASSOCIATE_ ORG_RESP, intra-cell maintenance, and inter-cell maintenance execution, $I_{2.1}$ still holds if it held before the execution of these modules.

❑

- **$I_{2.2}$**: Each boundary head i has less than 6 neighboring heads, and the distance between i and its neighboring heads is hexagonally bounded. That is,

  $(\forall$ boundary_head i: | neighboring_heads(i) | < 6) $\wedge$

  $(\forall$ boundary_head i $(\forall j \in$ neighboring_heads(i):

  $$\boldsymbol{(<CIC(i),ICP(i)> \;=\; <CIC(j),ICP(j)>)} \Rightarrow (\sqrt{3}R-2R_t \le dist(i,j) \le \sqrt{3}R+2R_t) \wedge$$

  $$\boldsymbol{(<CIC(i),ICP(i)> \neq <CIC(j),ICP(j)>)} \Rightarrow (\boldsymbol{dist(IL(i),IL(j))}-2R_t \le dist(i,j) \le \boldsymbol{dist(IL(i),IL(j))}+2R_t) \wedge \boldsymbol{(0 < dist(IL(i), IL(j)) \le 2\sqrt{3}R)}$$

)

Proof:

Since we have proved that $\mathbf{I_{2.1}}$ is an invariant, we just need to prove that $\mathbf{I}_{2.1} \Rightarrow \mathbf{I}_{2.2}$ in proving that $\mathbf{I}_{2.2}$ is an invariant. The proof of $\mathbf{I}_{2.1} \Rightarrow \mathbf{I}_{2.2}$ is as follows:

Boundary heads are generated and maintained (by intra-cell as well as inter-cell maintenance procedure) in the same way as inner heads. The only difference is that their cells are on the boundary of the system's geographic coverage such that there is no neighboring head in certain $(60+2\alpha)^o$ region around itself, where α denotes the angular deviation corresponding to the R_t head's location deviation. Since each inner head has exactly 6 neighboring heads in its $(\sqrt{3}R+2R_t)$ radius, each boundary head should have less than 6 neighboring heads in its $(\sqrt{3}R+2R_t)$ radius. And the distance between boundary head i and its neighboring heads is bounded in the same way as inner head does.

❑

- **$I_{2.3}$**: Each head, except for the big node, has no more than ***5*** children heads. The big node H_0 has 6 children heads if it is not on the system's boundary and it would have 1~5 children heads if it is on the boundary of the system but not disconnected from the small nodes. That is,

  $(\forall$ head i: | CH(i) | $\le$ ***5***) $\wedge$

  (H_0 is not on the boundary of system coverage $\Rightarrow$ (|CH(H0)| = 6)) $\wedge$

  (H_0 is on the boundary of system coverage but not disconnected $\Rightarrow$ ($1 \le$ | CH(H_0) | ≤ 5))

Proof:

Since both $I_{2.1}$ and $I_{2.2}$ are invariants, and $I_{2.1} \wedge I_{2.2} \Rightarrow I_{2.3}$, $I_{2.3}$ is an invariant too.

❑

- **$I_{2.4}$**: Each cell is of radius ($R+R_{random}$). ***When a cell i and all its neighboring cells are at the same <CIC, ICP> point,*** $|R_{random}|$ is no more than $\frac{2}{\sqrt{3}}R_t$ for inner cell i and no more than $((\sqrt{3}-1)R+2R_t+d_p)$ for boundary cells, with d_p being the diameter of the gap-perturbed area adjoining a boundary cell (d_p is 0 if there is no gap-perturbed area).; ***Otherwise, $|R_{random}|$ is less than (2R+Rt) for inner cell i***. This also means that each associate node is of ($R+R_{random}$) distance to its head. That is,

$(\forall$ inner cell C(i):

$\boldsymbol{(\forall j \in neighboring\_heads(i): \; (<CIC(i),ICP(i)> \;=\; <CIC(j),ICP(j)>)}$

$$\Rightarrow (\forall \text{ associate } i \in C : dist(i, H(i)) \le R+\frac{2}{\sqrt{3}}R_t))$$

$\wedge$

$\boldsymbol{(\exists\, j \in neighboring\_heads(i): \; (<CIC(i),ICP(i)> \neq <CIC(j),ICP(j)>)}$

$$\Rightarrow (\forall \text{ associate } i \in C : dist(i, H(i)) < 2R+R_t)$$

)

$\wedge$

$(\forall$ boundary cell C(i): $(\forall$ associate i $\in$ C(i) : dist(i, H(i)) $\le$ R')) $\wedge$ (R' $\le \sqrt{3}R+2R_t+d_p$))

Proof:

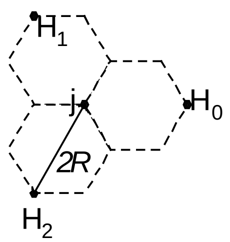
As for inner cell C(i), we could prove the claim is closed under program actions by proving that $I_{2.1} \wedge I_{2.2} \wedge I_3 \Rightarrow$ ($I_{2.4}$ for inner cell). The proof is as follows:

Case one: ∀j∈ neighboring_heads(i): <CIC(i), ICP(i)> = <CIC(j),ICP(j)>

The proof is the same as that for static networks, we neglect it here for simplicity.

Case two: ∃ j∈neighboring_heads(i): <CIC(i), ICP(i)> ≠ <CIC(j),ICP(j)>

In this case, the worst scenario for an associate node j in an inner cell is as the following figure shows:



That is, the three heads (H0, H1 and H2) are farthest away from one another ($2\sqrt{3}$R). But the maximum distance between j and any of these three heads is still no more than 2R with possible R_t location deviation by geometry calculation.

As for boundary cells, the set of actions that are possibly enabled at invariant state and that can affect $I_{2.4}$ for boundary cells are those of HEAD_ORG, ASSOCIATE_ORG_RESP, new node joins, intra-cell maintenance and inter-cell maintenance.

As proved in static networks, $I_{2.4}$ for boundary cells is closed under HEAD_ORG and ASSOCIATE_ORG_RESP operations.

$I_{2.4}$ for boundary cells is closed under new node join operation too, because new node join does not creat new heads, and the way SMALL_NODE_BOOT_UP and SMALL_NODE_JOIN work guarantees that the distance between the new node and its selected head is no more than MAX+($\sqrt{3}$ R+2R_t).

In intra-cell maintenance, the sub-modules that can affect $I_{2.4}$ for boundary cells are REPLACING_HEAD and STRENGTHEN_CELL. The result of REPLACING_HEAD is that one head j in a cell is replaced by a head candidate k in j's cell that is no more than R_t away from the ideal location of this cell. Thus, $I_{2.4}$ for boundary cells is closed under any round of REPLACING_HEAD. The result of STRENGTHEN_ CELL initiated by a head j is that j retreats to be an associate node and another associate node k previously in j's cell become the head of this cell. Before the execution of STRENGTHEN_CELL, the maximum distance between j's cell's original ideal location and any associate in this cell is no more than MAX+($\sqrt{3}$ R+2R_t). after the STRENGTHEN_CELL operation, the distance between k and this cell's original ideal location is no more than R. So after the STRENGTHEN_CELL operation, the maximum distance between k and any associate in its cell is no more than MAX+($\sqrt{3}$ R+2R_t)+R. So $I_{2.4}$ for boundary cells is closed under STRENGTHEN_CELL. In a word, $I_{2.4}$ for boundary cells is closed under intra-cell maintenance.

In inter-cell maintenance, the sub-modules that can affect $I_{2.4}$ for boundary cells are HEAD_ORG, SYN_CELL and PARENT_ SEEK. Since the claim is closed under HEAD_ORG, we only need to prove that it is closed under SYN_CELL and PARENT_SEEK. The result of SYN_CELL is that one head j is replaced by an associate k in j's cell, if such k exists. If such k really exists, the way SYN_CELL and its resulting REPLACING_HEAD work guarantees that $I_{2.4}$ for boundary cells still holds after the execution, in the same way as STRENGTHEN_CELL. Thus, $I_{2.4}$ for boundary cells is closed under SY_CELL and its resulting REPLACING_HEAD operations. If such k does not exist, any associate in j's cell goes back to boot-up state and acts as a new node joining the system, where $I_{2.4}$ for boundary cells is also closed. So $I_{2.4}$ for boundary cells is closed under SYN_CELL. As for PARENT_SEEK, it will be initiated at head j only if the sub-tree rooted at j is disconnected from the other part of G_h because of the removal of edge (j, P(j)) in G_h. The result of PARENT_SEEK will not add any new head to the system or change the location of any existing heads, even though it might remove some existing heads from the head level graph, which will initiate new node joins operation. But all this does not violate $I_{2.4}$ for boundary cells. Thus $I_{2.4}$ for boundary cells must be closed under PARENT_SEEK. Therefore, $I_{2.4}$ for boundary cells is closed under inter-cell maintenance.

❑

- **I_3**: <u>Inner Cell Optimality</u> (for associate nodes)

- Each associate node in an inner cell chooses the best (closest, most remaining energy, etc.) neighboring head to join. That is,

(∀ associate i in any inner cell: ∀ head j ≠ H(i) (H(i) <u>*better than*</u> j))

Proof:

The set of actions that are possibly enabled at invariant state and that can affect I_3 are those of HEAD_ORG, ASSOCIATE_ORG_RESP, new node join, intra-cell maintenance, and inter-cell maintenance.

As proved for static networks, I_3 is closed under HEAD_ORG and ASSOCIATE_ORG_RESP.

I_3 is closed under new node join operation because in SMALL_NODE_BOOT_UP and SMALL_NODE_JOIN, a new node always choose the best neighboring head to associate. In SMALL_NODE_BOOT_UP and SMALL_NODE_JOIN, a new node might join a neighboring head's HEAD_ORG process, but in this case, the new node is in a boundary cell, either at an inner gap or at real system boundary. Therefore, I_3 is closed under new node join scenario.

In intra-cell maintenance, the sub-modules that can affect I_3 are REPLACING_HEAD and STRENGTHEN_CELL. The result of REPLACING_HEAD is that one head j in a cell is replaced by a head candidate k in j's cell. At the moment k assumes the head role, it will broadcast a Head_intra_alive message that guarantees that any associate that k can serve as a better head will choose k as the new better head. Thus, I_3 is closed under any round of REPLACING_HEAD. The result of STRENGTHEN_ CELL initiated by a head j is the same as REPLACING_HEAD from the point of view of I_3. So I_3 is closed under STRENGTHEN_CELL. In a word, I_3 is closed under intra-cell maintenance.

In inter-cell maintenance, the sub-modules that can affect I_3 are HEAD_ORG, SYN_CELL and PARENT_SEEK. Since I_3 is closed under HEAD_ORG, we only need to prove that it is closed under SYN_CELL and PARENT_SEEK. The result of SYN_CELL is that one head j is replaced by an associate k in j's cell, if such k exists. If such k really exists, the result of SYN_CELL and its resulting REPLACING_HEAD is the same as that of REPLACING_HEAD from the I_3's point of view. Thus, I_3 is closed under SYN_CELL and its resulting REPLACING_HEAD operations. If such k does not exist, any associate in j's cell goes back to boot-up state and acts as a new node joining the system, where I_3 is also closed. So I_3 is closed under SYN_CELL. As for PARENT_SEEK, it will be initiated at head j only if the sub-tree rooted at j is disconnected from the other part of G_h because of the removal of edge (j, P(j)) in G_h. The result of PARENT_SEEK will not add any new head to the system or change the location of any existing heads, even though it might remove some existing heads from the head level graph, which will initiate new node joins operation. But all this does not violate I_3. Thus I_3 must be closed under PARENT_SEEK. Therefore, I_3 is closed under inter-cell maintenance.

❑

5) **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.. That is,

TRUE *<u>leads to</u>* Invariant ($\mathbf{I_1} \wedge \mathbf{I_2} \wedge \mathbf{I_3}$)

Proof:

In order to prove "TRUE *<u>leads to</u>* Invariant", we just need to prove "$\overline{Invariant}$ *<u>leads to</u>* Invariant" because "Invariant *<u>leads to</u>* Invariant" is obvious. The proof is as follows.

There are two cases where $\overline{Invariant}$ could be reached due to node failure, even though it cannot be reached just program actions.

a) A head j dies without following proper head retreat procedure

This kind of failure may make G_h disconnected temporarily, thus affects connectivity ($I_{1.2}$). This could be dealt with by inter-cell maintenance procedures. By inter-cell maintenance procedure, the parent head P(j) of head j will first initiate another round of HEAD_ORG process, trying to recover the failure of head j. If there is any node in head j's original R_t-radius range, the HEAD_ORG process will succeed in finding a replacing head k to play the role head j previously did. Thus this will make G_h connected again. If it fails, the sub-trees previously rooted at head j will initiate PARENT_SEEK. This will make these sub-trees connected to the remaining G_h if they are not completely disconnected from the remaining G_h. If they are really disconnected from the remaining Gh, all heads in the sub-tree will retreat to boot-up state, which also makes $I_{1.2}$ true again. Time complexity is $\boldsymbol{\theta(A)}$, where A stands for the size of the contiguously affected area.

This failure can also make all the associates originally in its cell live without a head, even though they still have their head pointer point to node j. So the inner cell optimality (I_3) is violated here too. This could be dealt with by intra-cell maintenance. Any associate k previously in head j's cell will go back to boot-up state after detecting that their head has died. Then SMALL_NODE_BOOT_UP process will be initiated at node k, which makes I_3 hold for node k again. On the other hand, before node k goes back to boot-up state, if it hears Head_intra_alive message from another head m other than j and k considers m being better than j, node k will choose head m as its head. Thus still makes I_3 holds again. The time complexity of this process is $\theta(C_1)$, where C_1 is a small constant related to the purely one-hop message exchange.

b) State corruption at existing head node or associate node j

This can affect connectivity ($I_{1.1}$) because the corrupted head j might choose a random number as P(j) and P(j) is the ID of another head, but P(j) and j are beyond the transmission range of one another. This could be dealt with by inter-cell maintenance. Since j cannot hear Head_inter_alive message from P(j), it will initiate PARENT_SEEK to find another head to associate, which will make $I_{1.1}$ true again. The time complexity of this process is $\theta(C_2)$, where C_2 is a small constant related to T_{phbt} and the purely one-hop message exchange in PARENT_SEEK (we assume head j's original parent is still there and j has heard Head_inter_alive message from it).

This can affect hexagonal head level structure (I_2) because head j's CH(j) might be corrupted, because head j will not hear Head_inter_alive message from its mistakenly-elected child head, which will make head j initiate a HEAD_ORG process even though it should not be. This will generate a head that should not be a head indeed. It could be dealt with by system state sanity check, as discussed in c). Time complexity is $\theta(A)$, where A stands for the size of the contiguously affected area.

This can affect inner cell optimality (I_3) because an associate j's H(j) could be corrupted such that H(j) is either not the best or is beyond transmission range. This could be dealt with by intra-cell maintenance at the moment associate j hears its neighboring heads' Head_intra_alive messages. The time complexity of this process is $\theta(C_3)$, where C_3 is a small constant related to the purely one-hop message exchange.

c) A new head j shows up due to state corruption, or due to HEAD_ORG initiated by a corrupted head, even though it should not be a head

It can affect the hexagonal head level structure (I_2), because an extra head shows up in the previously hexagonal head map, which corrupts the hexagon feature. This could be dealt with by system state sanity check. Time complexity is $\theta(A)$, where A stands for the size of the contiguously affected area.

d) Intra-cell maintenance at cell j fails to find a replacing head for this cell

The effect and the recovery process is the same as case a).

According to the analysis above, "TRUE <u>*leads to*</u> Invariant" hold.

❑

6) **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. That is,

Invariant ($\mathbf{I_1} \wedge \mathbf{I_2} \wedge \mathbf{I_3}$) <u>*leads to*</u> Fix Point ($\mathbf{F_1} \wedge \mathbf{F_2} \wedge \mathbf{F_3} \wedge \mathbf{F_4}$)

Proof:

a) Invariant ($\mathbf{I_1} \wedge \mathbf{I_2} \wedge \mathbf{I_3}$) <u>*leads to*</u> $\mathbf{F_3}$

We only need to prove that $I_1 \wedge I_2 \wedge I_3 \wedge \overline{F_3}$ leads to "F_3 for boundary cell", since F_3 naturally leads to F_3, I_1 as well as I_2 is invariant, and $I_3 \Rightarrow$ "F_3 for inner cell".

Same as that in static networks, for any associate node i in a coundary cell, the scenario where $\overline{F_3}$ could hold is when some better neighboring head j around it is still at state q_{head} and has not carried out the process HEAD_ORG yet. Because HEAD_ORG and HEAD_ORG_RESP guarantee that two neighboring heads within ($\sqrt{3}$ R+2R_t) range cannot initiate HEAD_ORG in parallel, associate i is able to hear the ORG messages from all its neighboring heads, including head j. The way ASSOCIATE_ORG_RESP works guarantees that associate i will choose the best (such as closest, highest remaining energy, etc.) head to associate with after all such better heads js finish their HEAD_ORG process. So $\overline{F_3}$ will be false and F_3 will be true after all the better neighboring heads around associate i finish their HEAD_ORG processes.

Suppose the number of better heads around associate i is BETTER_HEAD when $\overline{F_3}$ is true. Then BETTER_HEAD is no less than 0. When $\overline{F_3}$ is true, at least one HEAD_ORG process is enabled, and whenever a HEAD_ORG process finishes, the value of BETTER_HEAD will decrease by 1. Thus, it only takes BETTER_HEAD rounds of HEAD_ORG process for associate i to transfer from state $\overline{F_3}$ to state F_3, which is a finite procedure. Thus "$I_1 \wedge I_2 \wedge \overline{F_3}$ *leads to* F_3" holds.

Since it only takes finite time C_{head_org} for a HEAD_ORG process to finish, the state transition from $\overline{F_3}$ to F3 would only take BETTER_HEAD $\times$ ($C_{head\_org} + C_{gap}$) (i.e. θ**(BETTER_HEAD)**) amount of time, where C_{gap} denotes the maximum interval between two neighboring heads' HEAD_ORG process.

b) Invariant ($\mathbf{I_1 \wedge I_2 \wedge I_3}$) *leads to* $\mathbf{F_1 \wedge F_2}$

If we could prove that I_1 *leads to* F_1, I_2 *leads to* ($F_{2.1} \wedge F_{2.2} \wedge F_{2.4}$) and $I_2 \wedge F_{1.2} \Rightarrow F_{2.3}$, then "Invariant ($I_1 \wedge I_2 \wedge I_3$) *leads to* $F_1 \wedge F_2$" would hold too.

First, let us prove "I_1 *leads to* F_1".

Because F_1 is equal to ($I_1 \wedge (\forall v_i \in (V_h - \{H_0\})$: $D(v_i) = MIN(H_0, vi)$)) and I_1 is an invariant, we just need to prove that "I_1 *leads to* ($\forall v_i \in (V_h - \{H_0\})$: $D(v_i) = MIN(H_0, vi)$)".

This is proved by induction on $MIN(H_0, v_i)$.

Base: when $MIN(H_0, v_i) = 0$, v_i is the big node H_0. It is trivially true the $D(H_0) = 0$.

Hypothesis: the claim holds when $MIN(H_0, v_i) = d$

Induction: when $MIN(H_0, v_i) = (d+1)$

We only need to prove that for any head v_i, if $MIN(H_0, v_i) = (d+1)$, then $D(v_i) = (d+1)$. From HEAD_ORG_RESP and HEAD_INTER_CELL, a head v_i will choose the a head with lowest distance to H_0 as parent head, by listening to their Head_org or Head_inter_alive message. For any head v_i, if $MIN(H_0, v_i) = (d+1)$, then the closest neighboring head j must have a $MIN(H_0, j)$ value of d, which also means that D(j) = d by the hypothesis. Thus $D(v_i)$ must be (d+1) since j is v_i's parent head.

Then, let us prove "I_2 *leads to* ($F_{2.1} \wedge F_{2.2} \wedge F_{2.4}$)".

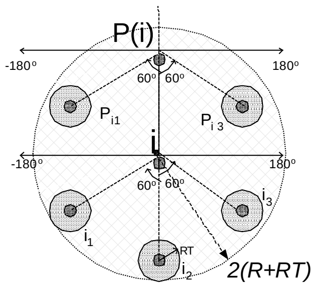
Let's first prove that I_2 leads to $F_{2.1} \wedge F_{2.2}$. Because of SYN_CELL, all heads in the system will have the same <CIC, ICP> value at fix point (stable system state), that is, I_2 leads to ($\forall$ head i ($\forall j \in$ neighboring_heads(i): (<CIC(i),ICP(i)> = <CIC(j),ICP(j)>). At the same time ($I_{2.1} \wedge I_{2.2} \wedge$ ($\forall$ head i ($\forall j \in$ neighboring_heads(i): (<CIC(i),ICP(i)> = <CIC(j),ICP(j)>)) $\Rightarrow F_{2.1} \wedge F_{2.2}$, and I_2 is an invariant. So "I_2 *leads to* $F_{2.1} \wedge F_{2.2}$" naturally holds due to transitivity of *leads to* operation.

Now let's prove I_2 leads to $F_{2.4}$. We achieve this by proving $I_2 \Rightarrow F_{2.4}$. Because $I_{2.4} \equiv F_{2.4} \wedge$ (There is no R_t -radius gap in the system $\Rightarrow R' \leq R+\frac{2}{\sqrt{3}}R_t$), we only need to prove that $I_1 \wedge I_2 \wedge I_3 \Rightarrow$ (There is no R_t -radius gap in the system $\Rightarrow R' \leq R+\frac{2}{\sqrt{3}}R_t$). According to the way HEAD_ORG works, the boundary cell would be no bigger than the inner cell, if there is no R_t -radius gap. Otherwise, the HEAD_ORG process will be continuously initiated. Thus the boundary cell's radius is still no more than $(R+\frac{2}{\sqrt{3}}R_t)$ according to $I_{2.4}$ that says any inner cell's radius is no more than $(R+\frac{2}{\sqrt{3}}R_t)$.

Finally, let us prove "$I_2 \wedge F_{1.2} \Rightarrow F_{2.3}$".

Since $F_{2.3} = (I_{2.3} \wedge (\forall \text{ head } i: |\ CH(i)\ | \leq 3\ ))$, we only need to prove that "$I_2 \wedge F_{1.2} \Rightarrow (\forall \text{ head } i: |\ CH(i)\ | \leq 3)$".

Let us consider any head (not big node) i without loss of generality. By I_2, there could be at most 6 neighboring heads around 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 also under the care of P(i) as shown in the picture below.



By $F_{1.2}$, p_{i1}, p_{i3} certainly will not choose head i as parent because of the existence of head P(i) that is closer to H_0 than head i. It is also trivially true that P(i) cannot be the child of head i. Therefore, there could be at most three heads (i_1, i_2, i3) that could serve as head i's children heads. Thus $I_2 \wedge F_{1.2} \Rightarrow (\forall$ head i: $| CH(i) | \leq 3)$.

c) Invariant ($\mathbf{I_1 \wedge I_2 \wedge I_3}$) *leads to* $\mathbf{F_4}$

We only need to prove that $I_1 \wedge I_2 \wedge I_3 \wedge \overline{F_4}$ leads to F_4, since it is obvious that $I_1 \wedge I_2 \wedge I_3 \wedge F_4$ leads to F_4. We need to prove that "$I_1 \wedge I_2 \wedge I_3 \wedge \overline{F_4}$ leads to F_4" holds both in the initial phase of head organization diffusion and in later new node joins.

In the case of new node j joining the system, SMALL_NODE_BOOT_UP and SMALL_ NODE_JOIN guarantee that node j will find an existing node as head or surrogate head if it is not completely disconnected from the system, i.e. being a visible node. Thus F4 will be reached within $\theta(C)$ amount of time, where C is a small constant related to the purely one-hop message exchange.

In the initial phase of head organization process, since the way HEAD_ORG works guarantees that all the visible areas of the system could be covered by the HEAD_ORG process in the end (see the proof of this claim later), there will be a HEAD_ORG process waiting to take place whenever $\overline{F_4}$ holds. Because the system's coverage is finite and every HEAD_ORG process is able to cover another ($\sqrt{3}$ R+2R_t)-radius circular area, the number of possible HEAD_ORG process occurrence is finite. Therefore, "$I_1 \wedge I_2 \wedge I_3 \wedge \overline{F_4}$ leads to F_4" holds.

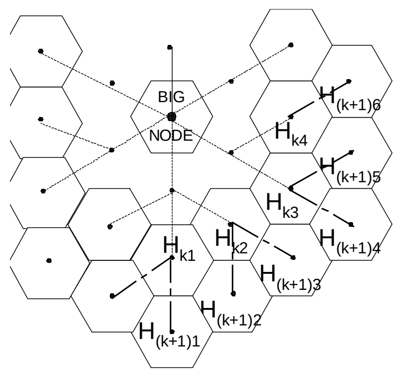
Now, let us prove that all the visible areas of the system could be covered by the HEAD_ORG process in the end. We prove it by induction on the area encircled by heads of i-band away from the big node, denoted Round Area *RA(i)* (i.e. area of radius ($\sqrt{3}$ R × i + R_t + R) and i is the number of hexagons away from the big node).

Case one: when there is no R_t circular region gap in node deployment

Base: when i = 0, 1, clearly holds

Hypothesis: the claim holds when i = k

Induction: when i = (k+1),



As we could see from the picture, any point that is in RA(k+1) but not in RA(k) will be covered by some (k+1)-band head. And each (k+1)-band head could be taken care of by some k-band head, either directly or indirectly, even though some of them might not be generated directly by a k-band head due to different progress speeds of the self-configuration process at different directions spreading from the big node. Thus the claim holds when i is (k+1).

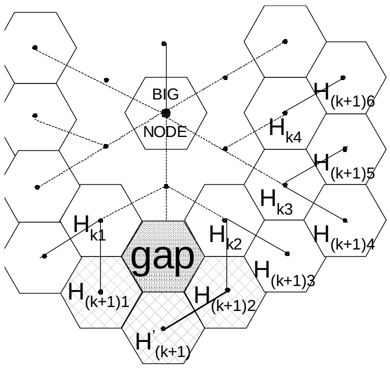
Case two: when there is some R_t circular region gap in node deployment

Base: when i = 0, 1, clearly holds

Hypothesis: the claim holds when i = k

Induction: when i = (k+1),

When there is only one R_t circular region gap at k-band, we could see that the three next-band heads, i.e. $H_{(k+1)1}$, $H_{(k+1)2}$ and $H'_{(k+1)}$, could be taken care of by the help of the two k-band heads that are neighboring the gap. Thus the claim still holds in this case.



When there are multiple R_t circular region gaps at k-band, we could see the same results just in the same way as above, and no matter whether these gaps are contiguous or not. For simplicity reason, we do not detail it here.

According to the proof in a), b), c), and d), the claim that Invariant ($I_1 \wedge I_2 \wedge I_3$) leads to the fix point ($F_1 \wedge F_2 \wedge F_3 \wedge F_4$) holds.

❑