An Underlay for Sensor Networks: Localized Protocols for Maintenance and Usage

Christo F. Devaraj
University of Illinois Urbana Champaign
Department of Computer Science
Urbana, IL 61801-2302
chdevara@microsoft.com

Mehwish Nagda
University of Illinois Urbana Champaign
Department of Computer Science
Urbana, IL 61801-2302
nagda@engineering.uiuc.edu

Indranil Gupta
University of Illinois Urbana Champaign
Department of Computer Science
Urbana, IL 61801-2302
indy@cs.uiuc.edu

Gul A. Agha
University of Illinois Urbana Champaign
Department of Computer Science
Urbana, IL 61801-2302
agha@cs.uiuc.edu

Abstract

We propose localized and decentralized protocols to construct and maintain an underlay for sensor networks. An underlay lies in between overlay operations (e.g., data indexing, multicast, etc.) and the sensor network itself. Specifically, an underlay bridges the gap between (a) the unreliability of sensor nodes and communication and availability of only approximate location knowledge, and (b) the maintenance of a virtual geography-based naming structure that is required by several overlay operations. Our underlay creates a coarse naming scheme based on approximate location knowledge, and then maintains it in an efficient and scalable manner. The underlay naming can be used to specify arbitrary regions. The overlay operations that can be executed on the underlay include routing, aggregation, multicast, data indexing, etc. These overlay operations could be region-based. The proposed underlay maintenance protocols are robust, localized (hence scalable), energy and message efficient, have low convergence times, and provide tuning knobs to trade convergence time with overhead and with underlay uniformity. The maintenance protocols are mathematically analyzed by characterizing them as differential equation systems. We present microbenchmark results from a NesC implementation, and results from a large-scale simulation of a Java implementation. The latter experiments also show how routing using the underlay would perform.[1]

1 Introduction

Wireless Sensor networking (henceforth simply *sensor networks*) applications in the future are likely to be supported by networking substrates. These substrates will provide services such as multicast, routing, data indexing (e.g., GHT [2], DIM [1]), aggregation [3]), etc. These protocols can be termed as *overlay* protocols, since they are all operations executed on the scale of the entire system (or parts of it) rather than at the level of individual sensor nodes [7] [2]. The main requirements from these overlay services are reliability, scalability, and energy-efficiency.

However, bridging the gap between the requirements of overlays on the one hand, and the inherent unreliability of sensor nodes and communication, as well availability of only approximate location knowledge (e.g., from GPS or localization algorithms) on the other hand, remains a challenge. For example, overlays for multidimensional range querying such as DIM and GHT [1, 2] require the network to be organized into a hierarchical structure (not necessarily just a spanning tree). Maintaining such a hierarchical structure that *underlies* the overlays has remained a difficult problem. Also, specifying arbitrary regions in the sensor network, and executing overlay operations on them (e.g., multicast, routing) requires a coarse naming scheme for the network that is only based on approximate location knowledge of sensor nodes.

In this paper, we propose protocols to maintain an **underlay** scheme that can be used to bridge the above-mentioned gap, while not compromising the reliability, scalability and energy-efficiency that the overlay operations seek to provide to the application. The underlay is called the Grid Box Hierarchy (GBH), and although the structure was proposed in [5], creation and maintenance of the underlay is an entirely different problem that was not addressed. In this paper, we focus on protocols for maintaining the GBH underlay, and evaluate their impact for a region-based routing protocol and a region-based multicast protocol. Aggregation using GBH was the focus of [5]. Supporting indexing schemes such as GHT and DIM over the GBH underlay is simple, and an evaluation is omitted due to space constraints.

The paper is organized as follows. Section 2 gives an overview of the GBH and the overview of our protocols. Section 3 presents two protocols to construct and maintain the grid

[1] This work was supported in part by NSF CAREER grant CNS-0448246 and in part by NSF ITR grant CMS-0427089

[2] This terminology is analogous to Internetwork-based overlays such as RON [4] and peer to peer systems.

box hierarchy. Section 4 presents two naming algorithms to name the grid boxes constructed. Section 5 analyzes the decentralized maintenance protocol. Section 6 describes an example overlay operation (routing) using the GBH. Section 7 presents our experimental results. Section 8 discusses related work. Section 9 concludes the paper.

2 Background

GBH Overview: The abstract structure of the Grid Box Hierarchy (GBH) is as follows [5]. The GBH for a sensor network of N sensor nodes consists of N/K *grid boxes*, each box containing an equal number of sensor nodes (K). K is a constant integer that is independent of N. Each grid box is assigned a unique ($log_K N - 1$) digit address in base K (i.e., each digit is an integer between 0 and $K - 1$). All these grid boxes lie only at the leaves of the virtual hierarchy. For all $1 \leq i \leq log_K N$, subtrees of height i in the hierarchy contain the set of grid boxes (actually, the sensor nodes inside them) whose addresses match in the most significant $(log_K N - i)$ digits - this is used to name the internal node of the GBH with a series of wildcards at the end.

For sensor networks, we require that (1) sensor nodes within each given grid box are physically proximate, and (2) each pair of grid boxes with close-by names, are physically proximate. Indeed, these conditions are loosely stated because this turns out to be sufficient for the overlay operations we are interested in. Condition (2) implies that the smaller the integer difference between two grid boxes, the closer they are in physical space. Internal nodes in the GBH now correspond to physical regions, and condition (2) implies that physical proximity also extends to regions spanned by internal nodes of GBH. Such a *GBH underlay* provides us a basis for building several important overlay operations. By virtue of the name-physical proximity relation (conditions (1) and (2)), if these overlay operations are designed in a manner that respects the hierarchy of the GBH, they will also be efficient in terms of actual message overhead within the wireless ad-hoc sensor network. Examples of overlay operations include anycasting, multicasting to a group of nodes and data aggregation as described in [5]. We discuss regions, routing and multicast operations using GBH in Section 6.

Creating and Maintaining the GBH Underlay: We study protocols to create and maintain the GBH underlay. Specifically, we wish to assign each sensor node to a grid box so that all grid boxes contain an equal number of nodes, and conditions (1) and (2) for the relation between name physical proximity is attempted to be maintained.

There are two components to our protocols: (a) *Balancing* protocols that ensure the balance of sensor nodes across grid boxes, and (b) *Naming algorithms* for maintaing conditions (1) and (2) above. The input to the creation algorithms

The GBH creation protocols take as input an approximate location for each node (obtained through a localization service) or GPS. These are used by the naming algorithm to assign names to grid boxes [3]. The balancing algorithm then takes over, and ensures that the grid boxes balance out. The balancing algorithm continues to run throughout the lifetime of the network and in fact constitutes the maintenance protocol.

The creation and maintenance protocols are required to be localized, energy-efficient, self-reorganizing and robust against node failures and rebirths.

3 Diffusion Based Balancing

In this section, we propose two new diffusion based balancing protocols for maintaining the GBH. These algorithms are similar to algorithms used for load balancing in multiprocessors. Sensor nodes transfer in between grid boxes (note that this is not physical movement) in order to restore balance.

3.1 Leader Based Diffusion

3.1.1 Direct Neighborhood (DN) Diffusion

This variant is based on leader-election. Figure 1 shows the pseudocode. The next section describes a decentralized variant. Each grid box G_i has a leader node L_i. L_i maintains a list of its grid box members, as well as a list of neighbor grid boxes, their leaders and their sizes. Every T_b time units, L_i checks its neighbor grid box sizes and picks a neighboring grid box G_j with maximum size difference and sends L_j a balancing request. If the request is accepted, then the leader of the larger box initiates a transfer of an appropriate number of nodes. The set of nodes transferred may include the leader L_i itself; however this node stays a leader for G_i; leaders do not move very far from their grid boxes since grid boxes do not ”move” large physical distances. Stale grid box size information in these messages does not cause inconsistency since each leader is participating in at most one transfer at a time. The communication between the leaders can be done through TTL-restricted flooding since they are likely to be close by. The nodes to be transferred may be chosen from among those that are close to the boundary of S_i and S_j, or those that are close to the centroid of G_j (this information is sent by L_j), or those that add maximum number of edges to G_j.

3.1.2 Average Neighborhood (AN) Diffusion

AN is an extension of DN whereby each grid box balances with more than one neighbor. Up to m neighbors may be used, where m varies from 1 to all neighbors. The AN algorithm uses the DN algorithm, where the m neighbors with highest differences are chosen, and are used for balancing. Implementation details are omitted since they are similar to DN in, and use $M_{req\_balance}$, $M_{accept\_req\_balance}$, and $M_{reject\_req\_balance}$ messages.

Failure of the leaders in these schemes can hamper the convergence properties of the protocol. This motivates decentralized schemes that do not reply on leaders. We discuss these schemes in the next section.

3.2 Decentralized Probabilistic Diffusion

The pseudocode for the Decentralized Probabilistic Diffusion Balancing protocol is shown in Figure 2. We explain the

[3] Some grid boxes may be nonexistent if the distribution of nodes is highly non-uniform - this simply results in an increase in the value of K in the distribution of nodes among the grid boxes.

```
Require: $L_g$ is node address, $g$ is initial GB address, $time_T = T_b$ and $state = DOING\_NOTHING$
  loop
    if $time > time_T \wedge state = DOING\_NOTHING$ then
      choose neighboring grid box $g'$ with maximum $|size(g) - size(g')|$ greater than 1
      send $M_{req\_balance}(g, L_g)$ to destination leader $L_{g'}$
      $state \leftarrow SENT\_REQUEST$
    end if
    receive $msg$
    if $msg = M_{req\_balance}(g', L_{g'})$ then
      if $state = DOING\_NOTHING$ then
        send $M_{accept\_req\_balance}(g, L_g)$ to $L'_g$
        $state \leftarrow BALANCING$
        if $size(g) > size(g')$ then
          choose $S$, a set of $\lfloor \frac{size(g)-size(g')}{2} \rfloor$ nodes in grid box $g$
          inform $S$ that their new grid box is $g'$
          inform $L'_g$ to add $S$ to grid box $g'$
          broadcast new size information to neighboring grid boxes
          $state \leftarrow DOING\_NOTHING$
        end if
      else
        send $M_{reject\_req\_balance}(g, L_g)$ to $L'_g$
      end if
    end if
    if $msg = M_{reject\_req\_balance}$ then
      $state \leftarrow DOING\_NOTHING$
      $time_T \leftarrow time + T_b$
    end if
    if $msg = M_{accept\_req\_balance}$ then
      $state \leftarrow BALANCING$
      if $size(g) > size(g')$ then
        choose $S$, a set of $\lfloor \frac{size(g)-size(g')}{2} \rfloor$ nodes in grid box $g$
        inform $S$ that their new grid box is $g'$
        inform $L'_g$ to add $S$ to grid box $g'$
        broadcast new size information to neighboring grid boxes
        $state \leftarrow DOING\_NOTHING$
      end if
      $time_T \leftarrow time + T_b$
    end if
  end loop
```

Figure 1. Direct Neighborhood Diffusion (DN).

```
Require: $i$ is node address, $g$ is initial GB address, $GS$ is the initial member set of $g$, $time_T = T_b$ and $N$ is set of known neighboring nodes and their grid box sizes
  loop
    if $time > time_T$ then
      choose $(j, g', GS')$ from $N$ with minimum $|GS'|$.
      if $|GS| > |GS'| + 1$ then
        if $rand < p$ then
          $GS \leftarrow GS' \cup i$
          flood $M_{entering\_gb}(g', i)$ and $M_{leaving\_gb}(g, i)$
          $g \leftarrow g'$
          broadcast $M_{my\_gb}(i, g, GS)$
        end if
      end if
      $time_T \leftarrow time + T_b$
    end if
    receive $msg$
    if msg = $M_{my\_gb}(j, g', GS')$ then
      update $N$ to contain neighbor $j$, its grid box address $g'$ and its member set $GS'$
    end if
    if msg = $M_{entering\_gb}(g, j)$ then
      $GS \leftarrow GS \cup \{j\}$
      broadcast $M_{my\_gb}(i, g, GS)$
    end if
    if msg = $M_{leaving\_gb}(g, j)$ then
      $GS \leftarrow GS - \{j\}$
      broadcast $M_{my\_gb}(i, g, GS)$
    end if
  end loop
```

Figure 2. Balancing through Decentralized Probabilistic Diffusion.

protocol below. Each sensor node initially knows its approximate grid box address based on its approximate location and by using a *naming algorithm* (described in Section 4). It then starts to maintain the current membership GS_i of its grid box G_i. This is achieved by having a newly joining node TTL-flood an $M_{entering\_gb}$ message, and receiving nodes in the grid box include this new node in their membership lists. Next we explain how this is maintained. After initialization is completed (specified by a timeout), each sensor node participates in the balancing protocol. Sensor nodes on the periphery of their grid boxes (those with neighbors in a different grid box) announce any changes in their grid box name and membership size to their neighbors through a $M_{my\_grid\_box}(G_i, GS_i)$ message, which are recorded at the recipients. Every T_b time units, sensor s_j selects a neighbor s_i such that $G_j \neq G_i$ and $|GS_j| > |GS_i| + 1$. Then, with probability P_T, s_j transfers itself from G_j to G_i. Nodes entering or leaving a grid box announce this by TTL-flooding $M_{entering\_gb}$ and $M_{leaving\_gb}$ messages respectively. The probabilistic choice P_T prevents migrations of large numbers of sensor nodes. This scheme should be chosen so as to minimize oscillations and assure convergence and a stable solution. We discuss different ways of setting this probability later in this section.

Maintenance: At regular intervals, every sensor floods (with a TTL enough to reach its grid box) a $M_{presence\_update}$ heartbeat message. Each entry in GS_i at s_i has a time to live which is initialized to slightly higher than the heartbeat interval. Entries time out if heartbeats are not received, thus gracefully removing failed nodes from the grid box and the system itself. Any such change will result in a subsequent balancing movement.

When a new sensor node joins (or rejoins) the network, it requests its neighbors for their grid box numbers. It chooses one of them and joins that box. The distribution is balanced out then by the balancing protocol. We discuss different ways of setting the probability P_T that determines the rate at which nodes move across neighboring grid boxes. The first technique is to use a constant probability. P_T is set to a constant value of $\frac{p}{|GS_i|}$. Choosing the right value for p is crucial to the protocol's success. A high value for p could result in a large number of nodes in the periphery of a grid box transferring to a neighboring grid box. If this movement is large enough to alter the order of grid box sizes, this could cause node oscillations between grid boxes. On the other hand, a very low value for p will result in slow convergence. The second technique is to weight a constant probability with an 'imbalance' factor. The probability P_T is set as $\frac{p\delta}{|GS_i|}$, where δ is the size of the imbalance (difference in number of nodes between grid box sizes). This ensures higher imbalances are balanced out faster (due to the higher probability of doing so).

The TTL-flooding used to spread information within grid boxes and across neighboring grid boxes can be replaced by a tree-based dissemination protocol to spread these updates in a more message- and energy-efficient manner. The basic idea involves each grid box maintaining a spanning tree containing all its nodes, as well as a few nodes from neighboring grid boxes. We omit description of the tree building/maintenance protocol due to its simplicity.

4 Naming Algorithms

A *naming algorithm* assigns an initial grid box address to a sensor node based on its knowledge of its approximate geographic location. For simplicity, we assume that all sensor nodes know the layout of the entire area, and this area is rectangular.

Let X represent the length of the area and Y represent the width. Assume that $n = N/K$ is a power of K. Consider two cases depending on whether n is an even or odd power of K.

- If $n = K^{2r}$, split the area into rectangular parts such that there are K^r boxes on each side. Each box is of length $\frac{X}{K^r}$ and width $\frac{Y}{K^r}$.
- If $n = K^{2r+1}$, split the area into rectangular parts such that there are K^r boxes on the smaller side and K^{r+1} on the larger side. If $X \geq Y$, each box is of length $\frac{X}{K^{r+1}}$ and width $\frac{Y}{K^r}$.

Let x represent the length of the system area in terms of boxes and y represent the width of the system area in terms of boxes. We model the naming scheme as a function f_{xy} that takes a grid box G_{ij} (where i and j represent the grid box's position along X and Y axes) and assigns it a number in base K. Two intuitive schemes are stated below,

Linear: In this scheme $f_{xy}(G_{ij}) = j \times x + i$. In other words, the boxes are numbered rowwise.

Recursive: Assume that we have to name K^n boxes. Split the area into $K \times K$ big boxes each of which has to house K^{n-2} grid boxes. Now number the big boxes in rowwise order from 0 to $K^2 - 1$. This needs 2 digits in base K and will act as prefix for the names of all boxes inside each big box. Now recursively split, name and add the prefixes. A simple analysis below shows that the recursive scheme produces clusters that are more squarish than that produced by the linear scheme. This results in better proximity between nodes in the same cluster.

First consider the case when $n = N/K = K^{2r}$, an even power of K. Consider the subtree (in GBH) comprising of boxes that match in the t most significant digits. Let us compute the squareness of this level (we call this the t-level for simplicity) for both schemes. For the linear scheme, we need to consider two cases $t < r$ and $t \geq r$. If $t \geq r$, then the t-level is a rectangle of size $K^{2r-t} \times 1$. If $t < r$, then the t-level is a rectangle of size $K^r \times K^{r-t}$. Consider the recursive scheme. We need to consider two cases: t is even and t is odd. If t is even, then the t-level is going to be the same as the 0-level of a hierarchy with K^{2r-t} boxes. This is of size $K^{r-\frac{t}{2}} \times K^{r-\frac{t}{2}}$. If t is odd, then the t-level is going to be the same as a linear arrangement of the 0-levels of K hierarchies with $K^{2r-(t+1)}$ boxes. This is of size $K^{r-\frac{t-1}{2}} \times K^{r-\frac{t+1}{2}}$. Let us define squareness as the ratio of the smaller side to the larger side. The closer it is to 1, the better it is. The linear scheme has squareness $\frac{1}{K^{2r-t}}$ if $t \geq r$ and $\frac{1}{K^t}$ when $t < r$. The recursive scheme has squareness 1 if t is even and $\frac{1}{K}$ if t is odd. It is easy to see that recursive scheme achieves much better squareness. It can be similarly shown for the case when n is an odd power of K.

5 Analysis

In this section, we analyze the decentralized probabilistic balancing protocol with linear probabilities of transfer.

Let us consider a grid box system with $N \times N$ regular grid boxes. The analysis can be easily extended to a system where the sides are not equal. Let the grid boxes be numbered in a rowwise fashion and let s_i represent the size (in number of nodes) of grid box G_i. Assume G_i and G_j are boxes that share a common side. The two boxes can diffuse nodes between them if $s_i \neq s_j$. Assume w.l.o.g that $s_i > s_j$. Now the probability of transfer for a node on the boundary of the two boxes but on the G_i side is given by $\frac{p}{s_i}(s_i - s_j)$. If f is the fraction of s_i that form the boundary, then the overall rate of transfer of nodes from G_i to G_j is given by $f \times s_i \times \frac{p}{s_i}(s_i - s_j)$ which evaluates to $f \times p(s_i - s_j)$. Let us w.l.o.g assume $f = 1$ which results in a node transfer rate of $p(s_i - s_j)$ from G_i to G_j.

Now, we figure out the total rate of transfer out of G_i by considering all 4 neighboring boxes (which may be 3 or 2 depending on edge/corner cases). This can be represented as $-\frac{ds_i}{dt} = p(s_i - s_{i-1}) + p(s_i - s_{i+1}) + p(s_i - s_{i-N}) + p(s_i - s_{i+N})$. More concisely, $\frac{ds_i}{dt} = p(s_{i-1} + s_{i+1} + s_{i-N} + s_{i+N} - 4s_i)$. Let NG_i represent the set of neighboring grid boxes of grid box G_i. Then the equation for rate of change of s_i can be given as,

$$\frac{ds_i}{dt} = p\{(\sum_{G_j \in NG_i} s_j) - |NG_i| \times s_i\} \tag{1}$$

We prove convergence properties of this system below. All theory behind the proofs can be found in [27]. Consider a general grid box system with $N \times N$ grid boxes whose sizes vary as given by the equations,

$$\frac{ds_i}{dt} = p\{(\sum_{G_j \in NG_i} s_j) - |NG_i| \times s_i\} \tag{2}$$

As mentioned in the previous section, this system of differential equations can be concisely represented by $\dot{\mathbf{s}} = \mathbf{A}\mathbf{s}$, where $\mathbf{A}$ is the coefficients matrix of equations 2.

Lemma 5.1. *$\mathbf{A}$ has real eigenvalues.*

Proof: It is known that a symmetric real matrix has real eigenvalues. We are done if we show that $\mathbf{A}$ is symmetric.

Recall that $\mathbf{A}$ is the coefficients matrix corresponding to the N^2 equations in N^2 variables. Hence, A_{ij} is the coefficient of s_j in the equation for $\frac{ds_i}{dt}$. Looking at Equation 2 it is obvious that all coefficients outside of the main diagonal are either -1 or 0. More precisely when $i \neq j$, A_{ij} is 1 *iff* G_j is a neighbor of G_i and 0 otherwise. Thus $A_{ij} = A_{ji}$ which concludes the proof. □

Lemma 5.2. *0 is an eigenvalue of $\mathbf{A}$.*

Proof: ¿From Equation 2, we can see the sum of coefficients in each equation is 0. This means the each row of $\mathbf{A}$ sums to 0. This further implies that,

$$\mathbf{A}.\begin{pmatrix} 1 \\ 1 \\ \dots \\ 1 \end{pmatrix} = 0 = 0.\begin{pmatrix} 1 \\ 1 \\ \dots \\ 1 \end{pmatrix}$$

Therefore 0 is an eigenvalue of $\mathbf{A}$ with eigenvector $\begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}$. Hence proved. □

Defn: A symmetric real square matrix $\mathbf{A}$ is negative semidefinite if for any nonzero vector $\mathbf{x}$, we have $\mathbf{x}^T\mathbf{A}\mathbf{x} \leq 0$. □

Lemma 5.3. *$\mathbf{A}$ is a negative semidefinite matrix.*

Proof: We will first see how this is proved for the 2×2 system given in the previous section. This is in spite of actually solving the system to illustrate a proof technique.

$$\begin{aligned} &\mathbf{x}^T\mathbf{A}\mathbf{x} \\ &= \mathbf{x}^T p \begin{pmatrix} -2 & 1 & 1 & 0 \\ 1 & -2 & 0 & 1 \\ 1 & 0 & -2 & 1 \\ 0 & 1 & 1 & -2 \end{pmatrix} \mathbf{x} \\ &= 2p(x_0x_1 + x_0x_2 + x_1x_3 + x_2x_3 - \textstyle\sum_i x_i^2) \\ &= -p[(x_0 - x_1)^2 + (x_0 - x_2)^2 + (x_2 - x_3)^2 + (x_1 - x_3)^2] \\ &\leq 0 \end{aligned}$$

So intuitively, the proof will proceed by proving $\mathbf{x}^T\mathbf{A}\mathbf{x}$ can always be written as the negation of a sum of squares.

Let $a = \mathbf{x}^T\mathbf{A}\mathbf{x}$ for a general $\mathbf{A}$. Notice that $\mathbf{A}\mathbf{x}$ is a column vector with the i^{th} row being $\frac{dx_i}{dt}$. Therefore the coefficient of x_i^2 in a is $-|NG_i|$ from Equation 2. Similarly the coefficient of x_ix_j in a is 2 if G_i and G_j are neighbors and 0 otherwise. There are no other terms in a. Thus a can be represented as,

$$a = -\sum_i |NG_i| x_i^2 + 2 \sum_{N(i,j)=1} x_i x_j \qquad (3)$$

where $N(i, j)$ is 1 if G_i and G_j are neighbors and 0 otherwise.

Because each x_i takes part in a product of the form $2x_i x_j$ exactly $|NG_i|$ times, we can rewrite the above equation as $a = -\sum_{N(i,j)=1}(x_i - x_j)^2$ which means $a \leq 0$. Thus we have proved $\mathbf{A}$ is a negative semidefinite matrix. □

Theorem 5.4. *$\mathbf{A}$ has only non-positive real eigenvalues.*

Proof: Lemma 5.1 proved $\mathbf{A}$ has only real eigenvalues. A negative semidefinite matrix has all non-positive real eigenvalues and we proved in Lemma 5.3 that $\mathbf{A}$ is a negative semi definite matrix. Therefore $\mathbf{A}$ has only non-positive real eigenvalues. □

Theorem 5.5. *A general system converges to a state where each grid box has an equal size.*

Proof: Theorem 5.4 states all eigenvalues of $\mathbf{A}$ are negative or 0. In general, the solution to a system of the form $\dot{\mathbf{s}} = \mathbf{A}\mathbf{s}$ can be written as $\mathbf{s} = \sum_i c_i \mathbf{v}_i e^{\lambda_i t}$ where λ_i are the various eigenvalues and $\mathbf{v}_i$ is a corresponding set of linearly independent eigenvectors. In the case when such a basis of linearly independent eigenvectors cannot be found, the exponentials just get scaled by appropriately computed polynomials in t ([27]). Lemma 5.2 showed that 0 is an eigenvalue. Therefore the constants in s_i are all equal to c_0 which is equal to the average grid box size upon solving the initial value problem. All non-constant terms are negative exponentials (proved by Theorem 5.4). Therefore all s_i converge to c_0 as $t \longrightarrow \infty$. □

6 Overlay Operations: Regions, Routing and Multicast

We have used the GBH underlay to build routing and multicasting operations. We have also added the ability to define arbitrary regions.

```
Require: $nodeid$ is node address, $gridbox$ is initial grid box address in base $K$, $neighborgb$ is the neighbor grid box set array, $neighbor$ is the neighbor node array, $to$ is the target grid box, $from$ is the source grid box
  if $msg$ = $M_{routing\_msg}(to, from)$ then
    if $gridbox = to$ then
      for all neighbor grid boxes $nbg$ in $neighborgb$ do
        if $nbg.gridbox = gridbox$ then
          send $M_{routing\_msg}(nbg, to, from)$
        end if
      end for
    else
      $closest_ngb \leftarrow closest(neighborgb, gridbox, to)$
      for all neighboring nodes $ng$ in $neighbors$ do
        if $ng.gridbox = closest_ngb$ or $nb.gridbox = gridbox$ then
          send $M_{routing\_msg}(nb, to, from)$
        end if
      end for
    end if
  end if
```

Figure 4. Routing algorithm

```
Require: $nodeid$ is node address, $gridbox$ is node grid box address in base $K$, $target$ is target grid box address in base K, $neighborgb$ is the neighbor grid box set array, $odd \leftarrow TRUE$
  $to\_odd \leftarrow digits(target, odd)$
  $to\_even \leftarrow digits(target, \neg odd)$
  $mine\_odd \leftarrow digits(gridbox, odd)$
  $mine\_even \leftarrow digits(gridbox, \neg odd)$
  $closest\_dist \leftarrow sqrt((to\_odd - mine\_odd)^2 + (to\_even - mine\_even)^2)$
  $closest\_ngb \leftarrow nodeid$
  for all neighbor grid boxes $ngb$ in $neighborgb$ do
    $nb\_odd \leftarrow digits(ngb, odd)$
    $nb\_even \leftarrow digits(ngb, \neg odd)$
    $temp\_dist \leftarrow sqrt((to\_odd - nb\_odd)^2 + (to\_even - nb\_even)^2)$
    if $temp\_dist < closest\_dist$ then
      $closest\_ngb \leftarrow ngb$
      $closest\_dist \leftarrow temp\_dist$
    end if
  end for
  {Returns $closest\_ngb$}
```

Figure 5. Closest Algorithm to choose closest neighboring grid box to target grid box

```
Require: $gridbox$ is node grid box address in base $K$, $odd$ for odd numbered digits
  for all digits $i$ in $gridbox$ do
    if $odd$ and $i\%2 = 1$ then
      append $gridbox[i]$
    else if $\neg odd$ and $i\%2 = 0$ then
      append $gridbox[i]$
    end if
  end for
  {Returns $final$}
```

Figure 6. Digit Algorithm to extract even and odd digits of a grid box address)

Routing and Multicast: Figure 4 shows the routing pseudocode. It assumes that each grid box node maintains a set of grid box's neighboring grid boxes. A routing message is forwarded to neighbors in either its own grid box or the closest grid box chosen by the closest() function shown in figure 5. The closest function extracts even and odd numbered digits for each grid box address (neighbors, target and own) and uses these as coordinates to calculate the Euclidean distance between two grid boxes. The neighbor chosen to route the message to is the neighbor with the smallest distance from the target grid box. A region of sensors, specified either as a set of sensors (e.g., close to a given object) or as geographical region, can be mapped to an aggregated region address. The region is comprised of the set of grid boxes that contain at least one sensor node intersecting with the region specified. The region can then be specified using the collection of names of these grid boxes. Any subset of grid boxes from this can be aggregated if they comprise all grid boxes that are descendants of an internal node in the GBH. For example, (1000, 1001, 1010, 1011, 0100, 0101) can be rewritten as "10**+010*". A region-based multicast protocol will anycast to a region such as this and follow up by either flooding or tree-based or gossip-based multicast among all grid boxes within that region.

7 Simulation Results

We have simulated the above protocols with $N = 512$, $K = 8$ which implies that there are 64 grid boxes. The area of simulation is 10×10 and the radio range is 1.0. We are assuming each node knows its location and thus knows which grid box it is in. Results for protocol performance under approximate locations are also studied. The simulation proceeds in rounds. During each round, all messages intended for each node are delivered and the node takes actions and sends messages which get delivered in the next round. Note that though we use round numbers to stand for running time of the protocol, the protocols proposed do not need time synchronization.

7.1 Leader-Based Diffusion

Variance in Grid Box Sizes: Figure 3(a) shows the variance in grid box sizes from K as the protocols (DN and AN) proceed. It can be seen that both protocols rapidly decrease the variance as time proceeds. AN uses a larger neighborhood information

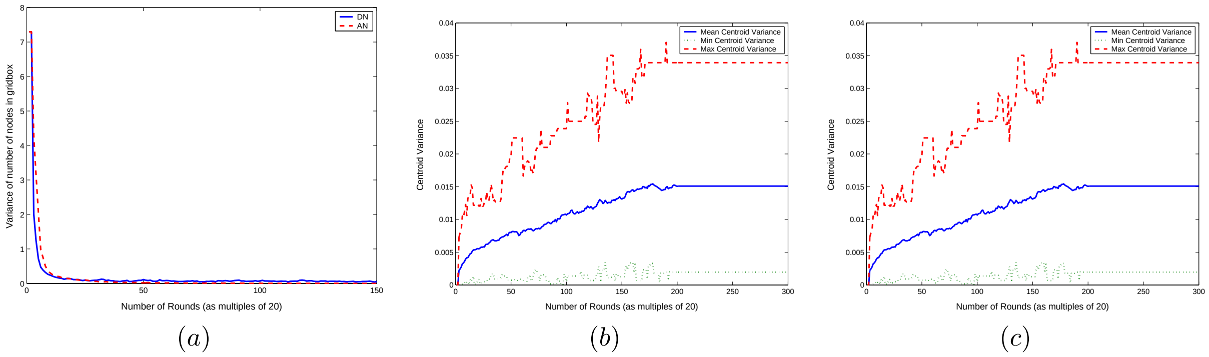

Figure 3. (a) Variance in grid box sizes vs. round number for DN and AN (b) Distance of final grid box centroids from the initial centroid positions for DN (c) Distance of final grid box centroids from the initial centroid positions for AN

than DN and converges faster. Note that here AN uses 4 neighbor (2 and 3 for edge and corner cases respectively) grid box information.

Movement of Grid Box Centroids: Figures 3(b) and 3(c) show movement of grid box centroids when compared to their initial positions. Three curves for maximum, average and minimum movement show that grid box movement is very small. Comparing Figures 3(b) and 3(c), we see that centroid movement is smaller in AN when compared to DN due to larger neighborhood information. Small grid box movement means lesser skewing of the initial grid box structure (based on locations) imposed on the system. This is very important for the naming scheme that was initially used on the grid boxes to be useful when the system reaches a balanced state.

7.2 Decentralized Probabilistic Diffusion

Grid Box Size Variance: Figure 10(a) shows the variance in grid box sizes from K. The different curves in the figure are the variance curves for different constant probabilities of transfer. Probability experiments that decide transfers are done every $T = 30$ rounds.

Frequency of Node Transfers: Figure 10(b) shows the number of node transfers that happen until variance becomes less than 1.0 (a common objective). More node transfers happen when a higher probability is used due to node oscillations. Node transfers need to be informed across two grid boxes and hence consume energy. This gives a natural application-dependent tuning factor viz., a higher probability results in faster convergence but larger energy loss. For a constant probability of 0.1 which has really fast convergence, only 82 broadcasts per node is required. Note that about 60 of these broadcasts happen only at the start of the protocol when nodes flood to announce their presence in a grid box.

Movement of Grid Box Centroids: Figure 10(c) shows the movement in final grid box centroids with respect to the initial box centroids. The graph shows that a higher probability results in larger centroid movement. This is due to a higher number of transfers which implies a higher expected distance moved by centroids. Again, we see a tradeoff between convergence rate and protocol correctness. Hence, with an appropriate probability, a good final state can be reached. However, in most cases, this would be at the cost of the convergence rate.

Dispersion of Grid Box Nodes: Another parameter that is important is how close nodes within a grid box are to each other. This is shown in Figure 7(a) as the average area of the bounding box of each grid box. Node dispersion increases with probability of transfer since a higher number of transfers generally disperses the grid boxes more.

Linear Probability: Figure 7(b) shows the variance in grid box sizes with protocol rounds for different linear probability functions. Note that the curves are plotted for different p. The number of transfers and hence message complexity, grid box dispersion and centroid movement is decreased. We do not show explicit graphs due to lack of space.

The Gap of 1 Problem: In previous simulations, node transfers do not happen when the size difference between two grid boxes is 1, because then the situation would only reverse. Hence, there are grid boxes that differ in at most 1 from each of their neighboring boxes and this gradient slowly builds across the network. This is the cause for the base variance of 0.5 below which the previous graphs could not venture. We call this the *gap of 1* problem. Now, we allow a node transfer across a *gap of 1* boundary with a certain small probability. This results in the gradient disappearing and most grid boxes are of size K at steady state. These results are shown in Figure 7(c).

Naming Algorithms: We now measure how good the naming scheme is on top of the decentralized protocol. Since the recursive and linear naming schemes are the same when $N = 512, K = 8$, we obtain results using $N = 512, K = 2$. Figure 8 shows the performance of linear probability based transfer protocol using linear and recursive naming schemes respectively. The curve shows average distance between nodes having certain maximum common prefix length in grid box addresses. This is important in aggregation protocols as pointed out in [5] because nodes that share a common prefix take part at the same level of aggregation and hence require to be proximal to each other. It can be seen that recursive naming scheme not only achieves lower average distance between nodes sharing a

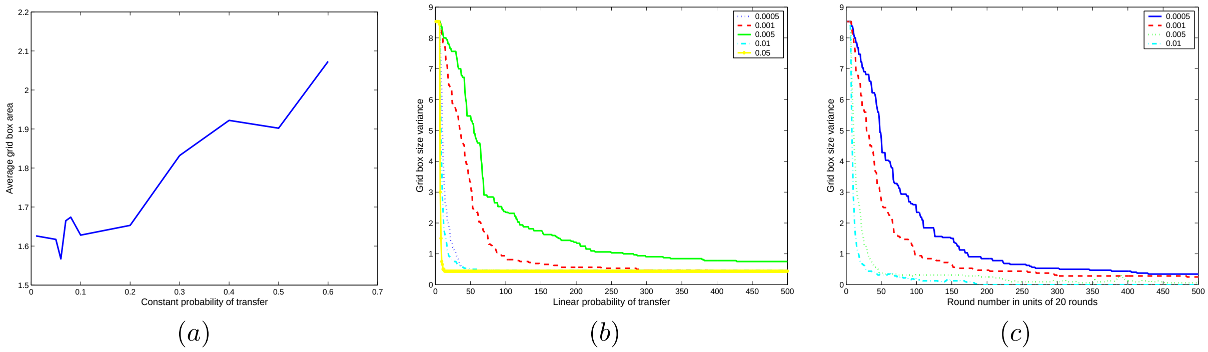

Figure 7. (a) Average final grid box area vs. constant probability of transfer (b) Variance in grid box sizes vs. round number for different linear probabilities of transfer (plotted for the various constants shown) (c) Variance in grid box sizes vs. round number for different linear probabilities of transfer (plotted for the various constants shown) with transfers across gaps of size 1

common prefix but also the lower maximum distance between such nodes.

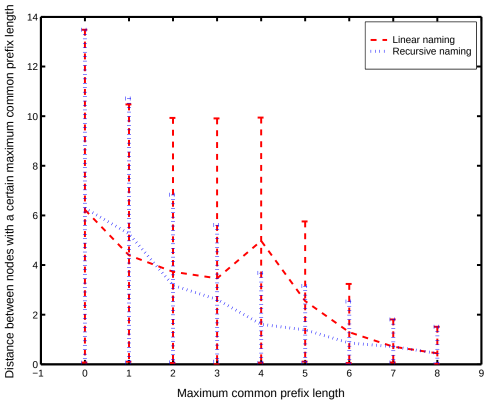

Figure 8. Average distance between nodes vs. length of maximum common prefix in grid box addresses for both naming scheme

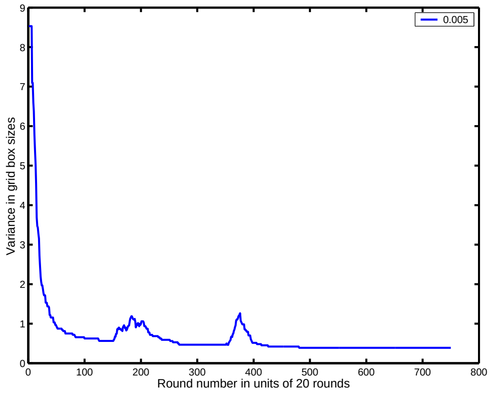

Figure 9. Variance in grid box sizes vs. round number for linear probability (0.005) of transfer under drastic node loss followed by node resurrection

Maintenance: Figure 9 shows the maintenance phase with node death between rounds 3000 and 4000 followed by resurrection of half of the dead nodes between rounds 7000 and 7500. Note that this is a drastic loss rate and results in about 100 sensors out of 512 being removed over a span of 1000 rounds and 50 added over 500 rounds. The variance initially goes up and then the maintenance mechanism kicks in stabilizing the system. During node losses (network failures), the behavior of the protocol is incorrect, but when the network returns to normal, the protocol stabilizes and returns to correct execution. Note that we have however ensured that network partitions do not occur. Testing resiliency of the protocol under network partitions is part of our future work.

Grid box Final State: For lack of space, we do not show figures that show the final grid box state. However, it was seen that some boxes are larger than necessary and there was considerable overlap. This might not good for the system because overlapping boxes implies inter box bandwidth contention and loss of locality for intra box operations. We then restrict transferrable nodes to be the set of nodes that are within 1.0 distance units from the destination box's centroid. This results in much better grid box shapes (smaller sizes and lesser overlap).

8 Related Work

Clustering algorithms have been proposed in ad hoc networks using the notion of a cluster-head. [18], ([19]), [20] and [21] are a few examples. The work of Corradi et all [23] investigates simple diffusion based policies for dynamic load balancing using only a local view of the system. This work forms the basis of the DN and AN algorithms presented in this paper. The problem of constructing the GBH is similar to a transportation problem. It has been shown in [24] that the transportation problem can be converted to the assignment problem. The resulting assignment problem can be solved in a distributed manner using auction algorithms.

9 Conclusion

Building and maintaining the GBH is a crucial step towards implementing hierarchical gossiping algorithms in wireless sensor networks. Further, this hierarchy can be used for geographic routing and geocasting. We have presented diffusion

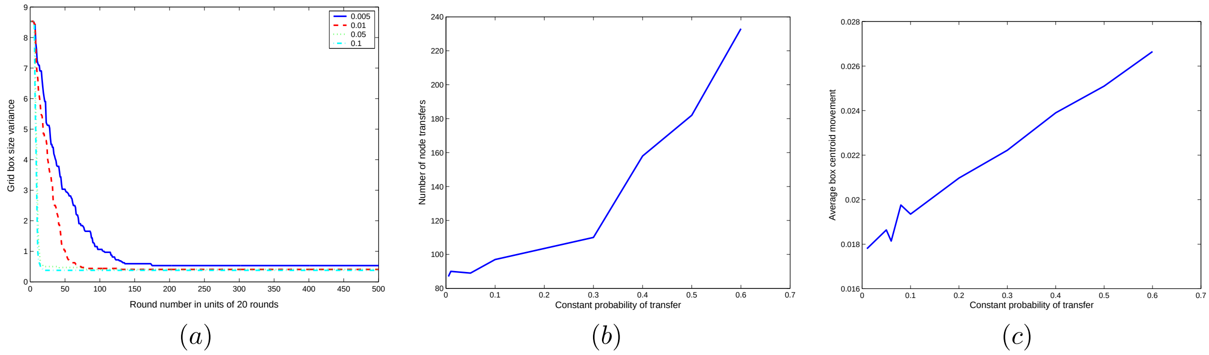

Figure 10. (a) Variance in grid box sizes vs. round number for different constant probabilities of transfer (b) Total number of node transfers in the system vs. constant probability of transfer (c) Distance of final grid box centroids from initial centroids vs. constant probability of transfer

based algorithms for constructing and continuously maintaining the GBH so that it is self-organizing and self-reconfiguring. In particular, we present two distinct approaches: one requiring a leader to be elected for each grid box and the other being completely decentralized relying on a probabilistic transfer function. However, the leader based approach is not fault tolerant and the probabilistic method stands out as a viable and efficient underlay self-assembly and self-reconfiguration protocol. Our results show that the **Diffusion Based Protocols** self-organize quickly and overcome the *gap of 1* problem. The recursive naming scheme achieves lower distances between nodes that share a higher common grid box address prefix length. In particular, the **Decentralized Probabilistic Diffusion Protocol** also recovers from node failures and node rebirths and stabilizes the variance in grid box sizes. Overall, it achieves a highly scalable, robust, energy efficient, application dependent manner of GBH self-organization and self-reconfiguration for wireless sensor networks.

References

[1] Xin Li, Young Jin Kim, Ramesh Govindan, “University of Southern California Wei Hong Multi-dimensional range queries in sensor networks”, Proceedings of the first international conference on Embedded networked sensor systems, Los Angeles, California, Pages: 63 - 75

[2] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, “GHT: A Geographic Hash Table for Data-Centric Storage in SensorNets”, In Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), Atlanta, Georgia, September 2002.

[3] Madden, Sam, Hellerstein, Joe, Hong, Wei, “TinyDB: In-Network Query Processing in TinyOS”, 10 Jan. 2003. 15 Jun. 2003

[4] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, Robert Morris, “Resilient Overlay Networks ”, Proc. 18th ACM SOSP, Banff, Canada, October 2001.

[5] I. Gupta, R. van Renesse, K. Birman, “Scalable Fault-Tolerant Aggregation in Large Process Groups”, The International Conference on Dependable Systems and Networks (DSN 01), Goteborg, Sweden, July, 2001

[6] I. Gupta, A. Kermarrec, A. Ganesh, “Efficient Epidemic-style protocols for Reliable and Scalable Multicast”, In 21st Symposium on Reliable Distributed Systems (SRDS 2002), pp. 180-189, October, 2002

[7] I. Gupta, K. Birman, “Holistic operations in large-scale sensor network systems: a probabilistic peer-to-peer approach”, Proc. International Workshop on Future Directions in Distributed Computing (FuDiCo), pp. 1-4, June, 2002

[8] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart and D. Terry, “Epidemic Algorithms for Replicated Database Maintenance, ” Proc. 6th ACM PODC,pages 1-12, August 1987

[9] D. Kempe and J. Kleinberg and A. Demers, “Spatial gossip and resource location protocols”, Proc. 33rd ACM STOC, July 2001, pages 163-172

[10] R. van Renesse, K. Birman, “Scalable management and data mining using Astrolabe”, Proc. 1st IPTPS, Mar 2002

[11] D. Estrin, R.Govindan, J. Heidemann, S. Kumar, “Next Century Challenges: Scalable Coordination in Sensor Networks”, Proc. 5th ACM/IEEE MobiCom, Aug 1999, pages 263-270

[12] L. Schwiebert, S.K.S. Gupta, J.Weinmann, “Research challenges in wireless networks of biomedical sensors”, Proc. 7th ACM/IEEE MobiComm, July 2001, pages 151-165

[13] D.C. Steere, A. Baptista, D. McNamee, C. Pu, J. Walpole , “Research challenges in environmental observation and forecasting systems”, Proc 6th ACM/IEEE MobiComm, Aug 2000, pages 292-299

[14] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris, “A scalable location service for geographic adhoc routing”, Proc. 6th Intnl. Conf. Mobile Computing and Networking, pages 120130, Aug 2000.

[15] J. Beal, “A Robust Amorphous Hierarchy from Persistent Nodes”, AI Memo 2003-012, April 2003.

[16] D. Coore, R. Nagpal, R. Weiss, “Paradigms for structure in an Amorphous Computer”, A.I. Memo No 1614, A.I. Laboratory, MIT, Oct 1997

[17] J. Beal, “Persistent Nodes for Reliable Memory in Geographically Local Networks”, MIT AI Memo 2003-011.

[18] S. Basagni, “Distributed and Mobility-Adaptive Clustering for Ad Hoc Networks”, Technical Report UTD/EE-02-98, July 98

[19] M. Gerla, J. T. C. Tsai, “Multicluster, Mobile, Multimedia Radio networks”, Wireless Networks, pp. 255 - 265, 1995

[20] O. Younis, S. Fahmy, “Distributed Clustering for Scalable, Long Lived Sensor Networks”, Proc 9th ACM/IEEE MobiComm 2003

[21] P. Basu, N. Khan, T. Little, “A Mobility based Metric for Clustering in Mobile Ad Hoc Networks”

[22] P. F. Tsuchiya, “The Landmark hierarchy: a new hierarchy for routing in very large networks”, Proc. Symp. Communications Architectures and Protocols, pages 3542, Aug 1988.

[23] A. Corradi, L. Leonardi, F. Zambonelli, “Diffusive Algorithm for Dynamic Load Balancing in Massively Parallel Architectures”, DEIS Technical Report No. DEISLIA -96-001, University of Bologna, April 1996.

[24] D. P. Bertsekas, “Auction algorithms for network flow problems: a tutorial introduction”, Comput. Optim. Appl., 1 (1992), pp. 7–66. 30 M. PATRIKSSON

[25] L. V. Kale, “Comparing the Performance of Two Dynamic Load Distribution Methods”, Proceedings of the International Conference on Parallel Processing, IEEE CS Press, 1988, 8-12.

[26] Y. Zou et al, “Sensor deployment and target localization based on virtual forces”, Infocom 2003

[27] Erwin Kreyszig, “Advanced Engineering Mathematics”, 8th edition, John Wiley and sons.