

# **STREAM: Sensor Topology Retrieval at Multiple Resolutions**

Budhaditya Deb, Sudeept Bhatnagar and Badri Nath<sup>\*</sup>

Computer Science Dept. Rutgers University.

{bdeb, sbhatnag, badri: @cs.rutgers.edu}

110 Frelinghuysen Road, Computer Science Dept. Piscataway, NJ-08854

Send Reprints to

Budhaditya Deb,

Computer Science Dept. Rutgers University

18 M ARI Drive

Somerset, New Jersey 08873

Phone Number: 732-398-9431

Email: bdeb@cs.rutgers.edu

## ABSTRACT

Large-scale sensor networks need energy-efficient mechanisms to extract topology for various aspects of sensor network management. Some network properties can be inferred from a relatively low-resolution representation of topology. Different topology resolutions suffice for different management applications to perform at a desired level. In these cases, it is an overkill to retrieve the entire topology of large-scale networks particularly because sensor nodes are energy constrained. In this paper, we describe a distributed parameterized algorithm for Sensor Topology Retrieval at Multiple Resolutions (*STREAM*), which makes a tradeoff between topology details and resources expended. We also define various classes of topology queries and rules for optimal parameter selection to support these queries.

***Key words: Sensor Networks, Multi-Resolution Topology, Spatial Sampling, Independent Dominating Sets, Network Management.***

## INTRODUCTION

Network Topology is an important attribute of the network state as it aids in network management and performance analysis. Accurate knowledge of network topology is a prerequisite to many critical network management tasks, including proactive and reactive resource management, server siting, event correlation, root cause analysis, growth characteristics and even for use in simulation for networking research.

In sensor networks (Kahn, Katz, Pister (1999)), where redundant, cheap nodes operating on battery, are used to form a large-scale dense network, energy-efficiency is the key criterion guiding the design of network protocols. Accordingly any topology discovery algorithm should also be energy-efficient. We show that different levels of partial topology suffice for determining different network properties. In fact, this observation opens up a new avenue for energy-efficient topology discovery where we can save a large amount of energy by aiming to retrieve only the partial topology without paying much in terms of inference of network characteristics.

In this paper, we describe a distributed parameterized algorithm for Sensor Topology Retrieval at Multiple Resolutions (*STREAM*) that trades between topology details and resources expended in large scale, dense networks. *STREAM* provides the flexibility to control the topology resolution and the overhead incurred in the process. The rest of the introduction section provides the motivation, main contributions and an overview of the algorithm.

### **The Need For Multi-Resolution Topology Discovery**

Consider the following observations about large-scale dense networks:

- We found that many topological properties in dense networks can be inferred accurately using a low-resolution representation of topology.

*For example, if the network administrator needs to test the robustness of the network by verifying that each node in the network has at least three disjoint paths (to a monitoring node), using *STREAM* she only needs to recover 17% of the edges (in a network of 1000 nodes with average degree 25)<sup>1</sup>. This results in reduction of 83% in energy consumed with respect to retrieving the complete network graph<sup>2</sup>.*

---

<sup>1</sup> Typically, Sensing Range  $\ll$  Communication range (E.g. in heat sensors and motion sensors). A network barely sufficient to provide sensing coverage could be sufficiently dense for providing communication coverage.

<sup>2</sup> Sensor network specific properties such as field exposure and coverage (Meguerdichian (2001a, 2001b)), also converge quickly towards the real values of the network even for at fairly low granularity of location information.

- Different topology resolutions are required for different applications to perform at a desired level.

*For example, estimating the overhead incurred in disseminating data to the monitoring node requires single source shortest path lengths (SSP). On the other hand, network-wide distribution of source coding symbols requires all pairs shortest paths lengths (ASP) (Scaglione, Servetto (2003)).*

*For SSP, on retrieving only 10% of the network edges (for a 1000 node network with average degree of 25) the average single source shortest path length increased by only 6% of the hops from the optimal. Similar bound (6% increase in hop length) for ASP required 25% of the network edges.*

- In some situations, network administrators may not be willing to spend more than a given amount of energy to retrieve certain network state. Given the energy budget we should be able to retrieve the maximum amount of information about the network. Note that the energy budget allotted to infer different network properties would also depend on the *importance* of the network property evaluated and how much the administrator is willing to tolerate in terms of the property degradation.

The above observations clearly highlight the need to have an ability to retrieve topology at multiple resolutions. STREAM provides us this ability.

### **Main Contributions**

The main contribution of this paper is a distributed parameterized Algorithm for **Sensor Topology Extraction at Multiple resolutions (STREAM)**. We introduce the notion of Minimal Virtual Independent Dominating Set (MVIDS), which is the minimal set of nodes required for extracting topology at a desired resolution. The MVIDS concept forms the basis for adaptive spatial sampling of the network. Retrieved network topology ranges from the backbone to the complete network graph.

The heuristics to approximate the MVIDS and its distributed implementation involves careful use of local timers. The timers are also responsible for creating an optimal topology aggregation tree and minimizing the message complexity of the process.

We also derive rules for optimal parameter selection under different constraints. We describe various types of topology discovery queries relevant for sensor networks and applications that can use these queries and use the rules to map the queries to the parameter values in STREAM.

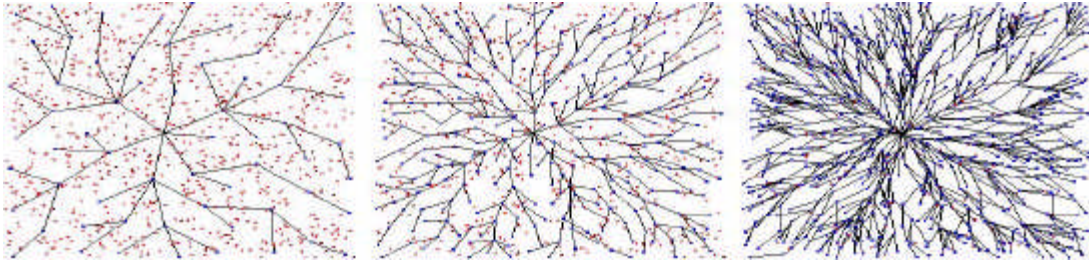


Figure 1: Topology of network with (1000 nodes in a 400x400 m<sup>2</sup> field, communication range 40m) retrieved at multiple resolutions. In all cases topology discovery is initiated at the center node. The nodes on the tree form the responding MVIDS. The effectiveness of STREAM in selecting the MVIDS is clearly depicted by the uniform distribution of the MVIDS across the sensor field.

### **Overview**

*STREAM* utilizes the broadcast property of wireless medium called the *Wireless Multicast Advantage* (Wieselthier, Nguyen, and Ephremides (2000)). A node can detect the presence of its neighbors by eavesdropping on the communication channel. Thus by selecting a subset of nodes, approximate topology can be created by merging their neighborhood lists. The resolution of the topology depends on the cardinality and structure of the chosen set of nodes. For example, to construct a minimal backbone tree of the network, we only need to merge the neighborhood lists of the minimum dominating set of the network graph.

*STREAM* runs in two stages. First a monitoring node, which requires the topology, sends a *topology discovery request* to all the nodes in the network by controlled flooding. The request contains two parameters called *virtual range* and *resolution factor*. These parameters are used to select a minimal set of nodes required to retrieve topology at a desired resolution. This set is termed as the *Minimal Virtual Independent Dominating Set (MVIDS)*. During this stage, the nodes are colored *red* or *black* such that each *red* node is a neighbor of some *black* node and the *black nodes* form the MVIDS. Further, at the end of the first phase, a *black* node tree rooted at the monitoring node is set up. In the second phase, the *black* nodes reply back to the request with a subset of its neighborhood list, determined by the resolution factor. Each black node aggregates the data received from its children black nodes and sends it to its parent in the tree. Figure 1 illustrates the results of applying *STREAM* to select multi-resolution MVIDS. The following factors make *STREAM* extremely suitable for sensor networks:

- The *MVIDS* is created using message complexity of  $N$  (number of nodes in the network), and compares well to a centralized scheme. Use of only one packet per node makes it highly energy efficient. The above is made possible by using local timer mechanisms (which do not require any *time synchronization*)

- The cardinality of the MVIDS is dependent only on the network field dimensions, the communication radius, and required resolution and is almost constant with respect to density of the network. Hence the message complexity does not increase with increase in density of the network.
- The entire message complexity to retrieve topology is proportionate to the resolution retrieved. If  $x\%$  of the topology (see definitions in section 2) is retrieved, the message complexity is  $N(1 + x + c)$ , ( $0 \leq c \leq x \leq 1$ ).

Rest of the paper is organized as follows. In section 2 we describe in detail our proposed algorithm. In section 3 we analyze the algorithm and the properties of retrieved multi-resolution topologies. In section 4 we describe various classes of multi-resolution topology discovery queries relevant for sensor network. We assume uniform network topology, and zero channel errors and node failures for the analyses in sections 2, 3 and 4. In section 5 we describe the changes required to make the algorithm adaptive to any arbitrary network condition. In section 6 we summarize related work and conclude the paper in section 7.

## 1. STREAM: SENSOR TOPOLOGY RETRIEVAL AT MULTIPLE RESOLUTIONS

In this section we describe STREAM in detail. We first give the basic assumptions and the topology resolution metrics. Next we describe the concept of MVIDS. We go into the details of request propagation involving the coloring algorithm and timer mechanisms to select the MVIDS. Finally we describe the second phase of the algorithm where topology information is aggregated back to the monitoring node.

### 1.1. ASSUMPTIONS AND NETWORK MODEL

The sensor network topology is a connected disk graph with a fixed circular communication range  $R$  (the radius of the disk). We assume that there are no node failures and channel errors, the request divergence is through controlled flooding<sup>3</sup> (each node forwards the discovery request exactly once) and nodes can collect their neighborhood lists by eavesdropping on the communication channel<sup>4</sup>. For complete evaluation of STREAM, we relax some of these assumptions in section 5.

The resolution of the topology is defined as:

---

<sup>3</sup> A probabilistic flooding as described in Haas and Halpern(2002) can be used as well. Ni et. al. (2002) discusses various heuristics for flooding in wireless networks.

<sup>4</sup> Note that each node must send at least one packet for other nodes to know its existence. A topology discovery request from each node ensures: 1) All nodes receive a packet; 2) Nodes have complete neighborhood lists

- **Edge Resolution:** Ratio of the number of *retrieved edges* and the actual number of edges in the network.
- **Node Resolution:** Ratio of the number of *retrieved nodes* and the actual number of nodes in the network.

To evaluate the performance of STREAM we describe two trivial approaches for multi resolution topology discovery for comparison.

- **Direct Response:** When a node receives a topology discovery request, it forwards this message and immediately sends back a response with its neighborhood list along the reverse path. In the probabilistic variation, nodes decide to reply back with some probability  $p$  and report all their edges.
- **Aggregated Response:** When a node receives a packet, it forwards the request immediately but waits to aggregate the information from its children before sending its own response. In the probabilistic variation, all nodes respond but report each of their edges with probability  $p$ .

The probabilistic variations can be used to retrieve topology at multiple resolutions. In the first case, since nodes respond back randomly, aggregation is not guaranteed. In the second case, since all nodes are reporting back, the number of responses is same as its non-adaptive counterpart, albeit with a reduced per-response cost. In the probabilistic variations, no guarantees can be provided that *each* node will be covered.

## 1.2. MINIMUM VIRTUAL INDEPENDENT DOMINATING SET (MVIDS)

STREAM selects a subset of nodes to reply to the topology discovery query with their neighborhood information. The cardinality of this subset determines the resolution of retrieved topology. The conceptual framework, which allows STREAM to control the cardinality, is a *Minimum Virtual Independent Dominating Set*.

Consider a graph  $G(V, E(R))$  where an edge exists between nodes  $v_i$  and  $v_j$  if their distance is at most  $R$ . For wireless networks, having communication range as  $R$  defines the network graph. We define the following terms on this graph:

- **Virtual Edge Set ( $E(r)$ ):** The subset of  $E(R)$  such that each edge in the set has endpoints at most distance  $r$  apart. In this case,  $r$  is called the *Virtual Range*.
- **Virtual Graph  $G(V, E(r))$ :** The sub-graph of  $G$  which has only edges from  $E(r)$ .

- **Minimal Virtual Independent Dominating Set  $MVIDS(r)$** : A minimal independent dominating set on  $G(V, E(r))$ <sup>5</sup>.

STREAM is a coloring algorithm, which creates an  $MVIDS(r)$  based on the *virtual range*  $r$ . Since a decrease in  $r$  causes a decrease in the number of virtual edges, the cardinality of  $MVIDS(r)$  increases as  $r$  decreases. STREAM selects the nodes in  $MVIDS(r)$  to respond to topology discovery queries with  $r$  providing a resolution control parameter. Note that finding a minimum independent dominating set of unit disk graph is NP-Complete (Clark, Colbourn and Johnson (1990)). STREAM is a distributed approximation algorithm to find  $MVIDS$ .

### 1.3. STREAM REQUEST PROPAGATION PHASE

The request propagation in STREAM uses controlled flooding. The algorithm takes three user-specified parameters that control the topology resolution from the minimal backbone tree to the complete network graph. The parameters are defined as follows:

- **Virtual Range ( $r\hat{I} [0,R]$ )**: The virtual range controls the cardinality of the  $MVIDS$  as described earlier.
- **Resolution Factor ( $f\hat{I} [0,1]$ )**: Each member of the  $MVIDS$  reports this fraction of edges originating from it. For example, if  $f=0.4$ , then a node should return 40% of its neighborhood list.
- **Query Type ( $Q$ )**: This defines the set of queries, which STREAM can support. The query type maps to specific filters and aggregating functions to support more sophisticated queries.

Thus, each query is of the form:  $STREAM(r, f, Q)$ .

To find the  $MVIDS$  we use four colors. As the topology discovery request propagates, different nodes are colored according to their definitions given below:

- **White**: Yet undiscovered node, or a node which has not received any topology discovery request packet.
- **Black**: A node in the  $MVIDS$  which replies to topology discovery request with its neighborhood set. After becoming *black*, a node discards all other request packets.
- **Red**: A node which is *virtually dominated* by at least one *black* node, i.e., it is inside the virtual range  $r$  of the *black* node. After becoming *red*, a node discards all other request packets. The node is said to be *attached-to* the corresponding black node.

---

<sup>5</sup> A minimum independent dominating set is the smallest set of nodes such that all nodes in the graph are either part of the set or neighbors of nodes in the set is also independent, i.e., no node in the set are neighbors of each other.

- **Blue:** A node which receives a packet from a *red* or *blue* node or a node which is within communication range of a *black* node but outside its *virtual range*. It waits for a time period for some other node in its virtual range to become *black*. Otherwise it itself becomes a *black* node.

Node ID	Color	Black Node Attached-To	Hops	Parameters: $r, f, Q$
---------	-------	------------------------	------	-----------------------

TABLE I. TOPOLOGY REQUEST PACKET.

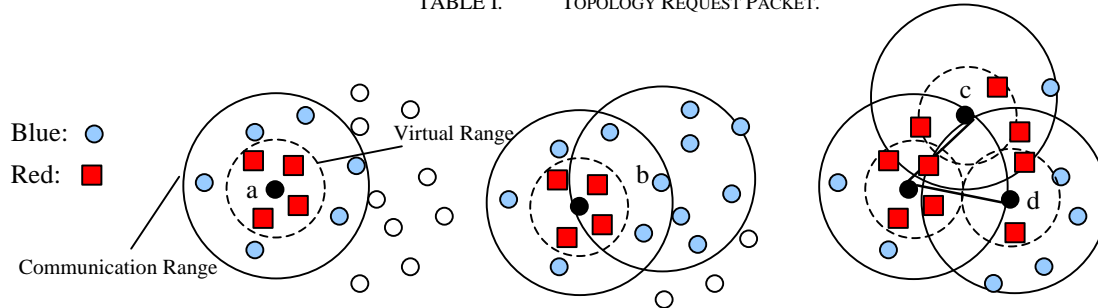


Figure 2: Illustration of the Coloring Algorithm. Node  $a$  becomes *black* and forwards a topology discovery request. Nodes within its virtual range  $r$  are colored *red* and nodes outside its virtual range but within its communication range  $R$  are colored *blue*. Node  $b$ , which is farthest from node  $a$  forwards the topology discovery request the earliest since forwarding delay is inversely proportional to the distance. Since node  $b$  was blue, all its *white* neighbors also become blue. All blue nodes start a timer to become *black*. Nodes  $c$  and  $d$  become *black* earlier than other blue nodes since they are closer to  $2r$  distance from previous *black* node  $a$ . They color their neighbors similar to earlier step. The *black* nodes  $c$  and  $d$  are children *black* nodes of node  $a$ .

```

RECEIVE_REQUEST_PACKET( recvColor, distance, r )
1.   if ( (recvColor==BLACK) & ( selfColor==WHITE) )
2.     if (distance < r)
3.       selfColor = RED
4.       FORWARD_REQUEST_PACKET(R_F_T(distance))
5.     if (distance>r)
6.       selfColor= BLUE
7.       B_F_T(distance)
8.       FORWARD_REQUEST_PACKET(R_F_T(distance))
9.     if ( (recvColor==BLACK) & ( selfColor==BLUE ) & (
distance<r ) )
10.    selfColor=RED
11.    cancel (B_F_T)
12.    if (a request packet has not been Forwarded )
13.      FORWARD_REQUEST_PACKET (R_F_T(distance))
14.    if ( (recvColor==RED) OR ( recvColor==BLUE ) )
15.      if (selfColor == WHITE)
16.        selfcolor = BLUE
17.        B_F_T(distance)
18.        FORWARD_REQUEST_PACKET(R_F_T(distance))

```

Figure 3: Coloring Algorithm when node receives a topology request Packet

The request packet is of the format given in Table 1. Figure 2 illustrates the coloring sequence. Figure 3 describes the action taken by a node on receiving a discovery request. Initially all nodes are *white*. The *monitoring*

*node* is colored *Black* and initiates the process by broadcasting a topology discovery request. As the request propagates, each node is colored *black*, *red* or *blue* according to the coloring algorithm. The algorithm is described as follows:

- Lines 1-13 describe the operations when a node receives packet from a *black* node. All *white* and *blue* nodes within virtual range of a *black* node become *red*. Other nodes that are outside *virtual range* become *blue*. After  $R\_F\_T(distance)$  time a node forwards the discovery request if it has not already done so. All *blue* nodes start a timer to become *black* with a *Black Node Formation delay* function  $B\_F\_T(distance)$ .
- Lines 14-18 describe operations when a node receives a packet from a *red* or *blue* node. When a *white* node receives a packet it becomes *blue*. It then starts a timer  $B\_F\_T(distance)$ , to become *black*.
- If any *blue* node receives a packet from a *black* node in its *virtual range*, it cancels its  $B\_F\_T$  and becomes *red*. Once nodes are *red* or *black*, they ignore other topology discovery request packets.

Ideally, a minimum number of nodes should reply back to provide a desired resolution (in this case the MVIDS). Since the problem of finding minimum dominating set is known to be NP-complete (MVIDS is a minimum dominating set on the *virtual graph*), we use a greedy approach for finding the set of nodes to reply back, i.e., at each step a node that covers the maximum number of *yet* uncovered nodes is chosen to become *black*. It is not possible to know the best candidate to become *black* without global knowledge of neighborhood sets. We use timer mechanisms to approximate the greedy approach<sup>6</sup>.

#### 1.4. TIMERS IN STREAM

We now describe the timer functions in STREAM.

- **Request Forwarding Timer ( $R\_F\_T(distance)$ ):** The time between receiving a discovery request and forwarding it, is the forwarding delay. The parameter  $distance^7$  is the distance between the receiving node and the sending node. The timer is inversely proportional to the distance, i.e., the farthest node forwards the earliest ( $R\_F\_T(distance) = c_1 / distance$ )

---

<sup>6</sup> For unit disk graphs the cardinality of any maximal independent set is within a factor of 5 of the minimum independent dominating set (Marathe (1995), Hunt (1998)). Since the black node set is at least a maximal independent set on the virtual graph, STREAM guarantees a constant factor approximation of the MVIDS.

<sup>7</sup> We assume that distance between nodes can be approximated by using GPS, signal strengths, etc.

- **Black Node Formation Delay ( $B\_F\_T(distance)$ ):** This is the time period after which a blue node changes to black. Thus a node whose *distance* is closest to the  $2r$  distance becomes black first.

$$B\_F\_T(distance) = c_2 / |2r - distance|$$

The timer mechanisms achieve the following:

- Maximal number of nodes become blue (*using  $R\_F\_T$  delay*).
- $R\_F\_T$  tries to reduce the collisions between forwarded packets (see appendix).
- $R\_F\_T$  is designed such that a node  $h$  hops away from the monitoring node forwards before a node  $h+1$  hops away (see appendix).
- The better candidates among the candidate blue nodes become black with a lower delay. (using  $B\_F\_T$  delay).
- The above timers work without *any global time synchronization* using only local knowledge

Please refer to appendix section 7.1 for discussion on how the timers achieve the above.

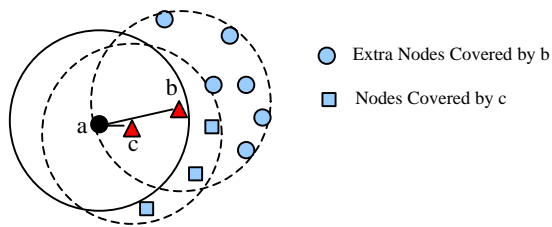


Figure 4: Illustration for the delay heuristic.

Consider the example in Figure 4 to illustrate the delay functions. Using  $R\_F\_T$  delay, node  $b$  forwards before node  $c$ , as it is expected to reach a larger set of uncovered nodes. Hence we have a larger candidate set to become black. Similarly a node at  $2r$  distance from the previous black node is expected to cover maximum number of uncovered nodes. Hence  $B\_F\_T$  delay is a heuristic to approximate the greedy selection of black nodes.

### 1.5. STREAM RESPONSE PHASE

During the first phase, the initiating node becomes the root of the *black* node tree where the parent *black* nodes are at most two hops away from their children *black* nodes. Each node has the following information at the end of this period:

- The *parent black node*, which is the last black node from which the topology discovery was forwarded.

- The *default node* to which it should forward packets in order to reach the parent black node. This node is essentially the node from which it had received the topology discovery request.
- All nodes have their *neighborhood information* by eavesdropping on the communication channel.

Using the above information, the response action is described below:

- When a node becomes *black*, it sets up an *acknowledgement timer* (described later) to reply to the discovery request. Each *black* node waits for this time period during which it receives responses from its children *black* nodes.
- It aggregates all topology information from its children and adds  $f\%$  of edges from its own region.
- When its acknowledgement timer expires, it forwards the aggregated neighborhood list to the *default node* to its *parent black node*.
- In the general framework for information retrieval, the parameter *query-type*, is used to filter or aggregate the information. However for topology discovery we just merge the neighborhood lists.

For the algorithm to work properly, timeouts of acknowledgements should be properly set. The *acknowledgement timer* of a *black* node should always expire before its *parent black node* so that each *black* node forwards only after receiving responses from its children. For this we set a timeout value inversely proportional to the number of *hops* a *black* node is away from the monitoring node (the number of hops is obtained from the discovery request packet). We need an upper bound on the number of hops between extreme nodes. If the extent of deployment region and communication range of nodes is known initially, the maximum number of hops can be easily calculated.

## 2. ANALYSIS OF STREAM

In this section we state some of the analytical results related to STREAM. All the results assume that the node density is uniform and that the network is connected. We verify our analytical results with simulations, for which we modified the NS-2 simulator to incorporate details of STREAM. We use the following terminology in this section:

$N$ =number of nodes

$A$ = area of sensor field

$R$ = communication radius

$r$ =virtual radius

$f$ = resolution factor

$d_r$  = average virtual degree for virtual radius  $r = \frac{Npr^2}{A} - 1$ . We omit the subscript if clear from the context.

$D$  = average node degree =  $d_r$ .

## 2.1. REQUEST PROPAGATION AND BLACK NODE TREE

**Proposition 1.** Each node receives the first request packet in minimum number of hops.

**Proof:** Consider a node which is  $h$  hops away from the initiating node. The timer mechanisms ensure that a packet that has traveled  $h-1$  hops is forwarded before one which has traveled  $h$  hops away. Thus the first packet the node receives is from a node  $h-1$  hops away.

**Proposition 2.** If the network has symmetric links, the aggregation tree is optimal in the number of hops. As a consequence every black node and intermediate node is optimal number of hops from the sink in the retrieved network graph.

**Proof:** The *next hop to parent* node for any node is the one from which it received the request packet first. Since the first packet is received in the optimal number of hops (proposition 1), by sending the topology response packet to the *next hop to parent* node, the initiating node can be reached in optimal number of hops. Since the aggregation tree is included in the retrieved topology, by default, every node in the black node tree is optimal number of hops away from the monitoring node in the retrieved network graph.

Next we consider the nature of request propagation in our algorithm. Reference (Cheng and Robertazzi (1989)) shows that the number of hops,  $H$ , a packet travels and the geometric distance,  $Dist$ , covered in a *broadcast percolation* scenario for high-density networks can be well approximated by the following relation:

1.  $H = \text{ceil}(Dist / R)$

Hence, from *proposition 1*, we can assume that the request percolates as a wave encompassing a circular region of radius  $iR$  in the  $i^{\text{th}}$  step. This is because each node at  $h$  hops forwards request before any node at  $h+1$ . We use the above approximation later to evaluate the expected overhead of our algorithm.

## 2.2. EXPECTED NUMBER OF BLACK NODES

To find number of black nodes ( $B$ ) for a given *communication range*  $R$  and *virtual range*  $r$ , we associate with each node a probability to become black based on the definition of node colors.

Let  $p =$  probability of each node to become black.

Each node would become black if no other node within its *virtual range* becomes black, i.e., if the expected virtual degree is  $n$ . Then for a given node, all its  $d$  virtual neighbors should not be black. If we assume that the nodes are colored with probability  $p$  independent of each other, equation 2 is satisfied. We note here that independence assumption used here is not true for the actual algorithm but serves as a good approximation for the number of black nodes formed.

$$2. \quad p = (1 - p)^d$$

For sufficiently large  $d$ ,  $p$  can be well approximated by the following:

$$3. \quad p = \frac{\ln(d) - \ln(\log(d))}{d}$$

We solve the above equation for  $p$  to get the probability of each node becoming black. Equation 2 is valid only when the expected number of nodes within a virtual region is greater than 1. As the virtual range reduces, this value can become less than 1. This means that each node has less than one node in its virtual range in the expected.

Let  $E =$  Expected virtual degree of a node

$$4. \quad E = \sum_{x=1}^{N-1} x \binom{N-1}{x} \left(1 - \frac{pr^2}{A}\right)^x \left(\frac{pr^2}{A}\right)^{N-x-1}$$

The total number of virtual edges in the network is half the sum of the expected degrees of all nodes, which is  $0.5NE$ . Since the number of *virtual edges* is less than the number of nodes (recall  $d < 1$  in this case), nodes with no edges would always become black. Out of the rest of the nodes, only half would become black with the remaining being neighbors of these nodes. The expected number of black nodes is given by:

$$5. \quad B = \left(N - \frac{NE}{2}\right) + \left(\frac{NE}{4}\right) = N - \frac{NE}{4}$$

Thus the expected number of black nodes is:

$$6. \quad B = \begin{cases} pN & \text{for } d > 1 \\ N - \frac{NE}{4} & \text{for } d \leq 1 \end{cases}$$

Note that Equation 6 does not take into account the heuristics and edge effects. However the heuristics in STREAM reduce the overlap in virtual regions. Hence the expected number of black nodes formed is lesser than that estimated by equation 6.

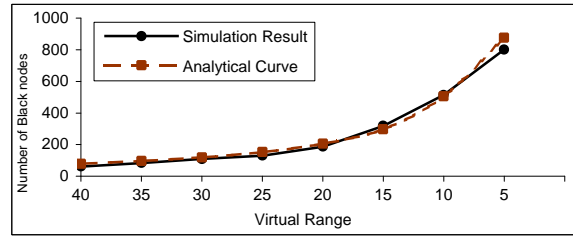


Figure 5: Comparison of analytical expectation of the number of black nodes and simulations. The simulation results are for 400x400m<sup>2</sup> field with 1000 nodes for a communication range 40m. The results show the number of black nodes formed for different virtual ranges

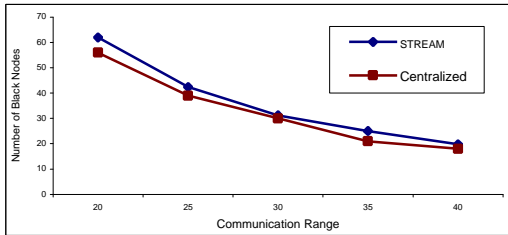


Figure 6: Number of *Black* nodes vs communication range for STREAM and centralized greedy algorithm. The size of sensor field is 200x200m<sup>2</sup> and the field has 1000 nodes.

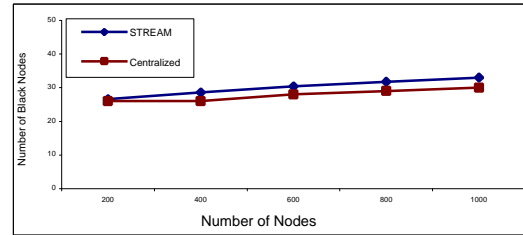


Figure 7: Number of *Black* nodes vs. total number of nodes for STREAM and Centralized greedy algorithm. The communication range is 30m and sensor field dimensions are is 200x200m<sup>2</sup>

Figure 5 plots the average number of *black* nodes formed in simulations compared to the number of *black* nodes from the analytical expectation. The analytical expectation is very close to the simulation results. In Figure 6 and Figure 7 STREAM is compared against centralized greedy algorithm (Cormen, Leiserson, Rivest (1990)) to find the black nodes. The nodes for this simulation are uniformly spread in 200m x 200m field and the virtual range is equal to the communication range. In Figure 6 and Figure 7 the impact of communication range and increasing the number of nodes in the field is shown for 1000 nodes in the field. In both cases, STREAM performs almost as well as the centralized solution which has global knowledge. This result shows that the heuristics used in STREAM timers perform well.

### 2.3. RETRIEVED TOPOLOGY RESOLUTION

STREAM takes two parameters, which control the resolution of the returned topology. Thus for a given *virtual range*( $r$ ) and *resolution-factor*( $f$ ), the returned topology resolution is computed as follows:

The number of black nodes formed for a *virtual range*  $r$  is given by equation 6. Each of these black nodes formed, returns  $f\%$  of its edges. Recall that two or more black nodes may be formed in the same communication range if *virtual range* is smaller than  $R$ . Since we assume edges to be symmetric, we should account for the edges which may be reported by both its endpoints.

$p(r)$  = probability of a node to be black, from equation 6 for a given virtual radius  $r$ .

Thus both nodes associated with any edge have probability  $p$  of becoming black. When a black node reports any edge with probability  $f$  (the resolution factor), the actual probability of that edge being reported is:

$$7. \quad X(p(r), f) = p^2(f^2 + 2f(1-f)) + 2pf(1-p) = 2pf - f^2p^2$$

Thus  $e$  = number of edges reported back =  $X(p, f)ND$ . Figure 9 shows the plot of expected topology resolution as a function of STREAM parameters *virtual range*( $r$ ) and *resolution-factor* ( $f$ ). The contours at the base are the *iso-resolution* curves obtained by intersection of a resolution plane with the curve. Each contour shows the set of  $(r, f)$  pairs which return the same resolution. We show in section 3, how these can be used for selecting the parameter values for specific queries.

### 2.4. OVERHEAD

First we consider the packet overhead of the algorithm. For the initial phase of the algorithm, since all nodes forward the topology discovery request at most once, the overhead is  $N$ . During the acknowledgement phase, a subset of nodes are involved namely the black nodes and the intermediate nodes between two black nodes. For a given virtual radius, the expected number of black nodes is given by equation 5.

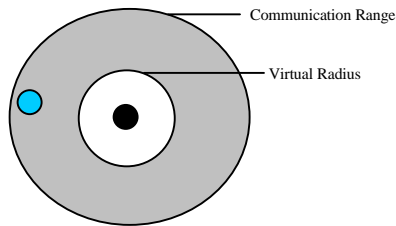


Figure 8: To Compute the Probability of an intermediate node to be black

To compute the expected number of intermediate nodes, we consider the probability that the intermediate node is also black. Consider Figure 8 with a black node  $a$  and an intermediate node  $b$ . We see that  $b$  is black with probability  $p$  only if it falls outside the virtual range of  $a$ . Thus the probability  $P$  that an intermediate node is *black* is given by:

$$8. \quad P = p \frac{R^2 - r^2}{R^2}$$

Hence the total number of packets transmitted ( $O_p$ ) in the acknowledgement phase is equal to the sum of number of black nodes and the number of intermediate red nodes. Since each black node except the monitoring node has one intermediate node, the total expected overhead is given by:

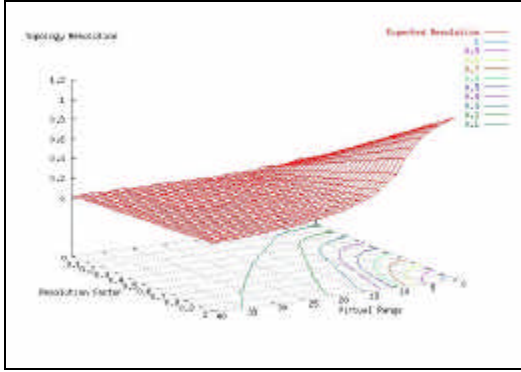


Figure 9: Expected Resolution of topology as a function of parameters Virtual Range and Resolution Factor. Each contour on the base shows the range of parameter values for topology returned at a particular resolution.

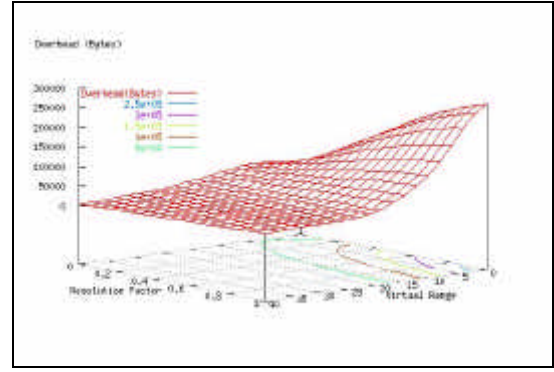


Figure 10: Expected Byte overhead of STREAM for different values of parameters ( $N=1000$ ,  $R=40m$ ,  $400 \times 400m$ ). Constant packet overhead  $C=5$  bytes, Overhead per edge information  $C_e=2$  bytes. The contours show the range of parameter values which require equal overhead to return topology.

$$9. \quad O_p \leq N + Np \left( 2 - p \frac{R^2 - r^2}{R^2} \right)$$

Next, the overhead incurred by STREAM is analyzed in terms of total bytes transmitted. Let,

$C_p = \text{constant header overhead per packet.}$

$C_e = \text{Overhead per edge information sent}$

Since the packet overhead is given by  $O_p$ , the overhead due to the constant packet overheads is simply  $C_p O_p$ . To compute the overhead of edge information, we need to estimate the number of hops the information about any retrieved edge travels, to reach the monitoring node. For simplicity of exposition we assume that the sensor field is

circular with radius  $R_A$  and the monitoring node is at the center of the deployed region. Generalization for any arbitrary shape of sensor region and position of monitoring node can be found in (Deb, Bhatnagar and Nath (2003a)).

From Cheng and Robertazzi (1989) we know that for sufficiently dense graphs, the maximum number of hops  $H$ , any node is away from the monitoring node is given by:

$$10. \quad H = \text{ceil}(R_A / R)$$

w.l.o.g. let  $R_A = RH$

We can divide the circular region into  $H$  concentric rings of width  $R$ . Thus the nodes in the  $i^{\text{th}}$  ring are  $i$  hops away from the monitoring node. Thus all edges which are reported by nodes in the  $i^{\text{th}}$  ring, travel  $i$  hops to the monitoring node (This is enforced by the aggregation tree properties from proposition 1 and 2). Let us now find the expected number of edges reported in the  $i^{\text{th}}$  hop.

$$\text{Expected number of edges in the } i^{\text{th}} \text{ ring} = \frac{\mathbf{p}(Ri)^2 - \mathbf{p}(R(i-1))^2}{\mathbf{p}R_A^2} \frac{ND}{2}$$

$$\text{Expected number of edges reported in the } i^{\text{th}} \text{ ring } E_i \leq X(p, f) \frac{ND}{2} \frac{R^2(2i-1)}{R_A^2}$$

Since each edge can be reported at most twice through independent paths, the total overhead due to the edges reported back is given by:

$$11. \quad \begin{aligned} O_E &\leq C_e 2 \sum_{i=1}^H i E_i = X(p, f) C_e ND \frac{R^2}{R_A^2} \sum_{i=1}^H i(2i-1) = \\ &= X(p, f) C_e ND \frac{R^2}{R_A^2} \left[ \frac{1}{6} H(H+1)(2H+1) - \frac{1}{2} H(H+1) \right] = C_e ND c X(p, f) \end{aligned}$$

Hence the total overhead  $O_T$  of the process is give by:

$$12. \quad O_T \leq O_p + O_E = C_p \left[ N + Np \left( 2 - p \frac{R^2 - r^2}{R^2} \right) \right] + C_e ND c X(p, f) = C_1 \left[ 1 + p \left( 2 - p \frac{R^2 - r^2}{R^2} \right) \right] + C_2 X(p, f)$$

Where,  $D$  is the average node degree,  $X(p, f)$  is given by equation 8, and  $C_1, C_2$  includes the constant terms which are not dependent on the STREAM parameters.

We state two important observations from the derivations above. These observations would later be used for optimal parameter selection for topology retrieval. Please refer to the appendix section 7.2. for a discussion on these observations.

**Observation 1:** For a given resolution, the overhead is a function of only the  $p$ , and independent of  $f$ .

We see that the second term in the overhead in equation is constant for a given topology resolution and the first part does not have the parameter  $f$ .

**Observation 2:** Given a resolution, the overhead is monotonically increasing and convex with respect to  $p$ .

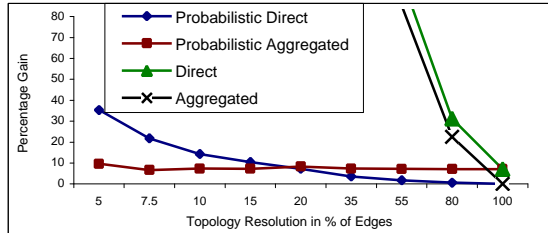


Figure 11: Relative Gain in Overhead of STREAM over Aggregated, Direct and their probabilistic variations.

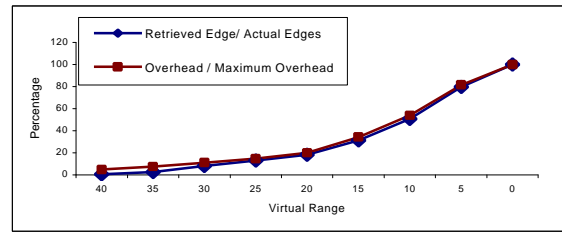


Figure 12: Graph shows the relative overhead (with respect to maximum overhead) incurred in retrieving topologies at different resolutions. (N=1000 nodes, communication range 40m, and field size 400x400 m<sup>2</sup>)

Next we consider the plots for the overhead functions. Figure 10 plots the STREAM overhead against different *virtual-ranges* and *resolution-factors*. Figure 11 and Figure 12 show the actual overhead incurred by STREAM. In evaluating the byte overhead, we assume a constant packet header of 5 bytes and an additional 2 bytes of information per edge reported in the packet. Total bytes transmitted during the entire operation characterizes overhead.

Figure 11 shows the percentage extra overhead incurred by direct response and aggregated response and their probabilistic variations over STREAM to retrieve equal resolution topologies. The probabilistic variations perform better than their non-adaptive counterparts, but are significantly worse than STREAM. Moreover, STREAM guarantees that each node would be reported whereas the probabilistic variations cannot provide such a guarantee.

Figure 12 shows the relative overhead for different virtual ranges. Observe that the relative overhead curve (as a percentage of the overhead incurred for retrieving the complete topology) closely follows the resolution. The relative difference at low resolutions is due to the constant overhead of the coloring phase of the algorithm.

### 3. MAPPING QUERIES TO PARAMETERS.

In this section we describe various types of queries that would be relevant in the context of sensor networks. While this is not meant to be an exhaustive list of queries, it would encompass a broad range of applications. STREAM algorithm can be invoked using three parameters: virtual range, resolution factor and query type. STREAM takes the following form:

– STREAM ( $r, f, Q$ )

To handle various queries and select parameter values we use the following theorems for multi-resolution topology retrieval using STREAM:

To handle various queries and select parameter values we use the following theorems for multi-resolution topology retrieval using STREAM:

**Theorem 1:** For a required resolution  $T_{res}$ , the minimum overhead is achieved by selecting  $r$  and  $f$  as follows:

- If  $T_{res} < X(p(R), 1)$  find  $f$  such that  $X(p(R), f) = T_{res}$
- If  $T_{res} \geq X(p(R), 1)$ , then put  $f=1$ , and find  $r$  such that  $X(p(r), 1) = T_{res}$ .

Thus the general rule is to first keep  $r=R$  and increase  $f$  up to 1 and then decrease  $r$  until we get the correct resolution.

**Proof:** Please refer to the appendix section 7.2.

**Theorem 2:** Given an overhead  $O_M$ , to maximize possible resolution  $X(p(r), f)$ , select parameter  $r$  and  $f$  as follows:

- Put  $r=R$ , and find  $f$  such that  $O_T(p(R), f) = O_M$
- If  $O_T(p(R), f) < O_M$ , then put  $f=1$ , and find  $p(r)$  such that  $O_T(p(r), 1) = O_M$ .

Thus the general rule is to first keep  $r=R$  and increase  $f$  up to 1 and then decrease  $r$  to get higher resolutions until the overhead constraint is exceeded.

**Proof:** Please refer to the appendix section 7.2.

The intuition behind the above rules is that they are maximizing the amount of information per packet so that the effect of constant overhead is minimized.

Note that, the retrieved topology at a desired resolution might be required to possess certain properties. The trade-off here is between the overhead and maintenance of graph properties. For example, to test whether each node has two edge-disjoint paths to the sink, the retrieved topology should have minimum node degree of two. However, exploring these tradeoffs is orthogonal to the focus of this paper and is part of our future work.

Note that, the retrieved topology at a desired resolution might be required to possess certain properties. The trade-off here is between the overhead and maintenance of graph properties. For example, to test whether each node has two edge-disjoint paths to the sink, the retrieved topology should have minimum node degree of two. However, exploring these tradeoffs is orthogonal to the focus of this paper and is part of our future work.

In this work, we try to *minimize the overhead* or maximize the *retrieved resolution* for which theorem 1 and theorem 2 are used to select the parameter values. We use simulations (averaged over 20 runs with randomly chosen initiating nodes) to show the effectiveness of STREAM in dealing with different types of queries.

### 3.1. NODE CONSTRAINED QUERY:

The node-constrained query is of the form: “Return a representation of the network with  $x\%$  of the nodes”. Such a query helps in determining the approximate density distribution of the network while not requiring all the nodes to be returned.

For this query, we construct the minimal backbone topology by setting the parameter *virtual range* ( $r$ ) equal to communication range  $R$  and *resolution-factor* ( $f$ ) equal to the desired fraction  $x$ . From theorem 1 we know that such a selection of parameter values will give the minimum overhead for the query. The following lists various node-constrained queries:

- STREAM( $R, x, \text{Node-Constrained-Query}$ ).
- STREAM( $R, 0, \text{Node-Constrained-Query}$ )
- STREAM( $r, 0, \text{Exposure-Query}$ )

When the parameter *Node-Constrained-Query* is passed, each black node finds its neighbor nodes, which have *not* been reported by its children black nodes. Out of these unreported neighbors, it picks  $x\%$  of nodes and aggregates with the children’s information before sending it upstream. The 2<sup>nd</sup> query is used to find the virtual backbone of a network. In the query-type *Exposure-Query*, each node just returns its location. The number of nodes responding is controlled by virtual range  $r$ . As has been observed in [13], the minimum exposure path does

not change significantly after a certain node density. STREAM can return the minimal number of nodes to calculate the exposure with desired precision.

Required Node Resolution	Returned Node Resolution		
	Average	Maximum	Minimum
0.8	0.813	0.832	0.791
0.6	0.621	0.646	0.605
0.4	0.439	0.456	0.416
0.2	0.238	0.269	0.213

TABLE II. PERFORMANCE OF NODE-CONSTRAINED QUERY ON A RANDOM 1000 NODE TOPOLOGY IN A 400x400M<sup>2</sup>, R =40M.

Table 2 gives the performance of node-constrained query using STREAM for the different required resolutions in the simulation setup described earlier.

### 3.2. EDGE CONSTRAINED QUERY:

The form of an edge-constrained query is: “Return  $x\%$  of the edges in the network”. This type of query is useful in determining the connectivity properties of the network. We evaluate properties such as shortest paths lengths, number of node disjoint paths, etc. for different resolutions of topology retrieved.

To extract topology at required resolution with minimum overhead, we use the solution in theorem 1 to find parameters  $r$  and  $f$ . The *resolution-factor* is set to 1. Note that setting resolution-factor to 1 ensures that at least one edge pertaining to each node is reported back. Let,

$$E = \text{expected total number of edges in the topology. } E = \frac{ND}{2},$$

Since every black node reports all edges originating from it, the total number of edges reported back only depends on the number of black nodes chosen to reply back. To get the required fraction of edges ( $x$ ) the following condition must be satisfied.

$$\frac{Bd}{2} = xE \quad \Rightarrow B = \frac{2xE}{D}$$

And also from equation 5,

$$B = pN \Rightarrow p = \frac{B}{N} = x$$

This gives the virtual radius as follows:

$$13. \quad r = \sqrt{\frac{A}{Np} \left( 1 + \frac{\ln(x)}{\ln(1-x)} \right)}$$

Now for equation 13 to be consistent with Equation 6, the expected number of nodes in virtual range should be greater than 1. Thus  $r$  is valid only if:

$\frac{\ln(x)}{\ln(1-x)} > 1$ , Otherwise  $r$  is given as:

$$14. \quad r = \sqrt{\frac{4A(1-x)}{Np}}$$

The topology discovery query takes the following form:

- STREAM( $r$ , 1, Edge-Constrained-Query).

The edge-constrained query is in essence, central to the multi-resolution topology extraction problem we are trying to solve. The overhead for this type of query given by equation 12 illustrates the exact nature of trade-off between the resolution of topology retrieved and communication overhead expended.

The edge-constrained query also illustrates the convergence properties of the multi-resolution network graph. Let us compute the average hop deviation for shortest paths from monitoring node in the retrieved graph. From proposition 2 in section 2.1 we know that every black node and intermediate node is optimal number of hops away from the sink in the retrieved graph. All other nodes are at most two hops away in the retrieved graph since they are always neighbor of some black node. Hence the average deviation is bounded by the number of blue nodes which are not intermediate nodes. For required edge resolution  $x$ , this is given by:

$$15. \quad N_{blue} = N - Nx \left( 2 - x \frac{R^2 - r^2}{R^2} \right)$$

$$16. \quad H_{deviation}(x) \leq 2 \left( 1 - x \left( 2 - x \frac{R^2 - r^2}{R^2} \right) \right)$$

Thus if we retrieve  $x\%$  of the edges using edge-constrained query the average deviation is bounded by equation 16. Thus STREAM can give guarantees on hop deviation and 100% node coverage which is not possible with probabilistic variations. Moreover the overhead incurred is lesser which makes STREAM suited for sensor networks.

Table 3 shows the mapping of required edge-resolution to the value of parameter  $r$  for the simulation setup described at the beginning of the section 3. Using the above queries, we see that STREAM is able to retrieve edges at resolution close to the desired values. Figure 11 showed that multi-resolution topology is retrieved at proportionate cost.

Required Edge Resolution	Virtual Radius	Returned Edge Resolution		
		Average	Maximum	Minimum
0.8	6.38	0.710	0.715	0.706
0.6	9.03	0.556	0.561	0.546
0.4	11.93	0.424	0.427	0.421
0.2	20.45	0.179	0.188	0.175

TABLE III. PERFORMANCE OF EDGE CONSTRAINED QUERY ON A RANDOM 1000 NODE TOPOLOGY IN A 400x400m<sup>2</sup>, R =40M.

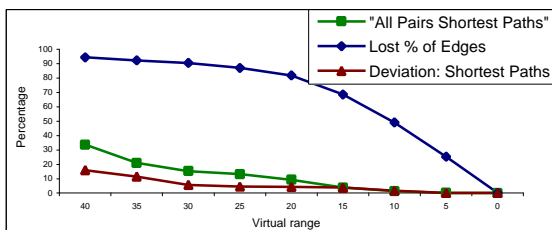


Figure 13: Effect of Edge Resolution: Deviation from optimal All-pairs and single source shortest path lengths, (N=1000, R=40, field size 400x400 m<sup>2</sup>)

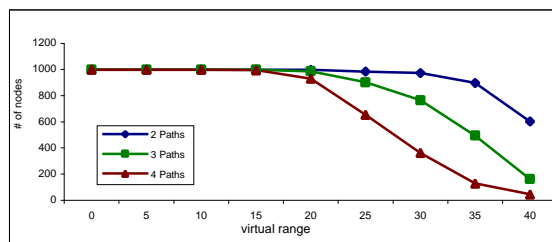


Figure 14: Effect of Edge Resolution on number of Node Disjoint Paths to Sink. (N=1000, R=40, field size 400x400 m<sup>2</sup>)

Figure 13 shows the effect of edge resolution on the length of single source shortest path lengths (rooted at initiating node) and all-pairs shortest path lengths. The graph shows the relative degradation in the single source shortest path lengths and all pair shortest path lengths as the topology resolution increases. We see that even for low resolutions, paths do not deviate significantly from the optimal. Figure 14 shows the number of nodes which have 2, 3 and 4 node and edge disjoint paths for different resolutions.

We note that the savings in topology discovery increases as the density of the graph increases, i.e., lesser percentage of edges is required as density increases for similar deviation from optimal behavior. Also as density increases lesser percentage of nodes become black due to which the relative message complexity decreases. Using STREAM we can attain significant savings by discovering topology at the required resolution in dense sensor networks.

### 3.3. OVERHEAD CONSTRAINED QUERY

This query has the form: “Return the best resolution topology using a given Overhead budget.” The total overhead in topology discovery is due to request forwarding (network-wide flooding) and the discovery acknowledgement. To support this query, we use solutions from theorem 2 to select the optimal parameter values. In general, the analytical computation of parameters by this method is not required if we assume uniform random

graphs. Setting *resolution-factor* to 1 and then selecting the minimum possible virtual range for the given overhead would retrieve the maximum resolution. However, other considerations may come into account when selecting the parameter values (as discussed at the beginning of the section).

#### 4. STREAM UNDER ARBITRARY NETWORK CONDITIONS

In this section we analyze the performance of STREAM under arbitrary network conditions involving non-uniform topologies, sleeping nodes, channel error rates and node failures. We also propose methods which make STREAM adaptive to these varying network conditions.

##### 4.1. NON-UNIFORM TOPOLOGY

We had assumed a uniform node distribution for the network topology. Although this was useful to analyze the behavior of STREAM, practical sensor networks may not be uniform and we may not have any information about the underlying distribution. For STREAM to be useful in practice, it should be able to function properly even without knowledge of node distribution.

Consider the network to have any arbitrary distribution where a node  $i$  has some degree  $d_i$  whose distribution is unknown. The key to getting the desired resolution in a uniformly distributed network was that having a constant virtual range across the network sufficed. When the topology is non-uniform, each node has to compute its own virtual range based on the local network density and node degree.

Consider the circular range of the node  $i$  with  $d_i$  neighbors. Let  $p_i$  be the probability of nodes inside this region to become black.

*Expected number of black nodes in the region =  $p_i d_i$ .*

Assuming that node distribution is uniform *inside the communication range*<sup>8</sup>, the number of edges reported for this region should be a *fraction*,  $x$  of the total expected edges in this region. Hence, we have to choose a virtual range so that  $p_i=x$  where  $x$  is the required edge resolution.. Let the virtual range for node  $i$  be  $r_i$ .

$$17. \quad r_i = \sqrt{\frac{R^2}{d_i} \left( 1 + \frac{\ln(x)}{\ln(1-x)} \right)}$$

---

<sup>8</sup> While the node distribution across the network is non-uniform, the distribution inside a node's communication range can be considered to be uniform when the range is much smaller than the network dimensions.

And for  $\frac{\ln(p)}{\ln(1-p)} < 1$ , we have

$$18. \quad r_i = \sqrt{\frac{4R^2(1-x)}{d_i}}$$

Thus every node computes its own virtual radius according to its local information. The coloring scheme and the algorithm remain the same while using these local values of virtual range.

We evaluate this scheme using two different non-uniform topologies. In the first one we consider a square field of dimension 400x400 m<sup>2</sup> and divide it up in four quadrants of size 200x200 m<sup>2</sup>. We put 100, 200, 300 and 400 nodes in the first, second, third and the fourth quadrant respectively resulting in different densities in each quadrant. In the second case we generate a gaussian distribution of 1000 nodes around the center of a 400x400 m<sup>2</sup> field and variance 75m.

Required Edge Resolution	Edge Resolution on Topology 1 assuming uniform	Returned edge resolution with local scheme	Edge Resolution on Topology 2 assuming uniform	Returned edge resolution with local scheme
0.8	0.66	0.706	0.503	0.71
0.6	0.497	0.56	0.353	0.56
0.4	0.38	0.411	0.23	0.43
0.2	0.160	0.184	0.087	0.208

TABLE IV. PERFORMANCE OF EDGE CONSTRAINED QUERY ON NON UNIFORM TOPOLOGIES OF 1000 NODES IN A 400x400M<sup>2</sup>, R =40M.

Table 4 shows the results of performing edge-constrained queries of the two non-uniform topologies. We see that by assuming the topology to be uniform and computing a single global value of virtual range results in large deviations in the returned edge resolution compared to the required edge resolution. By using local virtual ranges the returned resolution is very near to the required resolutions and performs as good as with the uniform topology case.

#### 4.2. IMPACT OF SLEEPING NODES

One of the distinguishing features of sensor networks is that nodes would sleep periodically. We compute the fraction of sleeping nodes that would be reported by the responding active nodes. Note that active nodes cache data about its neighbors and would respond with information about sleeping nodes as well. This discussion assumes that the network of active nodes is connected. Let,

$x$ =fraction of nodes sleeping,  $P(1-x)N = N'$  is the number of active nodes.

$p$  =probability of node to become black (equation 5)

$p_s$ =probability of a sleeping node being reported

$b_x$ =expected black neighbors of each node:

$$19. \quad b_x = p \sum_{i=1}^D \binom{D}{i} i(1-x)^i (x)^{D-i}$$

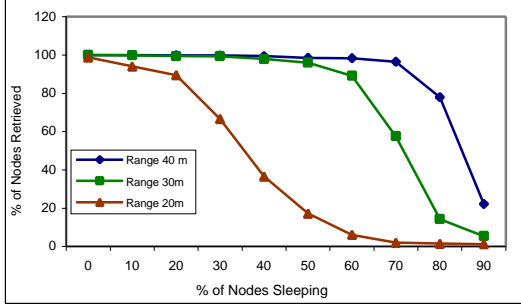


Figure 15: Effect of sleeping nodes on the Retrieved Topology. The network consists of 1000 node in a field of 400x400m<sup>2</sup>. Simulations are carried out for three different communication ranges with varying % of nodes sleeping.

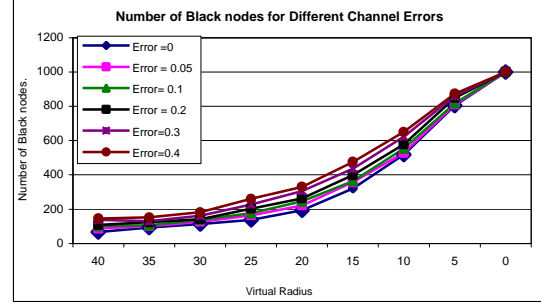


Figure 16: The number of black nodes formed for different channel error rates. 1000 node deployed in 400x400 m<sup>2</sup> field with R=40m.

Each active node is always reported because the network of active nodes is assumed to be connected. The probability of a sleeping node being reported is the probability of that any one of its  $b_x$  black neighbors report it. Since black nodes report  $f$  % of their edges, the probability is given by:

$$20. \quad p_x = \begin{cases} 1 - (1-f)^{b_x} & b_x \geq 1 \\ b_x f & b_x < 1 \end{cases}$$

We tested the performance of STREAM with varying percentage of sleeping nodes. The simulations are set up for  $r=R$  and  $f=1$ . The results in Figure 15 show that STREAM is able to retrieve information about nearly all the nodes if the number of active neighbors of each node is around 8. The results also imply that if the active node set is connected, then we get information about almost all nodes.

#### 4.3. CHANNEL ERRORS AND NODE FAILURES.

Sensor networks are envisioned to be deployed in hostile territories and hence would have significant channel errors and node failures. To make our proposed algorithm useful in practical scenarios, we have to ensure that it is fault tolerant. We first discuss the effect of channel errors and node failures during the request propagation phase.

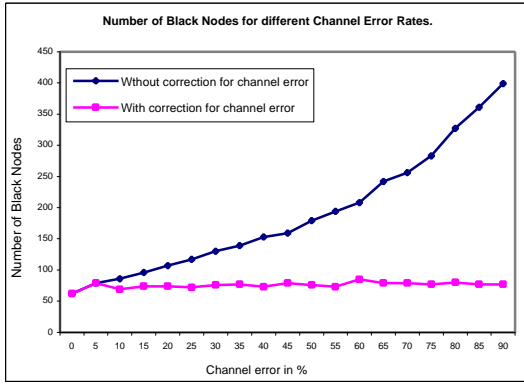


Figure 17: The number of black nodes formed for different channel error rates. 1000 node deployed in 400x400 m<sup>2</sup> field with R=40m, and r=40.

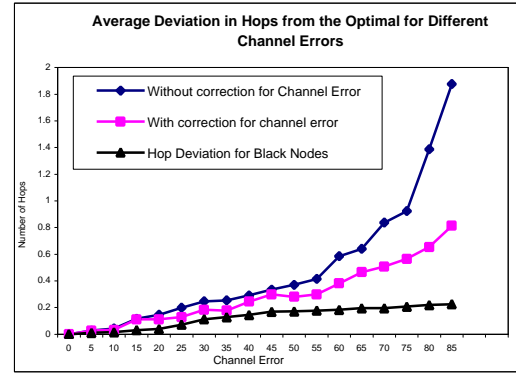


Figure 18: The average hop deviation of the received request packets for different channel error rates. 1000 node deployed in 400x400 m<sup>2</sup> field with R=40m, and r=40.

Channel errors affect the request propagation phase in the following ways:

- *Increase in number of black nodes:* Since packets may be lost when a black node forwards a request, a node may become black even if it is inside the virtual range. The effect is shown in Figure 16 and Figure 17.
- *Proposition 1 and 2 are no longer valid.* This means that request divergence may not have a wave like propagation. Figure 18 illustrates this effect.
- Timers may not work since nodes do not receive a topology discovery request packet in optimal hops.
- All nodes may not receive a request packet. However for dense networks, this effect is negligible

A node becomes black if no other node in its virtual range is black or if packets from one or more black nodes in its virtual range are lost due to channel errors. Thus equation 2 can be modified to incorporate the channel errors to get the analytical model for number of black nodes:

$$21. \quad p = \sum_{i=0}^d e^i p^i (1-p)^n$$

One may choose a value of virtual radius according to equation 21 to get the required resolution. We may also select timers such that it incorporates the expected deviation related to the error rate. However computing the values may be too complex for sensor nodes. We choose a simple method by which the number of black nodes and hop deviation is reduced significantly.

We see in Figure 16 and Figure 17 that increase in number black nodes and hop deviation is negligible for channel error rates lesser than 10%. This means that at such channel error rates, propositions 1 and 2 are valid with high probability and equation 6 is still valid approximation.

Let  $0 < c < 0.1$  be some channel error rate for which the effect is negligible. Suppose we send multiple copies of the request packet per node, neighbor nodes would receive the packet with higher probability because of this redundancy, thus reducing the apparent channel error rate. In particular to make the apparent channel error close to  $c$ , the number of copies  $m$  required is:

$$22. \quad m = \frac{\ln(c)}{\ln(e)}$$

However if each node sends  $m$  request packets the overhead would increase significantly. Instead if only a black node sends  $m$  copies, overhead is significantly lowered since black nodes are much lesser in number. We can use the scheme of only black nodes forwarding multiple packets if it is able to reduce the effect of channel errors significantly.

We assume that nodes have knowledge of their local channel error rates. We use  $c=0.05$  to compute the value of  $m$ . We use the scheme where only black nodes forward  $m$  copies. In Figure 17 and Figure 18 we show the effect of this scheme on the number of black nodes and hop deviation for different channel error rates. We see that the number of black nodes and average hop deviation is stabilized even for high channel error rates.

By stabilizing the number of black nodes and average hop deviation for any channel error rate, the analysis in section 2 remains a good approximation. Hence the parameters for edge and node constrained queries can still be mapped as described in section 3. The bounds on the properties of retrieved topology are also maintained with high probability. Since  $m$  increases sub-linearly with increase in  $e$  the overhead increases sub-linearly if the number of black nodes is small compared to the total number of nodes in the network.

The effect of node failures during the request propagation phase is equivalent to the effect of sleeping nodes since nodes do not participate in the topology discovery if they are dead or sleeping. However a node may die in between the topology request phase and the acknowledgement phase.

During the request propagation phase, black nodes send back their aggregated topology information upstream. Due to channel errors topology information is lost if an acknowledgement packet is lost. Similarly if any node in the black node tree dies, the information from its sub-tree would not reach the sink. *A passive acknowledgement*

*scheme* from (David Johnson and Maltz (1996)) is used to account for channel errors. We utilize the high density of sensor networks to account for node failures. These schemes are described as below:

After a black node sends its aggregated edge information to the next hop to its parent black node it eavesdrops on the communication channel to see if the next hop node has forwarded its packet. If it does not hear within a certain time period it forwards the packet again.

Since the next hop may be dead, the number of retransmissions cannot be allowed to go on indefinitely. We have to limit the number of retransmissions. From equation 24, we know that if a packet has been retransmitted  $m$  times, then with probability  $1-c$  it should have reached the next hop. If the black node does not receive the passive acknowledgement after  $m$  tries we may infer with high confidence that the next hop node is dead.

After  $m$  retransmissions, a black node selects another node from its neighborhood with a gradient towards the sink (if the black node is  $h$  hops away from the sink, it selects a node which is  $h-1$  hops away from the sink). We note that since the network density is high there are multiple such nodes with high probability. Each node just has to cache the information which it gathers during the request propagation phase.

Let  $k$  be the number of neighbors of a node with a gradient towards the sink. If the node failure rate is  $f$  and local channel error rate is  $e$ , the probability that a packet is forwarded successfully to a node next hop is given by:

$$23. \quad P(\text{success}) = 1 - (1 - (1 - e)(1 - f))^{mk}$$

Using the above method, information from active black nodes can be sent to the sink with high probability. For example with  $e=0.3$ ,  $f=0.1$ ,  $m=3$  and  $k=2$ , gives  $P(\text{success})= 0.999622$ . Thus we see that even if a node retransmits a packet at most three times, and has only two next hop nodes, then even for a channel error of 30% packets from downstream are lost with very low probability.

However the neighborhood information of a dead black node is lost. To compensate for this loss, when an intermediate node infers after  $m$  transmissions that its black parent is dead, it adds its own neighborhood list to a packet when forwarding it upstream. Thus we are able to retrieve information about nodes which are in range of both the black node and the intermediate node. The only problem with this method is that there is no way to compensate for the death of a node in the last hop.

We have to also take care of the timers for nodes which forward when some node in the black node tree dies. After receiving the topology request each node starts an acknowledgement timer in case it is required. With this method the layered aggregation semantics remain valid. This completes the description of our algorithm.

Virtual range	Expected resolution to be retrieved	Retrieved Edge Resolution, $e=0.15$ , $f=0.05$	# of nodes Retrieved $e=0.15$ , $f=0.05$	Retrieved Edge Resolution, $e=0.3$ , $f=0.1$	# of nodes Retrieved $e=0.3$ , $f=0.01$
40	0.07	0.16	982.46	0.182	978.952
30	0.15	0.262	980..5	0.28	963.047
20	0.219	0.304	995.57	0.326	994.5
10	0.62	0.706	999.95	0.731	999.578
5	0.81	0.859	999.71	0.91	999.6

TABLE V. PERFORMANCE OF EDGE CONSTRAINED QUERY WITH DIFFERENT CHANNEL ERROR AND NODE FAILURE RATE, 1000 NODES IN A 400X400M<sup>2</sup>, R =40M.

Table 5 gives the performance of the edge constrained query for two pairs of channel error and node failure rate. In the first case we look at mild conditions of 15% channel error and 5% node failure. In the second case we look at a harsher condition of 30% channel error and 10% node failure rate. The node failure rate is a relatively high since it is unlikely that 10% of the nodes would die within a time period of topology discovery. We see that retrieved edge resolution is always higher than the expected resolution although the average number of nodes, which are included in the retrieved topology, is not 1000. This is because with high failure rate, many black nodes die. To compensate for this, other intermediate nodes insert their own neighborhood lists. Thus although the edge resolution increases, all nodes are not covered.

We also see that at low resolution there is a large difference in the expected topology and the retrieved topology and the difference reduces for higher resolutions. This is because at higher resolutions percentage of blue intermediate nodes is very low and the extra edges added by them correctly compensates for the missing information due to dead black nodes. This result is also favorable for sensor networks since we would ideally like to have better control on retrieved as resolution increases because retrieved resolution directly affects the overhead incurred in the process.

## 5. RELATED WORK

Deb, Bhatnagar and Nath (2002) describes a rudimentary topology discovery algorithm, *TopDisc* for sensor networks. The basic description of STREAM proposing multi-resolution topology retrieval for sensor networks appears in Deb, Bhatnagar and Nath (2003a) and Deb, Bhatnagar and Nath (2003b).

Researchers have proposed different mechanisms for topology discovery of data networks in Downey (1999), Lowekamp, O'Hallaron, and Gross (2001), Breitbar et. al. (2000), Miller and Steenkiste (2000) primarily using probing techniques. Using routing tables for aggregating topology information is not feasible because traditional routing tables may not be available if data centric model of routing is used in Intanagonwiwat, Govindan and

Estrin (2000). Further, in ad-hoc deployments, routing tables are often inaccurate or incomplete. Probing techniques used for Internet mapping (Downey (1999)) is not possible in sensor networks since nodes operate in various levels of doze mode, may often be disconnected and the incurred overhead is unsuitable for energy constrained sensor networks.

In Bestavros, Byers and Harfoush (2002), given a set of network endpoints, end-to-end Bayesian probing is used to infer properties of IP networks using correlations. Our work focuses on *selecting a representative set* of nodes in a sensor networks to estimate network properties.

In RoyChoudhury, Bandyopadhyay and Paul (2000) authors propose a mobile agent based framework to distribute topology information. Here, the optimal number of agents required is half the number of nodes. The overhead with this approach would not scale for sensor networks where the number of nodes is large. Reducing the agents would increase the expected time that any agent takes to reach the initiating node.

References Das, Bhargavan (1997), and Ephremides, Wieselthier, and Baker (1987) introduced the concept of virtual backbones for wireless networks. References Das, Bhargavan (1997), Ephremides, Wieselthier, and Baker (1987), Gerla and Tsai (1995), Guha and Khuller (1996), Sinha, Sivakumar and Bharghavan (1999), Bannerjee, Khuller (2001), Das, Sivakumar and Bharghavan (1997) also describe routing in ad hoc networks using minimal connected dominating sets. Using each of the above methods, a simple topology discovery algorithm can be designed to query the dominating nodes, which provide their neighborhood lists. However, our algorithm differs significantly from the above in its *multi-resolution* nature. Hence, the topology returned is not limited to the minimal backbone. In fact, the topology that each of the above can provide is only a special case of lowest resolution topology recovered by our algorithm.

In Ganesan and Estrin (2002) a conceptual framework called DIMENSIONS was introduced for multi-resolution data-access in sensor networks. STREAM provides a framework to spatially sample a sensor network to select a subset of representative nodes for the network at a given resolution. The algorithm proposed in Ganesan and Estrin (2002) focus compression of data at each node using spatial and temporal correlations and maybe used in conjunction with STREAM after using STREAM to first select the subset.

## 6. CONCLUTIONS AND FUTURE WORK

In this paper, we have described an algorithm for sensor topology extraction at multiple resolutions (STREAM). STREAM may also be used as a general-purpose multi resolution information retrieval algorithm. The main issues are generalizations of the sampling parameters to select the subset.

For topology discovery the node selection process finds a suitable MVDS. In general, the node selection could depend on any characteristic of the information sought. For example, if the desired information has spatial gradient and neighborhood correlations, then such information can determine the *virtual-range*.

The parameters *resolution factor query-type* and determines the filtering and compression of neighborhood done at the sampled node set. The *resolution-factor* determines the number of neighbors about which a reporting node responds. Each node can collect information about its neighboring nodes by eavesdropping. The responding node can choose a fraction  $f$  of these neighbors and apply the filter specified by the query-type parameter. Even, in case of topology discovery a more sophisticated aggregation scheme as proposed in Awerbuch and Shavitt (2001) could be used. A different type of aggregation could be used for energy information as proposed in Zhao, Govindan and Estrin (2002). Finally use of schemes like smart messages as proposed in Borcea et. al. (2002) which would carry specific code along with the query to support infrequent queries could save valuable memory resources in sensors.

STREAM opens up a wide design space for multi-resolution information retrieval. This would require knowledge of the properties of queries and the corresponding aggregation/filter functions to compress that property. We intend to explore this design space for different types of information in future.

## 7. APPENDIX

### 7.1. TIMERS

In this section we derive the function  $R\_F\_T(\text{distance})$ . The function is derived keeping two important properties in mind: first it should ensure that Proposition is valid and second it should reduce the number of collision. These are described as follows:

**$R\_F\_T(\text{distance})$  ensures that node at  $i^{\text{th}}$  hop would always forwards before  $(i+1)^{\text{th}}$  hop**

Each node decides a time period  $Epoch_{Start}$ , to  $Epoch_{End}$  within which they should forward the request. Suppose the node decides to forward with delay  $T_{delay}$  in its Epoch. Then the next hop node should forward after the  $Epoch_{End}$  of previous hop. To ensure this, in the request packet we pass the value of  $(Epoch_{End} - Epoch_{Start} - T_{delay})$  which is added onto the  $T_{delay}$  of the next hop. We note here that here we don't need the actual time of epoch  $Epoch_{Start}$  and  $Epoch_{End}$  but only the time period  $Epoch_{End} - Epoch_{Start}$ . The above scheme would ensure that a node at  $i^{\text{th}}$  hop would always forward before  $(i+1)^{\text{th}}$  hop.

Epoch in  $R\_F\_T(\text{distance})$  given by equation 26 also reduces the *collisions* between forwarded packets. We now show how this is achieved and give an approximate computation of the Epoch. Let,

$T_g = \text{the granularity of the timer.}$

$t_p = \text{time to broadcast a packet based on the packet size.}$

$n = \text{number of neighbors in the neighborhood.}$

$d_i = \text{distance between sending nodes and the } i^{\text{th}} \text{ receiving node.}$

$R\_F\_T(d_i) = \text{request forwarding timer.}$

**$R\_F\_T(\text{distance})$  minimizes collisions among forwarded packets.**

We know that the distance  $d_i$  is mapped to a discrete value of the timer and to avoid collisions we must have unique timers for each node with high probability.

Let us consider an interval  $d_x$  such that

$$R\_F\_T(x + d_x/2) = R\_F\_T(x - d_x/2).$$

I.e. due to discrete nature of timers, all timer values in the interval  $d_x$  are mapped to the same value. To have unique values of timers, we require that expected number of nodes in an interval  $d_x$  should be less than one. The interval  $d_x$  corresponds to a ring of width  $d_x$  in the neighborhood of the node. Thus the expected number of nodes in a ring of width  $d_x$  at distance  $x$  is  $2\pi x d_x n$ .

Since the outermost ring at distance  $R$  would have the maximum number of nodes, by computing  $d_x$  such that it contains one node in the expected, we ensure that any ring of width  $d_x$  inside would have lesser than one nodes.

$$d_x = \frac{1}{2\pi R n}$$

Approximately, the number of rings of width  $d_x$  that are present in communication range  $R$  is  $= 2\pi R^2 n$ . I.e.

Epoch should be chosen such that due to discrete  $T_g$ , we have  $2\pi R^2 n$  discrete levels each separated by packet transmission times. Hence:

$$24. \quad \text{Epoch} = T_g (t_p + c) 2\pi R^2 n.$$

Where  $c =$  constant time delay that may be added to take processing delays at nodes into account.

Thus the function  $R\_F\_T(\text{distance})$  is given by:

25.  $R\_F\_T(\text{distance}) = \text{Epoch} / \text{distance}$

Where Epoch =  $T_g (t_p + c) 2\text{PIR}^2 n$

## 7.2. OVERHEAD

We state two important observations from the derivations above. These observations would later be used for optimal parameter selection for topology retrieval.

$$26. \quad O_T(p, f) \leq O_p + O_E = C_1 \left[ 1 + p \left( 2 - p \frac{R^2 - r^2}{R^2} \right) \right] + C_2 X(p, f)$$

**Observation 1:** For a given resolution, the overhead is a function of only the  $p$ , and independent of  $f$ .

We see that the second term in the overhead in equation is constant for a given topology resolution and the first part does not have the parameter  $f$ .

**Observation 2:** Given a resolution, the overhead is monotonically increasing and convex with respect to  $p$ .

The second term in overhead is constant for a given resolution. Hence we have to show that the first term is monotonically increasing. I.e.

$Np \left( 2 - p \frac{R^2 - r^2}{R^2} \right)$  is monotonically increasing in  $p$ .

$$27. \quad Np \left( 2 - p \frac{R^2 - r^2}{R^2} \right) = 2Np - Np^2 \left( 1 - \frac{r^2}{R^2} \right) = 2Np(1 - p) + Ap^2 \frac{d}{R^2}$$

Again the first term in equation 28 is monotonically increasing. We approximate the second term as follows:

$$28. \quad p^2 d \approx \left( \frac{\ln(d) - \ln(\ln(d))}{d} \right)^2 d = \frac{(\ln(d) - \ln(\ln(d)))^2}{d}$$

The above is monotonically decreasing for  $d$  (i.e. increasing w.r.t  $p$ ). Hence equation 28 is monotonically increasing w.r.t  $p$ . We omit the derivation for convexity of the overhead function for lack of space.

**Theorem 1:** To retrieve topology at a required resolution  $T_{res}$ , the minimum overhead is achieved by selecting parameter  $r$  and  $f$  as follows:

- If  $T_{res} < X(p(R), I)$ , put  $r=R$ , and find  $f$  such that  $X(p(R), f) = T_{res}$
- If  $T_{res} \geq X(p(R), I)$ , then put  $f=I$ , and find  $r$  such that  $X(p, f) = T_{res}$ .

Thus the general rule is to first keep  $r=R$  and increase  $f$  up to 1 and then decrease  $r$  until we get the required resolution.

**Proof:** The proof of theorem 1 is the solution to the following non-linear optimization problem:

$$\text{Min}(O_T(p, f))$$

$$29. \quad \begin{aligned} \text{s.t. } & X_T(p, f) = T_{res} \\ & 0 \leq f \leq 1, \quad p(R) \leq p \leq 1 \end{aligned}$$

We see here that constraints of the algorithm are actually hidden in the above formulation. The above formulation ensures that at least a spanning tree of the network would be formed (*since*  $\min(p)=P(R)$ ) and all the

From equation 28, we see that the above optimization can be reduced to the following:

$$\text{Min}[F(p)]$$

$$\text{where} \quad F(p) = p \left( 2 - p \frac{R^2 - r^2}{R^2} \right)$$

$$30. \quad \begin{aligned} \text{s.t. } & X_T(p, f) = T_{res} \\ & 0 \leq f \leq 1, \quad p(R) \leq p \leq 1 \end{aligned}$$

The monotonicity of  $F(p)$  makes the solution even simpler. We have to find minimum  $p$ , for which  $X(p, f)=T_{res}$ . Since  $X(p, f)$  is also monotonic, first put  $p=p(R)$  (the minimum value of  $p$ ), and find  $f$  such that  $X(p(R), f)=T_{res}$ . If  $X(p(R), 1) < T_{res}$  i.e. maximum value of  $X$  such that  $p=p(R)$ , is less than the required resolution, find  $p(r)$ , by solving  $X(p(r), 1)=T_{res}$  for  $p(r)$ . Since the overhead function is convex, the solution is also a global maxima.

**Theorem 2:** Given an overhead  $O_M$ , to retrieve topology at the maximum possible resolution  $T_{res}$ , select parameter  $r$  and  $f$  as follows:

- Put  $r=R$ , and find  $f$  such that  $O_T(p(R), f) = O_M$
- If  $O_T(p(R), f) < O_M$ , then put  $f=1$ , and find  $p(r)$  such that  $O_T(p(r), 1) = O_M$ .

Thus the general rule is to first keep  $r=R$  and increase  $f$  up to 1 and then decrease  $r$  to get higher resolutions until the overhead constraint is met..

**Proof:** The proof of theorem 2 is the solution to the following non-linear optimization problem:

$$\text{Max}(X(p, f))$$

$$31. \quad \begin{aligned} \text{s.t. } & O_T(p, f) = O_M \\ & 0 \leq f \leq 1, \quad p(R) \leq p \leq 1 \end{aligned}$$

We write the Lagrangian for the above formulation:

$$\begin{aligned} 32. \quad \max L(p, f, \mathbf{m}) &= X(p, f) + \mathbf{m}[O_T] = X(p, f)(1 + \mathbf{m}) + \mathbf{m}F(p) + \mathbf{m}C_1 \\ &\equiv \text{Max}[F(p) + \mathbf{m}X(p, f) + \mathbf{m}'C_1] \end{aligned}$$

Since the functions  $X(p, f)$  and  $F(p)$ , are convex and monotonic, we have reduced the optimization problem in equation 28 to the dual problem given in equation 33. From Lagrangian duality theorem for convex functions, if  $(p_o, f_o)$  is the solution to primal problem in equation 29 (*minimize*  $F(p)$ ), then there exists  $\mathbf{m}_0 \in 0$  such that  $(p_o, f_o, \mathbf{m}_0)$  solves the dual problem in equation 34. Hence we get the necessary rules for selecting parameters.

## 8. ACKNOWLEDGEMENTS

This research work was supported in part by DARPA under contract number N-666001-00-1-8953 and a grant from CISCO systems. We would like to thank Dr. S. Muthu Muthukrishnan, his invaluable comments to improve the technical quality and presentation of this work. We would also like to thank Dr. Mario Szegedy with whom we had discussions regarding some of the analysis in this paper.

## 9. REFERENCES

J.M. Kahn, R.H. Katz, K.S.J. Pister (1999), *Next century challenges: Mobile networking for 'smart dust'*, Proc. MOBICOM, 1999, Seattle, 271-278.

G.J. Pottie and W.J. Kaiser (2001), *Wireless Integrated Network Sensors*. Communications of the ACM, vol. 43 no. 5 (2001), 51-58.

A. Downey (1999), "*Using pathchar to estimate Internet link characteristics*," Proc. SIGCOMM 1999, Cambridge, MA, pp. 241-250, Sept. 1999.

Jerry Zhao, Ramesh Govindan and Deborah Estrin (2002), "*Residual Energy Scans for Monitoring Wireless Sensor Networks*", IEEE Wireless Communications and Networking Conference, 2002.

C. Intanagonwiwat, R. Govindan and D. Estrin (2000); *Directed diffusion: a scalable and robust communication paradigm for sensor networks*; in Proceedings of Mobicom '00, 2000.

Cormen, Leiserson, Rivest (1990) *Introduction to Algorithms*, MIT Press, McGraw Hill.

Bruce Lowekamp, David R. O'Hallaron, and Thomas R. Gross (2001), "*Topology Discovery for Large Ethernet Networks*," In Proceedings of ACM SIGCOMM August 2001 (San Diego, California), ACM Press.

Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri and A. Silberschatz.(2000) "*Topology Discovery in Heterogeneous IP Networks*" In Proceedings of IEEE INFOCOM, 2000.

Nancy Miller and Peter Steenkiste (2000), "*Collecting Network Status Information for Network-Aware Applications*"In Proceedings of IEEE INFOCOM 2000.

David B Johnson and David A Maltz (1996), "*Dynamic Source Routing in Ad Hoc Wireless Networks*", Mobile Computing, Volume 353, Kluwer Academic Publishers, Edited by Imielinski and Korth.

Seapahn Meguerdichian, Farinaz Koushanfar, Gang Qu, Miodrag Potkonjak (2001a), "*Exposure In Wireless Ad Hoc Sensor Networks.*" Proc.. of 7th Annual International Conference on Mobile Computing and Networking, MOBICOM 2001.

Seapahn Meguerdichian, Farinaz Koushanfar, Miodrag Potkonjak, Mani Srivastava (2001b), "*Coverage Problems in Wireless Ad-Hoc Sensor Networks,*"In Proc. of IEEE INFOCOM 2001.

Bhaskar Krishnamachari, Stephen B. Wicker, and Ramon Bejar (2001), "*Phase Transition Phenomenon in Wireless Ad hoc Networks*", Symposium on Ad-Hoc Wireless Networks, GlobeCom2001, San Antonio, Texas, November 2001.

L. Li, Z. Haas and J.Y. Halpern (2002), "*Gossip-Based Ad Hoc Routing*", IEEE INFOCOM, June 2002.

Romit RoyChoudhury, S. Bandyopadhyay and Krishna Paul (2000), "*A Distributed Mechanism for Topology Discovery in Ad hoc Wireless Networks using Mobile Agents*", Proc. of Workshop On Mobile Ad Hoc Networking & Computing (MOBIHOC 2000).

Baruch Awerbuch and Yuval Shavitt (2001), "*Topology Aggregation for Directed Graphs.*", IEEE/ACM Transactions on Networking, Vol.9, N0.1, Feb 2001.

B. Das, V Bhargavan (1997), "*Routing in Ad Hoc Networks Using Minimum connected dominating Sets*". IEEE International Conference on Communications (ICC '97), Jun, 1997.

- A. Ephremides, J.E. Wieselthier, and D.J. Baker (1987), "A design concept for triable mobile radio networks with frequency hopping signaling." Proceeding of IEEE, 75, 1 (Jan 1987) 56-73.
- M. Gerla, J.T. C. Tsai (1995), "Multicluster, mobile, multimedia radio networks," ACM J. Wireless Networks , 1, 3 (1995) 255-265.
- S. Guha and S. Khuller (1996), "Approximation Algorithms for Connected Dominating Sets," European Symposium on Algorithms. 179-193, 1996.
- P. Sinha, R. Sivakumar and V. Bharghavan (1999), "CEDAR: a Core-Extraction Distributed Ad hoc Routing Algorithm" In proceedings of IEEE Infocom 1999.
- Suman Bannerjee, Samir Khuller (2001), "A Clustering Scheme for Hierarchical Control in Wireless Networks", In Proceedings of IEEE INFOCOM 2001.
- B. Das, R. Sivakumar and V. Bharghavan (1997). "Routing in ad hoc networks using a virtual backbone." In Proc. IEEE (IC3N'97).
- Cristian Borcea, Deepa Iyer, Porlin Kang, Akhilesh Saxena, Liviu Iftode (2002), "Cooperative Computing for Distributed Embedded Systems", International Conference of Distributed Computing Systems, 2002.
- B. Clark , C. Colbourn and D. Johnson (1990), "Unit disk graphs", Discrete Mathematics, vol. 86, pp. 165--177, 1990.
- A. Bestavros and J. Byers and K. Harfoush (2002), "Inference and Labeling of Metric-Induced Network Topologies" In Proceedings of Infocom'02: The IEEE ICC, June 2002.
- J.E. Wieselthier, G.D. Nguyen, and A. Epremedes (2000), "On the construction of energy-efficient broadcast and multicast trees in wireless networks" In IEEE INFOCOM 2000, Tel Aviv, Israel 2000.
- Deepak Ganesan and Deborah Estrin (2002), "DIMENSIONS: Why do we need a new Data Handling architecture for Sensor Networks" , In Proceedings of the ACM Workshop on Hot Topics in Networks, HotNets-2002.
- Yuan-Chieh Cheng and Thomas G. Robertazzi (1989), "Critical Connectivity Phenomenon in Multihop Radio Models", In IEEE Transactions on Communications, Vol. 37, No. 7, July 1989.

Kleinrock, and J. Silvester (1978), "*Optimum Transmission Radii for Packet Radio Networks or Why Six is a Magic Number*," Proceedings of the IEEE National Telecommunications Conference, pages 4.3.1--4.3.5, Birmingham, Alabama, 1978.

Marathe, M. V., Breu, H., Hunt III, H. B., Ravi, S. S., and Rosenkrantz (1995), D. J. (1995), "*Simple heuristics for unit disk graphs*", Networks 25, 59-68.

Hunt III, H. B., Marathe, M. V., Radhakrishnan, V., Ravi, S. S., Rosenkrantz, D. J., and Stearns, R. E. (1998), "*Nc-approximation schemes for NP- and PSPACE-hard problems for geometric graphs*", J. Algorithms 26, 238-274.

A. Scaglione, S. D. Servetto.(2002) "*On the Interdependence of Routing and Data Compression in Multi-Hop Sensor Networks*. In the Proceedings of the 8th ACM International Conference on Mobile Computing and Networking (MobiCom), Atlanta, GA, September 2002.

S.Y. Ni, Y.C. Tseng, Y.S. Chen, and J.P. Sheu (1999), "*The Broadcast Storm Problem in a Mobile Ad hoc Network*," Int. Conf. on Mobile Computing and Networking (MobiCom'99), pp. 151-162, 1999.

Budhaditya Deb, Sudeept Bhatnagar and Badri Nath (2003a), "*STREAM: Sensor Topology Retrieval at Multiple Resolutions*", DCS Technical Report DCS-TR-515, Rutgers University Jan 2003.

Budhaditya Deb, Sudeept Bhatnagar and Badri Nath (2003b), "*Multi-Resolution State Retrieval in Sensor Networks*", In proceedings of First IEEE workshop on Sensor Network Protocols and Applications, SNPA, ICC May, 2003 .

Budhaditya Deb, Sudeept Bhatnagar and Badri Nath (2002), "*A Topology Discovery Algorithm for Sensor Networks with Applications to Network Management*", In IEEE CAS workshop, September 2002