

A Topology Discovery Algorithm for Sensor Networks with Applications to Network Management

Abstract—In this paper we describe a topology discovery algorithm (*TopDisc*) for wireless sensor networks with its applications to network management. The algorithm finds a set of distinguished nodes, using whose neighborhood information we can construct the approximate topology of the network. Only these distinguished nodes reply back to the topology discovery probes, thereby reducing the communication overhead of the process. These nodes logically organize the network in the form of clusters comprised of nodes in their neighborhood. *TopDisc* forms a Tree of Clusters (*TreC*) rooted at the monitoring node, which initiates the topology discovery process. We show how managing a complex network of sensor nodes is simplified using a *TreC*. This organization is used for efficient data dissemination and aggregation, duty cycle assignments and network state retrieval. The mechanisms proposed are completely distributed, use only local information and are highly scalable.

I. INTRODUCTION

Recent advances in MEMS technology have resulted in cheap and portable devices with formidable sensing, computing and wireless communication capabilities [1][14]. A network of these devices could be invaluable for automated information gathering and distributed micro-sensing in many civil, military and industrial applications. The use of wireless medium for communication provides a flexible means of deploying these nodes without any fixed infrastructure, possibly at some inhospitable terrain. Once deployed, they should require minimal external support for their functioning.

Wireless sensor networks pose many new challenges primarily because the sensor nodes are resource constrained. Energy is constrained by the limited battery power in sensor nodes. The form factor is an important node design consideration for easy operability and ad-hoc deployment of these nodes, which limit the amount of resources to be put in a node. Thus the protocols and applications designed for sensor networks should be highly efficient and optimized in terms of the resources they consume. A lot of work has already been done on various low power MAC Protocols [2], Power Aware Routing algorithms [3], Energy Efficient Communication Protocols [4][5] and lightweight system support [12].

Routing and data dissemination problem in sensor networks can be viewed as a special case of Mobile ad-hoc networks and many routing algorithms exist [9]. Use of location has also been used effectively for this problem [10]. However for sensor networks we need a much simpler and more scalable solution. Directed diffusion [11] deals with this aspect but can be computationally intensive

Emerging sensor network architectures envision massively distributed and highly complex network systems comprised of hundreds of tiny sensor nodes. These nodes would have various modes of operation, maintaining only local knowledge of the network for scalability and may also have networking functionalities like routing to cooperatively maintain network connectivity. The behavior of the network

would be highly unpredictable because of randomness in individual node state and network structure. Once these networks are deployed, we would be frequently faced with questions on how to manage such systems.

Performance analysis and management of these networks is a non-trivial task because of the above reasons. Network management protocols for wired networks work on certain network models and provide an administrator with information about the state of the network [6]. Similarly, managing sensor networks would involve various network administration activities based on certain network models to depict the network state. However, the underlying network model and the management functions would be significantly different since these networks have a fundamentally different architecture from normal wired networks. Thus functionalities provided in SNMP[6] may not be useful in these architectures.

In this paper, we introduce some of the issues associated with sensor network management. We describe possible network models on which the protocols can operate and different management functions particular to sensor networks. We propose an efficient Topology Discovery Algorithm (*TopDisc*) for sensor networks with its applications to data dissemination and aggregation, duty cycle assignments and network state retrieval. Network Topology is an important model of the network state as it implicitly gives a lot of information about the active nodes present and the connectivity/reachability map of the system. Many topology discovery algorithms exist for wired networks but most of them introduce a lot of additional traffic into the network [7]. Some others use network models suitable for the internet based on power laws of internet topology[8]. These may not be useful for sensor networks.

The *TopDisc* algorithm is based on the fact that wireless is a broadcast medium of communication. Nodes can know the existence of other nodes in its communication range, just by listening to the communication channel. The algorithm takes advantage of this fact to find a set of distinguished nodes; using whose neighborhood information we can construct the approximate topology of the network. Only distinguished nodes reply back to the topology discovery probes, thereby reducing the communication overhead of the process. These distinguished nodes form clusters comprised of nodes in their neighborhood. These clusters are arranged in a tree structure called *TreC*, rooted at the monitoring node (or the initiating node).

The *TreC* represents a logical organization of the nodes and provides a framework for managing sensor networks. Using only local information between adjacent clusters, information flows from nodes in one cluster to nodes in a cluster at a different level in the *TreC*. The clustering also provides a mechanism to assign node duty cycles such that a minimal set of nodes are active in maintaining the network connectivity. The cluster heads incur only minimal extra

overhead of setting up the structure and maintaining local information about its neighborhood.

The paper is organized as follows: In section 2 we discuss sensor network management issues, network models to represent states and other management functions. Section 3 classifies and details various methods of topology discovery. Section 4 describes clustered response scheme for topology discovery. We describe different applications of TopDisc for routing, data dissemination and aggregation and assigning duty cycles to nodes in Section 5. The performance issues of the proposed schemes are evaluated in section 6. We mention related work in section 7 and conclude by giving future directions for research in section 8.

II. SENSOR NETWORK MANAGEMENT

Sensor networks have fundamentally different architecture than normal wired data networks. Nodes are designed with low cost and small form factor for easy deployment in large numbers. Hence limited memory, processor and battery power is provided to them. Energy constraints also limit the communication range of these devices. These nodes would have various modes of operation with different levels of active and passive state for energy management. They would maintain only local knowledge of the network as global information storage would not be scalable and may have networking functionalities like routing to cooperatively maintain network connectivity.

The behavior of the network could be highly unpredictable because of the operating characteristics of the nodes and the random manner in which the network is set up. Hence all algorithms should consider failure of network as a rule rather than as an exception and should handle them more efficiently. Also algorithms for performance analysis and protocols for network management have to be looked with significantly different perspective.

The above factors imply that we need different models that depict the current state of the network. A sensor network model has to incorporate the specific features described above. Some of the possible models for sensors are:

- *Network Topology*: This describes the current connectivity/reachability map of the network and could assist routing operations and in future deployment of nodes.
- *Energy Map*: This gives the energy levels of the nodes at different parts of the network. The spatial and temporal energy gradient of the network nodes may also be modeled. Coupled with network topology, this could be used to identify “weak areas” of the network.
- *Usage Pattern*: This describes the network activity in terms of periods of activity for nodes, amount of data transmitted per unit time and tracking of hot spots in the network.
- *Cost Model*: This represents the network in terms of equipment cost, energy cost, and human cost for maintaining the network at desired performance level.

- *Non-deterministic Models*: Sensor networks are highly unpredictable and unreliable. Statistical and probabilistic models could prove to be much more effective in estimating network behavior than deterministic models.

The above models would form the Management Information Base (MIB) for sensor networks. To update the MIB with the current state of the network, a monitoring node needs to measure various network parameters. Measurements in general would have spatial and temporal error. To get more accurate state, measurement probes have to be done at a finer granularity. Since any probe would use up finite amount of energy from the system, this can actually change the state of the network. So the more precision we require the more the whole process is going to change the state. This we term as the *Uncertainty Principle for Sensor Networks*. This is true for all systems but since sensor networks are resource constrained, the effects could be quite significant.

The above models could be used for different network management functions. We define some of these functions as follows:

- *Deployment of sensors*: Typically sensors would be deployed at random with no prior knowledge of the terrain. Future deployment of sensors would depend upon the present state of the network.
- *Setting Network Operating Parameters*: This involves setting up of routing tables, node duty cycles, timeout values of various events, position estimation etc. These differ from normal monitoring operations because many protocols may require accurate information of the above for their optimal performance.
- *Monitor Network States using Network Models*: Take periodic measurements to obtain various states like network connectivity, energy map etc.
- *Network Maintenance*: By monitoring the network, regions of low network performance could be traced with reasons for such performance could be identified. Corrective measure like deployment of new sensors or directing network traffic around those regions could be useful. This is *Reactive Network Maintenance*.
- *Predict Future Network States*: From periodic measurement of network states it could determine the dynamic behavior of the network and predict future state. This could be useful for predicting network failures and preventive action could be taken. This is *Proactive Network Maintenance*.
- *Design of Sensor Networks*: The models on Cost Factor and Usage Patterns could be used for design of sensor network architectures.

The models and the functions enlisted in this section is a preliminary inspection of the issues involved. We expect a more formal and well-defined framework for sensor network management in future.

III. TOPOLOGY DISCOVERY

In this section we describe the topology discovery algorithms used in sensor networks. Our aim is to construct the topology of the whole network from the perspective of a single node. We divide the algorithms in three stages of execution.

- A monitoring node requiring the topology of the network initiates a "topology discovery request".
- This request diverges throughout the network reaching all active nodes.
- A response action is set up which converges back to the initiating node with the topology information.

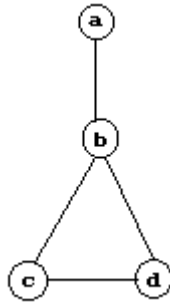


Figure 1. Example illustrating difference between topology discovery approaches

We assume that the request divergence is through controlled flooding so that each node forwards exactly one "topology discovery request". Note that each node should send out at least one packet for other nodes to know its existence. This also ensures that all nodes receive a packet if they are connected. However various methods may be employed for the response action. The different mechanisms described, differ only in this aspect and affect the overall efficiency of the process.

A. Overview of TopDisc Approaches

We consider three methods, which may be used topology discovery. We note that when topology discovery request diverges, every node gets information about neighboring nodes. In the response action each node can reply back with its neighborhood list. For illustrating the response action of these methods we consider the network in figure 1 with node *a* as the initiating node. The topology discovery request reaches node *b* from *a* and nodes *c* and *d* from node *b*. Requests are forwarded only once such that no action takes place even though node *c* and *d* may hear requests from each other.

1. Direct Response:

In the first approach we flood the whole network with the topology discovery request. When a node receives a topology discovery request it forwards this message and sends back a response to the node from which it received the request. The response action for the nodes in figure 1 is:

- Node *b* replies back to node *a*.
- Node *c* replies to node *b*; node *b* forwards the reply to node *a*.
- Node *d* replies to node *b*; node *b* forwards the reply to node *a*.
- Node *a* gets the complete topology

We note that even though parent nodes can hear the children while they forward a request (for example *a* can know about *b* when it forwards), this is not useful as its neighborhood information is incomplete. Hence an exclusive response packet is needed for sending the neighborhood information.

2. Aggregated Response:

Again all active nodes send a topology discovery request but wait for the children nodes to respond before sending their own responses. After forwarding a topology discovery request, a node gets to know its neighborhood list and children nodes by listening to the communication channel. Once this is set up it waits for responses from its children nodes. On receiving them it aggregates the data and sends it to its own parent. The response action for the nodes in figure 1 is:

- Node *c* and *d* forward request; node *b* listens to these and deduces them to be its children.
- Node *c* replies back to node *b*; Node *d* replies back to node *b*.
- Node *b* aggregates information from *c*, *d* and itself; node *b* forwards the reply to node *a*.
- Node *a* gets the complete topology

3. Clustered Response:

In this approach the network is divided into set of clusters. Each cluster is represented by one node (called the cluster head) and each node is part of at least one cluster. Thus each node is in range of at least one cluster head. The response action is generated only by the cluster heads, which send information about nodes in its cluster. Similar to aggregated response, cluster heads can aggregate information from other cluster heads before sending response. The response action for the nodes in figure 1 is:

- Assume that node *b* is a cluster head and nodes *c* and *d* are part of its cluster.
- Node *c* and *d* do not reply.
- Only node *b* replies to node *a*.
- Node *a* does not get link $c \leftrightarrow d$.

We note that information may be incomplete with the clustered response approach. Direct and aggregated response methods give a very accurate picture of the network topology. The clustered response gives a *reachability map* where if all cluster heads are reachable then all other nodes are reachable from at least one cluster head.

On the other hand the overhead incurred in topology discovery by clustered response approach would be significantly less than the direct or aggregated response

approaches. In the next section, we focus on the clustered response approaches. We first give a formal definition of the problem and then describe two variations the *TopDisc* algorithm.

IV. CLUSTERED RESPONSE APPROACHES

We consider the sensor network to be an undirected graph $G(V, E)$ with vertices V and edges E . In clustered response approaches, the nodes in the graph are divided into sets of nodes. A cluster head represents each set and the nodes in its set belong to its neighborhood.

Let C be the set of cluster heads.

Let V_i be the neighborhood list of node i , with $i \in C$

Then following conditions hold

1. $V = \cup V_i$
2. $\forall x \in V_i, \text{edge}(x, i) \in E$

We note that the communication overhead for clustered response approach is dependent on the number of clusters that are formed and the path length connecting the clusters. Thus for minimum communication overhead we need to solve the following problems.

- Find a minimum cardinality set of cluster heads, which has to reply back: *Set Cover Problem*
- Form a minimal tree with the set of the cluster heads: *The Steiner Tree Problem*.

The above are combinatorial optimization problems. Moreover for optimal solution we need to have global information about the network whereas the nodes only have local information. We give heuristics, which provide approximate solutions to the problems. Our proposed algorithm is simple and completely distributed and thus can be applied to sensor networks.

Our *TopDisc* algorithm for finding the cluster heads is based on the simple greedy *log(n)-approximation* algorithm for finding the set cover [13]. In our approach at each stage a node is chosen from the discovered nodes, which should cover the maximum remaining undiscovered nodes. Since in the case of topology discovery the neighborhood sets and vertices in the graph are not known at runtime we cannot have a straightforward implementation of the algorithm. Instead the neighborhood sets have to be generated as the topology discovery request propagates through the network. We use two different node-coloring approaches to find the set of cluster heads during request propagation: the first one uses three colors and the second one uses four colors. The response generation mechanism is same for both cases.

B. Request Propagation with three colors

In this version, to find the cluster head subset, we use three colors. The different colors and their definitions are given below. All nodes, which receive a topology discovery request packet and are alive, to respond are considered as discovered nodes.

- **White:** Yet undiscovered node, or node, which has not received any topology discovery packet.
- **Black:** Cluster head node, which replies to topology discovery request with its neighborhood set.
- **Grey:** Node which is covered by at least one black node i.e. it is neighbor of a black node.

Initially all nodes are white. When the topology discovery request propagates, each node is colored black or gray according to their definition. At the end of the initial phase of the algorithm, each node in the network is either a black node or a neighbor of a black node (i.e. grey node). All nodes broadcast a topology discovery request packet exactly once in the initial phase of the algorithm. Thus all nodes have the neighborhood information just by listening to these transmissions. Thus nodes have the neighborhood lists available before the topology acknowledgement is returned.

We use two heuristics by which we try to get the next neighborhood set determined by a new black node, which should cover maximum number of uncovered nodes: The first is using a node coloring mechanism to find the required set nodes. The second is using a forwarding delay inversely proportional to the distance between receiving and sending node. These two heuristics provide a solution quite near to the centralized greedy set cover solution. The process is described below:

- The node which initiates the topology discovery request is assigned color black and broadcasts a topology discovery request packet.
- All white nodes become grey nodes when they receive a packet from a black node. Each grey node broadcasts the request to all its neighbors with a random delay inversely proportional to its distance from the black node from which it received the packet.
- When a white node receives a packet from grey node, it becomes a black node with some random delay. In the meantime if it receives any packet from some other black node, it becomes a grey node. Again the random delay is inversely proportional to the distance from the grey node from which the request was received.
- Once nodes are grey or black, they ignore other topology discovery request packets.

A new black node should be chosen so that it covers the maximum number of yet uncovered elements. This is achieved by having a forwarding delay inversely proportional

to the distance between sending and receiving node. The heuristic behind having a forwarding delay inversely proportional to distance from the sending node is explained as follows: The coverage region of each node is the circular area centered at the node with radius equal to its communication range. Then the number of nodes covered by a single node would be proportional to its coverage area times the local node density. The number of new nodes covered by a forwarding node is proportional to its coverage area minus the already covered area. This is illustrated in Figure 2. Here node *a* makes nodes *c* and *b* grey. Node *b* forwards before node *c* due to which more new nodes get the request. The delay would make node *d* more likely to be black than node *e*. Also the new nodes covered by *d* is likely to be more than that covered by *e*. We also note that an intermediate node between two black nodes (node *b* in figure 2) is always within range of both the black nodes since three colors were used for their formation.

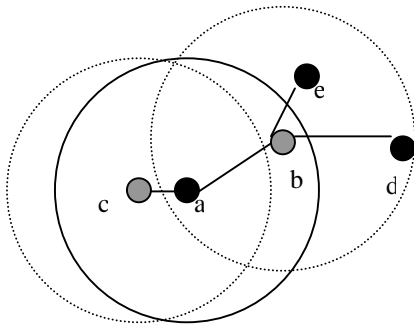


Figure 2. Illustration for the delay heuristic for 3 colors.

C. Request Propagation with four colors

To find the cluster heads we use four colors in this approach. The different colors and their definitions are given below. All nodes, which receive a topology discovery request packet and are alive, to respond are considered as discovered nodes.

- **White:** Yet undiscovered node, or node, which has not received any topology discovery packet.
- **Black:** Cluster head node, which replies to topology discovery request with its neighborhood set.
- **Grey:** Node which is covered by at least one black node i.e. it is neighbor of a black node.
- **Dark Grey:** Discovered node, which currently is not covered by any neighboring black node and hence is two hops away from a black node. White node changes to dark grey on receiving a request from grey.

This method propagates similar to the previous method. Initially all nodes are white. When the topology discovery request propagates, each node is colored black, gray or dark grey according to their definition. Thus at the end all nodes in

the network are either black nodes or a neighbors of a black nodes (i.e. gray nodes). This is explained as follows:

- The node which initiates the topology discovery request is assigned color black and broadcasts a topology discovery request packet.
- All white nodes become grey nodes when they receive a packet from a black node. These grey nodes broadcast the request to all their neighbors with a delay inversely proportional to its distance to the black node from which it received this request.
- When a white node receives a packet from grey node it becomes dark grey. It broadcasts this request to all its neighbors and starts a timer to become a black node. The forwarding delay is inversely proportional to its distance to the grey node from which it received this request.
- When a white node receives a packet from dark grey node, it becomes a black node with some random delay. In the meantime if it receives any packet from some other black node, it becomes a grey node.
- A dark grey node waits for some time so that one of its neighbors becomes black. When the timer expires it becomes a black node itself because there is no black node to cover it.
- Once nodes are grey or black they ignore other topology discovery request packets.

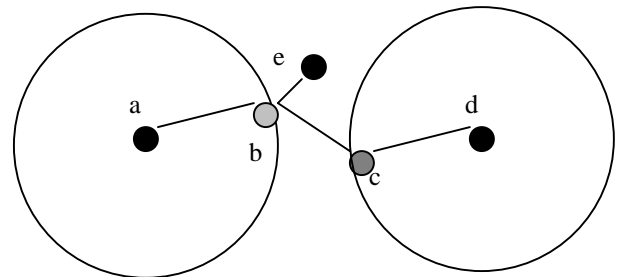


Figure 3. Illustration for the delay heuristic & 4 colors.

The heuristic behind having four colors for the algorithm is explained using Figure 3: A new black node should be chosen so that it covers the maximum number of yet uncovered elements. For this we want black nodes to be separated from each other by two hops so that nodes belong to only one black node neighborhood (like node *a* and *d*). This may not be possible in all cases and some black nodes are formed just one hop away from another (node *e* in figure 3). The heuristic behind the forwarding delay principle is same as for three colors.

The number of clusters formed by this approach is slightly less than with three colors. Clusters are formed with lesser overlap than the three-color approach. There are some solitary black nodes (dark grey nodes which time out to

become black), which do not need to cover any of its neighbors. Thus even though number of black nodes is similar to three-color case, the number of bytes transmitted is lower. However the three-coloring approach generates a *TreC*, which is more amenable to the network management applications we describe later. Hence we do not give detailed performance evaluation for this method.

D. TopDisc Response Mechanism

The first phase of the algorithm sets up the node colors. The initiating node becomes the root of the black node tree where the parent black nodes are at most two hops away (using 4 colors) and one hop away (using 3 colors) from its children black node. Each node has the following information at the end of this period:

- A clusters is identified by the black node, which heads it.
- A grey node knows its cluster id
- Each node knows its *parent black node*, which is the last black node from which the topology discovery was forwarded to reach it.
- Each black node knows the default node to which it should forward packets to reach the parent black node. This node is essentially the node from which it received the topology discovery request.
- All nodes have their neighborhood information.

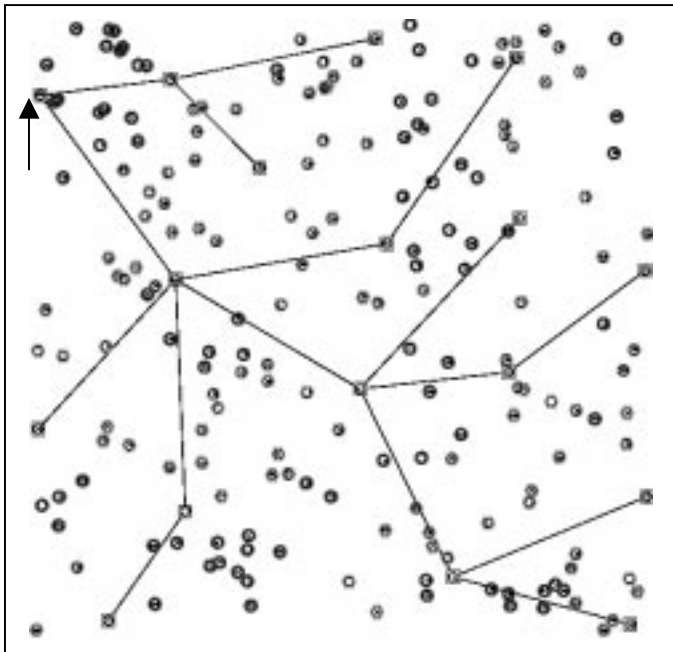


Figure 4. Illustration of typical TreC for 200 nodes and range 20m

Using the above information, the steps for *TopDisc Response* are described as follows.

- When a node becomes black, it sets up a timer 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 neighborhood lists from its children and itself and when its time period for acknowledgement expires, forwards the aggregated neighborhood list to the default node to its parent.
- All forwarding nodes in between black nodes may also add their adjacency lists to the list from black nodes.

Here we note that for the algorithm to work properly, timeouts of acknowledgements should be properly set. For example timeouts of children black nodes should always expire before a parent black node. For this we set a timeout value inversely proportional to the number of hops a black node is away from the monitoring node. Again, for this to work 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. However, if that information is not available to nodes, we can assume that topology discovery runs in stages where it discovers only a certain extent of area at each stage. A typical *TreC* obtained by *TopDisc* is shown in figure 4. The example shows a 100x100 sq.m area with 200 nodes and communication range 20m. The arrow represents the initiating node. We note the following characteristics of the clusters

- The total surface area and the communication range of nodes bound the maximum number of black nodes formed.
- Number of nodes in each cluster depends on the local density of network
- Depth of tree is bounded.
- Routing paths are near optimal for data flow between source (sensor nodes) and sink (monitoring node).

We compare the above characteristics with a centralized solution and find out to what extent approach deviates from optimal values.

E. Handling Channel Errors

The mechanisms described above assume a zero error rate for channels. However we have to make minor changes to the protocols to account for dropped packets due to channel errors.

We note that the topology discovery initially floods to the whole network. Hence channel error would not be a problem as long as topology discovery request reaches a packet from any paths. Since sensor networks under consideration here are dense with many paths existing between source and destination, channel error does not create a significant impact. The number of black nodes formed may be increased due to packet losses.

However topology acknowledgement packets are returned through single prescribed paths and hence packets may be lost. Also since our algorithm decreases the redundancy of topology information being propagated among different packets, the loss of one single packet might be quite significant. As the packets are aggregated while moving up the cluster tree, the magnitude of loss might increase.

This problem has a simple solution if we assume that all links are symmetrical. This means that if node a can listen to b when b is transmitting, then node b can listen to a when a transmits. When a topology discovery response has to be sent from node a , it forwards the packet to its default node (say node b). The node b upon receiving this packet would again forward to its own default next hop to parent. Now a can listen to any packet forwarded by b and hence would be able to know whether b forwarded the same packet. If a does not hear such a packet, it retransmits the packet assuming that b never received the packet due to channel error.

Thus again we see that eavesdropping can be used as an indirect *ack* mechanism for reliable transmission. The only added overhead for this simple method is that every forwarded packet has to be stored at a node till the packet is reliably transmitted. Also node spends some energy in listening to transmissions and cannot switch itself off immediately after forwarding a packet.

V. APPLICATIONS OF TOPOLOGY DISCOVERY

In this section, we describe some applications of *TopDisc* to certain sensor network management issues.

A. Retrieving Network State

The main purpose of the topology discovery process is to provide the network administrator with the network topology. The mechanisms described in this paper can provide four different types of network topology.

- *Connectivity Map*: The direct response and the aggregated response mechanisms provide the entire connectivity map of the region. Note that clustered response methods cannot provide this information.
- *Reachability Map*: The *TopDisc* mechanism provides a reachability map of the region. We note that a connectivity map is a superset of the reachability map.
- *Energy Model*: When a node forwards the topology discovery request, it can include its available energy in the packet. Each node can cache the energy information of all its neighbors. If a node doesn't become black, it can discard the cached value. Thus all black nodes have energy information about all their neighbors, which can be sent as part of the reply. A black node can also estimate the energy consumption of nodes in its cluster by listening to the transmitted packets.
- *Usage Model*: Like in the previous case, each node can transmit number of bytes received and transmitted by it during the last t minutes. A black node will have this information cached at the time it sends its response.

Thus our simple *TopDisc* algorithm can provide different views of the network to the user.

B. Data Dissemination and Aggregation

We assume that in sensor networks all information flow would be from sensor to monitoring node with some control

information being transmitted from monitoring node to sensors. The topology discovery process sets up a *TreC* rooted at the initiating node. Thus any data flow from a sensor to monitoring node has to flow up the *TreC*.

Each cluster has a minimal number of nodes, which are active to transfer packets between a parent-child cluster pair. We describe how duty cycles of active nodes are assigned in the next subsection. Whenever a sensor needs to send some data to the monitor, it can just wake up and broadcast. The duty cycle assignment mechanism ensures that there is at least one node, which is active and responsible for forwarding the data to the next cluster. Also there is at least one node in the next cluster active to receive this packet.

We see that each black node covers a region given by its communication range. The parent black node, logically also covers the area covered by its children black nodes. Thus the area covered propagates up the tree and the monitor covers the whole field. The area covered by each black node may be cached during the topology response phase. The parent black node gets such areas from its children and in turn makes larger area to approximate its logical coverage region. Region based queries from the monitor node can be channeled to the appropriate region by the black nodes using their coverage information. At the return path the data may be aggregated at the black nodes.

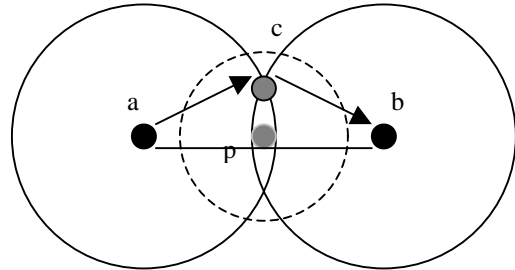


Figure 5. Assigning up the duty cycle with location information..

C. Duty Cycle Assignment

In this section we describe the mechanism for setting up the duty cycle of nodes for data forwarding. Each node in any cluster has at least two kinds of information: the *cluster id* and the *parent black node*, which is the last black node from which, the topology discovery request had been forwarded. In each cluster using this information, sets of nodes between two clusters are chosen which can forward packets between clusters. At least one node in each set is active at a given time to maintain a link between a parent/child cluster pair. We show two kinds of mechanisms by which this can be done: the first one assumes that nodes know their geographic location and in the second one we relax this condition.

1. Assignment with Location Information:

In this case we assume that nodes have knowledge about their geographical location. After a black node has sent a

topology acknowledgement, it has knowledge of both its parent black node and children black node. Here it also has location information about them. Using this information, sets of nodes for each parent/child pair, need to be set up such that in any set only one node needs to be active to transfer or receive packets from particular clusters. Consider Figure 5 for description of the mechanism.

Figure 5 shows a general case in which a cluster (with black node b) may be formed as child of another cluster (with black node a). Since we are using three colors to set up the *TreC*, there would be one intermediate node between the clusters (node c).

The communication range of nodes is equal to R . If we consider a circular area with radius $R/2$ (shown by the dotted circle), nodes inside this region would always form a completely connected graph, as each would be within communication range of others. We consider such a region centered at the midpoint (point p) of a parent/child cluster pair. If there is at least one node active in both clusters inside this region, then there is always a way to forward a packet from one cluster to the other. The various steps for setting up the sets of nodes are given below:

- Black nodes send a packet with information about its parent cluster and children clusters to all its neighbors. This also contains the locations of the black nodes heading the respective clusters.
- Nodes decide to be part of the packet forwarding set by considering a circular region of radius $R/2$ centered at mid point of the particular pair of black nodes.
- If node is inside such a region for a particular packet pair it becomes an active forwarding node for that cluster pair with some random delay.
- When it becomes a forwarding node it sends a packet to signal this event. All other nodes go sleep mode for that pair of clusters. However they may be in active mode for some other pair.
- A node may give up its active state for a cluster pair when it has spent a certain amount of energy. It sends a signal so that one of the other sleeping nodes can take over. When it gets a signal back from some node it goes to sleep mode for that cluster pair.
- Although the circular region of radius $R/2$ overlaps two clusters, there may not be nodes in both clusters. Since all nodes can receive transmission from each other in this region, when an active node in one cluster receives a signal about activation of some node in the other cluster, it signals the black node that there exists at least one node in each cluster and overlap regions may be used for packet forwarding
- The intermediate node between two black nodes is used for forwarding if overlap region does not have nodes in both clusters.
- While forwarding a black node listens to all packets and forwards only if the sending node from is out of range of the active forwarding node.

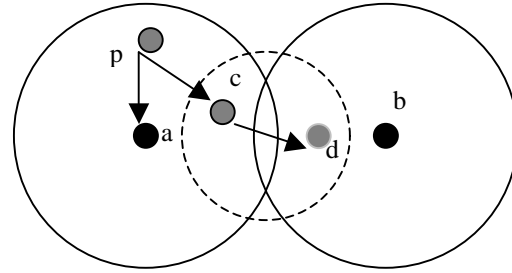


Figure 6. Node Forwarding Between Clusters.

Figure 6 illustrates how a packet would be forwarded from one cluster to the other. We have two clusters with black nodes a and b . The respective forwarding nodes in the overlap regions are c and d . When node p sends a packet, a determines if p is within range of c . If not it forwards the packet to c . Otherwise c can listen to the packet from p . Node c forwards it in the overlapping region where d receives it. Note that since c is in range of p the black node a does not need to forward this packet.

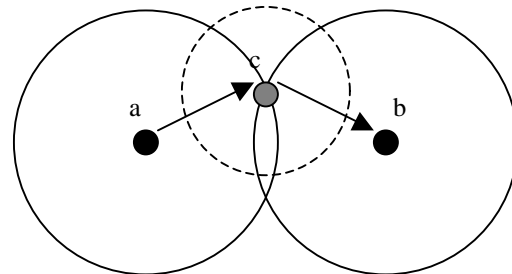


Figure 7. Assigning up the duty cycle without location information..

2. Assignment Without Location Information

In this we assume that nodes do not have location information. The cluster tree we obtain using three colors has the property that any parent/child pair is at most one hop away from each other. This means that there is at most one intermediate node between any two Black nodes.

In the previous routing algorithm, since we knew the locations of black nodes, the actual mid point could be calculated. The nodes inside a circular region of radius $R/2$ centered at this point were considered for forwarding. However this is not possible without location information. In this approach we consider the circle of radius $R/2$ centered at the intermediate node between two black nodes. Figure 7 illustrates the mechanism.

The intermediate node c sends out a message to set up the forwarding nodes. Nodes within a distance of $R/2$ (shown by dotted circle) from this consider themselves for forwarding between the particular pair of clusters. Rest of the procedure is exactly same as the approach with location information.

While forwarding packets there again has to be minor changes to the previous approach. Here again due to lack of location information a Black node cannot decide the

reachability of packets between forwarding nodes. The black node instead of forwarding packets immediately waits for random amount of time before forwarding it. In the meantime if it hears that the active forwarding node has forwarded the same packet, it does not forward the packet.

VI. SIMULATIONS AND RESULTS.

In this section we analyze the performance of our proposed schemes. We modified the NS-2 simulator to incorporate details of topology discovery algorithms. We assume that sensors are randomly deployed in a field of 100m x 100m. The number of sensors and the communication range of the sensors vary according to the experiment requirements. The results represent an average of these values for ten different runs.

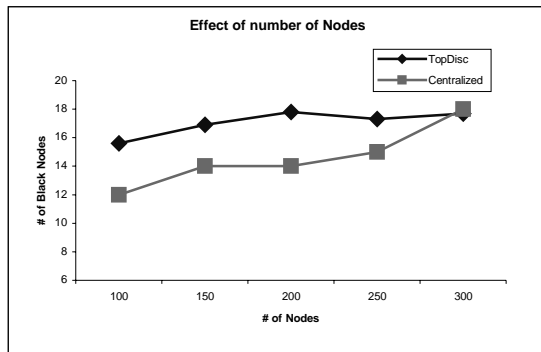


Figure 8. Number of Black nodes vs total number of nodes. Range 20m

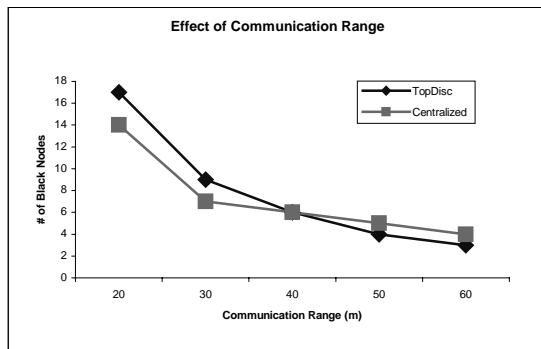


Figure 9. Number of Black nodes vs communication range. 200 nodes

Our first experiment is designed to find the number of black nodes formed during a topology discovery, when the number of nodes increases. We compare our results against the centralized $\log(n)$ -approximate solution provided by the greedy algorithm for set cover. We set the communication range to 20m and vary the number of nodes in the field from 100 to 300. Figure 8 shows the number of black nodes formed at the end of *TopDisc* and the corresponding values for the centralized algorithm. We observe that our algorithm works nearly as well as the centralized algorithm. Another important point demonstrated by this result is that the number

of black nodes formed, is similar for different total number of nodes.

The second experiment illustrates the effect of communication range on the number of black nodes. We perform the experiment with 200 nodes, and communication range increasing from 20m to 60m. The results for this experiment are shown in figure 9. The results are again compared with the centralized greedy set cover algorithm and performance is similar in each case. The number of black nodes decreases with increase in communication range.

Next we evaluate the overhead incurred in doing the topology discovery. We compare the *TopDisc* approach against direct response and aggregated response approaches. Overhead is characterized by the number of packets and the number of bytes transmitted during the entire operation. Since the request propagation phase is same for all the approaches, the graphs show the overhead incurred during the response phase only. The results are evaluated with varying number of nodes (with communication range 20m) and varying communication range (with number of nodes 200). In evaluating the number of bytes, we assume a constant packet header of 5 bytes and an additional 5 bytes of information per node reported in the packet.

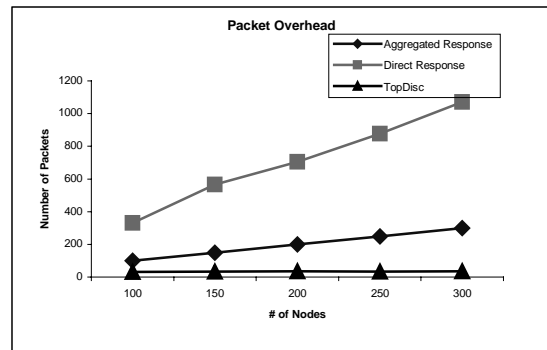


Figure 10. Packet overhead for topology discovery, range 20m

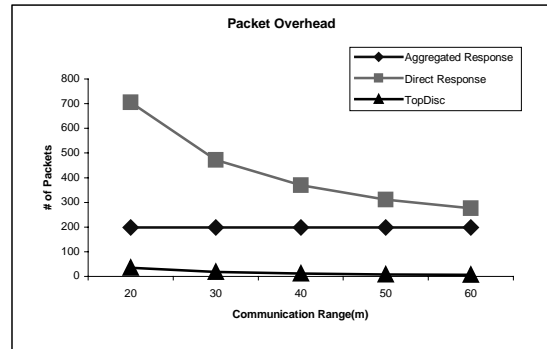


Figure 11. Packet overhead for topology discovery, 200 nodes

The number of packets transmitted for varying number of nodes with a communication range of 20m is shown in figure 10. In the direct response mechanism, each node transmits its reply towards the sink. Each node on the path has to forward

the packet and thus incurs an overhead of a packet. In the aggregated response mechanism, each node waits for its children to transmit and then aggregates their values and sends. Thus each node transmits exactly one packet. The *TopDisc* approach only requires the black nodes and default forwarding nodes to forward. As shown in figure 8, the number of black nodes (and hence the number of default forwarding nodes) is virtually unaffected by an increase in the number of nodes. Hence, the number of packets transmitted in the response phase by the *TopDisc* algorithm is nearly constant.

The number of packets transmitted for 200 nodes, the results with increasing range are shown in figure 11. An increased range with a constant number of nodes, results in the number of hops to the monitor node reducing. This causes a decrease in the number of packets transmitted by the direct response mechanism. The number of packets transmitted by aggregated response mechanism is constant as explained earlier. For *TopDisc*, the number of packets transmitted reduces because the number of black nodes (and thus the number of default forwarding nodes) reduces.

each node sending more bytes. In the aggregated response mechanism, each node transmits the bytes equivalent to the number of nodes which are in its sub tree. In *TopDisc*, only the neighborhood lists of the black nodes are transmitted and although the size of these lists increases, the cumulative increase is very less when compared with other approaches.

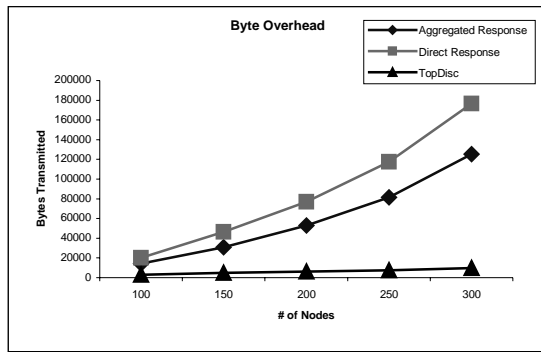


Figure 12. Byte Overhead for topology discovery, range 20m

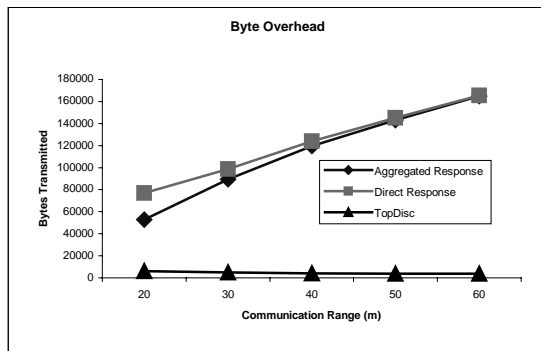


Figure 13. Byte Overhead for topology discovery, 200 nodes

The number of bytes transmitted for a varying number of nodes with a communication range of 20m, are shown in figure 12. As the number of nodes increase, the size of neighborhood of each node increases. In the direct response mechanism, each node sends its neighborhood information to the monitor node and each node on its path, forward the packet. An increase in the number of neighbors, results in

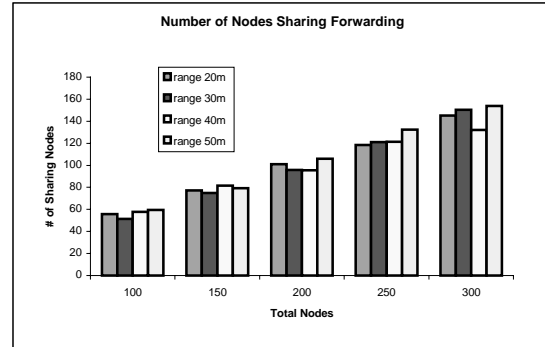


Figure 14. # of nodes sharing forwarding duty (with location information)

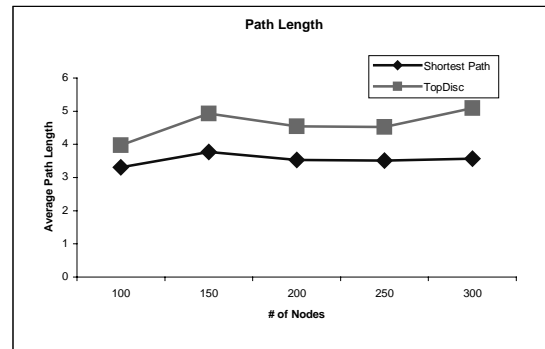


Figure 15. Average path length, range 20m

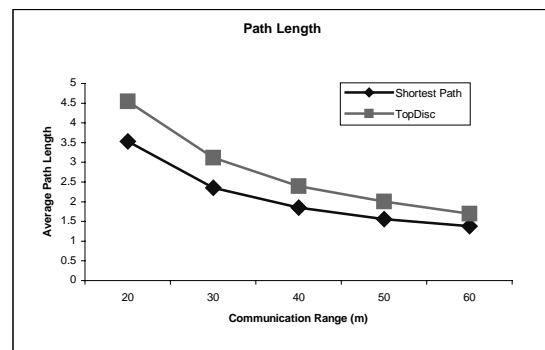


Figure 16. Average path length, 200 nodes

Figure 13 shows the number of bytes transmitted for 200 nodes and range increasing from 20m to 60m. As the range increases, the size of the neighborhood lists of nodes increases. At the same time, the average path length reduces. So the aggregated response mechanism approached the direct response mechanism. The overhead incurred by *TopDisc* tends to decline a bit because the reduction in the number of

black and forwarding nodes offsets the increase in the size of the neighborhood list.

The routing approach based on forwarding between clusters provides us a way of distributing load among different nodes. Figure 14 gives the number of nodes, which are involved in sharing the load in each cluster. Overall it shows that in all cases nearly 50% of the total nodes are part of the forwarding set although the minimum number required for reachability is much lower (number of black nodes + number of default nodes).

The *TopDisc* algorithm gives a reachability map based on *TreC*. Nodes that are part of the tree, comprising of black nodes and intermediate nodes between two black nodes, need to be active for reaching any part of the network. The average path length with our scheme is compared to the shortest path routing scheme. Figure 15 and 16 show that our approach works very close to the shortest path routing.

We provide another mechanism to distribute load of forwarding packets when location information is not known to nodes. Figure 17 gives the number of nodes, which are involved in sharing the load in each cluster. Overall it shows that in all cases nearly 40% of the total nodes are part of the forwarding set. It shows that even without location information it performs as well. An implicit result that may be observed is from this is that the *TreC* is not skewed.

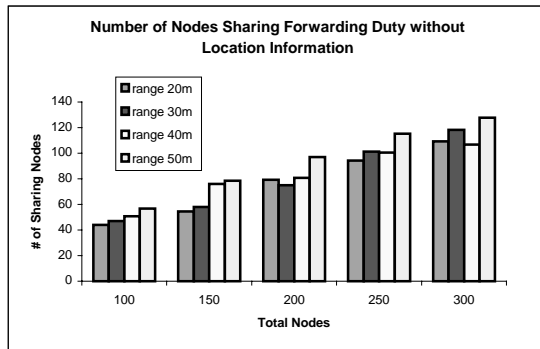


Figure 17. # of nodes sharing forwarding duty (without location information)

VII. CONCLUSIONS AND FUTURE WORK

In this paper we have described a topology discovery algorithm (*TopDisc*) for wireless sensor networks. *TopDisc* selects a set of distinguished nodes, and constructs a reachability map based on their information. *TopDisc* logically organizes the network in the form of clusters and forms a Tree of Clusters (*TreC*) rooted at the monitoring node. We showed the applications of *TreC* for efficient data dissemination and aggregation, duty cycle assignments and network state retrieval. *TopDisc* is completely distributed, uses only local information and is highly scalable.

This work presents a preliminary investigation into various aspects of sensor network management. *TopDisc* gives an efficient way to get approximate yet structured information about the topology. We would like to investigate further on what kind of information can be deduced from the

received data and how they could be used for estimating network performance. We plan to implement the protocols proposed here in a prototype sensor network. Future research would build upon the concepts and protocols described in this paper to culminate into a complete framework for a *Sensor Network Management Protocol (sNMP)*.

REFERENCES

- [1] J.M. Kahn, R.H. Katz, K.S.J. Pister, *Next century challenges: Mobile networking for 'smart dust'*, Proc. MOBICOM, 1999, Seattle, 271-278
- [2] S. Singh and C. Raghavendra; *PAMAS: Power Aware Multi-Access protocol with Signalling for Ad Hoc Networks* ACM Computer Communications Review, 1999.",
- [3] S. Singh, M. Woo, and C. S. Raghavendra; *Power-aware routing in mobile adhoc networks*; in Proceedings of Mobicom '98, pp. 181-190, 1998.
- [4] W. Heinzelman, J. Kulik, and H. Balakrishnan; *Adaptive protocols for information dissemination in wireless sensor networks*; in Proceedings of Mobicom '99, pp. 174-185, 1999.
- [5] Wendi Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan, *Energy-Efficient Communication Protocols for Wireless Microsensor Networks*, Proc. Hawaiian Int'l Conf. on Systems Science, January 2000.
- [6] J. D. Case, M. Fedor, M. L. Scho stall, and C. Davin. *RFC 1157: Simple network management protocol (SNMP)*. RFC, IETF, May 1990
- [7] A. Downey, "Using pathchar to estimate Internet link characteristics, Proc. SIGCOMM 1999, Cambridge, MA, pp. 241-250, Sept. 1999
- [8] A. Medina, I. Matta, and J. Byers, "On the origin of power laws in Internet topologies," ACM Computer Communication Review, vol. 30, no. 2, pp. 18--28, Apr. 2000
- [9] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva. *A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols*. In Proc. of the ACM/IEEE MobiCom, October 1998
- [10] Y.-B. Ko and N. H. Vaidya. *Location-aided routing (LAR) in mobile ad hoc networks*. In ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom'98), October 1998.
- [11] C. Intanagonwivat, R. Govindan and D. Estrin; *Directed diffusion: a scalable and robust communication paradigm for sensor networks*; in Proceedings of Mobicom '00, 2000.
- [12] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. *System architecture directions for networked sensors*. In Proceedings of the 9th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, pages 93-104, Cambridge, Massachusetts, Nov. 2000
- [13] Cormen, Leiserson, Rivest (1990) *Introduction to Algorithms*, MIT Press, McGraw Hill
- [14] Jaap Haartsen, Mahamoud Naghshineh, Joh Inouye, Oalf J. Joeressen, and Warren Allen, "Bluetooth: Vision, Goals, and Architecture," Mobile Computing and Communications Review, Oct. 1998, 2(4):38-45