TreeCast: A Stateless Addressing and Routing Architecture for Sensor Networks

Santashil PalChaudhuri, Shu Du, Amit K. Saha, and David B. Johnson
[santa, dushu, amsaha, dbj]@rice.edu
Department of Computer Science
Rice University,Houston, TX-77005

*Abstract— Recent advances in technology have made low-cost, low-power wireless sensors a reality. A network of such nodes can coordinate among themselves for distributed sensing and processing of certain phenomena. In this paper, we propose an architecture to provide a stateless solution in sensor networks for efficient addressing and routing. We name our architecture **TreeCast**.*

*We propose a unique method of address allocation, building up multiple disjoint trees which are geographically inter-twined and rooted at the data sink. Using these trees, routing messages to and from the sink node without maintaining any routing state in the sensor nodes is possible. Next, we use this address allocation method for scoped addressing, through which sensor nodes of a particular type or in a particular region can be targeted. Evaluation of our protocol using **ns-2** simulations shows how well our addressing and routing schemes perform.*

Keywords: Sensor network, System design, Simulation

I. INTRODUCTION

With the recent advancement of technology, processor size, memory size, and wireless antenna size have gone down considerably, and the trend is not likely to stop very soon. This enables the construction of low cost and low power computing devices that can communicate with other such devices or with conventional computers. When we add sensing capabilities to such devices we get what are called *sensor nodes*. Since these devices are usually very small and inexpensive, they can be deployed over an area in abundant quantities, thus creating a dense network of sensor devices known as a *sensor network*. The inherent redundancy of these networks helps in tolerating the low quality of such small and inexpensive devices. Such networks are potentially very powerful since they can sense the environment in which they are deployed and report sensory data to some processing center, all with little or no human intervention. Pottie and Kaiser [1] provide a good overview of the field of sensor networks.

Our design is primarily motivated by the following two observations:

1) *State in the sensors:* Though sensor networks are attractive due to the expected feature of little or no human intervention, there are various issues that need to be addressed before such networks become practically feasible. One of the more important among such issues is the robustness of the network even in the presence of node failures or temporary wireless failures. It is highly desirable for the sensor network to be stateless, so that node failures do not lead to incorrect or inefficient functioning of the sensor network.

2) *Communication pattern:* A sensor network has certain simplifying features that allow for a better and more efficient routing protocol than most existing wireless ad hoc network routing protocols. One of the basic features of sensor networks is that they consist of one or more sink nodes that are typically more powerful than the sensor nodes. Most of the communication in a sensor network fall under two categories: (1) The sink node sends out a query that floods the network. Whichever sensor can sense data relevant to the query replies back to the sink node with the sensory data. A query might also entail periodic replies from the sensors. (2) Even in the absence of an explicit query in the immediate past, a sensor can sense data and send it to the sink node. These characteristics suggest that there is seldom any end-to-end communication between individual sensor nodes themselves, and most of the communication is between the sensor nodes and the sink node.

In this paper, we propose a novel architecture called *TreeCast* to achieve robustness in the underlying sensor network through our stateless addressing and routing scheme. All the state necessary for routing is in the sink nodes, thereby eliminating any dependency of the functionality of sensor networks on the state within each sensor. We use the features of the communication pattern in sensor networks, enabling us to do routing that is robust in the presence of node failures, and does not require any state. We take a unique approach to generate a address for each sensor node (rather than depending on pre-assigned addresses) such that once the address assignment is completed, routing from any sensor node to the sink node is trivial. The addressing scheme in itself will include the path to the parent, such that a node only has to remember it's own address that is generated and assigned to it after deployment. We also use this addressing scheme for completely stateless scoped addressing of sensor nodes, so that certain queries are forwarded only towards certain regions or classes of the sensor network if necessary.

Previous work [2] have shown that considerable energy savings can be obtained by performing local computations to reduce and aggregate data before transmission. Though our paper does not take that approach for energy savings, such approaches are completely consistent with our work and can be applied in addition to our protocols. Data centric routing [3] has been proposed for sensor networks. Our architecture provides a stateless routing and scoping layer on top of which schemes like these can be built. We realize that explicit addressing of nodes is not necessary in sensor networks [4], but by using this inexpensive addressing scheme we assure considerable reduction in energy usage by intelligently putting some nodes to sleep while retaining coverage.

The rest of the paper is structured as follows. In Section II, we survey the related work that has been done in this area and point out the novelty in our approach. Then we describe our architecture for addressing and routing in Section III. In Section IV, we illustrate how we use the assigned addresses to implement a stateless scoped addressing of sensors. We evaluate our work in Section V, and finally in Section VI, we discuss our conclusions.

II. RELATED WORK

1) Data-Centric Routing: Because sensor networks usually have specific applications and the networks themselves are composed of a large number of resource/energy constrained sensor nodes, it is reasonable to believe that the requirements for sensor networks will be quite different from those of traditional wired or wireless networks. One of the important differences is that sensor networks can be data-centric [5]. For a data-centric network, the precise node where the data is generated is not important. Rather, data from different nodes could be aggregated on the path to the sink to yield energy efficiency and data reduction. Directed diffusion [3] is a data-centric routing mechanism in sensor networks. It uses the *interests* to tag all data flows. As the sink broadcasts the interest over the network, the interest entries, as well as the routing information called *gradients* are cached in each sensor node. The gradients are used for selecting routing paths when there is an *interesting* data generated or received. Multiple alternate routes can be maintained to make the algorithm robust and potentially energy efficient. Another data-centric routing algorithm, SPIN [6], is different from directed diffusion, as it requires the sensor node to broadcast its available data and wait for requests. It reduces energy consumption by sending a short description of the data before it sends out the real data. Our work is consistent with such data-centric protocols and such a protocol can run on top of our addressing and routing scheme. TreeCast provides such protocols with completely stateless routing and scoping.

2) Ad Hoc Network Routing: For the past several years, much research on routing protocols for wireless ad hoc networks has been done. Several protocols, pro-active as well as reactive, have been proposed and evaluated (e.g., [7], [8], [9], [10], [11]). Some hierarchical address-based routing strategies (e.g., [12], [13], [14]) have been developed, which use group-like addresses to stand for geographically proximate node clusters and therefore achieve better scalability. In general, these protocols consider the whole network to comprise of separate clusters. Nodes that are geographically proximate are negotiated as a cluster and the head of a cluster is elected as the gateway for traffic to the other clusters. However, all of these protocols are in general unsuitable for a resource and energy constrained large scale ad hoc network such as a sensor network, because a sensor node may not have enough memory to maintain the large routing table to every other node. For sensor networks, routing is much easier than in other wireless ad hoc networks: sensor nodes usually need to communicate the data back to the sink or vice versa. Sometimes, a node may contact neighbors to do local coordination such as data aggregation. Some stateless routing protocols in ad hoc networks that can be applied to sensor networks, such as Greedy Perimeter Stateless Routing (GPSR) [15] uses a localized algorithm to decrease the overhead of routing in large scale ad hoc networks. However, GPSR requires location information and also assumes that there exists an address-location mapping service in the network to help the source know the location of the destination. Sequential Assignment Routing (SAR) [16] is a routing protocol designed for sensor networks. The protocol generates multiple paths to avoid route recomputation overhead on route failure. Multiple paths from each node to the sink are implemented by building multiple trees, each rooted from a one-hop neighbor of the sink. Each node must remember the energy resources and cumulative quality of service parameters associated with each path, such that the node can choose its preferred path. However, it is not clear how to construct such trees. Compared with SAR, our trees are built totally disjoint with each other and therefore our trees are more load balanced and each node only needs to know its own address. Finally, LEACH [17] is a cluster-based protocol that achieves energy efficiency in communication. It is also separated into two phases: a set-up phase, in which cluster heads are selected by some predefined probability, and a steady phase, in which the cluster heads can aggregate the data and send it directly back to the base station.

III. TREECAST ARCHITECTURE

In this section, we describe our TreeCast architecture in detail. The architecture provides intelligent, automatic address assignment to the sensor nodes in the sensor network and uses these addresses for stateless routing. This approach differs from previous approaches where the addresses (or identifiers) for sensor nodes are predefined. Our address assignment constructs multiple trees with the property that a subset of the trees can serve the purpose of the sensor network equally well. We then describe how routing is easily achieved once the address assignment has been completed.

A. Address Allocation

A sensor network is characterized by one or more sink nodes and a large number of sensor nodes, where the sink node is much more powerful compared to the sensor nodes. We explain our protocol using a single sink node and later describe how this protocol is extended in the presence of multiple sink nodes.

1) Length of Address: The basic idea here is to create a tree like network such that the address of a node has some correlation with the distance of the node from the sink and the subtree it belongs to. We assume that the average density of sensor nodes over the given sensor network is ρ sensor nodes per unit area. Thus the average number of nodes in an area A is $N_A = \rho \times A$. Our address assignment assumes a maximum number of neighbors N (within wireless transmission range) that each sensor node in the network may have; for example, by assuming a wireless transmission range of each sensor node, the number of neighbors can be calculated based on the assumed sensor density ρ. The sink node then chooses an integer b such that $2^b \gg N$. (For correct functioning of the protocol, $2^b > N$ suffices, but $2^b \gg N$ improves the protocol's performance.) The same value of b is used throughout the sensor network. The base 2 in choosing $2^b \gg N$ could instead be any integer $j > 1$, with $j^b \gg N$, but for simplicity we choose the base to be 2. The sink node does not assign itself any address. If there are multiple sink nodes then each of the sink nodes can assign themselves an address and inform the other sink nodes of this address. From now on, we shall consider the sink node to have a *null* address. The addresses assigned by this protocol to any sensor node will be of the form $(0|1)^{bk}$, with the sink node having the null address ($k = 0$). We define a node with address of the form $(0|1)^{bk}$ to have a *level* of k (a node with a higher value of k is at a higher level, i.e. it is further away from the sink). The sink node hence is at level 0. The address le ^th is proportional to the log of the number of nodes in the netv
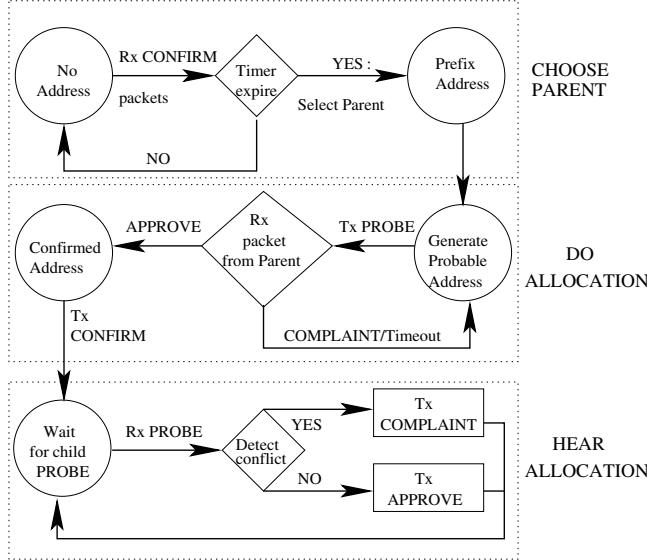
Fig. 1. Address Assignment Algorithm

2) Address Assignment: In our address assignment mechanism, after fixing a value for b (for the entire sensor network), the sink node initiates the address assignment protocol by locally broadcasting a packet that contains the value b and also contains the source address of itself, which in this case is a null address.

The address assignment protocol is divided in 3 parts as shown in Figure 1: *CHOOSE PARENT*, *DO ALLOCATION*, and *HEAR ALLOCATION*.

CHOOSE PARENT part of the address assignment protocol is done by each node when it does not yet have a parent associated. A *CONFIRM* packet signifies that a node has assigned itself a unique address.

1) In level $(m + 1)$, each node waits for a fixed multiple, k, of T_{wait} time after receiving the first CONFIRM packet, from all level m nodes it can hear. So that a node in level $(m + 1)$ hears as many of level m nodes as possible, the value of $k \times T_{wait}$ depends on the ratio of 2^b and N for the system.
2) Among the list of nodes at level m from which it has received a CONFIRM packet, the node then selects any one of them randomly with equal probability to be its parent node, preferably a choosing a parent sending the strongest signal. The parent's address becomes the node's prefix address.

DO ALLOCATION part is for the sensor nodes to choose a locally unique address for itself, after it has selected a parent. This works because of total ordering of address assignment of any node enforced by it's parent.

1) The node generates a random b-bit number. It concatenates this random number with the received address of it's parent to construct it's own address.
2) The node broadcasts its address locally with a *PROBE* packet, after a random jitter in order to avoid collisions. The PROBE packet also contains a random tag string to identify the packet.
3) The node waits for a fixed amount of time, T_{wait}. During this time, it listens for *COMPLAINT* or *APPROVE* packet from the parent node. If neither packet type is received, step 2 is repeated.

4) If the node hears a COMPLAINT from parent node in response to its PROBE, it chooses another address (chooses another random b-bit number and concatenates that with the address of it's parent). The node then repeats the steps above beginning with Step 2.
5) If the node hears a APPROVE message from the parent node, it sends out a CONFIRM message confirming it's address.

HEAR ALLOCATION part of the address assignment protocol is done by a node after it has a confirmed address.

1) A node waits to hear PROBE packet from all nodes whose probed addresses are direct children of itself.
2) If a node receives a PROBE, it checks if the address within it has already been allocated.
3) If it has not heard the address, it sends out an APPROVE message with the same random tag string present in the PROBE packet. Also, it remembers this approved address. Then the node goes back to Step 1. The addresses that a node remembers are only temporary and are deleted once the HEAR ALLOCATION step is done.
4) If that address has already been allocated, it sends out a COMPLAINT message with the same random tag string present in the PROBE packet. Then the node goes back to Step 1.

The address assignment algorithm thus constructs a unique address for each sensor node. While unique addresses may not be necessary for many of the current applications of sensor networks, it is useful to have this property for in-network collaboration or control.

3) Tree Construction: The CHOOSE PARENT procedure encourages the trees to be well-balanced and inter-twined. While this is not necessary for correctness, it will lead to better efficiency. Thus, each node tries to hear from all possible parents in the previous level before choosing one of them randomly as its parent. This is achieved as nodes at increasing distance from the sink are allocated addresses in a lock-step fashion facilitated by the wait time of $k \times T_{wait}$ between address allocations for consecutive levels. This ensures that only after all the addresses have been assigned for a node at a particular level, does the assigning of nodes in the next level start. The probability of the trees being balanced rather than being unbalanced is very high for uniform random distribution of the sensor nodes. But an adversary scenario with a non-uniform node distribution can be constructed such that the trees will not be well-balanced. This will lead to some amount of decreased performance without hurting the correctness of the protocol.

An example of address allocation is shown in Figure 2. Here node S starts allocating addresses. Nodes B_1, B_2, C_1, and C_2 randomly choose their parent to be B or C, respectively, after hearing *CONFIRM* packets from both of these nodes.

We state two properties of our address assignment protocol:
- *Property I:* A node can identify its own level simply by knowing its address and the constant value b, which a node remembers during address assignment.
- *Property II:* A node A, with address of the form $(0|1)^{bk}$, has a parent that has an address of the form $(0|1)^{b(k-1)}$, and the parent's address is a strict prefix of the address of A.

4) Multiple Sink Nodes: If the sensor network contains more than one sink node, then either of the following approaches can be taken:
- Each of the sink nodes will run the protocol independently, and hence the sensor nodes will have one address per node. This is the obvious solution when the sink node
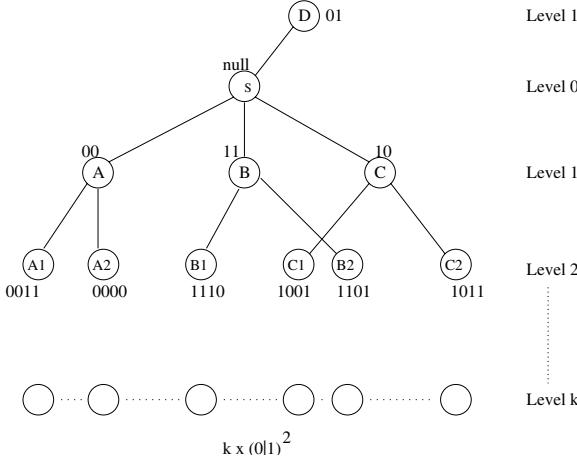
Fig. 2. An example sensor network with $b = 2$

intentionally kept independent of each other, for example because of reliability issues.

- If however, reliability of sink nodes is not a significant issue, then the sink node that initiated the address assignment protocol can act on behalf of the other sink nodes. All queries generated by other sink nodes will be routed through this particular sink node and all responses targeted towards other sink nodes will also be routed through the sink node that initiated the address allocation protocol.

The issue of routing between the sink nodes is of less importance since typically the network connecting the sink nodes will be much more capable and reliable and will involve traditional wired or wireless routing, thus enabling any of the many existing routing protocols to service the sink nodes.

B. Tree Maintenance

The above algorithm will work perfectly assuming there is no change in topology, sensor failures or addition of new sensors. However as these situations can occur, our architecture has a tree maintenance phase to counter this.

No parent: Nodes which lose their parent either due to topology change or node failure become an *Orphan*. When a node does not receive any traffic from it's parent for certain timeout period, it sends queries to the parent multiple times. The parent replies immediately when it receives such a query, informing the child of the availability of the parent. If no such reply is received by the child after multiple queries, the node becomes an Orphan. After a node has been an Orphan for a specified Orphan Timeout period, it will initiate tree maintenance. The length of Orphan Timeout period is a trade-off between cost of maintenance and consistency in terms of latency and reliability. Routing messages as in Section III-C will still take place though, albeit with increased latency.

The Orphan solicits for parents with a local broadcast packet. Any node receiving this solicitation responds with their respective addresses and the value of b. The Orphan selects a node randomly from all responding nodes with the lowest level. The Orphan then initiates the DO ALLOCATION algorithm for choosing it's complete address. The chosen parent node broadcasts locally the PROBE packet for the Orphan multiple times for increased reliability. Any children of the parent having the same address as the Orphan chose, will send COMPLAINT to the parent each time it gets the PROBE packet, which the parent forwards to the Orphan. Then the Orphan chooses

another address and repeats the above steps. If there is no COMPLAINT received, the parent sends out an APPROVE message for the Orphan. Upon receiving this, the Orphan gets an address and sends out a CONFIRM message. All children of the node receiving this CONFIRM message then redo the DO ALLOCATION algorithm to get a new address. This process ends when all the nodes in the subtree rooted at the Orphan node has changed address to conform to the new address of the Orphan.

Node Addition: New sensors might be added to the sensor network after the initial address assignment for the network. Whenever a new sensor node joins the network, it selects a parent in the same way as stated above, by sending out a solicitation message and choosing the best parent from the respondents. It chooses it's own address by the DO ALLOCATION algorithm and attaches to the parent. The new node is at one level above the parent it chose.

C. Routing Messages

Query Messages: When the sink node needs to send a query to all sensors, it floods the query through the trees. A node listens for packets from it's parent, and forwards the packet to it's children. This happens at every interior node of the tree. Each query packet has the address of the forwarder, which is used to determine the nodes which should forward this packet. If a node becomes an Orphan, it forwards all unique query packets till the time it remains an Orphan. Hence, efficient flooding is achieved using these trees. The case of addressing a specific class or region of sensors is handled in Section IV.

Response Messages: This part describes how a sensor node routes responses that it has to send to the sink node

Suppose a sensor node A at level k, with an address of $(0|1)^{bk}$, has a message to send to the sink node. Node A forwards the message to a sensor node B at level $k-1$ that has an address of the form $(0|1)^{b(k-1)}$, such that the address of a node B is a strict prefix of the address of A. If the network topology has not changed after the address allocation, then B will be in the transmission range of A, and by our address allocation protocol, B is the parent of A. The receiving node B now repeats the same procedure and forwards the packet that it received from A to a node C that has an address that is in level $(k-2)$ (thus having an address of the form $(0|1)^{b(k-2)}$) and whose address is a strict prefix of node B's address. The parent address is not explicitly specified but is implicitly contained in the address of A. If a node at level m receives a packet from another node at level k and $k - m > 1$, then the receiving node ignores the packet, since it assumes that there is another node at level $k-1$ that will be able to handle the packet sent by the node at level k.

For a node in level k that becomes an Orphan any node at level $k-1$ can receive the response from the node at level k and forward the packet to it's parent in level $k-2$. We indicate in the packet sent by a node in a lower level whether the packet needs to be forwarded *only* by the parent node or can be forwarded by *any node* at the same level as the parent node by setting the *ANY PARENT* flag. When the packet is sent for the first time by a node at the lower level, this flag is set to *false*, so that only the parent node attempts to forward the packet. However, if even after several attempts (say 3), the node at the lower level does not get an implicit acknowledgment from the parent node (hearing the parent node forward the packet), then the node retransmits the packet, this time with the ANY PARENT flag set to *true*, allowing any node at the level of the parent node to forward the packet. If no implicit acknowledgment (overhe

the packet being forwarded by the parent) is heard even after forwarding for the second time with the ANY PARENT flag set, then the packet is forwarded again with the *LATERAL* flag set. This enables nodes in the same level to forward the packet. Routing loops between siblings are prevented in the same way as in alternate routing in the DARPA Packet Radio (PRNET) routing protocol [18].

Control Messages: Applications for sensor networks might require communication between any sensor to a group of other sensors, for collaboration or in-network control and actuation. This is accomplished in our tree structure by the message going up the tree till the longest prefix match is reached. Then the message goes down the tree till the target group of sensors can be reached.

The data aggregation and data centric routing techniques as proposed in sensor networks literature [3] can be applied on top, since our protocol only creates a routing infrastructure on the basis of address assignment.

IV. STATELESS SCOPED ADDRESSING

The address assignment of the nodes, tree formation and routing of messages has been elaborated in the previous section, where it was assumed that all the sensors need to be reached from a specific sink. Hence, any message originating from the sink was flooded throughout the network. The assumption of global flooding might not hold true in a variety of circumstances. Among all sensors, if only a subset of sensors need to be reached, the query should cover only these sensors and not cover those sensors which will not respond to the query. There are two types of such queries:

- Sensors from a particular geographical area might be targeted. Only the sensors in the targeted region should respond to the query. Hence for optimization, the query should only be directed towards these sensors.
- Sensors sensing a particular attribute might be targeted. For optimization the query should only reach to all sensors sensing this attribute, and not reach an area where these sensors are not present.

The guiding principle we follow is to maintain statelessness of the sensor nodes in the network. All the state is maintained in the sink node, and necessary state information is carried by the query packet as piggy backed data. This stateless property of our protocol makes it robust under various sources of node failure and is in keeping with the inherent assumption in sensor networks that sensor nodes are cheap and unreliable devices and hence can fail in several ways. To efficiently process queries, we use a technique called Reinforcement Learning which is influenced by the technique used in Directed Diffusion [3].

A. Reinforcement Learning

The first time a sink sends a query directed to a particular class of sensors, the query needs to be flooded throughout network. This is necessary as there is no information present on the location of the particular class of sensors targeted. This is done by flooding the query along the trees constructed in Section III-A.3. This initial query should contain information about which type of sensors are being targeted in addition to the actual query. Intelligent querying is a separate issue altogether and we do not try to solve that problem in this paper. However, we do offer some properties that should be present in queries for processing the queries efficiently.

When a relevant sensor node receives the query and decides to respond to it, it constructs a response message which contains the response and its own address. The responding sensor
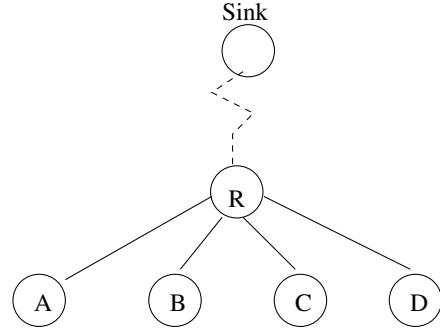

Fig. 3.   Problem with simple set approximation

node then routes this response message up the tree towards the sink using the techniques explained in Section III-C. When all the responses reach the sink the sink node has an exact idea of where the responses came from and where the query should be directed next time the query needs to be posed. This process has to be done once for each query. Once the sink node knows the relevant sensors for a query the sink node can direct the query to only the relevant nodes. We present an efficient way to do this in Section IV-C.

B. Address Aggregation

The idea as explained above is not efficient since listing all the addresses of responding sensors is not feasible in a large sensor network. Doing so would require large packets to be transmitted thus using up the sensors' energy. Here we present few techniques for efficiently representing a large number of sensors based on our addressing scheme.

- *Set Approximation*: The basic idea here is to replace the addresses of several responding sensor nodes with the address of a sensor node which is the root of the subtree that contains the sensor nodes. This of course involves a trade-off between the resolution with which the sink can identify the responding sensors and the size of the response message. Whenever this approximation occurs, a *level* associated with the address is incremented by 1. The level is initially 0, when no address aggregation takes place. It signifies the number of levels that the scoped broadcast should go, below the aggregated address. This technique prevents the scoped broadcast to always go to the leaf nodes of a branch, unless the leaves are part of a scoped broadcast. For example consider the scenario in Figure 3. Suppose sensors *A*, *B*, and *D* respond to a query and *C* does not. When the responses are routed towards the sink, instead of individually listing out the addresses of *A*, *B*, and *D*, sensor *R*, which is the root of the subtree containing *A*, *B*, *C*, and *D*, can simply send its own address instead of a list of addresses, thus reducing the size of the message that has to be routed to the sink. However, when the sink node receives the response the sink knows that the response came from *R* and does not understand that *C* is not a relevant sensor, i.e. we loose out on resolution but gain in packet size efficiency. However, such a decision has to be based on several factors and we list out some of the important ones. Based on these factors a node should decide whether it should do set approximation or not. Two approaches to set approximation are as follows:

  Threshold based: A very simple way to aggregate the addresses of responding sensors is the following. Each sor node maintains a upper threshold and a lower thre

(say MAXCOUNT and MINCOUNT). The exact values for these thresholds will depend on which level the sensor node is and what is the expected number of nodes in the subtree rooted at this sensor node. Hence the threshold can be calculated simply on the basis of how many bits are used for each level. If a sensor node finds that the number of responding sensors that belong to the subtree rooted at itself is greater than the MAXCOUNT then the node replaces all of those addresses with its own address. It also updates the level of the response address to be the maximum level of all the replaced addresses. If the difference between the levels of the responding addresses of its children are more than a maximum difference, then a set of addresses with various levels is passed up instead of a single address with one level. In this way, the scoping would not have a resolution more than the maximum level difference allowed. For example in Fig. 3, if the levels of *A* and *B* are 3 and 20 respectively, and the maximum difference allowed is 10, then the address of *A* and *B* would not be replaced by address of *R*, and will be separately grouped. Conversely, if the number of responding sensors in the subtree is less than MINCOUNT then the sensor node forwards none of them.

However, this does not solve the problem shown in Figure 3. In particular, any solution based on absolute thresholds will not be able to handle scenarios which involve unusually spare or dense regions of the network. To solve this problem we propose a simple technique. When a query is sent out for the first time each node reports to its parent the number of nodes in the subtree rooted at itself. When doing set approximation this number is used to decide whether to replace all the responding sensors with a single address or not. In order to decide whether to replace all the responding addresses with its own address a node calculates the following value:

(Number of nodes in subtree rooted in children nodes which did not respond to the query) / (Number of nodes in the subtree rooted at this node).

If this value is greater than a **threshold** then the node decides not to replace the addresses with its own address. If the value is less than the threshold then the node decides to replace the addresses with its own address. This technique is able to handle situations in which sensors are unusually sparse or dense in a particular part of the network. This threshold based address is evaluated later.

In the example of Figure 3 if *R* knows that *C* is the root of a large subtree then *R* would decide not to replace the addresses of *A*, *B*, and *D* with its own address and would pass up the addresses as is.

Query based: The query can dictate whether the response should have as many data points as possible or whether a response involving an aggregation of all responses would be enough. Hence, if the query were to ask for as many responding sensor nodes as possible then the sensor nodes have no other alternative but to forward as many responding sensor addresses as it receives. The node should not perform any set approximation in that case. Even under such situations, address coalescence, explained below will provide good compression.

There can be several other criteria based on which set approximation can be applied. However, we do not attempt to explore all such possibilities in this work.

- *Address Coalescence*: If there are multiple addresses in a response which has to be routed to the sink then the addresses are coalesced together according to some compression technique. Though there are several ways to compress the addresses that are present in a response, here we present a simple *delta compression* technique to compress the addresses. This compression technique provides good efficiency of compression as well as requires little processing complexity. Our compression technique borrows from the OID (Object Identifier) Compression technique used in SNMP (Simple Network Management Protocol) community [19].

The basic delta compression technique that we use is as follows. This algorithm reduces the amount of redundant information by calculating *delta* addresses with respect to a reference address. The delta address that will be generated from this algorithm consists of a value identifying the position in the current address at which the current address starts diverging from the reference address. Following this value will be the remaining part of the current address. For example, suppose a sensor node receives responses from sensor nodes *A*, *B*, and *C* with addresses *1.2.3.4.5.6.7.8*, *1.2.3.4.6.7.8*, and *1.2.3.5.6.7* respectively. These three addresses can be compressed together as *1.2.3.4.5.6.7.8*, *5*;6.7.8, and *4*;5.6.7. This is a very simple algorithm and hence has a good balance between the compression that it achieves and the complexity of the algorithm. Also, the algorithm is not limited to a single reference address and there can be multiple reference addresses allowing better compression. For example, if in addition to the previous 3 addresses there were another two addresses *D* (*1.5.4.3.2.3.4*) and *E* (*1.5.4.3.2.4*). If we allow only one reference address then the delta addresses for *D* and *E* with respect to the reference address *A* (*1.2.3.4.5.6.7.8*) would be very big since the addresses start diverging at a very early stage. However, if we allow multiple reference addresses then *E* can be compressed with respect to *D*. This also fits well with the higher probability of isolated groups of sensors responding to a query.

Both the above mentioned techniques are independent of each other and are also complimentary to each other. Using any of the techniques will decrease the size of the packet while using both the techniques will decrease the packet size even further.

When tree maintenance is done, a subtree might change the parent it is rooted at, thereby breaking the proper working of the scoping process. Whenever tree maintenance is performed, the node which changed it's parent informs the sink of this change. The sink which has the scoping information for all scoped queries, checks to see if the change infringes with any specific query (whether the subtree specified had any nodes corresponding with a scoped query). If it is indeed the case, the sink suitably changes those addresses in any query which was affected by the change of the root of the subtree.

Also, the approximation methods explained above is totally independent of the aggregation of the actual responses. Aggregation of the actual response values is outside the scope of this paper and we do not attempt to address this issue. Having said that, it suffices to say that our approximation algorithms do not infringe with response value aggregation in any way.

C. Query Optimization

Query optimization works almost exactly like address coalescence. The sink node sends out the query and targets the

| Parameter Name	Parameter Value
Number of runs	10
Timeout to wait for probable parents	12 s
Area	1000 m × 1000 m
Number of nodes	200
Transmission range	125 m
Node density per transmission area	Approximately 10
Bits per level	8 (2^8 addresses per level)

Fig. 5.   Levels assigned to nodes with increasing distance from sink

Fig. 6.   Reachability of sensor nodes under node failures

to a specific set of sensors. The addressing mechanism for targeting this set of sensor nodes is exactly the addressing mechanism used for compressing the addresses of responding sensors. For example, if the sink node has to send the query to A, B, and C with addresses *1.2.3.4.5.6.7.8*, *1.2.3.4.6.7.8*, and *1.2.3.5.6.7* respectively then the sink node compresses these addresses as *1.2.3.4.5.6.7.8*, *5*;*6.7.8*, and *4*;*5.6.7*. The sink node also has to mention the maximum level for each of the addresses. Each sensor node receiving a query packet does the following:

- The node receiving such a query looks at the list of addresses specified in the query packet. Each node only forwards the query packet if any of the addresses listed are part of the subtree rooted at itself and the corresponding level mentioned in the query packet is > 0.
- Before forwarding the query packet the node filters out those addresses from the packet which are not part of the subtree rooted at itself. However, this does not require re-running the address compression algorithm since filtering out those addresses from the packet which are not part of the subtree rooted at this node is essentially equivalent to filtering out one or more of the divergent addresses (starting with a position). Also, the level associated with the forwarded address is decreased by one.

So as the query goes out from the sink, the number of addresses contained in the query keeps decreasing until the query reaches the last of the destination sensor nodes.

In the previous description we have not described how the query itself is handled by each relevant sensor node. Actually processing a query (as opposed to routing query packets) is a separate issue and we do not attempt to address that issue in this work.

V. EVALUATION

In this section we present the evaluation of our protocol using the *ns-2* network simulator. We ran our simulations for sensor networks of size ranging from 100 to 1000 nodes which were placed according to a uniform random distribution in a two dimensional area of 1000 m×1000 m. We show all the results in this paper for 200 nodes. Results with different number of nodes show similar characteristics. Each node having a 125 m transmission range gives the average density per transmission area to be $\frac{200}{1000 \times 1000} \times (\pi \times 125^2) \approx 10$ nodes per transmission area. Table I shows the parameters used for the simulations. The results presented are averaged over 10 runs, each run being done on a different scenario of the same size.

A. Address allocation scheme

We present the evaluation of the address assignment protocol that was described in Section III-A.
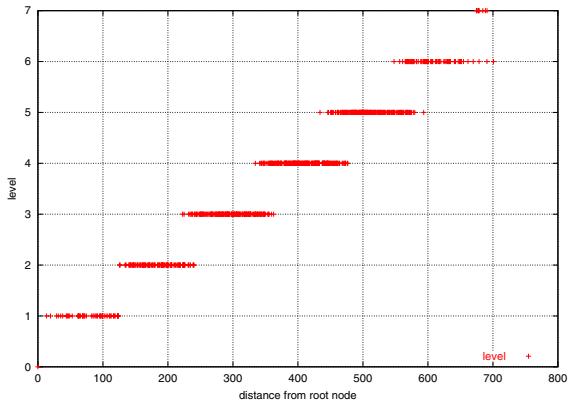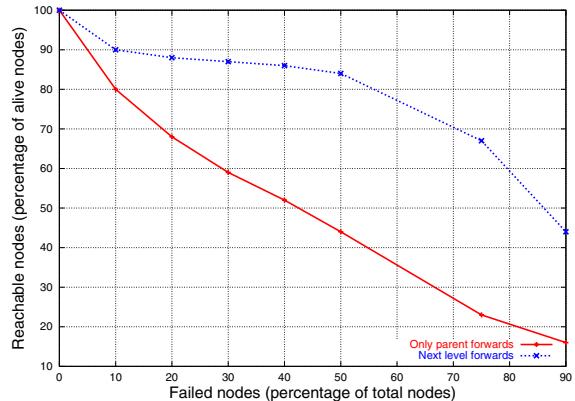
Figure 4(a) shows the distribution of nodes in the specified area for a sample topology, with the sink being in the center of that area. Figure 4(b) shows the tree formed by the address allocation scheme rooted at the sink. As can be qualitatively seen from the figure, the tree formed is balanced and intertwined, which are the properties sought with the Address Allocation scheme.

To show explicitly the quality of tree formed we use Figure 5, which shows the Euclidean distance of a node from the sink on the X-axis, and on the Y-axis is the level assigned to the node. The tree our algorithm formed, had most nodes in the minimum level possible according to transmission range of the node, and the rest of the nodes within 1 level of the minimum possible level. It demonstrates that our scheme does a very good job of assigning levels to each sensor node and the levels have very close resemblance to the Euclidean distance from the sink node. The existence of multiple levels at the same distance from the sink (for example at a distance of 230 m from the sink) has two reasons. Firstly, some nodes might not have received packets from the *ideally expected* levels due to the lack of intermediate nodes providing direct routes from the sink to the node, as the network is not uniformly dense. Secondly, packet collisions can lead to a sensor node not receiving the packet from the ideal level but later listening to packets from a higher level.

B. Routing under failures

The most common traffic scenario in a sensor network is query traffic from sink towards sensor nodes, and vice v
The routing strategy is trivial once the addressed tree is for
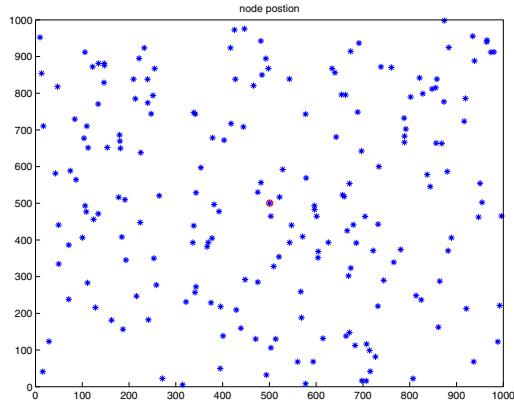
(a) Scenario 200 nodes, 1000 m X 1000 m area
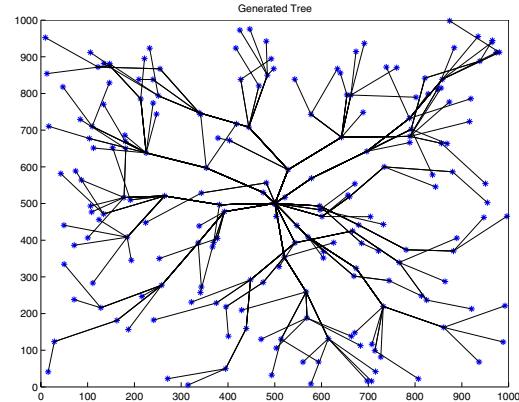
(b) Generated Tree for scenario on left

Fig. 4.   A sample scenario and the tree generated.

The interesting thing to observe will be the performance of Routing as described in Section III-C, before tree maintenance as detailed in Section III-B has started working. We show here that even in the time during which tree is not perfect, routing can still take place using the routing strategy described before. Figure 6 shows the behavior of the sensor network when failures are introduced into the system. In this simulation some percentage of randomly chosen nodes are put to sleep and then the reachability of the rest of the nodes is shown. The lower curve in Figure 6 shows the percentage of live sensor nodes that can reach the sink when *only* the parent of a sensor node is allowed to forward the packet forwarded by a node. The upper curve shows the percentage of live nodes that can reach the sink if any sensor node which is at the same level as the parent of the forwarding sensor node, forwards data. The figure shows that our routing strategy is able to sustain large number of failures. This is particularly due to the fact that on the event of the failure of a parent of a sensor node, other sensor nodes in the same level as that of the parent can take over the job of forwarding data packets. This evaluation is however pessimistic because we neglect the possible fact that in the presence of node failures even a global flooding protocol might not be able to deliver packets simply because the network has been partitioned.

VI. CONCLUSION

In this work, we have presented a novel way to auto-configure a deployment of sensor networks. Locally unique address allocation is done such that routing messages in the sensor network becomes stateless and trivial. This address allocation gives rise to multiple balanced trees. These trees are inter-twined such that each region is covered by more than one tree. Then we present a stateless addressing scheme for addressing a specific group or region of sensors using reinforcement learning, thus decreasing message overhead in the sensor network.

In the future, we plan to introduce a more scalable address allocation method for more efficient maintenance. In order to save energy, we plan to selectively use these multiple inter-twined trees. This scheme while saving power, will ensure adequate coverage.

REFERENCES

[1] G. J. Pottie and W. J. Kaiser,   "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[2] G.Pottie and W. Kaiser, "Wireless sensor networks," in *Communication of the ACM*, 2000.

[3] C. Intanagonwiwat, R. Govindan, and D. Estrin,   "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proceedings, Sixth Annual Int. Conf. on Mobile Computing and Networking (MobiCOM '00)*, Boston, Massachussetts, USA, 2000, pp. 56–67.

[4] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan,   "Building efficient wireless sensor networks with low-level naming," in *Proceedings of the eighteenth ACM symposium on Operating systems principles*. 2001, pp. 146–159, ACM Press.

[5] J. Heidemann D. Estrin, R. Govindan and S. Kumar, "Next century challenges: scalable coordination in sensor networks," in *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, Seattle, WA USA, 1999, pp. 263–270.

[6] J. Kulik W. R. Heinzelman and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, Seattle, WA USA, 1999, pp. 174–185.

[7] Charles E. Perkins and Pravin Bhagwat,   "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," in *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*. ACM, Aug. 1994, pp. 234–244.

[8] David B. Johnson and David A. Maltz,   "Dynamic Source Routing in Ad Hoc Wireless Networks," in *Mobile Computing*, Tomasz Imielinski and Hank Korth, Eds., chapter 5, pp. 153–181. Kluwer Academic Publishers, 1996.

[9] Charles E. Perkins and Elizabeth M. Royer, "Ad-hoc on-demand distance vector routing," in *Second IEEE Workshop on Mobile Computing Systems and Applications*, Feb. 1999, pp. 90–100.

[10] Zygmunt J. Haas and Marc R. Pearlman, "The performance of query control schemes for the zone routing protocol," in *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, Aug. 1998, pp. 167–177.

[11] M. Scott Corson and Anthony Ephremides, "A distributed routing algorithm for mobile wireless networks," *Wireless Networks*, vol. 1, no. 1, pp. 61–81, feb 1995.

[12] P. Krishna, N. Vaidya, M. Chatterjee, and D. Pradhan, "A cluster-based approach for routing in dynamic networks," 1997.

[13] Yu-Liang Chang and Ching-Chi Hsu,   "Routing in wireless/mobile ad-hoc networks via dynamic group construction,"   *Mobile Networks and Applications*, vol. 5, no. 1, pp. 27–37, 2000.

[14] G. Pei, M. Gerla, and X. Hong,   "Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility," 2000.

[15] Brad Karp and H. T. Kung,   "GPSR: greedy perimeter stateless routing for wireless networks," in *Mobile Computing and Networking*, 2000, pp. 243–254.

[16] K. Sohrabi, J. Gao, V. Ailawadhi, and G. Pottie,   "Protocols for self-organization of a wireless sensor network,"   in *IEEE Personal Communications Magazine, Vol.7, No.5*, Oct. 2000, p. 1627.

[17] Wendi Rabiner Heinzelman, Anantha Chandrakasan,   and Hari Balakrishnan,   "Energy-efficient communication protocol for wireless microsensor networks," in *HICSS*, 2000.

[18] J. Jubin and J. D. Tornow, ""the darpa packet radio network protocols"," in *Proceedings of the IEEE*, Jan 1987, vol. 75, pp. 21–32.

[19] Sandra McLeod and David Partain,   "SNMP Object Identifier Compression," Apr 2001.