Efficient Handling of Sensor Failures *

Andreea Berfield, Panos K. Chrysanthis, Alexandros Labrinidis
Advanced Data Management Technologies Laboratory
Department of Computer Science
University of Pittsburgh
{andreea, panos, labrinid}@cs.pitt.edu

ABSTRACT

Sensors provide unprecedented access to a wealth of information from the physical environment in real-time. However, they suffer from a variety of resource limitations, most importantly power consumption and communication bandwidth. Additionally, environmental conditions can contribute to sensor failures, disrupting the flow of query results. In this paper, we propose new techniques to deal with sensor failures based on the principles of partition and single path redundancy. Our experimental results confirm the efficiency of our techniques with respect to different performance metrics in general, and, in particular, high quality of data.

Keywords: fault tolerance; sensor networks; multiple routing trees.

1. INTRODUCTION

In today's pervasive environments, networked sensors provide unprecedented access to a wealth of information from the physical environment in real-time. The application space is very wide, ranging from habitat to critical device monitoring. The challenge of meeting the expectations for sensor network applications is compounded by the fact that sensors suffer from various crippling *limitations*, like power consumption and limited CPU and memory. Another limitation is the communication bandwidth, which implies the usage of infrequent and short messages. Lastly, a direct consequence of being exposed to the elements of nature are node and communication failures. The challenge becomes how to operate the sensors under these conditions with minimal impact on their usefulness, that is quality of data (QoD) and quality of service (QoS).

We assume that sensors are distributed across the area of interest and before transmitting any results towards the base station (BS), they self-organize into a hierarchical structure, called *routing tree*, in accordance to some query plan. The underlying idea of our fault tolerant (FT) scheme and proposed FT routing algorithms, called FDR-SN, is to partition the sensor network into *multiple routing trees* (MRT), which will ensure the desired properties of the sensor network (scalability, load balancing, reliability, response time).

*Funded in part by NSF ITR Medium Award (ANI 0325353).

3rd International Workshop on Data Management for Sensor Networks (DMSN'06), held in conjunction with the VLDB 2006 Conference, Seoul, Korea, September 2006

FDR-SN is a *single-path redundancy* scheme in which data is sent every round on a different path to increase the chance of the user obtaining a result, as opposed to *multiple path redundancy* which requires the information to be sent on different paths simultaneously [4, 12]. FDR-SN takes advantage of the MRT configuration of nodes to repair local broken communication links by re-routing query results across partitions whenever necessary. The principles of (data/query) partition and single-path redundancy [18] are not necessarily new, but our contribution lies in providing an efficient means to integrate them to achieve high QoD in the presence of sensor failures.

In this paper, we focus only on FDR-SN algorithms and specifically on how to route the data within a single routing tree or among MRT and how to synchronize the MRT. In [11], we discussed the issues of partitioning related to our "divide and conquer" approach to FT, i.e., how many partitions are necessary and what is the optimal placement of the BS for each partition.

The advantages of using our FT scheme include:

- no assumptions on the topology of the network,
- on-demand local and global reconstruction of the MRT (performed when needed, not periodically),
- no additional message overhead compared to basic schemes, and
- larger query space (no need to worry about duplicate sensitivity of aggregates, as in the case with multiple-path redundancy).

Our experimental results confirm the efficiency of FDR-SN to provide high QoD under different failure situations.

In the next section we introduce the MRT system model, followed by the presentation of FDR-SN in detail. We discuss our experiments in Section 4 and related work in Section 5.

2. SYSTEM MODEL

We are assume a sensor network consisting of nodes with typical communication and computation capabilities that process continuous queries (and possibly perform in-network aggregation) with a specific *collection window* such as:

SELECT count()
FROM sensors
WHERE Temperature $\geq$ *10*
COLLECTION WINDOW 15

The COLLECTION WINDOW represents the arrival rate of new results expected by the user. This is what is referred to as EPOCH DURATION in TAG [9] and as INTERVAL UNIT in Cougar [17].

Applying the principle of (data/query) partition, our *multiple routing trees* (MRT) system architecture configures the sensor network into multiple disjoint partitions of sensor nodes. Our system assumes an overlay network of base stations (BSs), with one BS for each partition, that cooperatively process continuous queries. Each partition is disjoined and within each partition, the query processing is done in a similar fashion as in TAG and Cougar. Each node sends its most recent reading and processing result during a specific *transmission period* or *transmission window* (called sub-epoch in TAG and time interval unit in Cougar) which is determined by the specified query collection window and the node's level in the routing tree.

When a query is submitted to the sensor network, it is forwarded to all BSs. Each BS in our system initiates the construction of a routing tree. Any routing tree construction algorithm can be used, from simple schemes such as those used in TAG and Cougar [17] protocols, to more sophisticated ones as GANC in Tina [14] and multi-criteria routing protocols [8]. The only possible extension to these routing protocols is in the selection of a parent from potential parents belonging to different routing trees (partitions). In such a case, a node could decide to join a parent which is situated on the lowest level across MRTs. This will ensure that the MRTs are shallower, which means that the nodes will be allocated larger transmission periods, hence minimizing the possibility of collisions. Collisions are a source of energy waste as well as of lost readings, since package collisions impact the quality of data. In any case, by joining a routing tree, a sensor node becomes part of the partition of the BS at the root of the tree. The cost of constructing an MRT is the same as for constructing a single larger RT.

The MRT system architecture allows for scalability, for faster response time (i.e., smaller collection windows) and ensures load balancing, which decreases the energy consumption per node. Furthermore, besides avoiding a single point of failure, it allows for faster recovery from failures as we discuss in the next section by assuming only that the communication links are symmetric, which means either both communicating sensors or none of them can receive each other's messages. Taking in consideration asymmetric communication links is subject of future research.

3. FAULT DETECTION AND RECOVERY IN SENSOR NETWORKS (FDR-SN)

The MRT system architecture handles failures dynamically, on demand in two different ways: using global and local reconstruction of the MRT. Global reconstruction means that the BSs resynchronize in order to reconstruct the MRT. Local reconstruction, as we will discuss below, repairs broken links by reconfiguring subtrees within a partition or across two neighboring partitions.

3.1 FDR-SN Algorithms

FDR-SN takes advantage of the MRT configuration of nodes by sending results towards the BSs through one of the sensor's parents, cohorts, or neighbors.

DEFINITION 1. *Neighbor* of a sensor node is any node that lies in its broadcast range and which belongs to a *different* routing tree.

DEFINITION 2. *Cohort* of a sensor node is any node that lies in its broadcast range and belongs to the *same* routing tree, but it is not a child, or parent of the node and also not a sibling, i.e., has a different parent.

The detailed rules for message handling and forwarding synthesize the three algorithms of FDR-SN: *Sending/Forwarding Message* (Algorithm 1), *Reintegrate in the Network* (Algorithm 2) and *Process Received Message* (Algorithm 3).

FDR-SN functions as follows: every time a sensor has to send or forward a message towards the BSs, it picks a destination for the message. First it tries to send data to one of its (possible) parents discovered during the routing tree construction phase, with the hope that it is still within the node's broadcast range. All possible parents of a node belong to the same partition (i.e., have the same routing tree ID). A node normally has only one *active parent*, but it also keeps track of a limited number of possible parents[1]. During the next time slot, the node checks if its parent has forwarded any message or not. If not, the node will consider this parent dead and it will send its next message to another parent.

Algorithm 1 Sending/Forwarding Message

```
1: Create message
2: if myNode.numberParents ≠ ∅ then
3:    Select parent as destination for the message
4:    messsage.type = NORMAL_MESSAGE
5:    Send Message
6: else
7:    if myNode.numberCohorts ≠ ∅ then
8:       Select cohort as destination for the message
9:       myNode.parentID = cohortID
10:      messsage.type = COHORT_MESSAGE
11:      Send Message
12:   else
13:      if myNode.numberChildren ≠ ∅ then
14:         Select child as destination for the message
15:         myNode.parentID = childID
16:         messsage.type = CHANGE_PARENT
17:         Send Message
18:      else
19:         if myNode.numberNeighbors ≠ ∅ then
20:            Select neighbor as destination for the message
21:            myNode.parentID = neighborID
22:            myNode.numberChildren = 0
23:            myNode.level = neighbor.level − 1
24:            messsage.type = NEIGHBOR_MESSAGE
25:            Send Message
26:         else
27:            Reintegrate in the network
28:         end if
29:      end if
30:   end if
31: end if
```

In the case in which the node has no parents alive that it is aware of, it will try to send its message in a similar manner through its cohorts. By having a different parent, a cohort increases the chances of a node within the same tree to successfully forward the message. If no cohorts are available either, then it will try to select one of its children. The last option for choosing a destination is to select one of its neighbors. Only the nodes located in the vicinity of other routing trees have neighbors (the system may even have overlapping routing trees, but this is not a requirement).

The nodes dynamically update their parents, neighbors, children and cohorts lists, based on the messages they hear transmitted in the network. At any given point, the choice of destination is made randomly, but it can be based on multiple criteria which are used

[1]Fractional TAG [9] protocol also keeps track of several parents, but unlike FDR-SR, it sends to each parent a fraction of its result.

Algorithm 2 Reintegrate in the Network

```
1: Snoop incoming messages for a period of time
2: if (possibleParent == ∅) or (all possibleParent.numberChildren > threshold) then
3:    Go to sleep
4: else
5:    Select parent as destination for message
6:    myNode.parentID = message.nodeID
7:    myNode.level = message.level + 1
8:    message.type = NEW_PARENT
9:    Send message
10: end if
```

Algorithm 3 Process Received Message

```
1: Receive message
2: if message.type == CHANGE_PARENT then
3:    Remove message.n from myNode.parents
4:    if message.parentID == myNode.nodeID then
5:       Add message.nodeID to myNode.children
6:    end if
7: else
8:    if message.type == COHORT_MESSAGE then
9:       if message.cohortID == myNode.nodeID then
10:         Remove message.nodeID from myNode.cohorts
11:         Add message.nodeID to myNode.children
12:      end if
13:   end if
14: else
15:   if message.type == NEIGHBOR_MESSAGE then
16:      if message.neighborID == myNode.nodeID then
17:         Remove message.nodeID from myNode.neighbors
18:         Add message.nodeID to myNode.children
19:      end if
20:   end if
21: else
22:   if message.type == NEW_PARENT then
23:      Add message.nodeID to myNode.children
24:   end if
25: end if
26: Update myNode.parents, myNode.children, myNode.cohorts, myNode.neighbors lists
27: Apply Sending/Forwarding message algorithm
```

in a general routing construction [14, 8]. Having a randomly selected, variable destination for the messages, FDR-SN ensures a higher probability to meet the randomness of the failures in a sensor network, while achieving better load balancing under normal conditions.

3.2 Implementation of FDR-SN

In this section, we discuss the key issues for implementing FDR-SN.

3.2.1 Message Overhead

Under all protocols, in each message, nodes always transmit their node ID and their readings. Our scheme needs to transmit the following information in each message: *node ID, parent ID, level ID* and *tree ID*. The sender's parent ID is needed to make the distinction among children, parents and cohorts for a node, all of them having the same tree ID. Compared to Cougar or TAG, our routing protocol transmits two additional pieces of information: (1) the tree ID and (2) one of the two variables, either parent ID or level ID.

In order to make message transmission more efficient, we propose to compact different pieces of information into a single message of normal size, instead of sending several messages. For example, the standard length of a message for the Mica motes is 36 bytes [16], from which 14 bytes represent the header (it can slightly vary with the TinyOS version and platform used). We can utilize the remaining 22 bytes as follows: the node ID and parent ID can be encapsulated in the message header. The tree ID, level ID can fit in one-two bytes, for a reasonable size network.

3.2.2 Routing loops

A key concern with the dynamic selection of destinations in the presence of failures is the construction of routing loops as in Fig. 1. In this example, node 1 transmits its message to node 2, node 2 chooses to forward it to node 3, one of its parents. Assuming node 3's parents are experiencing failures, node 3 has only two choices to forward the message, either to node 2 or to node 4. Because it does not want to send back messages to their originator, the only valid choice is node 4. In the same manner, node 4 forwards the message to its original source, node 1. Node 1 is now forced to find other path to send its message or to drop it. Even if node 4 would have selected node 2, its cohort, as the destination for its message, a routing cycle would have appeared anyway.

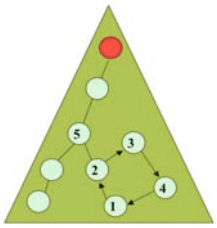

Figure 1: Forwarding Messages - Loops

One approach to prevent routing loops is by using a data cache in each node, which records the recently seen data items [7]. Instead of keeping an expensive data cache, we decided to modify the routing of the messages in order to avoid the routing cycles, so we do not need to keep an expensive data cache. In our case, the problem may appear whenever a node in the sensor network cannot forward the data to its parents and a local repair of the routing tree must take place if possible. We solve this potential problem by having a strict order in which a node tries to send its reading and special messages that inform a node that its parent has just became its child and it needs to find a new parent. These steps are shown in Algorithms 1-3. In short, a sensor will try to send its result to one of its possible parents. If none of its potential parents is alive, it will try recursively to select as destination for its message one of its cohorts, children or neighbors, in this order.

3.2.3 Synchronization of MRT

The two traditional synchronization protocols between parents and children in a routing tree are TAG [9] and Cougar [17]. TAG uses the notion of *epoch* (which corresponds to our collection window defined above), which represents the arrival rate of new results expected by the user. Nodes at a given level (depth) of the routing tree transmit their messages during a pre-defined period called *sub-epoch*.

Cougar delays the sending of a node's reading until it receives the values from its children nodes or its wait time interval expires.

The amount of time waited by a parent node is directly related to node's level in the network, with nodes close to the roots waiting a longer period of time than leaf nodes. The overall time between which readings are produced and propagation begins is defined by the collection window (as specified by the requested query).

While these synchronization schemes work well for individual routing trees, they cannot be used to synchronize MRT because the transmission periods in both schemes are directly related to the depth of the trees. If the routing trees have different depths, then it is not possible to communicate between trees easily. The purpose of synchronizing the MRT is not only to allow communication among them without inducing extra collisions, but also to present to the user data from the same collection window. Ensuring data consistency in sensor networks is as important as the very existence of the responses.

We have identified two ways to achieve the desired synchronization among nodes from MRT. First one is to choose a common collection window. The second way is to find a relationship between the transmission windows (i.e., TAG's sub-epoches or Cougar's time subintervals) available for nodes in different routing trees and use it when necessary. We will call these two approaches *collection window synchronization* and *hiccup* method for special cases. The"hiccup" method is less restrictive than the synchronization one, it allows the MRT to run with different response windows, as long as all the trees transmit their results within the same collection window.

We have also identified several different methods to implement the "collection window synchronization" approach. The most promising solution is:
$Common\_response\_window = Min(response\_window\_i)$**.**
In this case, all the trees act as if they have the same number of levels. Consequently, some trees might be inactive some short periods of time, because they do not have as many levels as others, but the overall expected response time will be the same as for the single tree.

The "hiccup" method can also be implemented in a number of different ways, all of which are based on the observation that all the trees loosely synchronize at the beginning of every collection window (more details in [1]). We assume that the leaf nodes of each routing tree are in the communication range of some of the leaf nodes from other routing trees (we have previously defined these nodes as neighbor nodes).

The proposed solutions are:

1. Every leaf node stores the *response window* values for its tree and its neighboring trees in the MRT construction phase. Based on that, it knows when to try to transmit its message to the neighboring nodes. Because the nodes store information only about the neighboring trees, not about all the routing trees in the network, the overhead required to store the additional information is insignificant.

2. Every leaf node that wants to transmit to a neighboring node listens until it hears it transmitting its message. This implies that the first node is able to transmit its message to the second node. The maximum wait time before transmitting a message is twice the value of the biggest response window of the two nodes.

3.2.4 Applying FDR-SN to TAG and Cougar

FDR-SN can be can be seamlessly integrated with Cougar and applied on top of TAG. Every node keeps only a short list of parents, children, cohorts and neighbors for routing purposes[2]. This list represents the next *possible hop* list.

FDR-SN-Cougar has all the information needed when the routing trees are constructed: children send special messages to the chosen active parent node to confirm their selection as parents. For the rest of the parents, children are on one level lower than the parents, in the same routing tree. Neighbors are easy to detect because they have a different tree ID number. The distinction between cohorts and possible parents can be made based on the level ID included in the message. The potential parent is situated on a lower level than the cohort. Because we also send the parent ID with the message, it is easy to detect the nodes which belong to the same tree, but have the same active parent , i.e., are not cohorts.

In Cougar, there is no expicit transmitting and listensing only phases, so it is not hard for the node to *snoop* and detect if the destination node of its most recent transmission has forwarded any message or not. If this destination is not active any more, it removes it from its next possible hop list and in the future it will choose another node as the destination for its message.

FDR-SN-Tag is mostly the same, except that children do not send confirmation messages to their parents. However, when they broadcast their parent advertisement, they include their parent ID, so that the actual parent can recognize its ID and add them to its children list. The problem here is how a node detects that its messages were not forwarded on a specific path, because the destination node has failed. In FDR-SN-Tag children sample the medium often enough so they can detect if the parent has transmitted a message or not. Another solution is to extend the notion of snooping in TAG in a way similar to FDR-SN-Cougar (mentioned above). In TAG, snooping is used by node to monitor its siblings to decide if it makes sense to transmit its value after a collision.

In implementing FDR-SN-Tag and FDR-SN-Cougar, we encountered the interesting issue of how long will the sensors wait for incoming messages when the routing tree is unbalanced or skewed. Neither Tag nor Cougar handle unbalanced trees very well. In FDR-SN-Cougar or FDR-SN-Tag, this problem is even more challenging because the nodes can receive a message from a potential child that it is not currently in their children list or even from their parent, cohorts or neighbors. Our solution is that each node has an estimate (count) of not only their immediate neighboring nodes, but also of the size of their sub-trees. This information can be transmitted along with the message.

Another interesting situation is when one sensor alone gets moved. In such case, it can be quickly reintegrated into the network. The node that needs to be reintegrated should snoop the packages sent in the network for a period of time, select a destination and send a message to it with its reading and NEW_PARENT header. The node can set its own level and tree ID based on the message it intercepted before, selecting the transmitting node as parent node. When receiving a NEW_PARENT message, a node should add this node to its children list and treat it accordingly. We do not introduce any further acknowledgments between the new parent node and its child in order to keep the number of messages sent in the network as low as possible.

4. EXPERIMENTAL EVALUATION

In our experiments, we adopted Cougar as the underlying synchronization scheme for data aggregation throughout the network. Notice, however, that our proposed routing scheme actually works independently of Cougar and will thus work with any synchronization scheme built on the parent/child concept. For the synchronization of the MRT, we used the epoch synchronization method (de-

[2]Not all of the parents, children, cohorts and neighbors are kept in the lists, just a fixed number of each type.

fined in Sec 3.2.3). We evaluated four protocols: **Cougar** (original protocol), **Fractional Cougar**, a variant of Cougar where nodes send partial data to each parent, **Fractional Cougar with Reconstruction**, a variant of the previous protocol that performs routing trees reconstruction periodically (every 10 epochs), and **FDR-SN-Cougar**, our proposed scheme (or FDR-SN for short).

Network We created a simulation environment using CSIM [13]. We experimented with grid topologies of different sizes (from 25*25 to 45*45). Due to space limitations, we are going to show only the results for the 45*45 network, the rest of the simulations on different size networks were similar. For collision avoidance, we used a contention-based MAC protocol (PAMAS)[15]. In this protocol, a sender node will perform a carrier sensing before initiating a transmission. If a node fails to get the medium, it goes to sleep and wakes up when the channel is free.

Topology In [11] we showed the numerous benefits obtained by using MRTs. Adding our routing scheme to the system architecture improves the fault tolerance of the system and extends over existing benefits. However, our fault tolerant routing is not dependent on the MRT topology; it functions equally well on a single routing tree system architecture. Our experiments in [11] concluded that four root nodes placed in the middle of the borders of the grid configuration give us the best case scenario for our given topology.

Metrics In this paper, we use the quality of the reported data as our performance metric. The *relative error metric* (REM) is a measure of how close the reported answer is to the exact answer, based only on the nodes that are still alive. We use REM as indication of the *quality of data* (QoD), where a high REM reflects a low QoD, while a low REM corresponds to a high QoD. Hence, we will be using both terms interchangeably.

We did not use energy as a performance metric in these experiments because FDR-SN has comparable energy consumption as Cougar. FDR-SN does not add any extra message transmissions, nor increases message size. The only penalty in energy FDR-SN has is due to the need to stay awake for snooping, i.e., incurring only listening cost, rather than transmitting cost. All intermediate nodes need to stay awake only an extra time slot. The leaf nodes also stay awake to hear any possible message from their neighbors. They will stay awake for as many time slots as the max difference in level between them and their neighbors.

Failures We have considered different distributions of the failed nodes. We studied different possible fail-stop sensor node failures, because node failures implicitly cover the case of communication failures and are more difficult to deal with. We varied the number of nodes that fail as a percentage over the total number of nodes (from 1% to 10%). In order to provide a complete analysis of the types of failures sensors suffer from and their effects upon the sensor network, we varied the distribution of failed nodes in two other ways: space and time. The *distribution of node failures in space* is achieved considering two possible cases: all failed nodes are in the same geographical area (area case) or the failed nodes are scattered through the network (point case). For the *distribution of node failures in time* we have also examined two cases: all nodes failing concurrently (simultaneous case) versus nodes failing over a period of time (incremental case). We performed experiments with all possible combinations of these parameters for all four protocols (Normal Cougar, Fractional Cougar, Fractional Cougar with Reconstruction and FDR-SN protocols). The experiments confirmed our intuition that the type of node failures influences significantly the performance of the sensor network.

Table 1 summarizes the system parameters used in our experiments whose results are reported next.

Parameter	Value	Default
Grid Size	25*25 to 45*45	45*45
Number BSs	1 to 4	1
Type of Node Failure	Area, Point, Simultaneous, Incremental	
Percent of Node Failed	1%, 5% and 10%	
Query Type	SUM, MAX, AVG	SUM
Randomness Degree	0.0 to 1.0	0.5
Random Step Size Limit	10% of domain	
Epoch Durations	280 - 1700 mSec	1500 mSec
Number of Epochs	100 Epochs	
Reconstruction Period	10 Epochs	

Table 1: System Parameters

4.1 Experiments: Area vs. Point Failures

We conducted our experiments for two different network layouts: grid network with one BS placed in the center of the grid (Fig. 2 and Fig. 3) and four BSs placed in the middle of the borders of the grid (Fig. 4, 5, 6, and 7, 8 and 9). For each column in the graphs, we executed five experiments and report the average value of REM.

If we compare the errors reported in Fig. 2 and Fig. 3 (single BS), with the corresponding results in Fig. 4 and Fig. 7 (four BSs), we can easily see that having an MRT architecture (bottom plots) helps Fractional Cougar protocol in any circumstance and Cougar protocol in most of the cases, especially for larger fractions of failed nodes. Fractional Cougar with Reconstruction protocol displays a spiky behavior, with peak values reaching the 100% error limit every time the routing trees are reconstructed and almost constant average REM. During the two cycles spent with the MRT reconstruction, the user cannot obtain any useful information from the system, therefore the error is maximum and it ends up out-weighting most of the benefits obtained by the reconstruction in the first place. As we had expected, FDR-SN out-performs all other routing protocols no matter what is the underlying network topology. In fact, the REM for FDR-SN is hardly noticeable on the plots (rightmost columns).

From the graphs presented in this paper and the additional experiments performed, we can also conclude that the system reacts much better to the area distribution for a given percentage of failed nodes (Fig. 3, Fig. 6 and Fig. 7) than to the point distribution of the same number of failed nodes (Fig. 2, Fig. 4 and Fig. 5). The explanation of this phenomena resides in the fact that for area failures, the failures are contained and have minimum effect over the other sensors in the network, whereas for point failures, the failures are scattered all over the network and more functional sensors find themselves unable to transmit their values towards the BS(s). If the number of failed nodes increases, then the differences disappear, because the percentage of failed nodes is so big, that the distribution of nodes cannot make a difference any more.

Another interesting question that we tried to answer is how does Cougar perform compared to Fractional Cougar. For simultaneous failures, Fractional Cougar performs better than Cougar. If failures are also point failures, than the difference between the REM is significant, whereas for the area failures, the difference in the reported REM is not that significant any more. Since Fractional Cougar (or TAG) was proposed as a solution to deal with failures in the network, this result is exactly what we had expected. However, the performance results are reversed in the case of incremental failures, when Cougar performs better than Fractional Cougar. This result should not be surprising either, because by sending the result

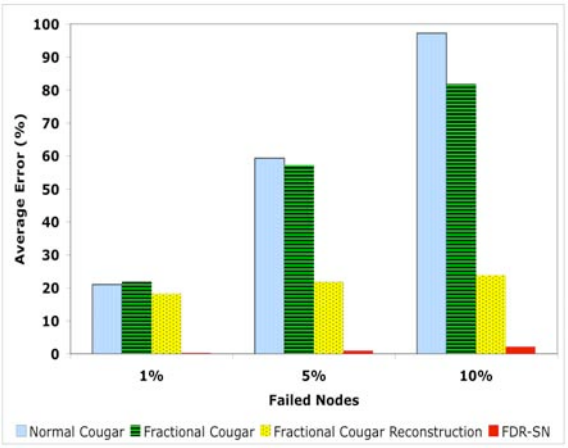

Figure 2: Simultaneous Point Failures under 1 Base Station

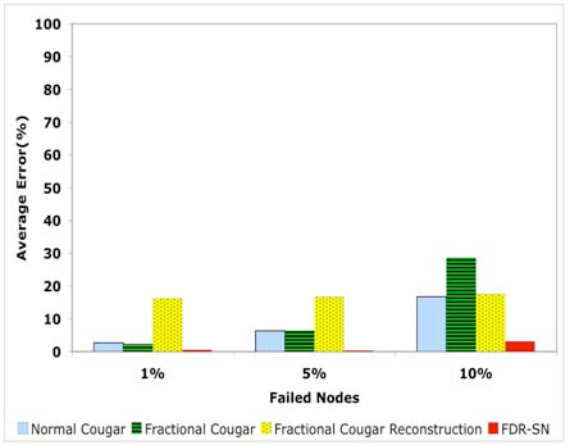

Figure 3: Incremental Area Failures under 1 Base Station

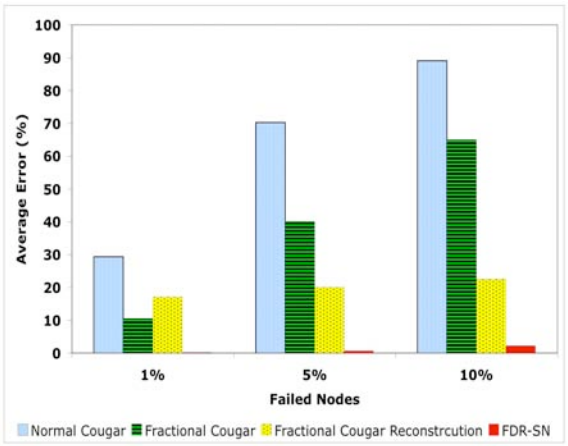

Figure 4: Simultaneous Point Failures under 4 Base Stations

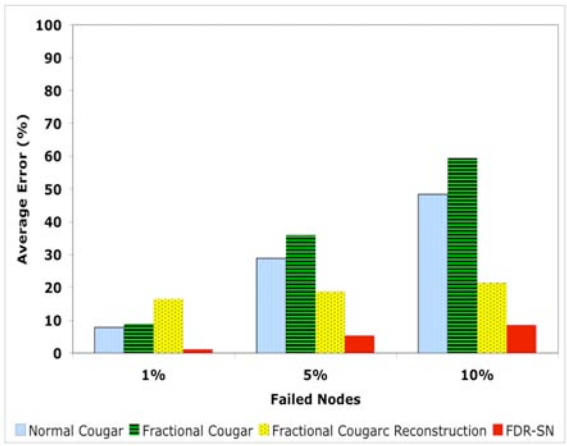

Figure 5: Incremental Point Failures under 4 Base Stations

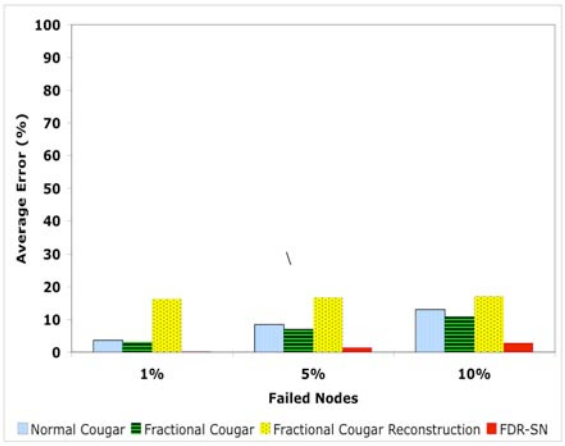

Figure 6: Simultaneous Area Failures under 4 Base Stations

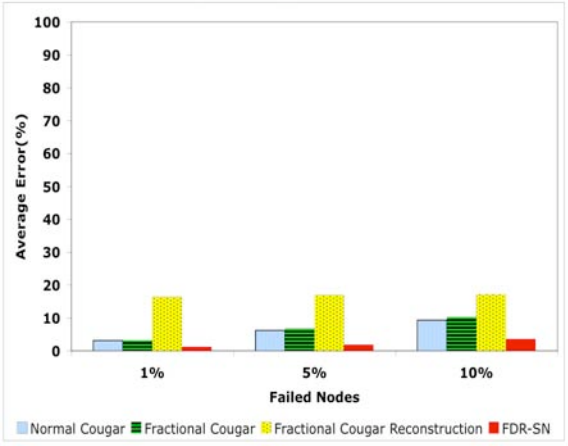

Figure 7: Incremental Area Failures under 4 Base Stations

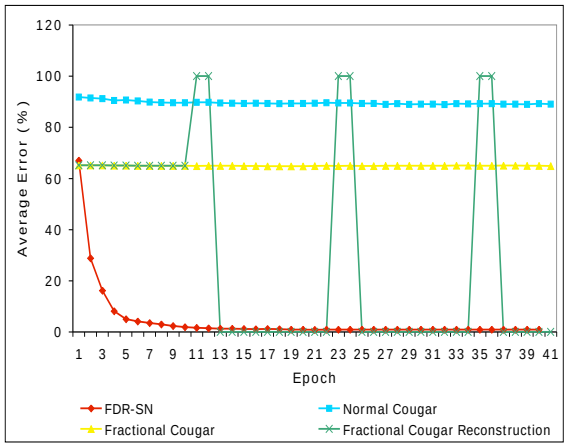

Figure 8: Network Adaptation under Simultaneous Point Failures (4 Base Stations, 10% failed nodes)

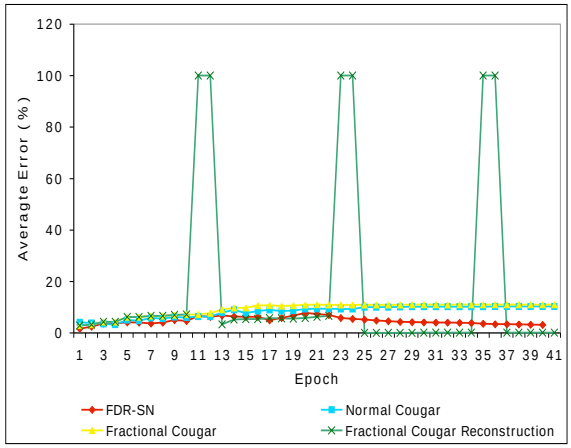

Figure 9: Network Adaptation under Incremental Area Failures (4 Base Stations, 10% failed nodes)

on multiple paths, more nodes may loose a fraction of their transmitted result, even though they still have a functional path to the BS. Therefore, Fractional Cougar is not the best way to deal with sensor failures in general.

4.2 Experiments: Adaptation over Time

Fig. 8 and Fig. 9 present the adaptation in time to the environment of all protocols. The graphs show the case of 10% of the nodes failures, distributed randomly in the network. For simultaneous failures, there is a big REM reported immediately after the simultaneous failure of the nodes, because nodes are not expecting the destination of their message to be dead. FDR-SN adapts over time, and learns by snooping that the destinations are not functional any more, so that the nodes pick another destination next time. The time needed by the network to stabilize is very short. For load-balancing purposes, the algorithm does not pick the same destination all the time, so mistakes may still appear and we see a slight variation of the reported REM. For the incremental failures case, the REM increases until we finish introducing failures in the network and the network can stabilize. Nevertheless, the error does not increase linearly, because nodes have a chance to learn which nodes are faulty on the fly. If we look at the other protocols, Cougar or Fractional Cougar protocols do not adapt in time.

5. RELATED WORK

Fault tolerance in sensor networks has been investigated with different approaches which can broadly be classified into *single-path redundancy* (SPR) and *multiple-path redundancy* (MPR). Besides Cougar with local repairs[18], another early SPR scheme is [5]. In this scheme, the exact locations where the sensors should be placed are computed in order to have maximum impact and a minimum number of sensors required, while supporting gracefully a number of K node failures.

A representative MPR scheme is "fractional parents" [9] which works by dividing the aggregate into fractions and sending each one to a different parent. Among the most successful MPR schemes are Sketches [4] and Synopsis [12], which use duplicate-insensitive multi-path forwarding. The main disadvantage of these approaches is that the communication cost and computation time requirements implied are very high for most of the aggregate functions. Even though both Sketches and Synopsis have a very low error rate, the results presented to the user always include an error, even when the network does not suffer from failures.

Tributaries and Deltas [10] have been proposed as an improvement over Synopsis, which are potentially very expensive and always return approximate answers, even in the absence of failures. Tributaries and Deltas use a hybrid of single-path and multi-path approaches, having the goal of combining the best of the two worlds. Whenever the number of participating nodes exceeds a threshold, the BS informs the rest of the network to expand or shrink the delta region. This scheme still suffers from some drawbacks: the BS is a single point of failure, the expansion and shrinkage of the delta region (similar with routing tree reconstruction) happens in a centralized fashion and during this period the user is unable to get results from the network. The initial performance of the scheme is highly dependent on the percentage of single-path to multi-path nodes in the network. Furthermore, if some failures happen at the leaves level in the routing tree, all the tree must become multi-path to deal with these failures. Even though the authors have proposed two variants of the scheme to cope with these problems, these either have limited applicability or require keeping track of huge amounts of information.

In order to put our contribution into perspective, we classified the related fault tolerant schemes across nine dimensions, capturing both functional and perfomance characteristics[1]. Due to space limitations, we did not include this taxonomy in this paper.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a new technique, *FDR-SN*, to handle sensor failures efficiently without much overhead or loss in quality of data. Our proposed scheme is based on the principles of data/query partitioning and single-path redundancy. We have shown how our scheme can be integrated with TAG and with Cougar. As a case study, we have experimentally shown that FDR-SN will outperform current extensions to Cougar for dealing with fault tolerance using single-path redundancy.

We envision FDR-SN working complementary to existing multi-path schemes, because the application space for which each scheme performs better is different, and intend to further study this as part of our future work.

The only underlying assumption of *FDR-SN* is that the communication links are symmetric. We are currently working on a scheme that handles asymmetric communication links where a sensor node can receive messages from another node, but the second node cannot receive from the first one.

Acknowledgments

We would like to thank Jonathan Beaver and Mohamed Sharaf for providing the initial simulator and for their useful feedback. We would also like to thank the anonymous referees for their helpful comments.

7. REFERENCES

[1] A. Berfield, P. K. Chrysanthis and A. Labrinidis. FDR-SN: Efficient Handling of Sensor Failures. TR-06-140, U. Pittsburgh, 2006.
[2] U. Cetintemel, A. Flinders and Y. Sun. Power-Efficient Data Dissemination in Wireless Sensor Networks. *DEWMA*, 2003.
[3] W. Choi, S. K. Das, and K. Basu. Angle-based Dynamic Path Construction for Route Load Balancing in Wireless Sensor Networks. *WCN*, 2004.
[4] J. Considine, F. Li, G. Kollios and J. Byers. Approximate Aggregation Techniques for Sensor Databases. *ICDE*, 2004.
[5] S. S. Dhillon and K. Chakrabarty. A Fault-Tolerant Approach to Sensor Deployment in Distributed Sensor Networks. *ASC*, 2002.
[6] G. Gupta and M. Younis. Load-balanced Clustering in Wireless Sensor Networks. *ICC*, 2003.
[7] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann and F. Silva. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. *MOBICOM*, 2000.
[8] Q. Li, J. Beaver, A. Amer, P. K. Chrysanthis, A. Labrinidis, and G. Santhanakrishnan. Multi-Criteria Routing in Wireless Sensor-Based Pervasive Environments. *JPCC*, 1(4): 313-326, 2005.
[9] S. Madden, M. J. Franklin, J. M. Hellerstein and W. Hong. TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks. *OSDI*, 2002.
[10] A. Manjhi, S. Nath and P. B. Gibbons. Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Streams. *SIGMOD*, 2005.
[11] A. Munteanu, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. Multiple Query Routing Trees in Sensor Networks. *DBA*, 2005.
[12] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis Diffusion for Robust Aggregation in Sensor Networks. *SenSys*, 2004.
[13] H. Schwetman. CSIM User's Guide. *MCC Corporation*, 1992.
[14] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. Balancing Energy Efficiency and Quality of Aggregate Data in Sensor Networks, *The VLDB Journal*, 13(4), 2004.
[15] S. Singhand and C. Raghavendra. PAMAS: Power Aware Multi-Access Protocol with Signalling for Ad Hoc Networks. *CC, Volume 28 , Issue 3*, 1998.
[16] R. Zheng. Design, Analysis and Empirical Evaluation of Power Management in Multi-hop Wireless Networks. *UIUCDCS-R-2004-2381*, PhD Thesis.
[17] Y. Yao and J. Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks. *Sigmod*, 2002.
[18] Y. Yao and J. Gehrke. Query Processing for Sensor Networks. CIDR, 2003.