

# Fault Repair Framework for Mobile Sensor Networks

Tuan Le<sup>1,2</sup>  
tuan.le@nicta.com.au

Nadeem Ahmed<sup>1,2</sup>  
nahmed@cse.unsw.edu.au

Nandan Parameswaran<sup>1</sup>  
paramesh@cse.unsw.edu.au

Sanjay Jha<sup>1</sup>  
sjha@cse.unsw.edu.au

<sup>2</sup>National ICT Australia (NICTA)  
Australian Technology Park, Sydney  
Australia.

<sup>1</sup>University of NSW, Sydney  
Australia.

**Abstract**—In this paper, we propose a framework for fault repair in mobile sensor networks. A hierarchical structure which consists of replacement module, management policy module, knowledge module, decision making module, and evaluation module is adopted. We also propose a solution for faulty sensor replacement problem. Through the numerical results, we show that our algorithm is more efficient and achieves higher energy savings than the greedy approach to sensor replacement. We believe that the problem of faulty sensor nodes can be solved efficiently through the cooperation and communication across different modules, such as evaluation decision making, knowledge management, and replacement.

## I. INTRODUCTION

In general, fault is the incorrect state of hardware or a program as a consequence of a failure of a component [1]. Permanent faults are the ones resulting from systems or communication hardware failure. For example, node may die due to battery depletion. An intermittent fault is one that has only incidental appearance due to unstable characteristic of the hardware. A transient fault is one that is the consequence of temporary environmental impact on otherwise correct hardware. For example, the change in environment may cause incorrect sensor reading. In this paper we consider only permanent fault, which once activated remains continuous until it is detected and repaired.

One common characteristic of nodes in wireless sensor networks is that they are prone to failure. Sensor nodes carry limited, generally irreplaceable, power sources. Nodes in sensor networks can fail for many different due to several reasons: their batteries may be depleted, they may be accidentally destroyed, and a malicious adversary may deliberately incapacitate them. As time progresses, faults will occur more often in sensor networks. Sensor network can continue to operate and provide services even with loss of some of the sensor nodes. However, the quality of offered services, such as coverage, is greatly degraded upon loss of few core nodes. The network loses utility when it does not provide the required coverage. Moreover, sensor failure may cause network topology changes and in extreme cases, network partitioning. Messages may still flow through the network despite these partitions. However, the resulting paths may have a longer delay, which

is unacceptable in some applications. Therefore, to overcome sensor node failure and to guarantee system reliability, faulty nodes should be detected and repaired promptly.

On the other hand, in most cases faulty sensors can not be easily replaced manually. Especially, in cases involving a polluted area or a hazardous chemical leak in a building [2], it is too dangerous for a human to access the site for sensor replacement. Other applications such as military surveillance and smart homes may not only require maintaining the original sensing topology but also extending the existing sensing coverage. In such cases, mobile sensors equipped with movement capabilities are a potential solution. For example, sensor nodes may be placed at the entrance of the building, allowed to proceed inside the building and find the desired position. However, the energy consumption for movement itself is costly. Hence, a method that minimizes such a cost is needed to improve system utility.

This paper presents our preliminary ideas on a fault repair framework for mobile wireless sensor networks. We introduce policy based management in conjunction with learning techniques. We also describe an off-line algorithm for replacement module of our architecture using Integer Linear Programming formulation and numerical results to support our intuition.

The rest of the paper is organized as follows. We propose fault repair architecture for sensor network in Section II. In Section III, we describe an initial solution for replacement module of our proposed architecture and show some numerical results. We discuss the related works in Section V. Section VI discuss some of the future works and concludes the paper.

## II. FAULT REPAIR ARCHITECTURE

The objective of fault repair is to maintain the overall health of a sensor network. The health of network here is the current sensing coverage. Assume that a sensor network is deployed to monitor a certain target area, which is divided into different sections. It is likely that the coverage requirement is different in different sections of the area. There are major sections which may require high coverage, while other sections may accept lower coverage. As sensor network is prone to failure, the major reason for coverage loss is faulty nodes. Therefore,

given a coverage distribution over the entire area, we want to maintain adequate coverage of the network.

Faults in sensor network have different characteristics than traditional networks. We discuss some of the distinguishing characteristics of fault repair for wireless sensor networks here.

### 1) Resource Limitation

Limited resources is a major issue for fault repair in sensor networks. As the lifetime of a sensor node is restricted to the limited battery power, an excessive communication burden on nodes to locate a faulty sensor is certainly unacceptable. So, fault detection and repair should spend as little energy as possible.

### 2) Response Time

To assure an application's reliability, the faulty node may need to be fixed quickly. For example, faulty sensors may reduce the coverage. As a result, the monitoring task may become unreliable. Faulty sensors should be replaced as soon as possible to guarantee continuing network performance. The response time for repair is thus also an important factor that should be considered.

### 3) Flexibility

As sensor networks contain a large number of nodes, the overall system behavior may not be affected considerably in the presence of few faulty nodes. Thus, faults may not need to be repaired unless they do cause a problem. If faulty sensors are located in an unimportant region, they can be ignored. Similarly, if faulty sensors have a minor impact on coverage, they can also be ignored. So, the fault repair should be smart enough to decide which nodes are eligible for replacement.

### 4) Adaptive

Fault repair framework should be adaptive to the operational surroundings. e.g. the coverage requirement in a section may change due to a variety of reasons. For instance, if the number of events increases suddenly in this section, this section needs to adjust the coverage resolution suitably. The framework should observe the dynamic changes, learn dynamic behavior (training) and make predictions. Adaptability is thus a desirable feature in fault repair for sensor networks.

### 5) Scalability

Scalability is a significant issue since a sensor network is usually deployed on a large scale.

Therefore, fault repair in sensor networks is more complicated and have different characteristics than in the traditional telecommunication network.

### A. Network Topology

We consider a randomly deployed sensor network that consists of a set of sensor  $S$  in a two dimensional area  $A$ .

$$S = \{S_1, S_2, \dots, S_n\}; \quad (1)$$

Each sensor  $S_i$  is located at coordinate  $(x_i, y_i)$  inside  $A$ . Let the sensors be grouped into  $m$  cluster

$$C = \{c_1, c_2, \dots, c_m\}; \quad (2)$$

each consisting of at most  $c$  sensors.

The sensor network is hybrid, consisting of both static and mobile nodes. Mobile nodes are initially considered as redundant nodes, not participating in the sensing/communication operation.

Each cluster maintains coverage above a certain coverage threshold  $TC$  with tolerance rate  $tr$ . The coverage in a section is thus to be maintained between the bounds given by Equation 3.

$$CoverageBounds = (TC - tr, TC + tr) \quad (3)$$

In Figure 1, region  $A$  may be divided into four clusters and each cluster may maintain different sensing coverage. For instance, cluster 1 may have  $TC = 70\%$  and  $tr = 5\%$ , while cluster 2 may require  $TC = 30\%$  and  $tr = 10\%$ . Coverage of cluster 1 is to be maintained between 65% and 75%, and that of cluster 2 between 20% and 40%, respectively. If sensing coverage requirement changes, e.g., environment change, application requirement, etc, the sensing coverage threshold  $TC$  is re-evaluated.

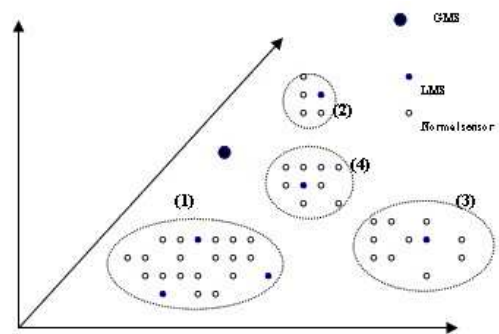


Fig. 1. Sensing Coverage

A set of sensors called *monitors* are deployed to observe the health of sensor network (Figure 1). Each cluster will contain local monitor sensor(s) (*LMS*) that observe its local health. It is assumed that the *LMS* has information about the current location of redundant sensor. The *LMS* thus form an overlay network on top of the sensing layer (Figure 2).

There is also a special monitor sensor, global monitor sensor *GMS*, that looks after the health of entire network. *GMS*

monitor the entire health of the network by accumulating the health of sub-regions reported from various *LMSs*. The *GMS* may be a micro server which is not resource constraint.

In a cluster, if the current sensing coverage is lower than a defined threshold due to faulty sensors, *LMS(s)* of the cluster will implement reasonable actions to recover the coverage. For example, mobile sensors in the network may be asked to relocate themselves in order to achieve a desired configuration. Since the cost associated with mobility is usually expensive, the total distance of sensor movement in the network should be minimized. Alternatively, a nearby redundant static sensor can be tasked to substitute a faulty sensor, if this sensor can help in maintaining the health of the network. *LMS* may also request a low energy sensor to reduce the number of communications, in order to extend this sensor's life. We discuss the fault repair actions in the next part.

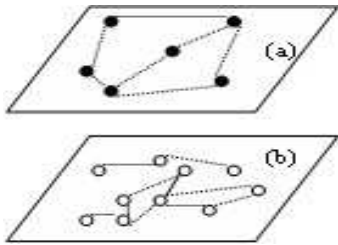


Fig. 2. a. Monitoring Layer b. Sensing Layer

### B. The Architecture

A hierarchical structure is adopted for fault repair, consisting of an evaluation module, a decision making module, replacement module, a management policy module, and a knowledge module (see Figure 3). The cooperation among these modules makes the framework efficient and adaptive. Following subsection discusses the individual role played by these modules in the fault repair framework.

1) *Evaluation Module*: Evaluation module is responsible for evaluating the health i.e. sensing coverage of the network. Based on the current status, it will report the coverage to the decision making module, in which an appropriate action to be performed is decided. Also, after a recovery action takes place, the final coverage is evaluated and validated in this module.

2) *Decision Making Module*: Decision making module is the central layer of fault repair architecture. Based on the report received from the evaluation module and the policies from management policies module, the scenario is analyzed and an appropriate action, for replacement module, is decided.

3) *Replacement module*: As the name suggests, the replacement module is responsible for sensor relocation that is decided in the decision making module. We define four main actions for the replacement module: *ignore*, *re-assignment*, *re-location*, and *replacement*. The first alternate action is ignore.

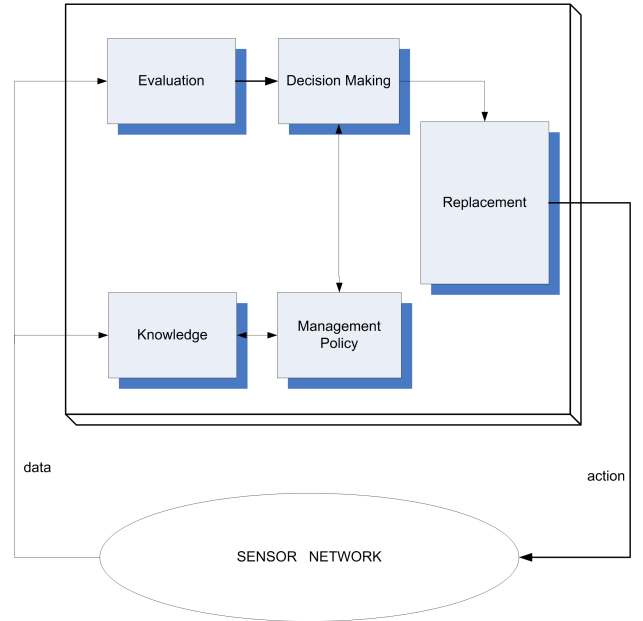


Fig. 3. Fault Repair Architecture

Obviously, if the coverage is still satisfied in the presence of a few faulty nodes, they can be ignored. The second action is re-assignment that is responsible for requesting a static sensor to substitute a faulty node's role. Some of sensors may be either in the idle state or working insignificantly, e.g., on low duty cycle. In this case, these sensors may be re-assigned new tasks so that they can substitute the faulty sensors. Thirdly, relocation action is responsible for moving a set of sensors to extend the existing coverage. However, as the cost of transportation itself reduces energy in the system, a movement schedule should be carefully planned such that the energy consumption is minimized and the total energy remaining is maximized. Finally, replacement is to replace a faulty node by a redundant sensor. Obviously, if a faulty sensor is one of the constituent sensors in the network, it should be replaced promptly. Therefore, based on the estimation of impact, an appropriate action will be determined and performed.

Replacing the faulty sensor by a redundant sensor is intuitively simple. However, the replacement should be energy efficient. Each faulty sensor can have different preferences for its replacement, e.g. response time, remaining energy etc. The best suitable redundant sensor is thus the one that satisfies its criterion. The cost of the movement depends on the distance between the faulty sensor and the replacement sensor and energy consumed per unit of distance. By intuition, the redundant sensor located closest to the faulty node is preferred. However, this is not always the best case. For example, it is obvious that a faulty sensor does not want to be replaced by a low energy sensor no matter how close and free it is. Instead, a faulty sensor would rather be replaced by a long distance sensor with adequate remaining energy.

4) *Management Policy Module*: The management policy module contains a set of rules that the decision making module and other modules should comply with them in order

to achieve common objectives, e.g., energy saving, coverage resolution, etc.

Policy is a set of rules or set of actions governing decisions that will be implemented to achieve the objectives [3]. Policy is not only defined by administrator, but also obtained and updated from the knowledge module, in which the learning on the network behavior is performed.

Policy contains four components: *event*, *condition*, *action*, and *scope*. Whenever an event occurs, the policy condition is evaluated. If the specified condition is true, the corresponding action is executed. The scope of a policy indicates its targets, i.e., at which nodes it should be enforced. Policy will support the decision of decision making module for selecting an appropriate action performed when a fault occurs. There are many factors such as the probability of faulty sensors, the battery status, the coverage problem, and the communication situations that should be in considered. For example, the policy may define the important level of a region. If faults occur in an important region, the repair algorithm may replace them as soon as possible. Otherwise, faults may be ignored. Using management policy will enhance the network performance by reducing energy consumption on unnecessary movements.

Since there are limited resources in sensor network, the management policy should be light-weight. In our architecture, three main classes of management policies are defined: coverage policies, resource policies, and performance policies. Each policy contains set of rules that support its objective. Coverage policies contain set of rules related to the degree of coverage over a region in the network. These rules may allow every location in a region be monitored by one node, for example. It also allows some regions to be maintained at a low degree of coverage. Resource policies are related to energy consumption issues. Energy is a paramount concern in sensor network that needs to operate for a long time on battery power. To reduce energy consumption, resource rules may be defined to allow a certain number of nodes to be inactive, while the remaining active nodes still provide continuous services. Performance policies are the policies about delay, priority, or resolution of a region. Again, a fundamental problem is to define the number of nodes that remain active, while still achieving acceptable a degree of coverage for applications.

There are several policy languages for network. For example, we can use Policy Framework Definition Language (PFDL) [4] to express various kinds of network policies. The PFDL can simply express lists of *IF* <condition> *THEN* <action> type of rules. List of the rules will form a policy. <condition> above is in fact a disjunctive normal form of single condition expressions, and <action> is a list of single action statements. If the evaluation of the condition expression request succeeds, the action list can be performed.

5) *Knowledge module*: Knowledge module represents intelligence of the network. It is motivated by the consequences of the environmental changes and resource limitations. To obtain the knowledge about faults, monitored sensors need to participate in accumulating knowledge, learn what occurs in

the surroundings and predict what will happen next.

Fault repair is used not only for current actions, but also for improving the ability to perform optimally in future to achieve the objective. Prediction of faulty sensor will save energy consumption on movement. For example, a redundant sensor may reject the request for replacement of a long distance faulty sensor, if it knows in advance that the nearby sensor may soon become faulty.

Reinforcement and supervised learning are the two main classes of learning that can be applied to this environment. A redundant sensor node would attempt to decide appropriate movement based on its current fault location information. In reinforcement learning, the machine can produce actions which affect the state of the world, and receive rewards (or punishment). Its goal is to maximize the rewards (or minimize punishment) in the long term [5]. Reinforcement learning determines actions to take as well as the possible outcomes. From the actions and outcomes, it tries to learn how to behave successfully to achieve a goal while interacting with an external environment. In other words, reinforcement learning learns via experiences. Reinforcement learning may be suitable in the case of soft fault sensor, in which the sensor node is still active but gives inaccurate-reading. However, it may require lot of message exchanges resulting in a high overhead. As we only consider permanent fault in our model, we prefer a simpler model called supervised model.

Supervised learning consists of one set of observations, called inputs, and another set of observations, called outputs. Supervised learning tries to determine the function that maps any input to an output such that disagreement with future input-output pairs is minimized [4]. In our case, supervised learner uses the faulty location information and target values as training data. After learners are trained, they would be able to make decisions based on system sensor readings. For instance, with such a model, sensor nodes could be able to compute the fault distribution based on faulty sensor information i.e. if the result of sensor readings is unusually different from the result of its neighbors, monitor sensors can predict that this sensor may be in faulty state. Further testing can be implemented to clarify the hypothesis.

However, supervised learning is not flexible, as the system will encounter the state that has not been previously observed. In this case, additional rules may need to be included to adjust the action to be adaptive to changes of environment.

### III. REPLACEMENT MODULE–INITIAL SOLUTION

We presented a preliminary framework for mobility based fault repair architecture in Section II. This work is in early stage. Here we provide our initial results where we developed an offline algorithm for the replacement module of the architecture. Our ambition is to develop a distributed decentralized algorithm for the replacement module. We also provide examples of basic policies for the management module that can help improve the fault repair system performance.

### A. Problem statement

Given a collection of sensors and a monitor sensor  $LMS$ , together with their locations and the energy of each sensor, find a replacement schedule with maximum energy remaining.

We assume the network topology as discussed in section II-A. All the data in the cluster is reported to its  $LMS$ . When the energy of a sensor is lower than a threshold bound,  $TE_{low}$ , it will alert its status to the  $LMS$ . Periodically the  $LMS$  checks the health of the sensors in its cluster. If there are some dying sensors, the  $LMS$  needs to make replacement schedule. Thus, at the  $LMS$ , the problem can be formulated as following:

There are  $a$  redundant sensors  $S_r$ , ( $a \leq n$ ):

$$S_r = \{S_{r1}, S_{r2}, \dots, S_{ra}\}; \quad (4)$$

There are  $b$  faulty sensors  $S_f$ , ( $b \leq a \leq n$ ):

$$S_f = \{S_{f1}, S_{f2}, \dots, S_{fb}\}; \quad (5)$$

Objective function:

"What is a replacement schedule in order to minimize energy consumption?"

This problem is similar to the bipartite matching problem that can be represented as a bipartite graph where one side is redundant sensors and the other is faulty sensors. Hence, the problem of fault replacement is transformed into the problem of finding perfect matching in a bipartite graph. The bipartite graph consists of two set of nodes  $S_r$  and  $S_f$ , representing redundant sensors and faulty sensor respectively. There is an edge from  $S_{rj}$  to  $S_{fi}$  and the weight of the edge is the actual energy remaining after movement.

### B. Problem formulation

Let's call matrix  $ER_{b \times a}$  energy remain matrix, for it specifies the remaining energy for the network.

$$ER_{b \times a} = \begin{pmatrix} er_{11} & er_{12} & \dots & er_{1a} \\ er_{21} & er_{22} & \dots & er_{2a} \\ \vdots & \vdots & \vdots & \vdots \\ er_{b1} & er_{b2} & \dots & er_{ba} \end{pmatrix}$$

Where,  $er_{ij}$  is the total energy remaining at redundant sensor  $j$  if faulty sensor  $i$  is replaced by sensor  $j$ . Thus,

$$er_{ij} = E_j - E_{move}; \quad (6)$$

where  $E_j$  is the initial energy of sensor  $j$  and  $E_{move}$  is the energy consumed for movement.

$$X = \{x_{ij}\}; i = 1, \dots, b; j = 1, \dots, a \quad (7)$$

where  $x_{ij} = 1$  if a faulty sensor  $i$  is replaced by redundant sensor  $j$ ,  $x_{ij} = 0$ , otherwise.

The replacement problem can be formulated as Integer Linear Program as following:

Objective: Maximize  $TR = \sum \sum er_{ij} x_{ij}$

Constraints:

- 1)  $x_{ij} \leq 1; i = 1, \dots, b; j = 1, \dots, a$
- 2)  $\sum x_{ij} \leq 1; j = 1, \dots, a$
- 3)  $\sum x_{ij} \leq 1; i = 1, \dots, b$

An optimal replacement can be found using an integer program with linear constraints. The integer program computes the TR subject to constraint (1) and additional linear constraints (2) and (3). Constraint (2) ensures that any redundant sensor  $j$  can only replace one faulty sensor. Similarly, constraint (3) is used to guarantee that any faulty sensor  $i$  can be replaced by only one sensor.

## IV. NUMERICAL RESULTS

### A. An example

In Figure 4, we randomly place 50 working sensors (filled circle) and 20 redundant sensors (hollow circle) in an area 100x100 meters. Among 50 working sensors, 10 faulty sensors (cross-marked) are selected arbitrarily. Redundant sensors are assigned different energy remaining and all faulty sensors are assumed to require a constant response time.

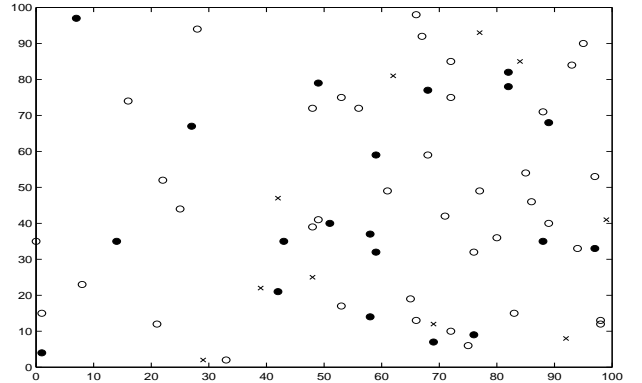


Fig. 4. Initial Network Topology

TABLE I  
REDUNDANT SENSORS WITH DIFFERENT REMAINING ENERGY

Redundant Node	Energy	Redundant Node	Energy
1	55	11	77
2	32	12	79
3	95	13	50
4	64	14	59
5	31	15	73
6	106	16	114
7	95	17	29
8	56	18	105
9	77	19	46
10	91	20	101

Firstly, the energy remaining matrix (redundant sensor  $j$  replaces faulty sensor  $i$ ) is calculated (Table II). Then by using ILP, we calculate the maximum total remain energy that comes out to be 1228 units of Energy.

TABLE II  
ENERGY REMAINING MARTIX

	$j_1$	$j_2$	$j_3$	$j_4$	$j_5$	$j_6$	$j_7$	$j_8$	$j_9$	$j_{10}$	$j_{11}$	$j_{12}$	$j_{13}$	$j_{14}$	$j_{15}$	$j_{16}$	$j_{17}$	$j_{18}$	$j_{19}$	$j_{20}$
$i_1$	0	5	51.9	20	0	42	84.7	11.3	63.3	66.2	40.9	26.6	0	33.3	25	99.2	0	44.8	0	82.6
$i_2$	13.4	0	14.8	0	0	68	40.9	50.6	35.8	37.2	3.7	33.1	0	19.1	0	75.5	0	37.9	3.8	37.9
$i_3$	15.3	0	18.9	0	0	60.3	0	0	4.7	0	0	42.8	3.4	0	2.7	19.9	0	76.3	7.8	12.5
$i_4$	35.3	0	24.1	0	0	80	14.3	0	19.6	0	0	60.4	6.9	0	6.3	36.8	0	79.8	27.9	24.1
$i_5$	1.5	0	71.7	35.3	0	47.9	62	0	58.6	39.7	19.3	40.4	25.4	7	48.2	73.2	13.4	67.9	0	77.5
$i_6$	11.9	0	50.2	11.1	0	56	25.9	0	28.1	4.8	0	50.6	34.7	0	34	41.3	6.8	99.1	6.7	41
$i_7$	41.9	0	26.5	0	0	92.5	34.1	20.9	38.1	21.6	0	61.5	0.3	0	5.6	61.6	0	64.7	33.6	38.4
$i_8$	36.9	0	19	0	0	91.6	31.4	26.8	33.7	21.3	0	52.3	0	1	0	61.3	0	55.5	26.7	33.6
$i_9$	18.8	0	57	18.8	0	64.2	46.1	0	49.7	26.1	2.1	58.9	31.5	0	37.3	63.1	22.9	8.5	13.9	58.4
$i_{10}$	0	0	58.7	31.8	0	18.4	76.9	0	41.5	58.9	51	6.7	0	15.9	29.4	78.1	0	32	0	88.1

The replacement schedule is as follows:

Assignment = [ 7 8 12 1 3 18 6 16 9 20 ]

This assignment matrix means the faulty sensor 1 will be replaced by redundant sensor 7; faulty sensor 2 is replaced redundant sensor 8, and so on. The following Figure describes the movement schedule:

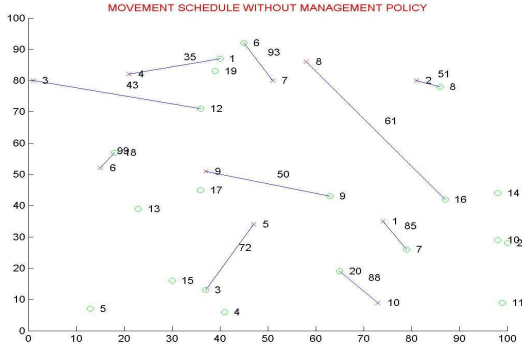


Fig. 5. Movement Schedule Without Management Policy

### B. With simple management policy

We provide a simple example of a resource rule and a performance rule that can be defined in the management policy module of the fault repair architecture:

1) *Resource rule*: It is obvious that a redundant sensor should only be used for replacement if it still has a minimum desired level of energy after the movement. This desired level of energy,  $Th_{accept}$ , guarantees that the sensor continues working at the new location after the movement. Therefore, a simple resource rule for management policy follows:

*A redundant sensor can only be chosen for replacement, if the remaining energy of redundant sensor after movement is greater than  $Th_{accept}$ .*

$$(E_i - E_{move}) \geq Th_{accept} \quad (8)$$

2) *Performance rule*: Each location usually has different working load. While there are major sections in which the sensing tasks occur continuously, in some other section sensors

may work occasionally. If a sensor fault occurs in an important section, it should be repaired soon. Therefore, each faulty sensor has different time delay preference for replacement. The management policy for this problem may be as follow: *If the redundant sensor response time is greater than the response time acceptable for a faulty sensor, then do not move that redundant sensor.*

$$\frac{d_{ij}}{v_j} \leq T_i \quad (9)$$

Where  $d_{ij}$  is the distance between redundant sensor  $j$  and faulty sensor  $i$ ,  $v_j$  is velocity of redundant sensor  $j$  and  $T_i$  is the response time required for faulty sensor  $i$ .

For our given network topology, we applied these management policy rules and re-calculated the movement schedule and the cost of movement (see Figure 6).

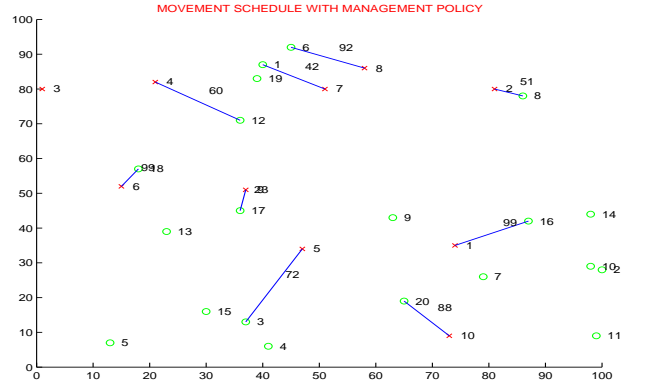


Fig. 6. Movement Schedule With Simple Management Policy

The amount of total energy remaining in the network with simple management policy (1321 unit) is better than no management policy defined (1228 unit of energy). This is because the resource rule of the management policy helps the replacement plan to avoid any long distance movements. For example, without the management policy, a monitor sensor will ask the redundant sensor 16 to replace the faulty sensor 8, and the energy remaining in the redundant sensor 16 after movement is 61 (Figure 5). However, it is not an optimal schedule, since sensor 16 travels a long distance and drains its energy considerably. With the inclusion of the management

policy, instead of replacing the faulty sensor 8 by the redundant sensor 16, it asks the redundant sensor 6 to replace 8 (Figure 6), and the remaining energy of the node 6 after the movement is fairly high (92 unit). Also, the redundant sensor 16 is scheduled to replace the faulty sensor 5, resulting in 99 units of energy after the movement. As a result, the total energy remaining improves. Therefore, the use of management policy enhances the network performance.

### C. Replacement algorithm performance

We want to investigate the performance of replacement algorithm.

1) *Greedy Heuristics*: To measure the performance of the replacement algorithm we implemented our algorithm and compared it to the greedy algorithm (heuristics algorithm). The idea of the greedy algorithm is that each faulty sensor will select one of redundant sensors that have the minimum cost for movement (e.g. physically closest.) In Figure 7, there are 90 working sensors, 10 faulty sensors which are randomly selected and the number of redundant sensor is increased from 10 to 30 nodes. In Figure 8, there are 100 working sensors, from which we selected the number of faulty sensors (from 1 to 20). We kept the number of redundant sensor constant at 30.

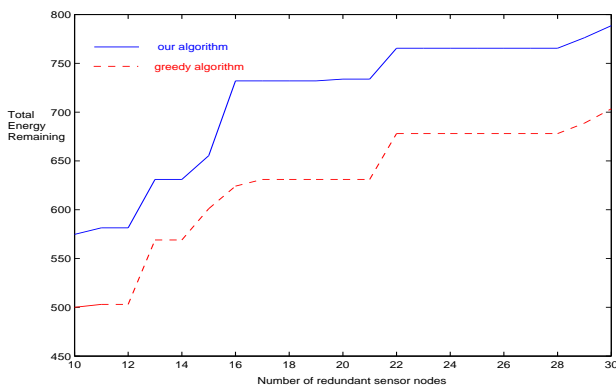


Fig. 7. Increasing the No of Redundant Sensors

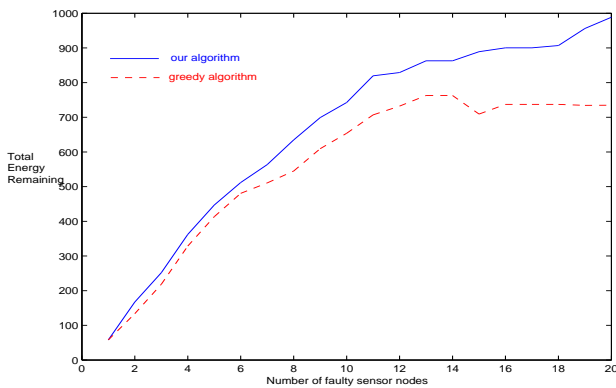


Fig. 8. Increasing the No of Faulty Sensors

Figure 7 shows that the greedy algorithm comes close to ILP results for a range of different number of redundant nodes as well as when the number of faulty nodes increase.

## V. RELATED WORK

Fault repair concept has been used widely for almost half a century in most of areas. In computer systems, proactive fault repair also has been attended. For example, Moore and Shannon [6] and von Neumann [7] use the redundancy to enhance reliability for the networks which are built from unreliable components. More recently, fault tolerance in Internet such as network availability and performance has been discussed in [8]. Recently, wireless sensor community has greatly focused on related research topic called fault tolerance. A reliable routing protocol in sensor network with an arbitrary network topology has been discussed in [9]. In [10], a distributed routing algorithm is proposed, which is able to deal with faults or holes presented in a sensor network. Moreover, due to harsh environmental conditions, majority of measurements in sensor networks are usually subject to errors. Techniques for measuring and adjusting uncertain values are presented in [11] and [12]. They guarantees reliable and accurate output when a large number of sensor measurement faults occurs. In [13] [14] and [15], several localized threshold based decision schemes to detect faulty sensor are proposed. The most recent readings of sensors are stored and statistically analyzed. Faults are detected based on any abnormal readings which are beyond an application-specific threshold. By exploring the correlation among neighboring sensor readings, faulty readings are separated from event readings. The intuition behind the approaches is that event readings are likely to be spatially correlated. The confidence are computed statistically based on the decision predicates from neighboring sensors. A different approach to detect failed nodes through route discovery and update is presented in [16]. A watchdog mechanism is used to identify misbehaving nodes and a path navigator is used for supporting routing protocols to avoid them. In general sensor readings are collected at a base station. An algorithm that is able to trace faulty nodes once these reading are received at the base station is proposed in [17]. However, there has been a limited research on fault repair. While there are several works on energy replacement by using mobile robots to recharge sensor nodes [18] [19], these new technologies have not been implemented yet. The author in [20] proposes a framework for replacing faulty sensor nodes by relocating mobile sensors. The framework consists of two phases, a Grid Quorum solution that locates the closest redundant sensor and the calculation of an efficient route for the relocation of mobile sensors. Cascaded movement is used to achieve good balance between energy efficiency and response time when determining a sensor relocation path. In [21], an algorithm called Coverage Fidelity maintenance algorithm (Co-Fi) uses mobility of sensor nodes for automated deployment and for repairing of coverage loss in the monitoring area. One of the limitations of these algorithms is that they are not able to

replace multiple faulty sensor nodes at a time. Hence, it is obviously not suitable for a long term maintenance, where a number of faulty sensors can be significantly large. To the best of our knowledge, there is no existing fault repair architecture in sensor networks in the presence of numerous faulty sensor nodes. Our proposed architecture provides robustness, adaptivity, and scalability.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a fault repair architecture. We introduce policy based management in conjunction with learning techniques. As a starting point we provide some examples of the management policies and an offline algorithm for the replacement module of our architecture using Integer Linear Programming formulation. We compare our ILP results with greedy heuristics. Our results suggest that the greedy algorithm performs very close to the optimal ILP results in terms of energy remaining. In its current stage our algorithm is intended to be implemented at the Base Station and cluster heads. We are also working on the distributed version of it at the moment. Moreover, in this work we presented the empirical evaluation of the replacement action in the replacement module of architecture. We are currently extending our work to include numerical and experimental evaluation of the entire system. Our future work will develop all other modules of the framework and provide experimental evaluation of a prototype system.

## REFERENCES

- [1] F.Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli, "Fault tolerance in wireless ad hoc sensor networks," *IEEE Sensors*, vol. 2, pp. 1491–1496, June 2002.
- [2] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *6th International Symposium on Distributed Autonomous Robotics Systems (DARS02)*, June 2002.
- [3] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," *AT&T Research*, 1996.
- [4] P. F. D. Language, "draft-ietf-policy-framework-pfdl-00.txt, <http://www.ietf.org/proceedings/98dec/i-d/draft-ietf-policy-framework-pfdl-00.txt>."
- [5] T. Runarsson and S. Sigurdsson, "The learning methodology, <http://cerium.raunvis.hi.is/tpr/courseware/svm/notes/chapter1.pdf>."
- [6] E. Moore and C. Shannon, "Reliable circuits using less reliable relays," *Franklin Institute*, vol. 262, pp. 191–208, 1956.
- [7] J. Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," *Automata Studies*, pp. 43–98, 1956.
- [8] D. Medhi, "Network reliability and fault tolerance," in *Wiley Encyclopedia of Electrical & Electronics Engineering*, University of Missouri, 1999.
- [9] S. Iyengar, M. Sharma, and R. Kashyap, "Information routing and reliability issues in distributed sensor network," *IEEE Transaction Signal Processing*, vol. 40, pp. 3012–3021, 1992.
- [10] Q. Fang, J. Gao, and L. J. Guibas, "Locating and bypassing routing holes in sensor networks," in *IEEE INFOCOM 2004*, June 2004.
- [11] V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak, "A collaborative approach to in-place sensor calibration," in *2nd International Workshop on Information Processing in Sensor Networks IPSN '03*, University of California, 2003.
- [12] K. Whitehouse and D. Culler, "Calibration as parameter estimation in sensor networks," in *ACM Workshop on Wireless Sensor Networks and Applications WSNA '02*, September 2002.
- [13] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized fault-tolerant event boundary detection in sensor networks," in *Infocom 05*, March 2005.
- [14] B. Krishnamachari and S. Iyengar, "Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Transactions on Computers*, vol. 53, pp. 241–250, March 2004.
- [15] M. Alanyali, S. Venkatesh, O. Savas, and S. Aeron, "Distributed bayesian hypothesis testing in sensor networks," in *American Control Conference*, 2004.
- [16] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Mobicom*, 2000.
- [17] J. Staddon, D. Balfanz, and G. Durfee, "Efficient tracing of failed nodes in sensor networks," in *ACM Workshop on Wireless Sensor Networks and Applications WSNA '02*, September 2002.
- [18] M. Rahimi, H. Shah, G. Sukhatme, J. Heidemann, and D. Estrin, "Studying the feasibility of energy harvesting in a mobile sensor network," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2003.
- [19] A. LaMarca, D. Koizumi, M. Lease, S. Sigurdsson, G. Borriello, W. Brunette, K. Sikorski, and D. Fox, "Plantcare: An investigation in practical ubiquitous systems," *Intel Research*, vol. IRS-TR-02-007, 2002.
- [20] G. Wang, G. Cao, T. Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *Infocom 05*, March 2005.
- [21] S. Ganeriwal, A. Kansal, and M. B. Srivastava, "Self aware actuation for fault repair in sensor networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2004.