A Localized Self-healing Algorithm for Networks of Moveable Sensor Nodes

Mohamed Younis, Sookyoung Lee, Sheetal Gupta and Kevin Fisher

Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
(younis, slee22, sheetal4, kfisher1@umbc.edu)

Abstract— **The effectiveness of wireless sensor networks (WSNs) deployed in search and rescue, battlefield reconnaissance, surveillance, and other applications depends on inter-node interaction and maintaining network connectivity. While connectivity can be provisioned at startup, then sustained through careful coordination when nodes move, the network can be partitioned if a node suddenly fails. This paper presents Recovery through Inward Motion (RIM), a distributed algorithm to efficiently restore network connectivity after a node failure. Instead of performing a network-wide analysis to assess the impact of the node failure and set a course of action, RIM triggers a local recovery process by relocating the neighbors of the lost node. RIM minimizes messaging overhead and reduces the distance that individual nodes travel during the recovery. Simulations validate RIM's performance.**

I. Introduction

Growing interest in wireless sensor network (WSN) applications has motivated lots of research work during recent years [1]. These applications, which include space exploration, coastal and border protection, combat field reconnaissance, and search and rescue, use a set of mobile sensor nodes to collaboratively monitor an area of interest and track certain events or phenomena. The sensors can operate unattended in harsh environments, avoiding risk to human life and decreasing the cost of the application. Sensors in these applications are typically battery operated and have limited processing and communication capabilities. Upon deployment, they form a network in order to share data and coordinate their actions. For example, in a disaster management task, the nodes collaborate to search for survivors, assess damages, and find safe escape paths. To enable such interaction, nodes need to stay reachable to each other.

Given the importance of inter-node connectivity, sensors usually inform their neighbors before moving so the network topology can be adjusted accordingly [2]. However, sudden node failures, caused by the harsh environment or depletion of the node's battery, can disrupt network operations. The loss of a node can break communication paths in the network and make some of its neighbors unreachable. In the worst case, the network may get partitioned into multiple disjoint blocks and become dysfunctional. Therefore, sensors should be able to detect and recover from the failure of one of their peers. Since WSNs operate autonomously and unattended, and their nodes are resource-constrained, the recovery should be a distributed, self-healing process. The network should remain responsive to detected events, so the recovery process should also be lightweight and work quickly, with minimal overhead. However, because some nodes may be unable to reach others, a well-orchestrated, non-centralized recovery procedure becomes very difficult.

This paper presents RIM: a distributed algorithm for Recovery through Inward Motion. RIM restores the connectivity of a WSN through the efficient repositioning of some of its nodes. RIM is a localized scheme that limits the scope of the recovery process. The main idea is that when a node fails, its neighbors move inward toward its position so they can connect with each other. The rationale is that these neighbors are the ones directly impacted by the failure, and when they can reach each other again, the network connectivity is restored to its pre-failure status. The relocation procedure is recursively applied to handle any nodes that get disconnected when one of their neighbors moves. RIM is simple and effective. It employs a simple procedure that recovers from both serious and non-serious breaks in connectivity, without checking to see if the failed node is a cut vertex. The entire recovery process is distributed, enabling the network to heal itself without external supervision.

This paper is organized as follows. The next section describes the considered system model and the implications of node's failure on network connectivity. Related work is covered in Section III. Section IV describes the proposed RIM algorithm in depth. Validation experiments are discussed in Section V. Section VI concludes the paper.

II. System Model and Problem Statement

RIM applies to networks that include sets of mobile sensor nodes, either in a flat network topology or the second tier of a hierarchical network architecture. When sensors are randomly deployed in an area of interest, they discover each other and form a connected network using techniques such as [3]. Nodes can also move on demand, in order to fill a hole in coverage or enhance the inter-node connectivity. RIM only assumes that nodes know their position relative to their neighbors [4], not necessarily their exact coordinates.

While node failures may affect both network coverage and connectivity, this paper focuses on maintaining the latter when a node is lost. The impact to network connectivity depends on the role of the lost node. The loss of a leaf node (such as (3) in the network topology depicted in Figure 1) does not impact inter-node reachability. The same applies to some non-leaf nodes, like (12) and (14), since alternate paths are available between the neighbors of the failed node. However, the failure of a cut-vertex like (9) partitions the network into two sub-networks and an orphaned node, (11). Though highly desirable, strategies that adapt the recovery process to the role of the failed node are often impractical in wireless sensor networks. First, centralized approaches for determining whether a failed node is a cut-vertex do not scale. In addition, their messaging and state maintenance functions impose a significant overhead of O(N+M) messages, for networks with N nodes and M links [5]. Instead, localized approaches are more practical in that context, especially in very dynamic environments where nodes move frequently and/or have a short lifespan. RIM exploits such a trade-off to effectively recover from a node failure.

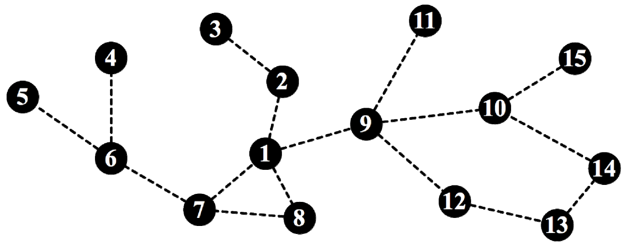

Figure 1: A connected network of mobile nodes.

III. Related Work

Node relocation is a well-studied problem [6]. Some approaches allow movement only between initial deployment and application startup (post-deployment), while others can arrange movement at any time (on-demand).

A. *Post-Deployment Repositioning*

Wang, Cao, and La Porta [7] proposed three related algorithms for post-deployment repositioning, called VEC, VOR, and Minimax. Each considers a sensor node's Voronoi polygon, the part of the sensing area to which it is the closest node. VEC emulates Coulomb's law, the equation that describes how electrostatic particles repel each other. If part of a node's Voronoi polygon is not covered, the node will be repelled from its neighbors by a force proportional to its distance from them and from the vertices of the polygon. A similar approach by Neo and Varshney [8] also mirrors Coulomb's law but does not consider the node's Voronoi polygon. Instead, each node moves a distance proportional to the density of nodes in its immediate area. VOR simply directs the node to move toward the most distant vertex of its Voronoi polygon. This movement serves to make the polygons more regular. However, VOR is prone to cause nodes to oscillate between multiple locations. The Minimax algorithm makes for shorter, more conservative moves and less oscillation. All three algorithms result in zig-zag motions, which waste energy and time. Unlike RIM, all these approaches focus on avoiding holes in coverage rather than maintaining connectivity.

B. *On-Demand Repositioning*

The above algorithms aim to place nodes in the most efficient formation. Once the network application has started, however, node failures may reduce overall efficiency, and changes in application requirements may change the definition of efficiency itself. In either case, nodes should move to maintain an effective network layout. Wang et al. [9] have proposed Cascaded Movement to replace a failed sensor by iteratively replacing a nearby node with a redundant node. While our work is similar in the sense that we use cascaded movements, connectivity is not considered in [9].

Some work has also considered connectivity. The approach described in [10] opts to sustain degree-two connectivity even under link or node failure based on moving a subset of the nodes. While the idea of movement of nodes is similar to ours, stressing the need for 2-connectivity may constrain the application-level functionality and may not be practical in large-scale networks of resource-constrained nodes. The approach most related to RIM that we have found in the literature is DARA [11]. Unlike RIM, DARA requires every node to maintain a list of their 2-hop neighbors and picks a neighbor of the failed node to relocate based on the number of communication links. RIM only requires the knowledge of 1-hop neighbors. In addition, DARA detects cut-vertices, something RIM avoids doing to simplify the recovery process.

IV. Restoring Connectivity Trough Inward Motion

This section describes in detail the RIM algorithm for restoring connectivity in WSNs that are severed due to node failure. Our approach aims to efficiently move nodes in order to restore the network connectivity to its pre-failure status.

A. *Detailed RIM Approach*

The proposed RIM approach exploits the node's mobility and requires only 1-hop neighbor information to recover from a node failure. The restoration process is both localized in scope and distributed in execution, requiring no coordination among nodes. Each node can independently decide when to start the restoration algorithm, and where to move. RIM only requires that each node know the locations of its 1-hop neighbors. The following describes the major steps.

Maintaining a One-hop Neighbor Table: At network setup, each node broadcasts a *HELLO* message to introduce itself to its neighbors, then builds a list of directly reachable nodes, i.e., 1-hop neighbors. The 1-hop neighbors table is maintained during network operation to reflect changes in the topology. Each table entry contains two parameters: {Node_ID, Relative position}. Nodes inform their neighbors before changing their position so they will not be wrongfully perceived as faulty.

Detecting a Failure and Initiating the Recovery Process: Nodes will periodically send heartbeat messages to their neighbors to ensure that they are functional. Missing heartbeat messages can be used to detect a failed node, hereafter referred to as S_f. RIM does not analyze the effect of S_f's failure on the network; it applies the same recovery process to cut-vertices and ordinary nodes.

The recovery starts with the 1-hop neighbors of S_f moving towards the position of S_f until they can reach each other and form a connected sub-network. To achieve that, they should move until they are a distance $r/2$ from the position of S_f, where r is the communication range of a sensor node. This connects all neighbors of S_f. (Proof omitted to save space.)

Cascaded Node Relocation: As explained above, the neighbors of S_f, known as Neighbors(S_f), initiate the RIM restoration process with their individual uncoordinated moves toward S_f. Every node $S_a \in$ Neighbors(S_f) sends a message to all of its 1-hop neighbors that includes a notification of the move, the new location of S_a, and the rank of S_a relative to S_f, which is equal to one for all Neighbors(S_f). Automatically, the neighbors of S_a have rank 2 unless they are in Neighbors(S_f). Such ranking enables the recovery process to successfully converge as explained below. The neighbors of S_a assess whether or not their link to S_a will be broken. If a node $S_{aa} \in$ Neighbors(S_a) loses connectivity to S_a, it will move toward the new position of S_a until it is within its communication range. We call this process *cascaded motion*. In Figure 2(a), nodes A and B move toward a failed node. Node A has no neighbors, but node B has a neighbor C, which is motivated to follow B in order to stay connected.

If S_{aa} is connected to more than one node that moves in response to the failure of S_f, which node will S_{aa} follow? Figure 2(b) and 2(c) illustrate the cases with two and three nodes, respectively. In Figure 2(b), two neighbors of node *D* (*B* and *C*) have moved. RIM relocates *D* to the closest point that lies within the communication ranges of *B* and *C*. This point is the closest intersection point of the two circles of radius *r* centered at *B* and *C*. Figure 2(c) shows where RIM places node *D* when three of its neighbors, namely *A*, *B* and *C*, move. RIM calculates the intersection points of every pair of circles around *A*, *B* and *C*. Node *D* is moved to the closest intersection point that also lies within the third (non-intersecting) circle. This idea also applies for more than three nodes since there must be an intersection point of two circles which lies within the communication ranges of all the moved nodes. (This proof is omitted due to space constraints.)

Cascaded motion can be applied recursively. For example, if S_{aa} moves and its neighbor S_{aaa} can no longer reach it, S_{aaa} will repeat the same steps that S_{aa} applied to repair its link with S_a. An interesting question is how a node differentiates between neighbors that move to tolerate the failure and those involved in the cascaded motion. For example, assume S_a and S_b are neighbors of S_f and move to tolerate the failure. S_{aa} and S_{bb} are neighbors of S_a and S_b, respectively, and are directly connected. Although S_{aa} and S_{bb} can perform a cascaded motion to follow S_a and S_b respectively, they will not know what to do about their link. If S_{aa} moves first, S_{bb} will have to follow it, and vice versa. This may prevent RIM from terminating, since nodes will keep performing a cascaded motion back and forth. The node rank, relative to S_f, solves this problem. Nodes will give priority to the neighbors with the lowest rank. Although it appears that RIM may break some links among peer nodes, i.e. those with the same rank, we have proven that the link between peer nodes will be maintained without any coordination. This proof is also omitted due to space, however in short, the cascaded motion always yields inward motion toward the faulty node and ensures network connectivity through higher-ranked nodes like *Neighbors(S_f)*.

B. *Pseudocode and Illustrative Example*

Figure 3 shows the pseudocode for RIM. Lines 1-14 detail the main procedure. Basically, a node *J* will check whether its neighbor *F* has failed. If so, *J* moves toward *F* after notifying its neighbors. The function *Notify_Movement(J)* informs all neighbors about *J*'s move, rank, and new position. If *J* instead needs to follow another node (line 4), it identifies a new position and notifies its neighbors before moving (lines 8-12). Nodes only move once (line 5-7), then set a flag showing their involvement in the recovery is complete. The procedure *Compute_newPosition(J)* identifies where node *J* must move to based on the notifications it received from other nodes.

Figure 4 explains how the RIM restoration process works using an example WSN. In Figure 4(a), *A*, *B*, *G* and *H* detect the failure of node *F*. First, each node notifies the neighbors who will lose their connection to it when it moves. G and H have no other neighbors, while *A* and *B* notify *{C}* and *{C, D}*, respectively. Then, nodes *A*, *B*, *G* and *H* move directly toward *F* and become connected, as seen in Figure 4(b). Since *C* receives two notification messages from *A* and *B*, it identifies a new position at the intersection of circles centered at *A* and *B*. Node *C* notifies node *D*, which will be disconnected when *C* departs, then moves as depicted in Figure 4(c). Node *D* also performs cascaded motion, but it determines its new location based only on *B*'s position. It ignores *C*'s message since its

```
1   IF a sensor node J detects a failure of its neighbor F
2       Notify_Movement(J);
3       J moves toward F until it becomes r/2 away from F
4   ELSE IF J receives (a) notification message(s) from F
5       IF I_Already_Reconnected
6           Done;
7       END IF
8       NewPosition ← Compute_newPosition(J );
9       IF NewPosition  != CurrentPosition
10          Notify_Movement(J);
11          J moves to NewPosition
12      END IF
13  END IF
14  I_Already_Reconnected ← TRUE;

Compute_newPosition(J) {
15  NUM_PriorNBR ← Number of notification messages that J
    have received with the lowest rank;
16  IF J stays connected with all PriorNBR with least rank
17      Return CurrentPosition
18  END IF
19  Location_Senders[] ← New locations of one or more
    senders from which J have received notification messages;
20  IF NUM_PriorNBR == 1
21      Return a point r units away from Location_Sender[0] on
        the direct path to Location_Sender[0];
22  ELSE
23      Define a circle of radius r around each of
        Location_Sender[];
24      IF NUM_PriorNBR == 2
25          Return the closest point between two intersection
            points;
26      ELSE IF NUM_PriorNBR ≥ 3
27          Return the closest point among intersection points
            which is located inside all other circles;
28      END IF
29  END IF }

Notify_Movement(J) {
30  Send notification message with rank value increased by 1 to
    notify all neighbors of J about the motion and new position;}
```

Figure 3. Pseudocode for the RIM algorithm.

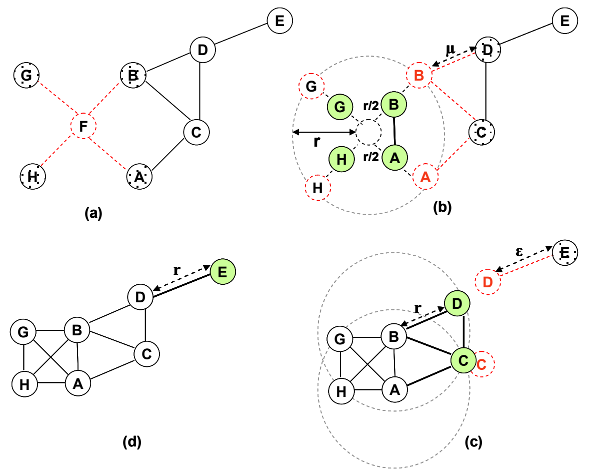

Figure 4: An example for how RIM works

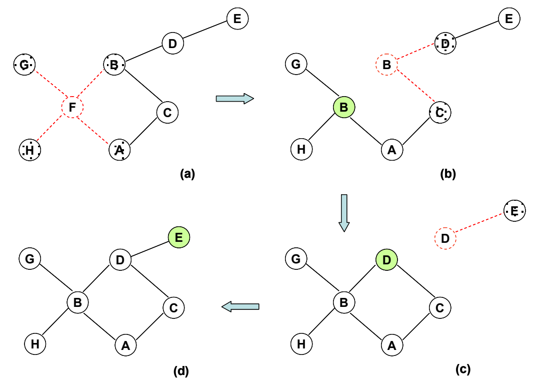

Figure 5: An example of how the NN algorithm works.

rank is 2, whereas the rank of *B* is 1. Nonetheless, when *D* arrives at its new position, it reconnects with *C* (Figure 4(c)). It is worth noting that the distance between *D* and *B* is now *r*, rather than μ, with $r \geq \mu$. Node *D* will notify *E* prior to moving. Finally, *E* travels and settles *r* units away from *D*, where $r \geq \varepsilon$. The restoration process ends when E settles; note that each node moves only once.

V. EXPERIMENTAL EVALUATION

This section describes the simulation environment, performance metrics, and experimental results that validate the effectiveness of the RIM approach.

A. *Experiment Setup and Performance Metrics*

In the experiments, 300 mobile sensor nodes are initially placed at random in a 1000m × 1000m area to form a WSN. We define the following RIM performance metrics:

- The total *distance moved* by all nodes involved in the recovery, which gauges efficiency and energy consumption.
- The *number of nodes moved* in the process, which serves as a measure of the scope of the recovery effort.
- The *number of messages exchanged* messages among nodes. This is a measure of recovery process overhead.

These parameters are used to vary the WSN topology:

- All nodes have the same *communication range (r)*. RIM should converge more quickly in highly-connected WSNs (ones with large *r*). This is because fewer nodes have to be engaged in the cascaded movement, reducing overhead.
- The *maximum inter-node distance (MaxiD)* constrains where nodes are positioned relative to others and limits over-spreading of nodes in the deployment region. A small *MaxiD* yields a dense topology with high connectivity among nodes, which should boost RIM's performance.

B. *Baseline Approach*

We compare the performance of RIM to the Nearest Neighbor (NN) algorithm which, like RIM, pursues greedy heuristics. When a node S_f fails, NN moves its closest neighbor, S_{NN}, to where S_f is located, repairing the severed connectivity around S_f. The neighbors of S_{NN} react to its departure, as the closest among them moves and settles where S_{NN} used to be, and so on. Although this process resembles the cascaded motion in RIM, the scope of relocation can be as wide as the entire network. NN terminates when there is no neighbor for a departed node (reaching the network periphery) or when all nodes in the network have already moved (which ensures NN eventually terminates). Unlike RIM, NN requires that every node is aware of its 2-hop neighbors so that the nearest neighbor will be known before the failure of S_F.

Figure 5 explains the NN algorithm through an example. Part (a) shows the initial topology and which links will be severed when *F* dies (red dotted lines). Nodes *A*, *G* and *H* are aware that node *B* is the nearest to *F* and will wait for *B* to move. Node *B* notifies *C* and *D* before it starts moving and *A*, *G* and *H* after reaching the new position. Parts (b) and (c) show the network topology after *B* replaces *D* and *F* replaces *B*. *E* will be the only affected node and will move to where *D* was, as seen in part (d).

C. *Simulation results*

We have simulated sparse and dense WSN topologies where *r* is chosen from the set {40, 60, 80, 100} and *MaxiD* is selected from the set {100, 150, 200, 250}. For every topology, RIM and NN are run after a randomly selected node fails. The results of the individual experiments are averaged over 10 trials. All results are subjected to 90% confidence interval analysis and stays within 10% of the sample mean.

<u>*Distance moved:*</u> Figure 6 shows the total distance that nodes had to travel during the recovery. As expected, RIM always outperforms NN. In RIM if a node has to change its position, it must only reach the boundary of the communication range(s) of its parent(s). Therefore, RIM minimizes the overhead incurred by the individual nodes engaged in the restoration process. On other hand, NN moves nodes to the exact position of the failed (or departed) node. Thus the total distance required for restoration is similar to the product of the number of moving nodes and the average distance between nodes.

Figure 6 also indicates that the performance of RIM degrades as *MaxiD* increases and/or *r* decreases because this creates non-dense topologies with low connectivity. This

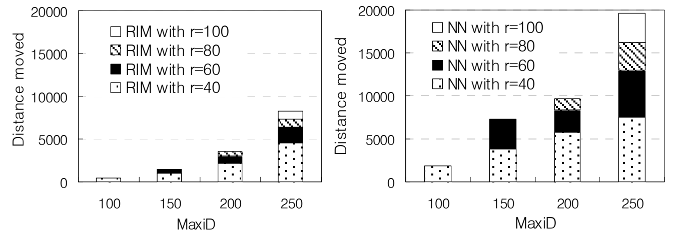

Figure 6: RIM causes less node movement than NN. Performance of both algorithms degrades as *MaxiD* increases or *r* decreases.

increases the distances traveled by the individual nodes during

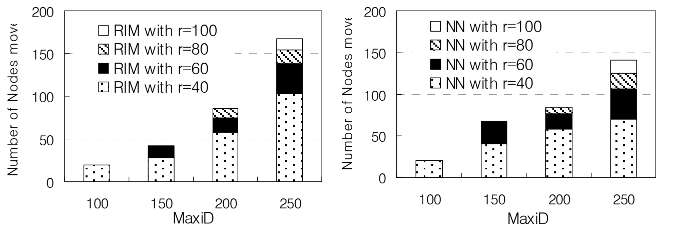

Figure 7: RIM moves more nodes than NN; increasing *MaxiD* or decreasing *r* causes both algorithms to move more nodes.

cascaded movement.

Number of moved nodes: Figure 7 shows that NN moves fewer nodes than RIM. Since RIM considers the links between nodes when it determines which nodes will move in each step, all neighbors that lose connectivity will be engaged in the cascaded movement process. On the other hand, the NN approach only moves a single node in each step. Therefore, RIM requires more nodes to relocate than NN, especially in sparse networks (ones with high *MaxiD* and/or low *r*). In these cases, links become very sensitive to nodes' proximity.

While at first this result seems negative (implying that

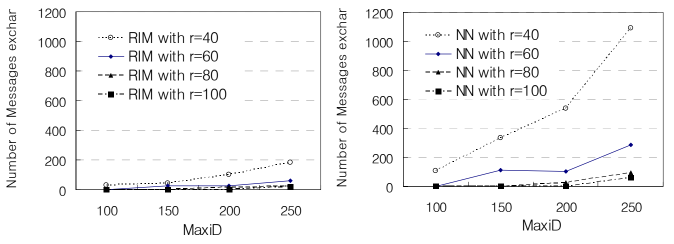

Figure 8: RIM creates far fewer messages than NN in all test cases.

RIM is inferior), it shows that RIM fairly balances the burden of movement among the individual nodes. Each node moves less on average, which helps conserve each sensor's energy supply and extends its lifetime.

Number of exchanged messages: As shown in Figure 8, RIM creates far less messaging overhead than the NN algorithm. This is because RIM only needs 1-hop information, rather than the 2-hop information NN requires. Although RIM engages more nodes than NN, most RIM messages are between a parent and its children and take the form of broadcasts which are counted only once. Figure 8 also indicates that messaging traffic is higher in sparse networks with high *MaxiD* and/or low *r*. Sparse networks have higher link losses, which motivate more nodes to move in order to sustain their connectivity to their neighbors, as also indicated by Figure 7. However, this affects the NN algorithm far more than RIM.

VI. CONCLUSION

RIM is a distributed algorithm for Recovery through Inward Motion. Unlike other schemes found in the literature, which perform a network-wide analysis to assess the impact of a node failure, RIM triggers a local recovery process by relocating the neighbors of the lost node. RIM minimizes messaging overhead and reduces the distance each individual node travels during the recovery. Simulations confirm RIM's effectiveness and demonstrate its ability to reduce overhead.

Future work includes extending the validation for RIM and comparing it with more sophisticated approaches that require significantly larger portions of the network state. We opt to assess the pros and cons and categorize the suitability of the available solutions to various application setups.

Acknowledgement: This work is supported by the National Science Foundation, contract # 0000002270.

REFERENCES

[1] I. F. Akyildiz W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: a survey", *Computer Networks*, Vol. 38, pp. 393-422, 2002.

[2] L. Bao and J. J. Garcia-Luna-Aceves, "Topology Management in Ad Hoc Networks," in the *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003)*, Annapolis, MD, June 2003.

[3] K. Akkaya and M. Younis, "COLA: A Coverage and Latency aware Actor Placement for Wireless Sensor and Actor Networks", in the *Proceedings of IEEE Vehicular Technology Conf. (VTC-Fall'06),* Montreal, Canada, September 2006.

[4] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less Low-cost Outdoor Localization for Very Small Devices," *IEEE Personal Communications*, Vol. 7, No. 5, pp. 28–34, Oct. 2000.

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, second edition, *Introduction to algorithms*, MIT press, 2001.

[6] M. Younis and K. Akkaya, "Strategies and Techniques for Node Placement in Wireless Sensor Networks: A Survey," *The Journal of Ad-Hoc Networks*, (to appear).

[7] G. Wang, G. Cao and T. La Porta, "Movement-Assisted Sensor Deployment," in the *Proceedings of the 23rd Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'04)*, Hong Kong, Mar. 2004.

[8] N. Heo and P. K. Varshney, "Energy-Efficient Deployment of Intelligent Mobile Sensor Networks," *IEEE Trans. on Systems, Man, Cybernetics*, Part A, Vol. 35, No. 1, pp. 78-92, Jan. 2005.

[9] G. Wang, G. Cao, T. La Porta, and W. Zhang, "Sensor Relocation in Mobile Sensor Networks," in the *Proceedings of the 24th Annual IEEE Conf. on Computer Communications (INFOCOM'05),* Miami, FL, Mar. 2005.

[10] P. Basu and J. Redi, "Movement Control Algorithms for Realization of Fault-Tolerant Ad Hoc Robot Networks," *IEEE Networks*, Vol. 18, No. 4, pp. 36-44, Aug. 2004.

[11] A. Abbasi, K. Akkaya and M. Younis, "A Distributed Connectivity Restoration Algorithm in Wireless Sensor and Actor Networks," in the *Proceedings of the 32nd IEEE Conf. on Local Computer Networks (LCN'07)*, Dublin, Ireland, Oct. 2007.