Dependable Wireless Sensor Networks by Dynamic Redundant Deployment

Nancy Alrajei

Palestine Polytechnic University, Department of Information Systems and Multimedia, Hebron, Palestine

***Abstract*—It is essential for Wireless Sensor Networks (WSN) to be deployed in a dependable manner to extend the network life time and to stay functional as long as possible. Redundancy has always been the key ingredient used to provide Dependability. The concepts of k-connectivity have been used in the recent years to quantify this redundancy and fault tolerance and dependability. In this paper we analyze in depth the concept of dependability and its threats in WSN. Furthermore propose deployment algorithm based on redundancy. The redundancy is not uniform but concentrated around areas of the network that are either more vulnerable or whose failure is more consequential.**

Keywords- Dependability; redundancy; k- connectedness; Fault tolerance; WSN; deployment

I. INTRODUCTION

A sensor network is a collection of low cost, small form factor, embedded devices called sensor nodes. Sensor network can provide access to information anytime, anywhere by collecting, processing, analyzing and disseminating data. Sensor nodes are great for deployment in hostile environments or over large geographical areas. This exposes them to attackers who capture and reprogram individual sensor nodes. Once in control of a few nodes inside the network, the adversary can extract private sensed information from sensor network readings [1].

Wireless sensor networks (WSN) could be deployed in both civil and military applications such as volcanic eruption monitoring, target monitoring, security and remote surveillance [2]. Their deployment in remote and frequently hostile environments combined with the device constraints, makes them particularly vulnerable to attacks from adversaries causing the nodes to fail.

Besides outsider attacks, these nodes are human made, and human-made systems in general are fault-prone. We cannot predict if it will continue to work happily ever after or not. These systems are vulnerable to failure and it is hard to predict when a failure will happen. Human-made systems are designed so as to minimize the occurrence of faults, yet it is understood that faults are bound to occur caused by wear and tear or by some overlooked conditions, design errors, or usage errors.

It is critical to design systems, including wireless sensor networks, to be dependable , i.e., to be intrinsically equipped to recognize the occurrence of faults, and react to them so that their behavior remains if not correct, at least safe. The idea of dependability is have the property that enables a system to continue operating properly in the event of the failure of some of its components.

Dependability has been studied extensively in a variety of systems from those that are primarily mechanical, electrical, electronic, or software, to the increasingly complex and hybrid systems that have all aspects combined. Dependability is a desirable feature for any computing system, by dependability we mean the system will not have any faults that affects its operational functions.

In General, Redundancy has always been the key ingredient used to provide dependability. Sensor networks are no exception In fact, they are by design redundant since they consist of a set of interchangeable, functionally overlapping sensor nodes. Upon deployment, a sensor network is generally highly redundant through the use of more nodes than strictly necessary. Its level of redundancy decreases as nodes fail, get depleted of their power [3], or fall victims of unintentional or malicious attacks. An important issue when deploying a sensor network is thus in determining the initial density required that will ensure that the network would remain alive throughout the lifetime of the application,from weeks to months or longer. [4].

A challenge in designing a dependable wireless sensor networks is to know the right network redundancy; the amount of nodes needed to deploy to achieve full converge and connectivity.[5] We surveyed the field of dependability and fault tolerance in WSN specifically, we found that the key to fault tolerance is redundancy. The general idea is that if we put enough nodes, the network should be resilient and dependable since whatever nodes die there will be others in their neighborhood.

It reasonable to expect that uniformly deploying more nodes is a quasi optimal way to increase the lifetime of the network. When the risk is malicious attacks, on the other hand, just adding more nodes becomes a very weak strategy because node failures are not random. A malicious attacker can choose which nodes to attack and can concentrate efforts at breaking the network by targeting specific areas. In this paper we discuss the issue of using redundancy and managing redundancy in a way to maximize Dependability. The question we address is: “Suppose that I am willing to devote h times more nodes than needed. What is the best use of the additional nodes? More specifically, given an area A of interest, given a set M of nodes, (1) What is the optimal organization for these nodes? (2) How can we get the nodes to self-organize in a way that would be a good approximation of this optimal organization? (3) How can we get the nodes to autonomously keep adapting their organization to faulty nodes in the network?”

This paper is organized as follows: In Section two, we discuss threats to dependability; faults , errors, and failures. In Section Three, we presented different dependability techniques from the literature. In section Four, we present traditional measures of redundancy that we concluded they are not sufficient to achieve dependability. In Section Five we propose a redundant based distributed deployment algorithms run on the individual nodes that will make the network organize more strategically and reorganize to reflect node failures. We summarize and conclude in Section Six.

II. Dependable Threats

The definition for a dependable system is different from one application to another. The dependability of a system is the ability to avoid service failures that are more frequent and more severe than is acceptable to the user(s) [6].

What is acceptable to each user and what is not vary, but all users agree on some attributes of a dependable system. Generally speaking, we can say that a dependable system should satisfy one or more from the following attributes depending on the type of the application and the environment this application works in:

(A) *Availability*: service should be there when it is needed. WSN's capability in sustaining its service continuity even during failure times.

(B) *Reliability:* continue to provide correct service.

(C) *Confidentiality*: only authorized people should access the information.

(D) *Integrity*: ensures that the message will not be altered during communication.

There are three threats to dependability; Fault, Error and Failure.

A threat starts out as a fault, which is a defect that takes place in some parts of the system. It could be a hardware defect, or a software defect. It could be accidentally made or purposefully forced into the system. No matter what the category of this fault is, we need to contain these faults and not to have them propagate into something bigger and even more dangerous.

Faults could be active or passive. An active fault could be noticed, such as dead battery in a sensor. A passive fault may not be noticed. A mistake or a bug in the code is a passive fault. If a fault has emerged and was not taken care of, it might propagate and affect other parts of the system, so it is no longer a defect, it will become an error. An Error is a noticed phase that leads the system into a state of not doing things the right way. So a fault is active when it causes an error, otherwise it is passive. If errors propagate they lead to a failure. So a failure is the deviation from correct service may assume different forms that are called service failure modes. Therefore Faults propagate into Errors, and Errors propagate into Failures, as shown in Figure 1.

Figure 1 Fault Propagation

An example of an error is lost connection between two nodes due to the dead battery in one of them. If an error propagates it leads to a failure. That is when the system starts to behave in a way that not what it should be doing. Following our example of the dead sensor node, the failure here is not being able deliver the data from that dead node.

Faults are classified into these categories[7]:

(A) *Internal faults:* the location of that fault originates inside the network boundary.

(B) *External faults:* those originate outside the system boundary and then propagate errors into the network by interaction or interference.

(C) *Natural faults:* caused by natural phenomena without human intervention.

(D) *Human- made faults:* faults result from human actions.

(E) *Malicious faults:* the objective of introducing the faults could be malicious, introduced by a human with the objective of causing harm to the system.

(F) *Non-malicious faults:* faults that are introduced without a malicious objective.

(A) Deliberate faults: the intention of the human who caused the faults is a harmful decision.

(B) Non-deliberate faults: Those are introduced without awareness.

(C) Accidental faults: the capacity of the human who introduced the faults is inadvertently or incompetence faults, that result from lack of professional competence by the authorized human.

(D) *Permanent fault:* the temporal persistence of the fault whose existence is assumed to be continuous in time.

(E) *Transient faults:* whose presence is bounded in time.

So, the existence of faults might or might not lead to a failure in the end. We should have these faults under control and not have them end up in a system failure when repair is harder or impossible.

III. Dependability techniques

The dependability in the system is to eliminate the occurrences of the threats in the system by predicting all the possible faults that are likely to happen. If we have faults under control that means we can prevent system failures. The more faults we prevent the less failures might occur. This step is called fault prevention. Fault prevention is to prevent the occurrence or introduction of faults.

Another mean to achieve a dependable system, is fault tolerance, which is to avoid service failures in the presence of faults. Fault tolerance techniques have some similarities

with fault prevention techniques, in the sense that they both should know (predict) beforehand what faults the system might face and provide some mechanism to avoid them. Fault tolerance concern is not only preventing faults from happening but it also assumes that these faults will actually happen and find solutions on how the system will survive and continue working according to its goals without any deterioration in its performance of services it provides.

What comes after fault tolerance? Should the system continue working in the presence of faults? The step that comes after fault tolerance is fault masking which means hiding faults and prevent them from resulting in errors. Finally, fault isolation; this moves the fault from the system either manually or automatically. Every technique in handling faults is a topic of research on its own.

Wireless sensor networks are fault prone because of the limited resources, and because nodes are deployed in harsh environment they are vulnerable to different threats. Figure.

A. Classifications of Fault in WSN

Faults may happen and can lead to life threatening failure. For example physical damage to the node itself, or delay in sending or receiving the data, or corrupted data. To understand the type of faults that could happen, we categorized the faults in each of the system layers as in Figure

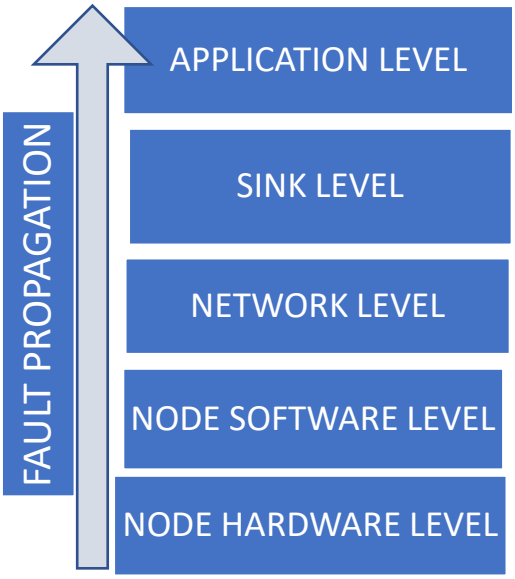

Figure 2 FAULT CLASSIFICATION

- ***Node level*:** Each node consists of hardware and software components. The hardware consists of sensors, CPU, memory, and battery. And the software: routing protocol, and Mac protocol. Hardware failure will generally lead to software failure in the node. Low power level in a node can give incorrect readings. The node as a whole might fail if that node was in a critical position such as if that node lies on the routing path to the sink.
- ***Network level:*** The radio communication between nodes is also vulnerable. When these nodes communicate wirelessly, two nodes might send their data at the same time which will end up in losing or colliding of data. The main reason of collision is congestion, when sensing nodes are allowed to transfer as many packets as possible at the same time, resulting in packet loss.
- ***Sink level***: The sink or the base station is responsible for collecting all the data from the sensors. This component could fail due to many reasons. It could be the sink node itself corrupted, presence of bugs or other problems in the hardware or the software of the sink. Or it could be that it is not possible to supply power to the sink permanently.
- ***Application level:*** The malfunction in nodes in the wireless sensor network that does not perform the intended operation according to user requirements, or it performs other operations replace sending incorrect information to the base station. Hence the faulty sensor node must be identified and separated from being a communication node that is responsible for routing data to the base station.

B. Classifications of Failures in WSN

Failures are results of the faults that happened in one of the previously explained category. These failures could be either one of the followings:

(A) Crash failure: occurs when the service doesn't respond to the request. It could be because of message loss or physical damage that broke the network apart into different disjoint components that the sink can no longer communicate with.

(B) Time failure: if the application has a strict time interval, the node responds to a request with possibly the correct value but the response was received either too early or too late.

(C) *Incorrect value failure:* generating an inaccurate value, by either a software malfunction, corrupt message, or a malicious. This failure results in lack of accuracy in the aggregated final value.

IV. TRADITIONAL MEASURES OF REDUNDANCY

We consider a sensor network used to monitor some parameters (e.g., detect human/animal presence) over an area (assume a square) A of interest. We further assume that all nodes are identical with a sensing range radius Rs and a communication range Rc. There are no larger than Rs. There are typically two concerns: Coverage and Connectivity. Coverage refers to the fact that every point of the area A is within sensing range (distance no larger than Rs) of some node. Connectivity refers to the fact that between any pair of nodes (n, n') there is a sequence n0=n,n1,n2,...nk=n' such

that the distance between any ni, ni+1 is no longer than the communication range Rc; in other words, any two nodes are able to communicate with each other. The network created by the set of nodes defines a graph whereby the nodes are the vertices of the graph and there is an edge between any two nodes that are within communication range of each other. When coverage is the only concern, the minimal number of nodes required to cover area A is proportional to the ratio L2/Rs where L is the length of the sides of the square area. This minimal number of nodes can be obtained by organizing the nodes in a lattice of equilateral triangles of side √3 Rs as is shown in Figure 3. When we are concerned with both connectivity and coverage, the optimal configuration depends on the relationship between Rs and Rc. If the communication range is equal to or larger than √3 Rs, the optimal configuration for coverage more than satisfies connectivity [8],[9]. The lattice, in fact will ensure that every node is at the center of a hexagon, and thus is connected to the six corners of that hexagon thus providing a high level of connectivity. In many applications, notably those where the phenomenon being monitored is continuous and not discrete, R_c becomes the most critical parameter and connectivity the most critical issue. This is the subject of our focus here.

To ensure connectivity, when the sensing range is large enough, it is sufficient to organize the nodes in a chain that meanders through the surface. Resilience and fault tolerance require more than simple coverage and connectivity. We need connectivity even after nodes fail, exhaust their power, or get corrupted. In other words, we need to ensure that nodes are connected in more than one way so that when some of these ways fail others remain available. This concept of multiple and redundant connectivity has been identified in the literature under the name of k-connectivity and formally defined below

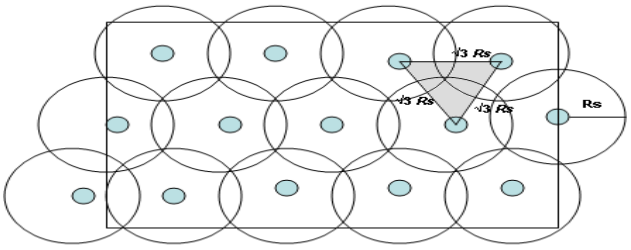

Figure 3 Optimal organization for overage

The vertex connectivity Kv(G) of a connected graph G is the minimum number of vertices whose removal can either disconnect G or reduce it to a 1-vertex graph. A graph G is k-vertex-connected if G is connected and Kv(G)>= k. If G has non-adjacent vertices, then G is k-connected if every vertex-cut has at least k vertices.

The edge connectivity Ke(G) of a connected graph G is the minimum number of edges whose removal can disconnect G. A graph G is k-edge-connected if G is connected and Ke(G)>=k. G is k-edge-connected if every edge cut has at least k edges. In the case of sensor networks, failures happen at the vertex level rather than at the edge level, thus we are interested in k-vertex-connectivity rather than k-edge-connectivity. In the reminder of this paper we will simply talk about k-connectivity to refer to k-vertex-connectivity. In a k-connected graph, between any two nodes there are at least k disjoint paths [10] In a k-connected graph, every cut has at least k vertices.

The concept of k-connectivity has been introduced in graph theory and predates the widespread of sensor networks. With the emergence of sensor networks and their applications, many researchers identified the potential of this property for fault tolerance and fault repair [11]. One key challenge in organizing a sensor network so as to obtain the highest connectivity with the minimal number of nodes is that most of these problems are NP hard [12]. For this reason, many researchers in sensor networks, in addition to resorting to simplifying, but application-adequate models [11], focus on finding linear time approximation algorithms. Bredin et al. [12] investigated heuristic algorithms for deciding which nodes to wake when some nodes fail and for identifying locations where additional nodes should be placed. Wu and Li [13] and Alzoubi et al. [11] focus on the connectivity, not of the whole network, but of the subset of nodes that play a key role in routing. Specifically, they propose approximate algorithms for k-connected m-dominating sets. In [8], Bai et al. calculate the optimal deployment for the purpose of achieving k-connectivity. They discuss various patterns such as the triangular lattice and square grids. In [18], Xing et al. discuss protocols for ensuring multiple coverage and connectivity. In particular, by using a communication range twice as large as the sensing range, they distribute the nodes so as to ensure coverage, and by the same token have a high level of connectivity. In [14], Khelifa et al. discuss the inadequacy of traditional k-connectivity as a true measure of the level of redundancy and level of fault tolerance of a network, notably for large networks. In effect, as the network grows in size, the probability that the k nodes that fail belong to the same cut becomes very small. Conditional connectivity measures the level of fault tolerance in a network by focusing on the probability of failure of all the nodes of the same cut. While Khalifa et al. focus on the quantitative aspects, we address here the pragmatic aspects of organizing and reorganizing the network to maximize fault-tolerance. The approach proposed in this paper shares the underlying premise of Ammari and Das [15] by accounting for the conditional probability that the failure of a node will indeed lead to a failure of the network. We motivate our approach by first introducing three intuitive concepts.

Given a square surface area of side length L, and given a set of nodes N, what configuration would statistically maximize the length of time that the network is connected? The rationale here is that given a value k (e.g., 5), it may not be the best strategy to strengthen connectivity in a uniform way as not all vertices are equally useful and not all vertices are equally vulnerable. We discuss multiple considerations when considering the importance and vulnerability of the vertices.

(A) The level of redundancy of a node.

In a graph that disposable) than nodes that belong to smaller cuts. In Figure 2, if we only focus on the two cuts shown, node a is less critical than nodes d and e. Node a belongs to a cut with three nodes, should it fail the connectivity will be ensured through c and e whereas a failure of d makes the network connectivity dependent on only one node: e. is 1- connected, there is no redundancy. Every node failure can disconnect the graph. In a k-connected graph, the k-1 first failures are safe because every vertex-cut has at least k vertices. When all cuts have exactly k vertices, and if we call p the probability that any node fails within a time period P, the probability that the network gets disconnected within that same time period from that specific cut is p^k. In practice, different cuts have different sizes. The larger is the size of the cut the lower is the conditional probability that the network gets disconnected when a node of that cut fails. Nodes that belong to large cuts have a lower conditional probability of causing the network to fail when they fail. They are therefore "less critical" or "lower impact" (more disposable) than nodes that belong to smaller cuts. In Figure 4, if we only focus on the two cuts shown, node a is less critical than nodes d and e. Node a belongs to a cut with three nodes, should it fail the connectivity will be ensured through c and e whereas a failure of d makes the network connectivity dependent on only one node: e

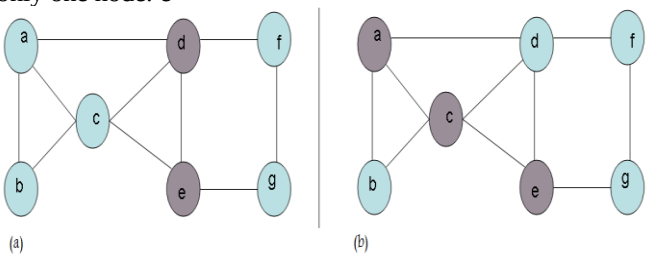

Figure 4. In (a) cut size is 2, in (b) cut size is 3

(B) Centrality of a node

Chipping vs. chattering the network. Given a connected graph, given a cut that divides the graph in connected components, the number and relative sizes of the resulting connected components are important in determining the impact of such a cut. A cut that barely "chips" the network by isolating a very small connected component from the rest of the network is less damaging than a cut that breaks it apart in many pieces or in two large pieces. The type of breakage that results from a cut characterizes the impact of losing all the nodes of a cut. Nodes in a cut that barely chips are less critical than nodes in cuts that chatter the network. This is illustrated in Figure 5. Losing node d results in disconnecting g but leaving the rest of the network together. By contrast, losing node c breaks the network into 4 components.

(C) "Diameter" of a node:

The intensity of the flow that goes through it. The network's functionality is captured by the flow circulating through it. Underlying every non directed sensor network, there is in fact a directed network where most there is in fact a directed network where most communication takes place from and towards the base station.

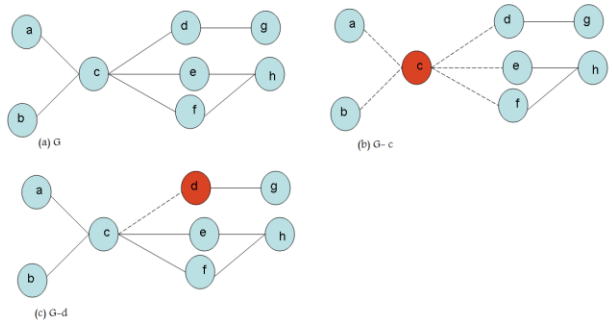

Figure 5 (a) Connected Graph G, (b) G after failure of c, (c) G after failure of d

For example, in the graph in Figure 6 even though all nodes have similar connectivity degrees with communication flowing in both directions, there is an (many) underlying tree showing the routing of information from the base station to the rest of the network and from the rest of the network towards the base station. Thus, what is even more important than the connectedness of a node is the amount of information (flow) that that node is responsible for. For example, the flow of the incoming information (blue routing tree) is shown in the graph where the number associated with every node represents the number of measurements that node transports (effectively, it is the size of the routing sub-tree rooted at that node).

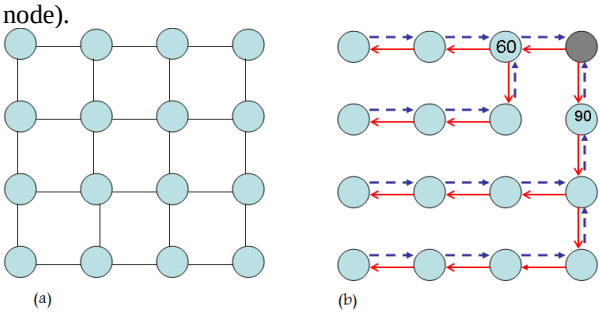

Figure 6. Flow through the network

Before we discuss these 3 concepts further we have to recognize that the first two concepts characterize a node based on "the" cut under consideration. Generally, a node belongs more than one cut, therefore a full characterization of a node would need to combine all cuts. We will not do that.

These concepts are presented here simply as an intuitive motivation for the following section. We are not planning on quantifying them or computing them. What we discuss here instead is the way in which they are inter-related. The level of redundancy of a node is the redundancy that is formally captured by the concept of k-connectivity except that k-connectivity sets an initial lower bound and captures the minimal level of redundancy present in the network. The conditional probability that the network gets disconnected when a node fails on the other hand, is associated with a node is a local measure of how much back up a node has. The centrality and diameter of a node are related in the sense that the flow that goes through a node is typically the flow from the sub-tree defined by that node towards the base station. The smaller is the sub-tree, the more the failure is likely to result on a chipping; the larger is the sub-tree, the more the

failure is likely to result in a chattering. Therefore, both concepts correlate. Measuring one or the other will do. The key is to organize the network so that nodes with a high flow have a high level of redundancy.

V. Redundancy-Based Deployment Algorithm

Regardless of how efficient is the original organization of the network, such organization is only statistically quasi-optimal. If there are no malicious attacks and nodes only fail because they exhaust their power it is possible that network will remain connected until it collectively dies out. In other words, the networks will deplete uniformly and die graciously chipping one bit at a time. Such scenario, while possible, is not very likely. There are enough unknowns and enough randomness to almost guarantee that whatever organization is selected in the beginning, it does not match exactly the scenario that will take place. For this, we want the organization to be dynamic and adaptive by allowing nodes from robust areas to cover for areas that get depleted. This is the concept of elasticity: whenever an area of the network is under excessive pressure, whenever possible, we would like it to stretch rather break. In fact, elasticity consists of stretching much earlier than the time when the network is at the point of breaking.

For example, consider a connected network; the flow goes from x to y, then z. Initially x, y, and z have cuts of size 3, 4, and 5 respectively. As 2 nodes from the cut of y fail, one node from the cut of x moves up and one node from the cut of z moves down resulting in cut sizes of 2, 4, and 5 (the nodes that migrates from z down to y is part of the cut of both y and z.

we want the More nodes to gather in the direction of the flow, this is called stretching the network rather than breaking it.

(A) Autonomous Self Organizing and Re-organizing Algorithms

We recall that the objective is to have the nodes self-organize so as to achieve a differential connectivity whereby nodes with higher flow benefit from the fact that they are part of larger sized cuts. We recall also that the intention is to use k*N nodes where only N would be sufficient for coverage and connectivity. We propose here a two-step process:

1) A sufficient number of nodes are deployed by spreading them using a uniform random distribution to ensure full coverage and connectivity.

2) This initial set of nodes collaborates to establish a reasonable flow pattern and identify the amount of flow traversing each of the nodes.

3) The remaining nodes are deployed in such a way that they are attracted to "cut areas" with an intensity proportional to the flow in that area. In other words, we want areas with high flow to attract a relatively large number of nodes, resulting in larger cuts whereas areas of small flow would attract fewer nodes resulting in smaller cuts.

We discuss each of these three stages:

a) Initial Uniform node deployment: This is the usual method to deploy most large sensor networks. The nodes are sprayed from a distance. A sufficient number of nodes are sprayed to maximize the likelihood that the network will cover the area of interest and will be connected. Because of the vulnerability of nodes (limited-life time battery, among others), most networks are deployed with sufficient redundancy. In other words, when only N nodes are sufficient to cover the area when placed in an optimal configuration, 2(or more)*N nodes are placed. In this case, we target to provide enough redundancy to ensure connectivity and coverage, but we start with a number close enough to the minimal.

b) Establishing the flow pattern. A number of algorithms and approaches have been proposed to guide the pattern of communication (routing) between the nodes of a network. It is without loss of generality that we will assume that

- The overall routing takes place using a routing tree, sub-graph of the overall network
- The tree is rooted at the base station or some other representative sink node.
- The communication can be decomposed into two flows: from the root of the tree towards the leaf nodes to broadcast the "query", i.e. the nature of the data requested from the nodes and from the leaf nodes up towards the root to send the data back to the base station.
- The two flows follow the same links in opposite directions.
- Finally, we assume that a quasi-minimum spanning tree rooted at the sink node minimizing the total communication cost (and thus the total length of communication links used is a good approximation of the actual routing structure that will be used.

The spanning tree will be constructed; the flow traversing each of the nodes based on this minimum spanning tree will then be used to estimate the actual flow going through those same nodes in the sensor network. The spanning tree has the advantage of being easily computed using a greedy distributed algorithm:

When the network becomes quiescent we have established two things:

- The spanning tree rooted at the sink node has been constructed.
- The flow going through each of the nodes has been computed.

Deploying the remaining nodes: The key now is to deploy the rest of the nodes so that they assemble around nodes in a way proportional to the flow through those nodes. The general idea is to disperse the nodes and then subject them to attractive forces by the existing nodes whereby the intensity of the attractive forces is correlated with the flow in the nodes. One way to visualize this process is to see the nodes with their flow as points in 3D space where their positions on the plane represent their x and y coordinates and their flow represent their elevation (elevation=large value-flow to be more accurate). For example, given the spanning tree and its flow shown in Figure 7. the resulting is shown in Figure 8.

In other words, whereas the first batch of nodes establishes the topography, the remaining nodes are dispersed and let initial random positions, gravity forces, and inertia determine their final positions. We discuss each of these three elements in turn:

(A) *Initial positions*: In the same way that the initial batch was deployed at random to cover the area, we can do the same thing with the remaining nodes. The x- and y-positions will be decided at random; the z-position will be initialized to unknown. In the simulation and validation we will experiment with different initial scenarios and assess their impact on the quality of the final configuration and on the amount of energy required to reach equilibrium.

(B) *Gravity*: The forces used to organize the node are calculated based on local information about gravity forces. The mobile nodes in this new batch have a node id, an x- and y- position, but have no flow (yet), i.e. they are not aware of their elevation. This latter information is obtained from the nodes in their communication disk.

Each node rom the first batch broadcasts a message m=<topic= "position", id=self.id, x=self.x, y=self.y, z=self.f>. Each node from the new batch listens to all of the incoming messages, identifies nodes with the "lowest

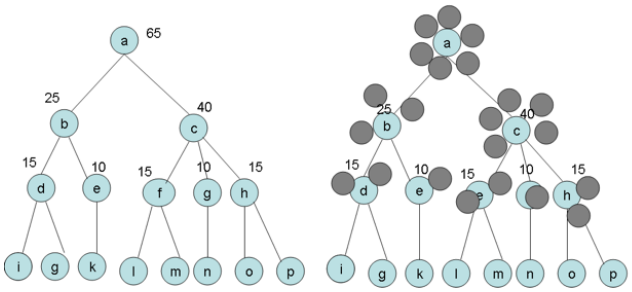

Figure 7. Flow in the Network, (a) Flow Size in each Sub-tree (b) Additional Nodes

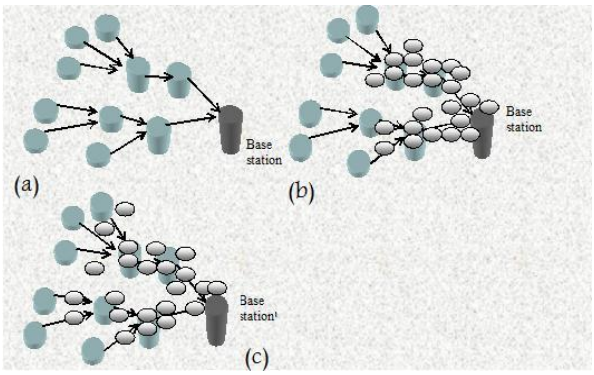

Figure 8. Deployments of Nodes (a) Initial Deployment, (b) Gravity Attracts Additional Nodes, (c) Inertia Force to Spread the Nodes

elevation", i.e, highest z, and moves towards them. The relative positions of the neighbors and their elevations determine the direction of movement and its intensity. More specifically: If there is only one neighbor, move towards that neighbor (this is unlikely to happen given the level of connectivity but we need to make provision for it) if its elevation is lower than self. We will assume that any value is lower than unknown

1. If there are multiple neighbors with lowest elevation than self, identify two neighbors with the lowest elevation, move towards the middle between them, calculate own elevation to be higher than the highest of the two.

2. If there are multiple neighbors but all of them have a higher elevation then self, do not move.

Each of the movements prescribed above is a small step after which the node listens again and may keep moving.

Inertia: Using the gravity alone, nodes stop only when they reach local minima. Because the flow grows rather uniformly from the boundaries of the area towards the sink node, there is a risk that all the nodes will end up flocking towards the lowest elevation point, i.e. the sink node. We will add an inertia force to ensure a more distributed spreading of the nodes. The form and intensity of this inertia force will be decided experimentally. We will test two different forms: a repulsion force based on the crowdedness of the neighborhood. This will consist of a repulsion force that will counteract the gravity attraction force. A node attracts mobile nodes when it has a low elevation. On the other hand a node also repulses mobile nodes with intensity proportional to the crowdedness of its neighborhood

VI. Conclusion

Redundancy is an integral part of sensor networks. The general idea being that if we put enough nodes, the network should be resilient since whatever nodes die there will be others in their neighborhood.

The concepts of k-connectivity have been used in the recent years to quantify this redundancy and fault tolerance. The concept of k-connectivity for example states that in a k-connected network, the network will remain connected after the failure of any k-1 nodes. While very useful, this concept does not fully capture how fault tolerant a network is. In particular, we want to know "how likely is a k-connected network to be disconnected after the failure of k, or k+1, or 2k nodes?" Another important and more pragmatic question is the one we address here: Suppose that I am willing to devote h times more nodes than needed. What is the best use of these additional nodes? Should we organize them so that the network has the highest k-connectivity level? We argue here that instead of uniform connectivity, we would be better off increasing the connectivity where it matters the most rather than uniformly.

Furthermore, we propose distributed algorithms that allow the nodes to organize themselves in an autonomous manner and to reorganize themselves as dictated by the changing needs of the network. The proposed algorithms are based on forces exercised by nodes over other nodes in their communication range. The interplay between the different forces generates the desired collective behavior.

REFERENCES

[1] Al-Karaki, J.N., Kamal, A. E. 2004. Routing Techniques in Wireless Sensor Networks: A Survey. IEEE Wireless Communications, Dec. 2004, Vol 11, No. 6, pp.6-28, Dec. 2004.

[2] I. Akyildiz and V. Mehmet, "Wireless sensor networks," Vol. 4. John Wiley & Sons, 2010.

[3] DelaRose, J. Y. Liu, L. Mili, A. Phadkee, L. Davilva, 2005. Catastrophic failures in power systems: Causes, Analyses, and Countermeasures. Invited paper. Proceedings of the IEEE. Vol. 93, No. 5, pp. 956-964.

[4] Lambrou, Th., Panayiotou, Ch., Felici, S., Beferull, B. 2010 Exploiting Mobility for Efficient Coverage in Sparse Wireless Sensor Networks. Wireless Personal Communications, Vol 54, pp. 187-201, doi:10.1007/s11277-009-9717-0.

[5] N. Alrajei,F. Mili "Differential Elasticity for Network Resilience", IEEE NAECON Bioinspired Systems and Biomedical Applications, Dayton Ohio, July 2010. IEEE Press.

[6] Soon. Oh, Mario Gerla, and Abhishek TiwariRobust, "MANET routing using adaptive path redundancy and coding," first international conference on communication systems and network India, 2009.

[7] Alrajei,H Fu, Y.Zhu "A Survey on Fault Tolerance in Wireless Sensor Networks" 2014 American Society For Engineering Education North Central Section

[8] Bai, X., Kumar, S., Xuan, D., Yun, Z., and Lai, T. H. 2006. Deploying wireless sensors to achieve both coverage and connectivity. InProceedings of the 7th ACM international Symposium on Mobile Ad Hoc Networking and Computing(Florence, Italy, May 22 - 25, 2006). MobiHoc '06. ACM, New York, NY, 131- 142. DOI= http://doi.acm.org/10.1145/1132905.1132921.

[9] Lambrou, Th., Panayiotou, Ch., Felici, S., Beferull, B. 2010 Exploiting Mobility for Efficient Coverage in Sparse Wireless Sensor Networks. Wireless Personal Communications, Vol 54, pp. 187-201, doi:10.1007/s11277-009-9717-0.

[10] Zhang, H. and Hou, J. 2004. Maintaining Sensing Coverage and Connectivity in Large Sensor Networks. In NSF International Workshop

[11] Alzoubi, K. M., Wan, P., and Frieder, O. 2002. Message-optimal connected dominating sets in mobile ad hoc networks. InProceedings of the 3rd ACM international Symposium on Mobile Ad Hoc Networking &Amp; Computing (Lausanne, Switzerland, June 09 - 11, 2002). MobiHoc '02. ACM, New York, NY, pp. 157-164. DOI= http://doi.acm.org/10.1145/513800.513820.

[12] Bredin, J. L., Demaine, E. D., Hajiaghayi, M., and Rus, D. 2005. Deploying sensor networks with guaranteed capacity and fault tolerance. In Proceedings of the 6th ACM international Symposium on Mobile Ad Hoc Networking and Computing (Urbana-Champaign, IL, USA, May 25 - 27, 2005). MobiHoc '05. ACM, New York, NY, pp. 309-319. doi= http://doi.acm.org/10.1145/1062689.1062729.

[13] Wu, Y. and Li, Y. 2008. Construction algorithms for k-connected m- dominating sets in wireless sensor networks. In Proceedings of the 9th ACM international Symposium on Mobile Ad Hoc Networking and Computing (Hong Kong, Hong Kong, China, May 26 - 30, 2008). MobiHoc '08. ACM, New York, NY, pp. 83-90. doi= http://doi.acm.org/10.1145/1374618.1374631.

[14] Khelifa, B. Haffaf, H. Madjid, M. and D. Llewellyn-Jones 2009. Monitoring Connectivity in Wireless Sensor Networks. International Journal of Future Generation Communication and Networking Vol. 2, No. 2, June, 2009

[15] Ammari, H. M. and Das, S. K. 2009. Fault tolerance measures for large-scale wireless sensor networks. ACM Transaction on Autonnmous and Adaptive Systems 4, 1 (Jan. 2009), pp. 1-28 doi= http://doi.acm.org/10.1145/1462187.1462189.

[16] N. Alrajei, G. Corser, H Fu, Y. Zhu "Energy Prediction Based Intrusion Detection in Wireless Sensor Networks," International Journal of Emerging Technology and Advanced Engineering (IJETAE, ISSN 2250–2459 Online, ISO 9001:2008 Certified Journal) Volume 4, Issue 3, March 2014

[17] Alrajei, N. "Implementation of Information Systems Security Threats: The Byzantine Generals and Wormhole Attacks." Journal of Communications Technology, Electronics and Computer Science 16 (2018): 6-11.Technology, Electronics and Computer Science, 16, pp.6-11.

[18]Xing,G.,Wang,X.,Zhang,Y.,Lu,Ch.,Pless,R.,andCh.Gill.2005. Integrated coverage and connectivity configuration for energy conservation in sensor networks, ACM Transactions on Sensor Networks (TOSN), v.1 n.1, p.36-72, August 2005.