Topology Maintenance of Wireless Sensor Networks in Node Failure-prone Environments

Lisa Frye, Liang Cheng, Shenfu Du, and Michael W. Bigrigg

***Abstract*—A critical issue in wireless sensor networks is topology management, and maintaining connectivity throughout the network and having a sufficient network lifetime in particular. There have been many topology management schemes proposed. Most of them conserve energy by periodically putting nodes to sleep. One consideration often overlooked is the fact that sensor nodes will fail, especially in harsh environments. In this paper, a modification to an existing topology management scheme, Naps, is proposed to ensure very minimal data loss while also handling node failures. The idea is to conserve energy to prolong the lifetime of the nodes and the sensor network while maintaining a constantly connected sensor network. Simulation results demonstrate that failed nodes do not have a significant impact on network connectivity or lifetime, at least in a random, uniform deployment. The paper also illustrates that two different approaches taken in the Naps modification for compensating for failed nodes produce similar results.**

I. INTRODUCTION

A. Node-Failure Prone Environments

WIRELESS sensor networks (WSN) may be used in various situations, such as habitat monitoring, environmental monitoring, first responder situations, homeland security applications, and military scenarios, to relay communication and environmental information, hop-by-hop, among users. Current communication systems tend to degrade quickly in many environments, including inside a structure whose walls contain significant amounts of metal, in terrain with mountains, and underground or in tunnels. By deploying a WSN that maintains line-of-sight or short-range communication between adjacent nodes, each node is capable of receiving information from other nodes and is then able to relay that information to other nodes until the information is communicated to the destination node.

The nodes used in wireless sensor networks have a limited lifetime due to the fact they are powered by batteries. Nodes may fail due to hardware failure. In many domains utilizing WSNs, conditions are not optimum. There may be environmental conditions, such as electromagnetic noise and physical destructions, which may cause a node to fail, or at least temporarily cease to participate in the current network topology. With these node failures, topology management schemes must consider the fact that some nodes may randomly fail in the network. There may be a need for other active nodes to compensate for these failed nodes. Another alternative is to deploy additional nodes into the network; however, this is not always possible due to the deployment constraints.

B. Topology Management Challenges

Using a sensor network in a practical domain requires the solution to some current technical obstacles. One of these obstacles is the topology management of a WSN. The essential concept within topology management for most domains is the ability to maintain a fully connected network at all times during the lifetime of the WSN. "Often sensor nodes are densely deployed, are prone to failures, and the topology of the network can dynamically change" [13]. With the limited power in the nodes, due to small batteries, sensor networks must be concerned about power conservation. One way to conserve power in a node is to allow it to sleep periodically, when not in use. Topology management schemes coordinate which nodes turn their radios off and when such that the traffic forwarding remains satisfactory while minimizing the network energy consumption [11, 12]. The selection of which node to be allowed to sleep and the duration it may sleep are critical issues and must be done while maintaining a connected network so time-sensitive data is not lost or delayed.

There have been many proposed solutions for topology management of a ad hoc WSN. Some of these solutions are algorithms such as STEM [11, 12], GAF [16], CEC [16], and Naps [5]. These algorithms turn off radios exploiting redundancy in order to conserve energy. One of the major disadvantages of most of these algorithms is that they trade energy conservation for increased routing latency [5].

This research has been partly financed by a grant from the Commonwealth of Pennsylvania, Department of Community and Economic Development, through the Pennsylvania Infrastructure Technology Alliance (PITA).

L. Frye is with the Department of Computer Science, Kutztown University, Lytle 267, Kutztown, PA 19530 (e-mail: frye@kutztown.edu) and the Laboratory Of Networking Group (LONGLAB), Department of Computer Science and Engineering, Lehigh University, 19 Memorial Drive West, Bethlehem, PA 18015.

L. Cheng and S. Du are from the Laboratory Of Networking Group (LONGLAB), Department of Computer Science and Engineering, Lehigh University, 19 Memorial Drive West, Bethlehem, PA 18015 (e-mail: cheng@cse.lehigh.edu). S. Du is a Visiting Research Scholar in LONGLAB, Lehigh University and is affiliated with the Network Information Center, North China Institute of Computing Technology (email: dusf@nci.ac.cn).

M. Bigrigg is from the Institute for Complex Engineered Systems, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh PA 15213 (e-mail: bigrigg@cmu.edu).

One energy-saving algorithm is STEM [11, 12] or Sparse Topology and Energy Management. This algorithm is based on time rather than network density. STEM takes advantage of the fact that a sensor network spends a majority of its time monitoring something, like the environment, and not forwarding data. A sensor network node's radio, which is the largest energy consumer, only needs to be awake when there is data to forward. STEM will put nodes to sleep (turn the radio off) to conserve energy. Nodes will periodically wake up to see if another node is trying to communicate with it. A second radio is used for sending wakeup messages (a message sent periodically to neighbor nodes so they know that this node is trying to communicate with them). This radio is a low duty cycle radio consuming little energy. If there are no nodes trying to send it data, the node will go back to sleep. The problem with STEM is that it sacrifices latency in communication because time is lost in forwarding data while waiting for nodes to wake up. This lost time is critical in some domains, like first responder, homeland security, or military scenarios.

Two algorithms that use localized, distributed algorithms are GAF and CEC. GAF [16] self-configures redundant nodes into small groups, or virtual grids, based on location information, typically provided by a GPS. If a node is determined to be redundant then it is put to sleep to save energy. GAF is limited by its dependency on global location information and performs badly in a network with low mobility or low density. There is less energy consumption achieved in GAF because instead of directly measuring connectivity, it guesses at it requiring more nodes remain awake. Another problem with GAF is that it operates independently of the routing protocol and may turn off a node that is actively routing packets. This caused periods of interrupted communication and high latency. These conditions are not tolerable in some domains.

CEC [16] is similar to GAF but does not rely on location information. Instead, CEC directly measures network connectivity thus finding network redundancy more accurately. This allows CEC to conserve more energy than GAF. CEC organizes nodes into clusters that overlap with each other. Each cluster has a node identified as a cluster-head that can be reached by all other nodes in the cluster in one hop. The responsibility of cluster-head will periodically change so all nodes have the chance to save energy. Some nodes are a member of multiple clusters and serve as gateway nodes. These nodes connect all clusters together to ensure overall network connectivity. CEC does not perform well if nodes are constantly mobile. CEC also does not perform well with data delivery if there are frequent topology changes.

II. NAPS SOLUTION

The topology management algorithm for wireless sensor networks to be studied in this paper is Naps [5]. It was developed to ensure reliable delivery of information in a wireless ad hoc sensor network while preserving power in the nodes; however, Naps compromises agility. The algorithm will decide when nodes will nap and when they should wake up in order to maintain sufficient node density to ensure a reliable network. Naps will change the set of waking nodes so that all nodes can nap. The Naps algorithm is a simple protocol that does not rely on many things that other protocols rely on, such as location information and clock synchronization. It is an algorithm that maintains good connectivity and energy conservation while scaling to very high node densities without using location information [5]. The Naps algorithm assumes that (*i*) each node is able to broadcast a message to all listening nodes within its transmission radius; (*ii*) each node will receive a broadcast message whenever it is awake; and (*iii*) message delivery is reliable and messages do not collide.

The Naps algorithm is based on two tunable parameters. The first one is a time period T that controls the rate of when nodes nap. The other parameter is the neighbor threshold c that controls the target density of waking nodes. The primary statistic analyzed is the maximum component accessibility (MCA). The MCA is "defined as the fraction of nodes that are either in or have an edge to the largest connected component of the waking subgraph" [5]. In order for an ad hoc WSN to be effective the waking subgraph must be well connected, indicated by an MCA of one.

Each node first waits a random amount of time and thereafter operates in time periods of length T. At the start of each period, a node broadcasts a "HELLO" message and will listen for "HELLO" messages from neighboring nodes. If fewer than c messages are received before the end of the period, the node will remain awake. When the node receives the c^{th} message, it will nap from that point until the end of the period. The period length is randomly chosen from $[0, T)$ every 10th iteration in order to prevent poor random choices being repeated every period. If the period length was not changed periodically, certain nodes would remain awake for a disproportionate amount of time.

The major advantage of the algorithm is its simplicity. All topology management algorithms, including Naps, require that higher layers of software can adapt to a changing topology without consuming inordinate amounts of energy [5]. Naps can be used for any application that requires selecting a set of nodes with a specified target density and which can adapt to dynamic changes in the topology. It is shown that Naps effectively saves power, especially at high densities.

III. ADAPTIVE NAPS FOR FAILED NODES

Wireless sensor networks are very dynamic as nodes are moved, added, and deleted within the network. Nodes in a WSN may move due to terrain changes, washed away by water, or other environmental conditions. Nodes may also be attached to PDAs or laptops, which are mobile by nature. Nodes may be added by a human or an automated device, like a robot, in order to compensate for failed nodes, and to increase network density or expand the coverage of the

WSN. Some nodes may be deleted from the network since node failure is a common occurrence in sensor networks deployed in harsh working environments. Some nodes will consume all their available power resulting in node failure. A node may also be destroyed, thus deleted from the network, due to events such as a building collapse, fire, vehicle, or accidentally by a human. A failed node may result in a disconnected network which can be dangerous in some application domains.

The Naps algorithm does not take into account that nodes will fail. Considering an ad hoc WSN in practical domains, we modify the Naps algorithm to make it adaptive to the node failures. The basic Naps algorithm with the introduction of node failures is shown in Fig. 1. In our simulation studies, a failed node is identified by using a random number generator. The random number will identify which node in the sensor network will fail during each cycle in the test. The node is marked as being dead and will no longer be a part of the sensor network. In order to maintain a connected network another node must compensate for the failed node. To do this, a neighbor node of the failed node will increase its energy level. The neighbor node selected for energy increase is also done at random for this particular version of the modified Naps algorithm. This will extend the node's communication range so it can reach more nodes and ensure a connected network. The random selection of a neighbor node allows the algorithm to operate in a distributed manner. It is possible that the failed node will not affect the connectivity of the network thus requiring no increase in energy of a neighbor node. This will be a future consideration.

```
Naps (Neighbor threshold $c \in Z^+$, time period $T \in R^+$)
    Wait random time $t_v$ chosen uniformly from [0, $T$)
    If random_number < total_to_fail then
        Fail a random node
    while true do
        Broadcast HELLO message
        Start timer $t$
        $i \leftarrow 0$
        while $t < T$ and $i < c$ do
            Increment $i$ for each HELLO message received
    Nap until $t = T$
```

Figure 1: The original Naps algorithm [5] with the introduction of failed nodes.

In order to extend the lifetime of the network, other factors should be considered when selecting this node. One factor is the amount of energy remaining in each neighbor node. The modified algorithm was enhanced so that the node selected to increase its energy, compensating for a failed node, is the neighbor node with the most energy remaining. This enhancement was done in the hopes of extending the lifetime of the WSN. One possible disadvantage of this enhancement is the centralized nature of this algorithm since the selection of the appropriate neighbor node is now done from a centralized location. Even though topology management is usually performed at a central station, a centralized approach may not be a scalable and robust solution in WSNs, mostly due to the fact that the decision-making node is simply another node on the network and it may fail. A centralized approach can also generate additional network traffic which will add complexity to the network.

With a node increasing its energy level to ensure a connected network, the lifetime of the sensor network should increase due to having a connected network for a longer period of time. Some of the simulation runs determine power or energy usage of each node to know if a node has energy left. These simulations will help predict the lifetime of the network and will indicate if the increase in power to compensate for failed nodes will extend the lifetime of the sensor network.

Another possibility is that the increase in energy level of a compensating node will consume more energy and may fail more quickly due to no energy remaining. If this occurs, another neighbor node will increase its energy level. However, if the node failure pattern has been clustered such that enough failed nodes reside in the same general area within the sensor network, it is possible the lifetime of the sensor network will decrease due to the network disconnection in that area. In this paper, only random, non-clustered node failures are considered.

IV. THRESHOLD SELECTION

From the simulation of adaptive Naps for a variety of densities λ (from 0.1 to 8.0 in increments of 0.1) in an area of 625 and for several integral neighbor threshold values c (from 3 to 6), as shown in Fig. 2, the average MCA for $c \leq 4$ is qualitatively low in the resulting subgraph. Here the node failure is 25%. The MCA for $c = 6$ roughly coincides with the largest connected component of the underlying graph. In other words, the result is very similar to the algorithm without node failure for the choice of c.

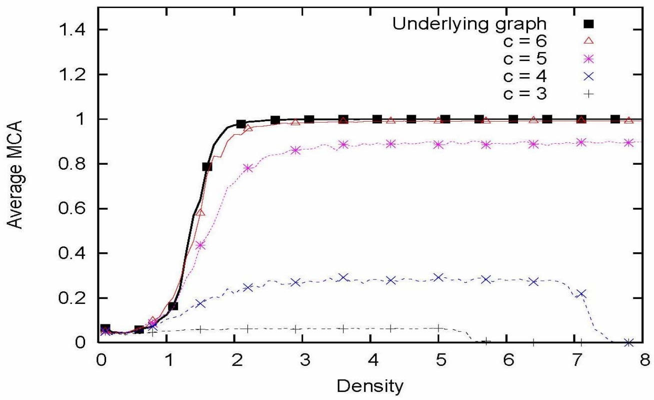

Figure 2: Average MCA as a function of density, averaged over 20 trials, for A = 625 when run with various c and the modified version of Naps that compensates for failed nodes by randomly selecting a neighbor node.

However, when $c \leq 4$ and λ is sufficiently large, the MCA drops to zero. That means no node is in or has an edge to the largest component of the waking subgraph. Because c is too small and λ is large enough, the duration of waking should be shorter. If a node fails and its selected neighbor to compensate is sleeping, the network may be disconnected. Nevertheless there is no impact on the threshold selection; the details may be a future consideration. Let us consider the number of nodes that are awake at any given time. Given any node v, like [5],

$$\begin{aligned}
&\Pr[\text{node } v \text{ awake}] \\
&\quad = \Pr[\text{node } v \text{ awake and node } v \text{ alive}] + \\
&\qquad\quad \Pr[\text{node } v \text{ awake and node } v \text{ dead}] \\
&\quad = \Pr[\text{node } v \text{ awake and node } v \text{ alive}] \\
&\quad = \Pr[\text{node } v \text{ awake} \mid \text{node } v \text{ alive}] * \\
&\qquad\quad \Pr[\text{node } v \text{ alive}] \\
&\quad = \{\Pr[\deg(v) < c] + c/(\pi\lambda) * \Pr[\deg(v) > c]\} * 3/4
\end{aligned}$$

This indicates that the probability of any node that is awake at any given time is larger than and trends to $3c/4(\pi\lambda)$ as c is constant and $\lambda \rightarrow \infty$. Here ¾ represents the rate of nodes that did not fail (since the node failure rate is 25%). Since the uniform distribution of our simulations approaches the Poisson distribution as $A \rightarrow \infty$, this means that the fraction of waking nodes should be slightly larger than $3c/4(\pi\lambda)$.

V. SIMULATION RESULTS

The Naps algorithm did not make use of an existing simulation tool but instead was tested with a C program written by the authors of the Naps algorithm. The code used for the simulations in this paper was obtained from one of the authors of the original Naps algorithm (P. Brighton Godfrey) and used with his permission. The Naps test code will establish a WSN with nodes placed uniformly at random in a square of a specified area. The code then executes the Naps algorithm. The node failure rate is set as 25%.

Simulations used to test results will vary numerous parameters. The primary variations are in the area of the network, the density of nodes in the network, the number of nodes in the network, and the neighbor threshold (c). When analyzing results for the average MCA, twenty trials are run through the simulation code and individual results are averaged. The simulation is run for five trials with results averaged for the analysis of the network lifetime. The Naps algorithm is simulated on a new random graph for each trial run. The results analyzed for this paper use the same starting values as the original Naps paper [5] in order to properly compare the algorithms.

In sensor networks, the goal is to achieve an MCA of one which will result in a connected dominating graph. An MCA of less than one is not desirable since this indicates a disconnected graph and some inaccessible nodes. According to the results of the original Naps algorithm [5], an area of 625 would require a minimum density of three and a neighbor threshold (c) greater than six. A neighbor threshold less than six will never achieve the goal MCA of one for an area of 625. For smaller networks (lower area) the neighbor threshold may need to be larger than six.

Several simulations were run to compare the energy conservation of the original and modified algorithms. With simulations that compare time, each node is given enough energy to survive and remain awake for time $100T$. Recall that T is an algorithm parameter representing a time period. The sleeping-to-waking power ratio r is parameterized so a node can sleep for $100/r$ time periods [5]. For Figures 3-5, ORIGSAFE corresponds to the case that the original Naps algorithm runs in a safe environment without node failures; ORIGNF is the case that the original Naps algorithm runs in a node-failure prone environment, i.e. with 25% node failure rate; NFIR corresponds to the modified algorithm that compensates any failed node by increasing the power in a randomly selected neighbor node; and NFIM compensates for failed nodes by increasing the power in the neighbor node with the most remaining power.

The first behavior analyzed examines the average MCA for a variety of densities (from 0.1 to 8.0 in increments of 0.1) in an area of 625. The results of the four versions of the Naps algorithm are shown in Fig. 3. When the algorithm introduces failed nodes, a slightly larger density is required to achieve the desired MCA of one due to some nodes failing thus resulting in a lower density when considering only alive nodes. The algorithm that compensates for failed nodes by randomly selecting a neighbor node and increasing its power, thus increasing its communication range, is able to achieve the same results as the original Naps algorithm. The results for this enhanced version is better than the version that selects the node with the most energy remaining to increase power, indicating that the selection of the compensating node may not be critical.

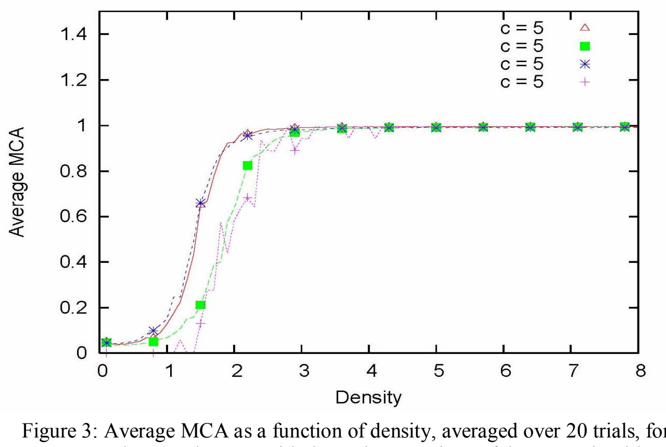

Figure 3: Average MCA as a function of density, averaged over 20 trials, for A = 625 and c = 6 when run with the various versions of the Naps algorithm.

The next simulation was done with an area of 625, c = 6 (the minimum neighbor threshold) and r = 0.1. The graph shows the MCA of the network over time. "The sharp decline towards the end of the system's lifetime implies that Naps effectively distributes energy consumption among nodes" [5]. Fig. 4 shows a comparison of the four cases in terms of the average MCA over time for a density of 20. The goal is to maintain an MCA of 1 so no part of the network is disconnected. Again, the original algorithm outperforms the modified versions, but does not consider node failures. As expected, when node failures are introduced into the algorithm, the MCA becomes less than 1 more quickly. The two algorithms that compensate for failed nodes perform about the same and both out-perform the algorithm with node failures and no compensation.

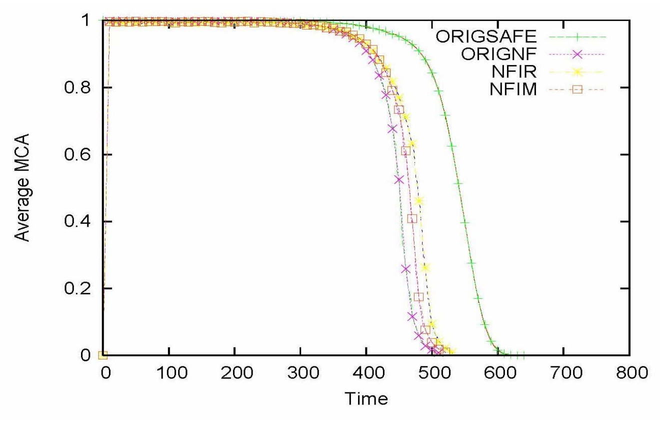

Figure 4: MCA of the Naps algorithm as a function of time for an initial density of 20, averaged over 20 trials.

The last simulation used to compare the different versions of the Naps algorithm examined power-saving results. This was done by examining the network lifetime which is the amount of time the MCA is at least 0.9. The simulation is run for an area of 900 and a variety of densities. In this simulation the sleeping-to-waking power ratio r is also modified. The factor increase in network lifetime from using the Naps algorithm is compared to a network utilizing no topology management. Fig. 5 shows the network lifetime for the four versions of the Naps algorithm, with r = 0.01. Again, the original version does not consider node failures and has a longer network lifetime. As node failures are introduced, the network lifetime drops, with a larger difference as the density increases. The network lifetime increases slightly over the failed node version when power is increased in neighbor nodes to compensate for the failed nodes. This is because the compensating nodes increase their power so their signal will reach more nodes, thus keeping a connected network for a longer period of time. When consider network lifetime, both algorithms that compensate for node failures perform similarly, indicating that compensating node selection may not be critical in a randomly uniformly deployed WSN.

VI. DISCUSSIONS

When a node fails, it is possible for the graph to become disconnected. A disconnected graph in some domains can be disastrous. There are two possible solutions to a disconnected graph. One option is to increase the power output in a neighbor node to increase the signal distance. A longer signal distance will allow the node to reach more nodes and allow the graph to become connected again. The current algorithm either randomly selects a neighbor node of the failed node to increase the power for a stronger signal or

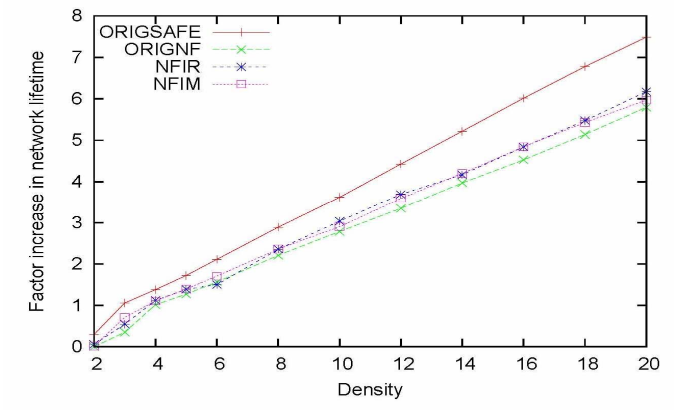

Figure 5: Factor increase in network lifetime vs. density for r = 0.01, averaged over 5 trials for the Naps algorithm. A = 900, c = 6, and a waking node survives for time 100T. [5]

it selects the neighbor node with the most energy remaining. The algorithm can be modified further to use a variety of factors to determine which node should increase power in order to compensate for the failed node. Additional criteria in compensating node selection may show more improvement in network connectivity and lifetime. Also, when a node sleeps, if that node is one with an increase in power, putting it to sleep may result in a disconnected network. More care must be given when allowing nodes to sleep or when a node sleeps; if it is using the increased energy, another node may have to increase its energy.

A second option when a node fails is to introduce an additional node or nodes in the area where the graph is disconnected. The general area can be determined from nodes at the edge of the connected graph adjacent to the disconnected portion. The difficult decision here is how to determine the exact location to introduce the new node or nodes. To find a more exact location would require relative location or positioning information for all nodes as well as some network topology information. A central location must maintain previous network topologies so the failed node can be identified and the ideal location for the additional node(s) determined. A centralized approach is not ideal in many WSNs.

The modified Naps algorithm makes the assumption that all node failures are permanent. In wireless ad hoc sensor networks some node failures may be temporary. An extension of this new algorithm may be to introduce some temporary node failures. If the node failure causes the graph to become disconnected, this must be addressed immediately so the length of the failure does not matter. A permanent node failure would always require action in order to ensure a connected graph with the desired density. If the node failure does not create a disconnected graph, a temporary node failure may indicate that no immediate action is required; the short-term lower density may not be an issue as long as the network remains connected.

Sensor nodes do not always fail cleanly. The two conditions stated above are when sensor nodes fail-fast and fail-silent, assumptions that we make throughout this paper. Deploying sensor nodes en masse, we have discovered there are two other forms of sensor node failure. If the sensor node does not fail-quiet, then a degraded node may broadcast a signal in an erratic manner, causing interference and preventing other nodes from communicating. The implication of such a node means that it affects several nodes in a graph. The use of additional nodes will not be able to mitigate the effect. If the graph is still fully-connected, a new route may have to navigate around the affected area.

If the node does not fail-fast, such as when the battery is weak on a sensor node, it may send a signal that is only barely detected. This is different from a situation where the sensor node is down or unavailable. If a node is down, the other nodes in the network will stop attempting to connect to it, though they may periodically attempt contact as the problem may only be transient and recoverable. When a node sends a weak signal, the other nodes will retry in an attempt to receive a signal. A degraded signal may still provide sporadic data fooling others into attempting to overcome any interference that may be causing the erratic signal. In a network where nodes may vary its power output to attempt to overcome a disconnected network, the weak signal can be worse than no signal at all. A WSN will not continue to contact a node that is down, but may spend a considerable amount of resource to communicate with a node that cannot contribute.

VII. CONCLUSIONS

Wireless sensor networks may become a critical technology used to overcome some current problems in various domains. Topology management and energy conservation are important issues in WSNs. Some energy conservation is obtained by using the Naps algorithm to allow some nodes to sleep while still maintaining a connected network. There are periods in the lifetime of the WSN where the network may not be connected.

The Naps algorithm needs modifications by addressing random node failures to better simulate practical applications of wireless sensor networks. As was expected, when node failure is introduced into the WSN the lifetime of the network decreases and the network connectivity is affected. However, the decrease in network lifetime and connectivity is negligible. This indicates that node failure does not significantly impact sensor network connectivity or network lifetime in the case of uniformly and randomly deployed wireless sensor networks. The node failure introduced is random failure; these results will change and may impact the overall connectivity and lifetime when the node failures are localized and clustered.

The Naps algorithm was modified in order to compensate for a failed node to hopefully increase the network lifetime and offer better performance. There were two ways to compensate when a node fails. A neighbor node can be selected at random and the energy in this random node increased in order to increase the communication range of the node. Another alternative is to be more selective in determining which node to increase the energy to offer compensation for a failed node and pick the neighbor node which has the most energy remaining. Both of these enhanced algorithms performed comparably in the simulations. This shows that either enhancement would help to increase network connectivity and lifetime in the presence of node failures. One simulation result showed that selecting a random node may be the better option, which also allows the algorithm to be run in a distributed manner.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A survey on sensor networks", *IEEE Communications Magazine*, August 2002.

[2] L. Bao and J. J. Garcia-Luna-Aceves, "Topology management in ad hoc networks", *MobiHoc*'03, Annapolis, Maryland, ACM, June 1-3, 2003.

[3] C.-Y. Chong and S. P. Kumar, "Sensor networks: evolution, opportunities, and ahallenges", in *Proceedings of the IEEE*, Vol. 91, No. 8, August 2003.

[4] D. Culler, D. Estrin and M. Srivastava, "Overview of sensor networks", *Computer*, August 2004.

[5] B. P. Godfrey and D. Ratajczak, "Naps: scalable, robust topology management in wireless ad hoc networks", *ISPN* '04, Berkeley, CA, ACM, April 26-27 2004.

[6] J. Hill, M. Horton, R. Kling and L. Krishnamurthy, "The platforms enabling wireless sensor networks", *Communications of the ACM*, Vol. 47, No. 6, June 2004.

[7] X. Jiang, N. Y. Chen, J. I. Hong, K. Wang, L. A. Takayama and J. A. Landay, "Siren: context-aware computing for firefighting", in *Proceedings of Second International Conference on Pervasive Computing*., Vienna, Austria, April 21-23 2004.

[8] X. Jiang, J. I. Hong, L. A. Takayama, and J. Landay, "Ubiquitous computing for firefighters: field studies and prototypes of large displays for incident command", *CHI Letters* [*Human Factors in Computing Systems*: *CHI* 2004], 2004.

[9] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors", *Communications of the ACM*, Vol. 43, No. 5, May 2000.

[10] V. Raghunathan, C. Schurgers, S. Park and M. B. Srivaastava, "Energy-aware wireless microsensor networks", *IEEE Signal Processing Magazine*, March 2002.

[11] C. Schurgers, V. Tsiatsis, S. Ganeriwal and M. Srivastava, "Optimizing sensor networks in the energy-latency-density design space", *IEEE Transactions on Mobile Computing*, Vol. 1, No. 1, January-March 2002.

[12] C. Schurgers, V. Tsiatsis, S. Ganeriwal and M. Srivastava, "Topology management for sensor networks: exploiting latency and density", *MOBIHOC*'02, Lausanne, Switzerland, ACM, June 9-11 2002.

[13] J. A. Stankovic, T. F. Abdelzaher, C. Lu, Lui Sha and J. C. Hou, "Real-time communication and coordination in embedded sensor networks", in *Proceedings of the IEEE*, Vol. 01, No. 7, July 2003.

[14] M. Tubaishat and S. Madria, "Sensor networks: an overview", *IEEE Potentials*, April/May 2003.

[15] M. Augusto, M. Vieira, C. N. Coelho, Jr., D. C. da Silva J., and J. M da Mata, "Survey on wireless sensor network devices", in *Proceedings of the 9th IEEE International Conference on Emerging Technologies and Factory Automation* (*ETFA*'03), 2003.

[16] Y. Xu, S. Bien, Y. Mori, J. Heidemann and D. Estrin, "Topology control protocols to conserve energy in wireless ad hoc networks", *Technical Report* 6, *Center for Embedded Networked Computing*, University of California, Los Angeles, January 2003.