2-2013

Fault-tolerant Coverage in Dense Wireless Sensor Networks

Akshaye Dhawan
Ursinus College, adhawan@ursinus.edu

Magdalena Parks
Ursinus College, maparks@ursinus.edu

Fault-tolerant Coverage in Dense Wireless Sensor Networks

Akshaye Dhawan and Magdalena Parks
Department of Mathematics and Computer Science, Ursinus College, 610 E Main Street, Collegeville, PA, USA
{*adhawan, maparks*}*@ursinus.edu*

Keywords: Wireless Sensor Networks:Coverage:Fault-tolerance

Abstract: In this paper, we present methods to detect and recover from sensor failure in dense wireless sensor networks. In order to extend the lifetime of a sensor network while maintaining coverage, a minimal subset of the deployed sensors are kept active while the other sensors can enter a low power sleep state. Several distributed algorithms for coverage have been proposed in the literature. Faults are of particular concern in coverage algorithms since sensors go into a sleep state in order to conserve battery until woken up by active sensors. If these active sensors were to fail, this could lead to lapses in coverage that are unacceptable in critical applications. Also, most algorithms in the literature rely on an active sensor that is about to run out of battery waking up its neighbors to trigger a reshuffle in the network. However, this would not work in the case of unexpected failures since a sensor cannot predict the occurrence of such an event. We present detection and recovery from sensor failure in dense networks. Our algorithms exploit the density in the recovery scheme to improve coverage by 4-12% in the event of random failures. This fault tolerance comes at a small cost to the network lifetime with observed lifetime being reduced by 6-10% in our simulation studies.

1 INTRODUCTION

Wireless Sensor Networks (WSNs) have attracted a lot of research interest due to their applicability in security, monitoring, disaster relief and environmental applications. WSNs consist of a number of low-cost sensors scattered in a geographical area of interest and connected by a wireless RF interface. Sensors gather information about the monitored area and send this information to an external node known as the base station. The radio on board these sensor nodes has a limited range and allows the node to transmit over short distances. In most deployment scenarios, it is extremely expensive for each node to communicate directly to the sink and hence, the model of communication is to transmit over short distances to other peers.

In order to keep their cost low, sensors are equipped with limited energy and computational resources. The energy supply is typically in the form of a battery and once the battery is exhausted, the sensor is considered to be dead. Sensor nodes are also limited in terms of memory and processing capabilities. Hence, harnessing the potential of these networks involves tackling a myriad of different issues from algorithms for network operation, programming models, architecture and hardware to more traditional networking issues. For a more detailed survey on the various computational research aspects of Wireless Sensor Networks, see the survey papers (Akyildiz et al., 2002; Chong and Kumar, 2003; Iyengar and Brooks, 2004a; Schmid and Wattenhofer, 2006; Römer and Mattern, 2004).

In this paper, we examine the problem of sensor coverage. Many intended applications of Wireless Sensor Networks involve having the network monitor a region or a set of targets. To ensure that the area or targets of interest can be covered, sensors are usually deployed in large numbers by randomly dropping them in this region. Deployment is usually done by flying an aircraft over the region and air dropping the sensors. Since the cost of deployment far exceeds the cost of individual sensors, many more sensors are dropped than needed to minimally cover the region. This leads to a dense network and gives rise to an overlap in the monitoring regions of individual sensors. A simplistic approach to meet the coverage objective would be to turn on all sensors after deployment. But this needlessly reduces the lifetime of the network since the overlap between monitoring regions implies that not all sensors need to be on at the same time. This can also lead to a very lossy network with several collisions happening in the medium access control (MAC) layer due to the den-

sity of nodes. In order to extend the lifetime of a sensor network while maintaining coverage, a minimal subset of the deployed sensors are kept active while the other sensors can enter a low power sleep state. Several distributed algorithms for coverage have been proposed in the literature (Lu and Suda, 2003; Prasad and Dhawan, 2007; Dhawan and Prasad, 2009; Cardei and Du, 2005; Kumar et al., 2004; Cardei et al., 2005).

Faults are of particular concern in coverage algorithms since sensors go into a sleep state in order to conserve battery until woken up by active sensors. If these active sensors were to fail, this could lead to lapses in coverage that are unacceptable in critical applications. Also, most algorithms in the literature rely on an active sensor that is about to run out of battery waking up its neighbors to trigger a reshuffle in the network. However, this would not work in the case of unexpected failures since a sensor cannot predict the occurrence of such an event. In a denser network, more sensors should be available to compensate for the failing sensor. In this paper, we exploit the density to recover from failure. This robustness comes at a small cost to the network lifetime.

The remainder of this paper is as follows. In Section 2, we present related work in the literature on both coverage and fault-tolerance in Wireless Sensor Networks. In Section 3 we discuss our failure recovery scheme. Section 4 presents our simulation set-up and results. Finally, we conclude in Section 5.

2 RELATED WORK

Coverage Algorithms: In this section, we briefly survey existing approaches to maximizing the lifetime of sensor networks, while meeting certain coverage objectives. (Cardei and Wu, 2006) gives a more detailed survey on the various coverage problems and the scheduling mechanisms they use. (Sahni and Xu, 2004) also surveys the coverage problem along with other algorithmic problems relevant to sensor networks.

The coverage problem has been shown to be NP-complete in (Abrams et al., 2004; Cardei and Du, 2005). The approach taken in the literature has evolved around algorithmic techniques commonly used to work with NP-complete problems. This is indicated by the use of solutions to related problems such as that of finding domatic partitions, coloring, set covers, etc, as applied to coverage. Initial approaches to the problem in (Slijepcevic, 2001; Cardei and Du, 2005; Abrams et al., 2004) considered the problem of finding the maximum number of *disjoint* covers. A number of distributed heuristics are presented in (Lu and Suda, 2003; Prasad and Dhawan, 2007; Dhawan and Prasad, 2009; Cardei and Du, 2005; Kumar et al., 2004; Cardei et al., 2005). We only discuss one specific algorithm below in the interest of space. Our algorithm improves upon this algorithm by adding failure recovery to it.

Face-based Coverage (Berman et al., 2004) present a coverage algorithm that models the network as a planar graph and then uses the faces of this graph to determine which sensors should be active in order to cover the area of interest. Each face represents a geographical area that can be sensed by one or more sensors. These faces are found by tracing the outer edge of the range of each sensor. The intersection points of all of these ranges are found and placed as nodes on a graph, with the lines along the edge of each sensor becoming edges in the graph. A sensor then simply needs to keep track of each of its faces and which neighboring sensors are capable of covering those faces.

Each sensor can be in one of various states at any given time. For every sensor, the basic states include *Active, Idle, Permanent, Vulnerable* and *Terminated*. Active sensors sense data for the faces they cover and transmit that data to the base station. Idle sensors neither sense data nor transmit data but instead eneter a low power sleep state. Permanent sensors, are a variation on active sensors and they also sense data and transmit data, but are distinguished from active sensors because they have at least one face for which they are the only covering sensor. Vulnerable sensors are sensors that are still in the process of deciding a state. Finally, a Terminated state refers to sensors that are no longer capable of contributing to the network.

Sensors periodically go through state transitions, where they will change from one state to another. The algorithm is broken into rounds and at the beginning of each round, if an idle or active sensor has a vulnerable neighbor, it will become vulnerable. This allows a sensor to trigger a change in the cover set by becoming vulnerable. This is called a reshuffle. Vulnerable sensors become idle if all of its faces are covered by either an active sensor, a permanent sensor, or a vulnerable sensor with a larger power supply. Vulnerable sensors become active if they have a face not covered by an active or vulnerable sensor. If a sensor's battery level goes low enough to only last a few more rounds, it will transition to vulnerable to trigger a reshuffle so that it may be replaced. Finally, any sensor becomes terminated if its battery level goes to zero, and any non-terminated sensor becomes permanent if it has a face that is covered by no other non-terminated sensor. See (Berman et al., 2004) for more details.

Fault-tolerance: Fault Tolerance has been extensively studied in the broader context of distributing computing (Pradhan, 1996), and also in the context of Wireless Sensor Networks. (Iyengar and Brooks, 2004b) examines the connection between classical fault tolerance techniques and sensor networks and provide two case studies. (Gupta and Younis, 2003) examines fault tolerance in clustering, but only looks at heterogeneous sensor networks where clustering is performed by special high energy gateway nodes that are much more powerful than regular sensor nodes. (Kuhn, 2006) examines fault tolerant clustering by formulating clustering as the k-fold dominating set problem. They give a probabilistic algorithm for a unit disk graph network. The authors go on to present fast approximation algorithms for the special cases of graphs with low arboricity in (Lenzen and Wattenhofer, 2010). In (Raj and Ramesh, 2008), the authors consider clustering in a sensor network to be deployed in landslide detection applications. However, they define a fault as a sensor that gives incorrect values as opposed to a failed sensor.

3 FAILURE RECOVERY

We started with the basic face-based coverage algorithm of (Berman et al., 2004) and looked at ways to detect sensor failure in dense deployment scenarios. Failures can cause this algorithm to break down since the algorithm depends on sensors being able to trigger a reshuffle before going to the terminated state. In the case of sensor failure, this may not happen. A failing sensor does not know in advance that it is going to fail, so it has no opportunity to trigger a reshuffle. When an active or permanent sensor fails, the faces it was responsible for will go uncovered until the next reshuffle happens, which may be a while. An ideal fault tolerant algorithm would be able to immediately activate enough of those sensors to negate the effects of the sensor failure.

In the case of dense deployments, the degree of the graph representing the network (i.e., the number of neighbors a given sensor has) is larger than that for less dense deployments. This leads to a network organization wherein, as the density increases, the number of active/permanent sensors within communication range of each other also likely increases. This coupled with the inherent broadcast property of the wireless channel would allow neighboring active/permanent sensors to detect the failure of other active sensors since upon failing these sensors would cease broadcasting. We modified the base algorithm to detect failure by having each active sensor construct a list of 1-hop neighbors that were either active or permanent. The sensor then listened for broadcast activity from these neighboring nodes. If at the expiration of a timer, no activity was detected for one of these neighboring, the sensor monitoring for activity would trigger a reshuffle for the network by going to a vulnerable state.

Without adding this mechanism for fault tolerance, the faces the failed sensor was responsible for may go uncovered for a length of time depending on the frequency of reshuffles in the system. If the reshuffles are frequent, the sensors will take over the failed sensor's faces more quickly, but will not be able to immediately recover from the failure. Faces that the failed sensor was responsible for would be uncovered until all neighboring sensors became vulnerable for the next reshuffle. In most cases, there are some active or permanent neighboring sensors, regardless of the number of vulnerable neighboring sensors. Transitioning these sensors to vulnerable would help ensure that recovery for the failed sensor happens more quickly. The biggest impact on the network's coverage after a sensor failure is immediately after the sensor fails. Sensors responding to the failure would be able to respond immediately to cover any faces that they can reach and the failed sensor was responsible for. This allows the initial impact of the failure to be minimized. It is worth mentioning that there will always be at least one face that cannot be immediately recovered in this way, since the failing sensor was active or permanent because no other sensor was covering one of its faces.

4 SIMULATION RESULTS

In order to study the performance of the proposed fault-tolerance model, we have conducted some preliminary simulations. We implement the face-based algorithm as described in (Berman et al., 2004) in $C++$ and then modify this implementation to include the failure recovery process described in Section 3.

For our simulation setup, we randomly create dense networks of sensors. We varied the number of sensors in a $30x30$ area. We ran the simulation with 25, 50, 75, and 100 sensors scattered randomly in this area. For any given network size, we created five different randomly generated configurations. Additionally, each sensor was given a random battery level between 10 and 20 units and a communication range of 10 m. We used a reshuffle threshold of 2. The reshuffle threshold is the maximum drop in battery level before triggering a reshuffle. In order to inject failure into the simulation, we randomly determined

Table 1: Failing sensor statistics.

Sensor count	Type	Active neighbors	Vulnerable neighbors	Faces covered
25	basic	5.810	1.776	19.724
25	fault tolerant	5.879	7.690	17.672
50	basic	5.771	8.593	49.008
50	fault tolerant	5.372	15.947	39.796
75	basic	5.536	15.762	64.548
75	fault tolerant	5.205	22.217	58.843
100	basic	7.421	18.345	66.817
100	fault tolerant	6.528	27.549	53.724

how many sensors should fail each round based on a failure chance of 5%, up to a maximum of 5% of the total sensors failing in one round. For every sensor that failed, we made the sensor enter a terminated stated immediately. In the fault tolerant version, any non-terminated sensors immediately became vulnerable. We ran this both with the normal version and the fault tolerant version, and ensured that the same sensors failed at the same time for both the normal version and the fault tolerant version, in order to better compare the two. The simulation was terminated once the coverage dropped to below 80% of the maximum possible coverage from the beginning of the simulation.

Our first goal was to track the number of active sensors each failing sensor had as a neighbor. At the time of each sensor's failure, we recorded the number of active neighbors, vulnerable neighbors, the number of faces the sensor was covering, and the number of faces with no other non-terminated sensor. The average over each of the five configurations can be seen in Table 1. These active neighboring sensors become vulnerable in reaction to the detection of the sensor failure, triggering a reshuffle, and are very important in the fault tolerant version. By design all these networks are extremely dense and hence we expect to see multiple active neighbors for each failing sensor.

The implementation of the failure recovery algorithm also led to an improvement of coverage as expected since reshuffles triggered by the detection of a failed sensor improved the amount of coverage. We tracked the total time for which the network was 100% covered and the fault tolerant version covered between 4-12% more area than the basic algorithm. This improved coverage came at a slight cost to the network lifetime. We tracked the time by which network coverage dropped below 80% and used this to terminate the simulation. Figure 1 shows this data for the different network sizes and for sensors with different ranges. As can be seen from the figure, the improved coverage gained through fault-tolerance comes at a 6-10% decrease in lifetime in the worst case (which happens at the higher range). At the lower ranges, our fault tolerant algorithm provides similar network lifetime to the basic algorithm.

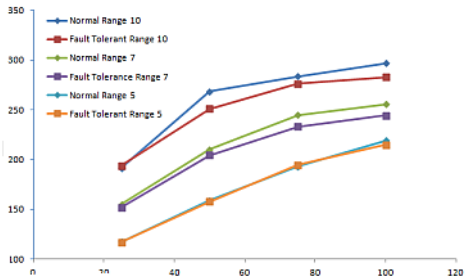

Figure 1: Lifetime of the Network in rounds

5 FUTURE WORK AND CONCLUSIONS

In this paper, we present a fault-tolerance recovery scheme for coverage in wireless sensor networks that exploits the density of the network to detect and recover from failures. The algorithm improved coverage by 4-12% at the cost of reducing lifetime by 6-10%. Future work on fault-tolerant coverage solutions includes exploring the use of a heterogeneous collection of sensors, where some sensors have significantly more battery power than the rest. We are also exploring a modified algorithm to determine which sensor is best suited to cover certain faces based on how many faces it would be responsible for, instead of basing the decision entirely on battery levels.

REFERENCES

Abrams, Z., Goel, A., and Plotkin, S. (26-27 April 2004). Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*, pages 424–432.

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). A survey on sensor networks. *IEEE Commun. Mag.*, pages 102–114.

Berman, P., Calinescu, G., Shah, C., and Zelikovsky, A. (2004). Power efficient monitoring management in sensor networks. In *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, volume 4, pages 2329–2334. IEEE.

Cardei, M. and Du, D.-Z. (May 2005). Improving wireless sensor network lifetime through power aware organization. *Wireless Networks*, 11:333–340(8).

Cardei, M., Thai, M., Li, Y., and Wu, W. (2005). Energy-efficient target coverage in wireless sensor networks. In *IEEE Infocom, 2005*.

Cardei, M. and Wu, J. (2006). Energy-efficient coverage problems in wireless ad hoc sensor networks. *Computer Communications*, 29(4):413–420.

Chong, C.-Y. and Kumar, S. (2003). Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256.

Dhawan, A. and Prasad, S. K. (2009). A distributed algorithmic framework for coverage problems in wireless sensor networks. *International Journal of Parallel, Emergent and Distributed Systems(IJPEDS)*.

Gupta, G. and Younis, M. (2003). Fault-tolerant clustering of wireless sensor networks. *IEEE Wireless Communications and Networking, 2003*, pages 1579–1584.

Iyengar, S. S. and Brooks, R. (2004a). Computing and communications in distributed sensor networks. *Special Issue, Jr. of Parallel and Distributed Computing*, 64(7).

Iyengar, S. S. and Brooks, R. (2004b). Computing and communications in distributed sensor networks.

Kuhn, F. (2006). Fault-tolerant clustering in ad hoc and sensor networks. In *In 26th International Conference on Distributed Computing Systems (ICDCS*.

Kumar, S., Lai, T. H., and Balogh, J. (2004). On k-coverage in a mostly sleeping sensor network. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 144–158, New York, NY, USA. ACM.

Lenzen, C. and Wattenhofer, R. (2010). Minimum dominating set approximation in graphs of bounded arboricity. In *Distributed Computing*, volume 6343 of *Lecture Notes in Computer Science*, pages 510–524.

Lu, J. and Suda, T. (20-21 Oct. 2003). Coverage-aware self-scheduling in sensor networks. *Computer Communications, 2003. CCW 2003. Proceedings. 2003 IEEE 18th Annual Workshop on*, pages 117–123.

Pradhan, D. K., editor (1996). *Fault-tolerant computer system design*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Prasad, S. K. and Dhawan, A. (2007). Distributed algorithms for lifetime of wireless sensor networks based on dependencies among cover sets. In *HiPC '07: Proceedings of the 14th International Conferenmce on High Performance Computing*. Springer.

Raj, R. and Ramesh, M. (2008). Fault tolerant clustering approaches in wireless sensor network for landslide area monitoring. In *Proc. of the 2008 International Conference on Wireless Networks (ICWN'08)*, pages 107–113.

Römer, K. and Mattern, F. (2004). The design space of wireless sensor networks. *IEEE Wireless Communications*, 11(6):54–61.

Sahni, S. and Xu, X. (2004). Algorithms for wireless sensor networks. *Intl. Jr. on Distr. Sensor Networks*.

Schmid, S. and Wattenhofer, R. (2006). Algorithmic models for sensor networks. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 11 pp.–.

Slijepcevic, S.; Potkonjak, M. (2001). Power efficient organization of wireless sensor networks. *Communications, 2001. ICC 2001. IEEE International Conference on*, 2:472–476 vol.2.