Fault Tolerance in Wireless Sensor Networks

Sushruta Mishra   **Lambodar Jena**   **Aarti Pradhan**
Dept. of CSE, GEC, BBSR   *Dept. of CSE, GEC, BBSR*   *Dept. of CSE, GEC,BBSR*
India   India   Indi

Abstract- Reliability is an important issue in a network and when the network is energy constrained as in a wireless network then it becomes a high priority to ensure that this network is highly reliable. But this reliability is severely affected by errors and faults that occur due to multiple reasons such as Hardware malfunctioning, bugs in software, environmental hazards etc. Thus a sensor network should be fault resistant so that it can deal with these erroneous conditions effectively. This paper tries to cover issues related to fault detection and recovery mechanisms in wireless sensor networks in both theoretical and application oriented research.

Keywords- Fault Tolerant, Wireless sensor network, Multi- hop Routing, Replication, Aggregation

I.   INTRODUCTION

Advances in embedded systems technology have made it possible to build wireless sensor nodes, which are small devices with limited memory, processing power, and energy resources [1]. Due to the low cost associated to these devices, it is possible to conceive the deployment of large scale wireless sensor networks (WSN) with potentially thousands of nodes [7]. In critical environment a high degree of dependability is required. In order to be considered dependable, WSNs must offer characteristics such as: reliability, availability and maintainability. Availability to a large extend depends on fault tolerance to keep the system working as expected. Availability on the service level means that the service delivered by a WSN (or part of it) is not affected by failures and faults in underlying components such as single nodes or node subsystems. In WSNs, the failure of such components is almost unavoidable. Most detection and recovery techniques therefore aim at reducing MTTR (the amount of time required for detecting and recovering from a failure) as much as possible. In these WSN deployments, it is common to havea node providing functionality to its neighbors. Multi hop routing is a simple example of such a service,where nodes forward messages on behalf of each other.Often, nodes assume dedicated roles such as cluster head, which implies the responsibility for certain tasks. For example, a cluster head could aggregate sensor data beforeit is forwarded to a base station, thereby saving energy.Nodes with stronger hardware capabilities can performoperations for other nodes that would either have to spenda significant amount of energy or would not be capableof performing these operations.These services, however, may fail due to various reasons, including radio interference, de-synchronization ,battery exhaustion, or dislocation. Such failures arecaused by software and hardware faults, environmentalconditions, malicious behavior, or bad timing of a legitimateaction. In general, the consequence of such an eventis that a node becomes unreachable or violates certainconditions that are essential for providing a service, forexample by moving to a different location, the node canno further provide sensor data about its former location.In some cases, a failure caused by a simple softwarebug can be propagated to become a massive failureof the sensor network. This results in application trialsfailing completely and is not acceptable in safety criticalapplications. Hence, our aim is to clarify the requirementsfor maintaining high level availability in WSNs, and toinvestigate the tools and mechanisms utilized in WSNresearch and engineering for fault detection and recovery.

        In this paper, we have made a detail analysis of various types of faults in a sensor network and have focused on enhancements ofservice availability in WSNs through the use of fault tolerance techniques.We present a survey of approachesto fault detection and recovery techniques in WSNs. Weprovide taxonomy of faults and classify the investigatedapproaches according to their ability to detect and toleratefaults.

II.   FAILURES IN WIRELESS SENSOR NETWORKS

To comprehend fault tolerance mechanisms, it is important to point out the difference between faults, errors, and failures. Various definitions of these terms have been used [3], [40]. This paper refers to the definition given in [40]:

A fault is any kind of defect that leads to an error.

An error corresponds to an incorrect (undefined) system state. Such a state may lead to a failure.

A failure is the (observable) manifestation of an error, which occurs when theSystem deviates from its specification and cannot deliver its intended functionality. Figure 2 illustrates the difference between fault, error, and failure. A sensor service running on node A is expected to periodically send the measurements of its sensors to an aggregation service running on node B. However, node A suffers an impact causing a loose connection with one of its sensors. Since the code implementing node

A's service is not designed to detect and overcome such situations, an erroneous state is reached when the sensor service tries to acquire data from the sensor. Due to this state, the service does not send sensor data to the aggregation service within the specified time interval. This results in a crash or omission failure of node A observed by node B.In the scenario explained above, the fault is the loose connection of the sensor.
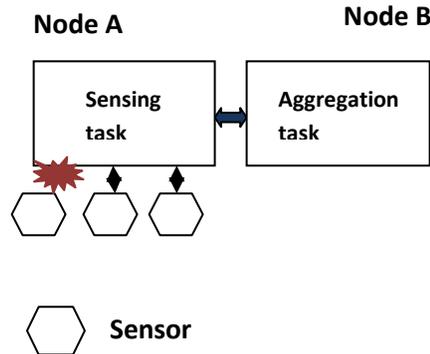
Node A   **Node B**

```
Sensing        Aggregation
task           task
```


Sensor

Fig. 1 A loosely connected sensor causing Failure

The error is the state of the service after trying to read the sensor data and the failure occurs when the application does not send the sensor data within the specified time interval. To provide resilience in faulty situations two main actions must be performed:

Fault detection. To provide any countermeasures, the first step a system must perform is to detect that a specific functionality is or will be faulty.

Fault Recovery. After the system has detected a fault, the next step is to prevent or recover from it. Themain technique to achieve this goal is to replicate the components of the system that are vital for its correct operation.

III.   Source of Faults in Real WSN Applications

Ultimately, if the application in the back-end which presents the WSN data to the users suffers a fault due to some software bug or hardware failure the entire system is considered faulty. In this paper, however, we will concentrate on faults that can happen in the sensor nodes up to the sink.

1) Node Faults: Nodes have several hardware and software components that can produce malfunctions. For example, the enclosure can suffer impacts and expose the hardware of the sensor node to the extreme conditions of the environment. In [27][41][39] due to stress from the environment and inadequate enclosures, the sensor nodes were exposed to direct contact with water causing short circuits. The report of a large-scale deployment in a potatoes field [20] indicated that the antennas from the nodes were quite fragile and would become loose when inserting the node into the packaging. When the battery of a node reaches a certain stage, sensor readings may become incorrect. This has been observed in [43] where many outlier readings were generated in the network caused by imminent battery failure. As demonstrated in Figure 3, hardware failures will generally lead to software failure. A Data Acquisitionapplication will not perform properly if the underlying sensors are providing incorrect readings. Nevertheless,some hardware failures do not affect all the services in asensor node. In the example discussed, although the nodecannot be used to provide correct sensor readings it still can be used to route packages in the sensor network.Software bugs are a common source of errors in WSNs. In [44], the researchers reported that a software bug caused the longest continuous network outage taking the system offline for three days until the nodes could be reprogrammed manually. Organizing a network in clusters is an approach used in many applications, for example to extend the lifetime of the network [15]. A small number of nodes are selected to become clusterheads. They are responsible for coordinating the nodes in their clusters, for instance bycollecting data from them and forwarding it to the base station. In case that a clusterhead fails, no messages of its cluster will be forwarded to the base station any longer. The clusterhead can also intentionally or due to software bugs forward incorrect information. Depending on the application case, the impact of such a failure can vary from quality degradation of measurements to alarm messages not being delivered to a back-end system. While forwarding messages, nodes can aggregate data from multiple other nodes in order to reduce the amount of data sent to the base station. One common simple approach is to calculate the average of correlated measured values such as temperature, humidity and pressure, sending only one message to the back-end. If a node generates incorrect data, the data aggregation results can suffer deviations from the real value. Also, if a node responsible for generating the aggregated data is subject to a value failure, the base station will receive incorrect information of an entire region of the network.

2) Network Faults: Routing is one of the fundamental building blocks in a WSN. It is essential for collecting sensor data, distributing software and configuration updates, and for coordination among nodes. Additionally, there may be application-specific routing protocols required, for example for tracking and "following" moving objects. Faults on the routing layer can lead to dropped or misguided messages, or unacceptable delays. In WSNs, communication links between nodes are highly

volatile. WSNs not always yield the same delivery rate of messages in field trials as in lab trials. For instance, in [38] a delivery rate of only 58% of the messages was observed, in [36] the instability of the links between nodeslead to constant changes in the routing paths. In several scenarios of sensor networks nodes have a certain degree of mobility. In a glacial expedition [27] the experiment assumed a one hop network. The connection of the nodes to the sink was calibrated during deployment with a reliable link connection. Nevertheless, after some time the nodes moved to a different location where the node was unreachable resulting in complete loss of data from this node. Radio interference can also cause the link between nodes to become faulty. For instance, in agricultural fields the placement of the nodes must be carefully planned to take into consideration that when plants start growing the link range is considerably reduced, as discussed in [42]. Another source of link failure is the collision of messages. In [39] researchers observed a potential for collision of messages of nodes in close proximity due to a phase change and overlap. In other situations, however, nodes may have perfect link connections but the messages are not delivered to their destination due to path errors. A software bug in the routing layer can generate circular paths or simply deliver messages to the incorrect destination.

3) Sink Faults: On a higher level of the network a device (sink) that collects all the data generated in the network and propagates it to the back-end system is also subject to faults of its components. When this device fails, unless fault tolerant measurements are present, a massive failure of the network happens given that the data from the sensor nodes cannot be accessed. The sink can be deployed in areas where no permanent power supply is present, in such applications batteries together with solar cells are commonly applied [27][20][43] to provide the amount of energy necessary. In the glacial expedition reported in [27], this traditional technique has proven to be inefficient. Although this worked perfectly for other expeditions, in this glacial environment the sink suffered a power failure due to snow covering the solar cells for several days. Network infrastructure is usually also not available in the area where the sink is deployed, therefore alternative solutions such as a satellite connection are used, which cause fluctuations in the back-end network can interface. In [24] researchers indicated that during periods of severe thunderstorm activity the satellite connection becomes unavailable. Finally, the software that stores the data collected from the network, processes it and sends it to the back-end system, is subject to bugs that when present can lead to loss of data within the period where the fault occurred

IV. Failure Classification

As discussed in Section IV-A, several faults could lead to failures in wireless sensor networks: a node could be moved to a different region, causing a link failure; nodes can suffer power failure and stop responding to requests, or they can start sending arbitrary values either intentionally (after a security breach) or due to a malfunction. Here, we classify the failures that a WSN is susceptible to, which are: crash, omission, timing, value and arbitrary. These failures are the observable manifestation of underlying faults presented on Section IV-A.

1) Crash or omission: A failure by omission is determined by a service sporadically not responding to requests. For instance, this could be caused by radio interference that leads to occasional message loss. A crash failure occurs when the service at some point stops responding to any request. An omission degree f can be defined which imposes a limit to the amount of omission failures a node might have before being classified as crashed.

2) Timing: Services might fail due to a timeout in processing a request or by providing data too early. Such timing failures occur when a node responds to a request with the correct value, but the response is received out of the time interval specified by the application. Timing failures will only occur when the application specifies timing constraints.

3) Value: A service is considered having failed due to an incorrect value when the service sends a timely response, however with lack in accuracy of the delivered value. For instance, a service performing aggregation of data generated by other nodes could forward a result value to the base station that does not correctly reflect the input data. Such situations could be caused by malfunctioning software, hardware, corrupt messages, or even malicious nodes generating incorrect data.

4) Arbitrary: Arbitrary failures include all the types of failures that cannot be classified in previously described categories. In [19], Lamport has introduced the Byzantine Generals Problem in the context of distributed systems. Recent work shows how to deal with this problem in the domain of wireless sensor networks [16]. Byzantine failures describe a type of arbitrary failures that are in general caused by a malicious service that not only behaves erroneously, but also fails to behave consistently when interacting with other services and applications. In sensor networks, an aggregation service could start sending both incorrect and correct values to the sink, ora node routing messages could not forward a messagedespite sending an acknowledge back to the sender.

V. FAULT DETECTION TECHNIQUES

The goal of fault detection is to verify that the services being provided are functioning properly, and in some cases to predict if they will continue to function properly in the near future. The simplest way to perform such a task is through visual observation and manual removal of incorrect values. This technique has obvious drawbacks: human interaction leads to errors, it has a high cost and it is not efficient. Hence, we investigated automatic fault detection techniques for WSN. We classified the techniques we investigated according to the parties involved in the process. Through selfdiagnosis the node itself can identify faults in its components. With group detection, several nodes monitor the behavior of another node. Finally, in hierarchical detection the fault detection is performed using a detection tree where a hierarchy is defined for the

identification of failed nodes. Often in a hierarchical detection the detection is shifted to a more powerful device such as the sink.

A. Self-Diagnosis

In many cases, nodes can identify possible failures by performing self-diagnosis. In [14], the authors propose an approach where a node would perform a self-diagnosis based on the measurements of accelerometers to determine if the node suffers from an impact that could lead to hardware malfunctions.

Using a similar approach, nodes could detect when they are being moved to a different location. Another

approach would be to keep track of the identities of the nodes in the neighborhood. A considerable change in the neighborhood could indicate that either the node itself or some of its previous neighbors have been moved. Faults caused by battery exhaustion can be predicted when the hardware allows the measurement of the current battery voltage [2][29]. By analyzing the battery discharge curve and the current discharge rate, an algorithm can determine an estimation of the time to death of the battery. Nodes can also identify that their current connection to surrounding nodes is unreliable by probing the link connection therefore identifying that it is isolated.

B. Group Detection

The detection of services failing due to incorrectly generated values is only possible if a reference valueis available. In [18] and [6], detection mechanisms are proposed to identify faulty sensor nodes. Both algorithms are based on the idea that sensors from the same region should have similar values unless a node is at the boundary of the event-region. The algorithm start by taking measurements of all neighbors of a node and uses the results to calculate the probability of the node being faulty. Another approach proposed in the literature is to let consumer nodes observe whether the service provider is in fact performing the operations that it is supposed to. In [26], a misbehavior detection algorithm to aide the routing layer is proposed. The misbehavior detection mechanism is based on the idea of monitoring thecommunication of the service provider to verify whether messages are forwarded correctly. Focusing on providing a fault-tolerant approach for clusters in WSNs, in [12] it is proposed to support the dynamic recovery of failed gateways (high-energy devices that act as clusterheads). The proposed protocol assumes that a gateway has failed only when no other gateways can communicate with it. The fault detection mechanism is based on constant status updates being exchanged between gateways and further use of a consensus algorithm.

C. Hierarchical Detection

The definition of a detection tree enables a scalable fault detection algorithm in WSN. Memento [33] proposes the usage of the network topology to forward the fault detection results of child nodes to the parent nodes and up to the sink. Each node forwards the status of the child nodes that it is monitoring to its parent node. The parent performs an aggregation (bitwise OR) operation on the results of the child nodes together with its on results and forwards it to the next level. The approach proposed by Memento scales well with the network size, however it consume resources of the network. Shifting the fault detection task to a more powerful device is an alternative that can help to increase the lifetime of the WSN. In [37], the authors propose an algorithm that puts the burden of detecting and tracing failed nodes to the base station. At first the nodes learn the network topology and send their portion of the topologyinformation to the base station. With this information the base station learns the complete network topology which is used to send route updates as soon as it detects that nodes become silent. This approach is not applicable to event-driven WSN because in such a network sensors only send messageswhen there is an event that should be reported, for instance when the temperature goes above a certain limit. In [35], the authors focus on providing a solution in this context. The proposed mechanism uses a hierarchical network topology where cluster heads monitor ordinary nodes, and the base station monitors the cluster heads. To perform the monitoring, the base station and the clusterheads constantly ping those nodes that still have battery power left and that are under their direct supervision. If a node does not respond, it is marked as failed. Sympathy [30] is a debuging tool that also utilizes the hierarchical detection approach. This tool instruments the WSN with monitoring software on the sensor nodes that generates metrics data that is forwarded to a centralized sink location for analysis.With this information Sympathy is able to detect crash, timeout and omission failures and identify the fault that generated the failure.

VI.   FAULT RECOVERY TECHNIQUES

Fault recovery techniques enable systems to continue operating according to their specifications even if faults of a certain type are present. As discussed in section IV, there are many potential sources for faults in WSNs. Fault tolerance techniques have been proposed in various contexts that increase the reliability of the functionality of sensor nodes in their specific domain. We attempt at giving an overview of this scattered work. The most common of these techniques is the replication of components. Although redundancy has several advantages in terms of high reliability and availability, it also increases the costs of a deployment. As an alternative, according to the specification of the project, the quality required from the WSN can be downgraded to an acceptable level. In this paper we classify the recovery techniques forWSN into two major approaches: Active and Passive replication. Active replication means that all requests are processed by all replicas, while with passive replication, a request is processed by a single instance and only when this instance fails, another instance takes over.

A. Active replication in WSN

Active replication in wireless sensor networks is naturally applied in scenarios where all, or many, nodes provide the same functionality. One example is a service that periodically provides sensor data. Nodes that run this service activate their

sensors and forward their readings to an aggregation service or to a base station. When some nodes fail to provide that information, the recipient still gets the results from other nodes, which is often sufficient. Fault recovery in the presence of active replicas is relatively straightforward. Nevertheless, for a consistent survey we present some of these approaches here:

1) Multipath routing: Usually, it is desirable to avoid that a single failing node causes the partitioning of asensor network. Thus, a network should be k-connected, which allows $k - 1$ nodes to fail while the network would still be connected [22]. Multipath routing [10] can be used to actively replicate routing paths. In [4], Bredin et al. proposes an algorithm that calculates the minimum amount of additional nodes and their positions to guarantee k-connectivity between nodes.

2) Sensor value aggregation: Sensor value fusion [28][5] is a reasearch area that seeks to provides highlevel information derived from a number of low-level sensor inputs. There, the inherent redundancy of sensor nodes can be used to provide fault-tolerant data aggregation.This is achieved through a tradeoff between the precision (the length) of the resulting sensor reading interval and the number of faulty sensors. This ensures that despite of node failures, the resulting reading interval will contain the correct sensor reading of a region.

3) *Ignore values from faulty nodes*: A simple but efficient solution to not propagate a failure of one specific node to the entire network is to ignore the data that it is generating, as applied in [14]. The major challenge in this case is the identification of the malfunctioning nodes.

B. Passive replication in WSN

When passive replication is applied, the primary replicareceives all requests and processes them. In order tomaintain consistency between replicas, the state of theprimary replica and the request information are transferredto the backup replicas. Given the constraints of WSNs, applicationsshould be designed to have only little or no stateat all, which minimizes the overhead for transferring stateinformation between nodes or eliminates it altogether.The process of recovering from a fault when usingpassive replication is illustrated in Figure 4 and consistsof three main steps: fault detection (discussed in SectionV, primary selection and service distribution.

1) *Node selection*: After it has been established thata certain functionality is not available any longer due toa failure in the primary replica, a new service providermust be selected. After this selection phase, one or severalnodes become service providers. Several approaches tohow the selection is performed have been proposed. Wedifferentiate them according to who makes the decisionon which party should become a service provider.

a) *Self election*: In LEACH [15], nodes periodically execute a probabilistic algorithm to establish whether they should serve as clusterhead to their neighbors. In this probabilistic rotation system, nodes keepchanging their role in the network. When a clusterhead node fails, it will take only one rotation period until another node starts providing the functionality of the failed or absent node.
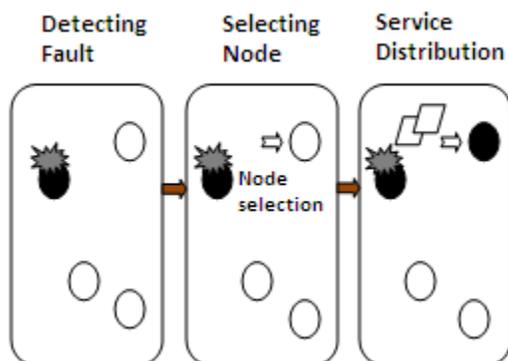

Fig. 2 Failure recovery in Passive replication mode

b) *Group election*: In [12], a reallocation of nodes that were part of a cluster that suffered a clusterhead failure is proposed. The clusterhead, called gateway, is considered to be a resourceful node. The solution presented considers that all the gateways in the network maintain a list of the nodes that are currently in their cluster and another backup list of nodes that could become part of their cluster. When a gateway fails, the nodes from its cluster are reallocated to the other gateways that have the nodes in their backup lists. If more than one gateway has a specific node in its backup list the node is assigned to the clusterhead that has the smallest communication cost.

c) *Hierarchical election*: In a hierarchical election, a coordinator selects the new primary node. This applies to the rebuilding of routing paths [37] as well as the selection of a new clusterhead [13]. The former describes an algorithm to select the node that is closest to the base station. The latter approach applies fuzzy logic in the base station to select which node will become a clusterhead. This algorithm makes use of a fuzzy descriptor, the node concentration, energy level in each node and its centrality with respect to the entire cluster.

2) Service Distribution: During this phase, nodes elected to become service providers must activate the

service. In some cases the service is already available on the nodes and a simple configuration change to inform the node that this service should be activated is required. However in some cases, for instance when nodes do not have enough memory to store the code of all potential services, it is necessary to inject code into the node through some technique. There are different techniques that can be used for service distribution: completely reprogramming the node, sending entire blocks of executable code, or sending small pieces of code such as scripts.

Pre-Copy: Pre-copying as described in [9], consists in making the code of all services available on all nodes before deployment. This allows nodes to change their behavior according to the role that they are assigned to.

b) Code distribution: Several approaches have been proposed for disseminating code throughout the network. Mat´e [21] is an example for a bytecode interpreter for TinyOS where code is broken into capsules of 24 instructions. These capsules can be distributed through the network and installed on nodes, which start to execute the new code. Agilla [8] is a Mat´e-based mobile agent middleware for programming wireless sensor networks. These mobile agents can be programmed to move through the network or replicate themselves to other nodes according to changes in the environment.

c) Remote Execution: On the one hand code migrationis an approach that reduces the amount of memoryrequired in the entire network since not all nodes needto have the application pre-installed, on the other handit consumes energy on the nodes exchanging the codeand is susceptible to link failures, which could causelong delays until the code update is completed. RemoteExecution [34], [32] is an alternative approach where lowpowerdevices transfer tasks to more powerful deviceswithout transferring the entire application code. Instead,only the required state information is transmitted. Suchan approach is especially suited for heterogeneous sensornetworks with at least some resourceful nodes.

VII.   CONCLUSION

In this paper we provided a thorough investigation of faults that occured in real WSN deployments. This concise investigation provides a valuable knowledge input for future application to prevent the same kind of issues from happening. By focusing only on the faults, the lessons learned from the different deployments can be used by any application even if the investigated trial had a different research focus. We proposed taxonomy to classify faults and failures that occur in WSN. We also studied the problem of fault detection and recovery, surveying the different techniques currently applied in WSN research. A classification of the available fault tolerance techniques for wireless sensor networks has been proposed considering the various mechanisms adopted by the solutions. To our knowledge this is the first work that provides a concise survey and classification in this area. Through the classification proposed it is possible to compare the different solutions identifying the strong and weak points of each of them. This allows for a correct selection of the techniques that are more suitable to specific applications.

 REFERENCES
[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. ASurvey on Sensor Networks. IEEE Communications Magazine, 40:102–114, August 2002.
[2] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi.A Discrete-Time Battery Model for High-Level Power Estimation. In Proceeding of the Design, Automation and Test in Europe Conference and Exhibition 2000, pages 35–39, 2000.
[3] A. Birolini. Quality and Reliability of Technical Systems:theory, practice, management. Springer, 1997.
[4] J. L. Bredin, E. D. Demaine, M. Hajiaghayi, and D. Rus. Deploying Sensor Networks with Guaranteed Capacity and Fault Tolerance. In Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing, pages 309–319, 2005.
[5] D. Desovski, Y. Liu, and B. Cukic. Linear randomized voting algorithm for fault tolerant sensor fusion and the corresponding reliability model. In IEEE International Symposium on Systems Engineering, pages 153–162, October 2005.
[6] M. Ding, D. Chen, K. Xing, and X. Cheng.Localized fault-tolerant event boundary detection in sensor networks.In INFOCOM, 2005.
[7] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: scalable coordination in sensor networks. In MobiCom'99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, pages 263–270,
New York, NY, USA, 1999. ACM Press.
[8] C.-L. Fok, G.-C.Roman, and C. Lu. Mobile Agent Middleware for Sensor Networks: an Application Case Study. In Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN), 2005.
[9] C. Frank and K. R¨omer. Algorithms for Generic Role Assignment in Wireless Sensor Networks. In Proceedings of the 3rd international conference on Embedded networked sensor systems, pages 230–242, 2005.
[10] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly- Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks.Mobile Computing and Communications Review, 1(2), 1997.
[11] R. Guerraoui and A. Schiper.Fault-Tolerance by Replication in Distributed Systems. In Proceedings of the 1996 Ada-Europe International Conference on Reliable Software Technologies, pages 38–57, 1996.

[12] G. Gupta and M. Younis.Fault-Tolerant Clustering of Wireless Sensor Networks. Wireless Communications and Networking, 3:1579–1584, 2003.

[13] I. Gupta, D. Riordan, and S. Sampalli. Cluster-Head Election Using Fuzzy Logic for Wireless Sensor Networks. In Proceedings of the 3rd Annual Communication Networks and Services Research Conference, pages 255–260, 2005.

[14] S. Harte and A. Rahman. Fault Tolerance in Sensor Networks Using Self-Diagnosing Sensor Nodes. In The IEE International Workshop on Intelligent Enviroment, pages 7–12, June 2005.

[15] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In Proceedings of the 33rd Hawaii International Conference on System Sciences, volume 8, page 8020, 2000.

[16] C.-Y. Koo. Broadcast in Radio Networks Tolerating Byzantine Adversarial Behavior. In Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing, pages 275–282, 2004.

[17] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentell. Fault Tolerance Techniques for Wireless Ad hoc Sensor Networks. In Proceedings of IEEE Sensors, volume 2, pages 1491–1496, 2002.

[18] B. Krishnamachari and S. Iyengar. Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks. IEEE Transactions on Computers, 53:241–250, March 2004.

[19] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems, 4:382–401, 1982.

[20] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. In IPDPS 20th International Parallel and Distributed Processing Symposium, 2006.

[21] P. Levis and D. Culler.Mat´e: A Tiny Virtual Machine for Sensor Networks. In ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems, pages 85–95, New York, NY, USA, 2002. ACM Press.

[22] N. Li and J. C. Hou. FLSS: A Fault-Tolerant Topology Control Algorithm for Wireless Networks. In Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, pages 275–286, 2004.

[23] T. Liu and M. Martonosi. Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems. In PPoPP '03: Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming, pages 107–118, New York, NY, USA, 2003. ACM Press.

[24] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, pages 88–97, New York, NY, USA, 2002. ACM Press.

[25] D. Marculescu, N. H. Zamora, P. Stanley-Marbell, and R. Marculescu. Fault-Tolerant Techniques for Ambient Intelligent Distributed Systems. In Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design, page 348, 2003.

[26] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad hoc Networks. In Proceedings of the 6th annual international conference on Mobile computing and networking, pages 255–265, 2000.

[27] K. Martinez, P. Padhy, A. Riddoch, H. Ong, and J. Hart. Glacial environment monitoring using sensor networks. In REALWSN' 05, 2005.

[28] K. Marzullo. Tolerating failures of continuous-valued sensors. ACM Transactions on Computer Systems, 8(4):284–304, 1990.

[29] D. Rakhmatov and S. B. Vrudhula. Time-to-Failure Estimation for Batteries in Portable Electronic Systems. In Proceedings of the 2001 international symposium on Low power electronics and design, pages 88–91, 2001.

[30] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy: a debugging system for sensor networks. In IEEE International Conference on Local Computer Networks, 2004.

[31] M. Ringwald, K. R¨omer, and A. Vitaletti.Snif: Sensor network inspection framework. Technical Report 535, ETH Z¨urich, Institute for Pervasive Computing, October 2006.

[32] P. Rong and M. Pedram. Extending the lifetime of a network of battery-powered mobile devices by remote processing: A markovian decision-based approach. In DAC '03: Proceedings of the 40th conference on Design automation, pages 906–911, New York, NY, USA, 2003.ACM Press.

[33] S. Rost and H. Balakrishnan. Memento: A health monitoring system for wireless sensor networks. In SECON, 2006.

[34] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning. The Remote Processing Framework for Portable Computer Power Saving. In SAC '99: Proceedings of the 1999 ACM symposium on Applied computing, pages 365–372, New York, NY, USA, 1999. ACM Press.

[35] L. B. Ruiz, I. G. Siqueira, L. B. e Oliveira, H. C. Wong, J. M. S. Nogueira, and A. A. F. Loureiro. Fault Management in Eventdriven Wireless Sensor Networks. In Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems, pages 149–156, June 2004.

[36] T. Schmid, H. Dubois-Ferri`ere, and M. Vetterli. Sensorscope: Experiences with a wireless building monitoring sensor network. In REALWSN' 05, 2005.

[37] J. Staddon, D. Balfanz, and G. Durfee. Efficient Tracing of Failed nodes in Sensor Networks. In Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, pages 122–130, 2002.

[38] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems, pages 214 226, New York, NY, USA, 2004.ACM Press.

[39] R. Szewczyk, J. Polastre, A. M. Mainwaring, and D. E. Culler. Lessons from a sensor network expedition. In EWSN, pages 307– 322, 2004.

[40] A. S. Tanenbaum and M. van Steen. Distributed Systems: Principles and Paradigms. Prentice Hall, 2002.

[41] J. Tateson, C. Roadknight, A. Gonzalez, T. Khan, S. Fitz, I. Henning, N. Boyd, and C. Vincent. Real world issues in deploying a wireless sensor network for oceanography. In RealWSN' 05, 2005.

[42] J. Thelen, D. Goense, and K. Langendoen. Radio wave propagation in potato fields. In First Workshop on Wireless Network Measurement, 2005.

[43] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems, pages 51–63, New York, NY, USA, 2005. ACM Press.