Volume 5, Issue 8, August 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Review on Fault Detection and Recovery in WSN

[1]Nagalgaonkar Pramod, [2]Dhanraj Biradar, [3]Gaikwad Ranjit Sharnappa
[1, 3] Mtech Student, [2]Asso. Prof.
[1, 2, 3] Dept of ECE, BKIT Bhalki, Karnataka, India

Abstract - In recent due to advance research, applications of wireless sensor networks (WSNs) have been increased in different applications. In wireless sensor networks, sensor nodes are operated in unattended mode. In WSNs, it is essential to maintaining the communication between sensor nodes for all the time. Failure of the node affects the consistency of the network. One of the design challenge efficient fault management solutions to recover network from unanticipated failures. In this paper, we discuss different types of faults, detections techniques and fault recovery algorithms.

Key Words: WSN, fault detection, Fault node recovery algorithm,

I. INTRODUCTION

Latest advances in MEMS (Micro-electro-mechanical systems) [1] wireless sensors are made of small, inexpensive, low power tiny devices. Sensor nodes are equipped with capability of local processing, wireless communication, internal memory and battery source. Sensors are generally have limited capabilities due to internal battery source. Thousands of sensor node are usually deployed to operate in attended mode to sense physical parameters and transmit this information to the base station through wireless communication called as wireless sensor network. Following are the examples applications of WSN include environmental monitoring such as air soil and water, condition based maintenance, habitat monitoring, seismic detection, military surveillance, inventory tracking, smart spaces [2] etc.

As the sensor nodes are tiny this technology offers some drawbacks also. One of the major issues is limited battery source. Failure of sensor node occurs when the battery gets completely discharged. Due to this the topology of the network changes and degrades the quality of the network services. Following are causes sensor node; failure of any hardware module, environmental reason, enemy attacks [4], incorrect communication, and congestion in network.

II. TYPES OF FAULTS IN WSN

Following are the some points that give the differences between faults, errors, and failures in order to understand fault [6]. Fault is an unintentional defect that finally channelizes to the cause of an error. An error corresponds to an inaccurate system state. Such a state may lead to a failure. A failure is the sign of an error, which occurs when the system deviate from its specification and cannot deliver its intended functionality.

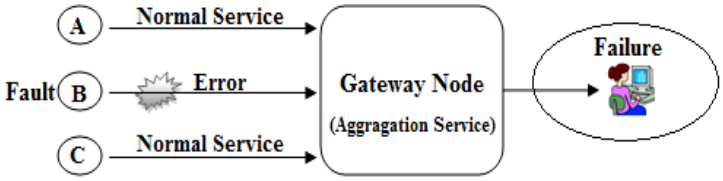

Figure 1: Relationships between Fault, Error and Failure

Figure 1 shows the basic difference between fault, error, and failure. The principle operation of sensor node A, B and C are reporting periodical sensed data to the gateway node which aggregates different generic sensor data for future analysis. Each sensor service is normal until node B suffers a fault. Thus, the immediate occurrence of fault causes an error in performing normal service by node B. Due to the occurrence of fault on node B, it provide an error service to the gateway node. These error services contain inappropriate information to the analysis of entire system. The faulty service provided by node B ultimately causes system failure.

In sensor network every node has one of state failure or working. Most of the sensor networks work in hierarchical structure due to power limitation. The issues are network environment unpredictable and dynamic topology. Duration Based Faults [3] [4]:

Transient faults: these faults are short duration and can dissolve by analysing optimum retry period.

Intermittent faults: The unpredictable faults which are appearance and disappearance are as called as intermittent faults. They take place by a number of factors such as malfunction.

Permanent faults: If the fault gets occurred in network it remains as it is until it repaired.

© 2015, IJARCSSE All Rights Reserved

Based on the performance

Soft Fault: These are also known as dynamic faults. When some of the sensors are capable to sense the environment and communicate with the other sensors but transmit some incorrect messages at a particular time, such sensor nodes are known as soft faulty sensors.

Hard fault: These are permanent or static faults. When sensor not succeeds to communicate with the other nodes in the network, that node is called as hard faulty sensor node.

In WSN all sensor nodes have the equal failure probability [5] and therefore the data delivery in sensor networks is faulty and unexpected. In most of the applications solid data delivery required to base station. Subsequently, it is basic to give fault tolerant techniques to distributed sensor network applications. Faults occurrence is unavoidable in Wireless Sensor Networks because nodes are operated in unattended mode. To differentiate between faulty nodes and alive node, numerous techniques are proposed. The main aim in wireless sensor network is to enhance the fault tolerance of every node furthermore give an energy productive fast data routing service.

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. The main technique to achieve this goal is to replicate the components of the system that are vital for its correct operation.

III. 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 method is to perform visual observation and manual removal of incorrect values. This technique offers drawbacks, human interaction leads to errors, it has a high cost and it is not efficient. Hence automatic fault detection techniques for WSN used. Through self diagnosis 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.

i. Self-Diagnosis

In many cases, nodes can identify possible failures by performing self-diagnosis. In [7], 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 [8]. 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.

ii. Group Detection

The detection of services failing due to incorrectly generated values is only possible if a reference value is available. In [9] a detection mechanism is proposed to identify faulty sensor nodes. Algorithm is 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. In [10], a misbehavior detection algorithm to aid the routing layer is proposed. The misbehavior detection mechanism is based on the idea of monitoring the communication of the service provider to verify whether messages are forwarded correctly. Focusing on providing a fault-tolerant approach for clusters in WSNs, in [11] it is proposed to support the dynamic recovery of failed gateways. 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.

iii. Hierarchical Detection

Memento [12] proposed the usage of the network topology to forward the fault detection results of child nodes to their parent nodes and finally to sink nodes. Every node forwards the condition of the child nodes that it is monitoring to their parent node. The parent node carries out an aggregation operation on the outcomes of the child nodes together with its own results and forwards it to the further process.

The Memento's proposed approach scales well with the network size; but it consumes resources of the network. Shifting the fault detection job to a more powerful device is an alternative that can help to increase the lifetime of the WSN. In [13], the authors propose an algorithm that puts the burden of detecting and tracing failed nodes to the base station. At first the nodes find out the network topology and send their part of the topology information to the sink node. With this information the base station learns the complete network topology which is used to send route updates quickly as it detects that nodes become silent.

This approach is not applicable to event-driven WSN because in such a network sensors only send messages when there is an event that should be reported, for instance when the temperature goes above a certain limit. In [14], 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 cluster heads constantly ping those nodes that still have battery power left and that are under their

© 2015, IJARCSSE All Rights Reserved

direct supervision. If a node does not respond, it is marked as failed. Sympathy [15] is a debugging 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 consideration is able to detect faults and failures and identify the fault that generated the failure.

IV. FAULT RECOVERY TECHNIQUES

Fault recovery techniques allow systems to continue operating according to their specifications even though certain types of faults 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 consistency 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 for WSN 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.

4.1 Active replication in WSN.

Active replication in WSN is applied in scenarios where many nodes provide the identical functionality. One example is a service that periodically provides sensor data. Nodes that run such service activate their sensors and send their analysis to an aggregation service or to a base station. When some of the nodes fail to offer information, the receiver still gets the results from other nodes, which is often sufficient. Fault recovery in the presence of active replicas is relatively simple. Nevertheless, for a consistent survey we present some of these approaches here:

1. Multipath routing: Multipath routing can be used to actively replicate routing paths. In [16], 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 [17] is a research area that seeks to provide high level 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 [18]. The major challenge in this case is the identification of the malfunctioning nodes.

4.2 Passive replication in WSN

When passive replication is applied, the primary replica receives all requests and processes them. In order to maintain reliability between replicas, the state of the primary replica and the request information are transferred to the backup replicas. Given the constraints of WSNs, applications should be designed to have only little or no state at all, which minimizes the overhead for transferring state information between nodes or eliminates it all together.

1. Node selection: After it has been established that certain functionality is not available any longer due to failure in the primary replica, a new service provider must be selected. After this selection phase, one or several nodes become service providers. Several approaches to how the selection is performed have been proposed. We differentiate them according to who makes the decision on which party should become a service provider.

a) Self election: In LEACH [18], nodes periodically execute a probabilistic algorithm to establish whether they should serve as cluster head to their neighbors. In this probabilistic rotation system, nodes keep changing their role in the network. When a cluster head node fails, it will take only one rotation period until another node starts providing the functionality of the failed or absent node.

b) Group election: In [19], a reallocation of nodes that were part of a cluster that suffered a cluster head failure is proposed. The cluster head, 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 cluster head 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 as well as the selection of a new cluster head. 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 cluster head. 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

© 2015, IJARCSSE All Rights Reserved

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.

a) Pre-Copy: Pre-copying as described in [20], 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 byte code interpreter for Tiny OS 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. c) Remote Execution: On the one hand code migrations an approach that reduces the amount of memory required in the entire network since not all nodes need to have the application pre-installed, on the other hand it consumes energy on the nodes exchanging the code and is susceptible to link failures, which could cause long delays until the code update is completed. Remote Execution [22] is an alternative approach where low power devices transfer tasks to more powerful devices without transferring the entire application code. Instead, only the required state information is transmitted. Such an approach is especially suited for heterogeneous sensor networks with at least some resourceful nodes.

V. CONCLUSION

In this paper we provided a thorough investigation of faults that occurred in real WSN deployments. 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] K. Padmanabhan and Dr. P. Kamalakkannan "Energy-efficient Dynamic Clustering Protocol for Wireless Sensor Networks", International Journal of Computer Applications (0975 – 8887), Volume 38– No.11, January 2012.

[2] Abolfazl Akbari, Arash Dana, Ahmad Khademzadeh and Neda Beikmahdavi, "Fault Detection and Recovery in Wireless Sensor Network Using Clustering", International Journal of Wireless & Mobile Networks (IJWMN) Vol. 3, No. 1, February 2011.

[3] Heena , Samarth Kapoor, "Survey of Fault Detection Algorithm in WSN", SSRG International Journal of Computer Science and Engineering (SSRG-IJCSE)-EFES April 2015.

[4] Peng Jiang, "A New Method for Node Fault Detection in Wireless Sensor Networks", Sensors 2009, 9, 1282-1294; doi:10.3390/s90201282.

[5] Vibha Paradkar, Gajendra Singh Chandel ,Kailash Patidar, "Fault Node Discovery and Efficient Route Repairing Algorithm for Wireless Sensor Network, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (2) , 2015, 1710-1715.

[6] Sushruta Mishra, Lambodar Jena, Aarti Pradhan, "Fault Tolerance in Wireless Sensor Networks", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 10, October 2012.

[7] 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.

[8] B. R. Tapas Bapu, K. Thanigaivelu, A. Rajkumar, "Fault Tolerance in Wireless Sensor Networks – A Survey", International Journal of Electrical, Computer, Electronics and Communication Engineering Vol:9, No:2, 2015.

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

[15] 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.

© 2015, IJARCSSE All Rights Reserved

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[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] 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.

© 2015, IJARCSSE All Rights Reserved