Fault Diagnosis Algorithms for Wireless Sensor Networks

Arunanshu Mahapatro
(509CS302)

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India

Fault Diagnosis Algorithms for Wireless Sensor Networks

Dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science and Engineering

by

Arunanshu Mahapatro

(Roll- 509CS302)

under the guidance of

Prof. Pabitra Mohan Khilar

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India

Dedicated to my parents...

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Orissa, India.

Dr. Pabitra Mohan Khilar
Assistant Professor

1^{st} November, 2012

Certificate

This is to certify that the work in the thesis entitled *Fault Diagnosis Algorithms for Wireless Sensor Networks* by *Arunanshu Mahapatro* is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy in Computer Science and Engineering in the department of Computer Science and Engineering, National Institute of Technology Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Pabitra Mohan Khilar

Acknowledgement

This dissertation, though an individual work, has benefited in various ways from several people. Whilst it would be simple to name them all, it would not be easy to thank them enough.

The enthusiastic guidance and support of *Prof. Pabitra Mohan Khilar* inspired me to stretch beyond my limits. His profound insight has guided my thinking to improve the final product. My solemnest gratefulness to him.

It is indeed a privilege to be associated with people like *Prof. S. K. Jena*, *Prof. S. K. Rath*, *Prof. B. Majhi*, *Prof. R. Baliarsingh*, *Prof. D. P. Mohapatra*, *Prof. A. K. Turuk*, and *Prof. B. D. Sahoo*. They have made available their support in a number of ways. My humble acknowledgement to *Prof. S. K. Patra*, and *Prof. S. Meher* for nourishing my intellectual maturity.

Many thanks to my comrades and fellow research colleagues. It gives me a sense of happiness to be with you all. Special thanks to *Ratnakar*, *Saroj*, *Pankaj* and *Hunny* whose involvement gave a new breath to my research.

Finally, my heartfelt thanks to my wife *Alaka*, my sister *Nibedita* and my daughter *Ayushi* for their unconditional love and support. Words fail me to express my gratitude to my beloved parents and parents-in-law who sacrificed their comfort for my betterment.

Arunanshu Mahapatro

Abstract

The sensor nodes in wireless sensor networks (WSNs) are deployed in unattended and hostile environments. The ill-disposed environment affects the monitoring infrastructure that includes the sensor nodes and the links. In addition, node failures and environmental hazards cause frequent topology change, communication failure, and network partition. This in turn adds a new dimension to the fragility of the WSN topology. Such perturbations are far more common in WSNs than those found in conventional wireless networks. These perturbations demand efficient techniques for discovering disruptive behavior in WSNs. Traditional fault diagnosis techniques devised for wired interconnected networks, and conventional wireless networks are not directly applicable to WSNs due to its specific requirements and limitations.

System-level diagnosis is a technique to identify faults in distributed networks such as multiprocessor systems, wired interconnected networks, and conventional wireless networks. Recently, this has been applied on ad hoc networks and WSNs. This is performed by deduction, based on information in the form of results of tests applied to the sensor nodes. Neighbor coordination-based system-level diagnosis is a variation of this method, which exploits the spatio-temporal correlation between sensor measurements. In this thesis, we present a new approach to diagnose faulty sensor nodes in a WSN, which works in conjunction with the underlying clustering protocol and exploits spatio-temporal correlation between sensor measurements. An advantage of this method is that the diagnostic operation constitutes real work performed by the system, rather than a specialized diagnostic task. In this way, the normal operation of the network can be used for the diagnosis and resulting less time and message overhead. In this thesis, we have devised and evaluated fault diagnosis algorithms for WSNs considering persistence of the faults (transient, intermittent, and permanent), faults in communication channels and in one of the approaches, we attempt to solve the issue of node mobility in diagnosis.

A cluster based distributed fault diagnosis (CDFD) algorithm is proposed where the diagnostic local view is obtained by exploiting the spatially correlated sensor measurements. We derived an optimal threshold for effective fault diagnosis

in sparse networks. The message complexity of CDFD is $O(n)$ and the number of bits exchanged to diagnose the network are $O(n \log_2 n)$.

The intermittent fault diagnosis is formulated as a multiobjective optimization problem based on the inter-test interval and number of test repetitions required to diagnose the intermittent faults. The two objectives such as detection latency and energy overhead are taken into consideration with a constraint of detection errors. A high level ($> 95\%$) of detection accuracy is achieved while keeping the false alarm rate low ($< 1\%$) for sparse networks. The proposed cluster based distributed intermittent fault diagnosis (CDIFD) algorithm is energy efficient because in CDIFD, diagnostic messages are sent as the output of the routine tasks of the WSNs.

A count and threshold-based mechanism is used to discriminate the persistence of faults. The main characteristics of these faults are the amounts of time the fault disappears. We adopt this state-holding time to discriminate transient from intermittent or permanent faults. The proposed cluster based distributed fault diagnosis and discrimination (CDFDD) algorithm is energy efficient due to the improved network lifetime which is greater than 1150 data-gathering rounds with transient fault rates as high as 20%.

A mobility aware hierarchal architecture is proposed which is to detect hard and soft faults in dynamic WSN topology assuming random movements of nodes in the WSN. A test pattern that ensures error checking of each functional block of a sensor node is employed to diagnose the network. The proposed mobility aware cluster based distributed fault diagnosis (MCDFD) algorithm assures a better packet delivery ratio ($> 80\%$) in highly dynamic networks with a fault rate as high as 30%. The network lifetime is more than 900 data-gathering rounds in a highly dynamic network with a fault rate as high as 20%.

Keywords: WSNs, fault, persistence of fault, fault diagnosis, channel fault, multi-objective optimization, node mobility, test pattern.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rapid advancement in technology, particularly in Micro-Electro-Mechanical systems has facilitated the development of smart sensors (e.g., Mica motes from Crossbow, Tmote Sky from Moteiv, the MKII nodes from UCLA, etc.). This made it possible to connect independent sensor nodes together to create a Wireless sensor networks (WSNs) with greater monitoring and target tracking [2–6]. Smart sensor nodes are low power devices subject to tight communication, storage and computation constraint. A variety of sensor nodes can be deployed in huge numbers in order to monitor, detect and report time-critical events such that the urgency of the situation can be evaluated, and efforts are coordinated in a timely manner. The WSNs have the potential to enable a substantial class of applications [3,7–14]. In military target applications; a WSN can assist in intrusion detection and identification. Sensor nodes can sense and detect the environment to forecast disasters before they occur [7]. Surgical implants of sensors can help to monitor a patient's health in biomedical applications (body sensor networks). Deployment of sensors along the volcanic area can detect the development of earthquakes and eruptions.

Sensor nodes are expected to operate autonomously in unattended and hostile environments for applications with short mission time and applications that last for months to years. Infact, WSNs are prone to have faults compared to traditional wireless networks where faults are likely to occur frequently and unexpectedly. Faults may range from simple crash faults where a node becomes completely inactive to faults where the node behaves arbitrarily or maliciously [15]. The

fault is an incorrect state of hardware or software as a consequence of a failure of a component [16]. As faults are inevitable in WSNs, it is crucial to determine which nodes of the network are working and, which are faulty. As shown in Fig. 1.1, faults can be at each level of six individual components of a node: computing engine, transceiver and memory subsystems, energy source, sensors, and actuators which in turn results in node failure.

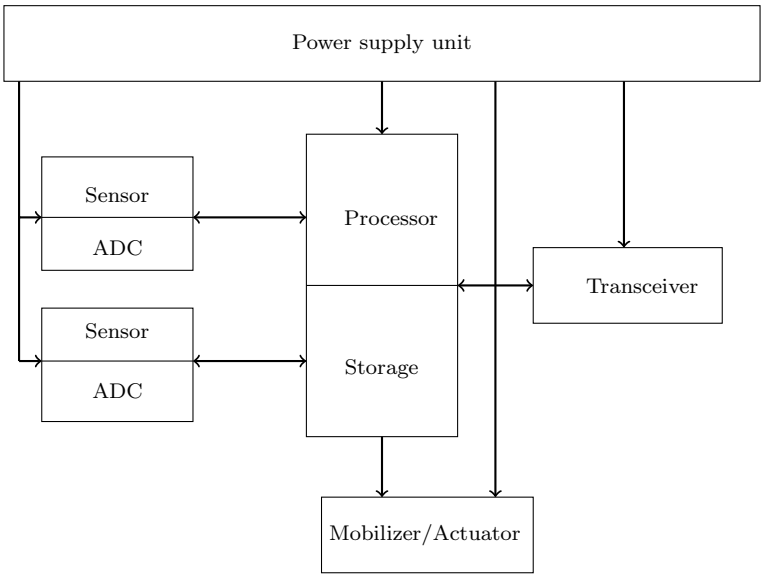


Figure 1.1: Typical sensor node.

If faults occur, consequences can be severe in terms of human life, environmental impact or economic loss. The erroneous outputs from faulty sensors might result in wrong interpretation or undesirable alarms. This may lead to life-threatening events to occur as a significant percentage of WSNs will be involved in safety critical applications. For example, faulty sensor nodes in a WSN, embedded around beams or columns of a railway bridge for detailed building constructional monitoring, may not give early warning of any structural weakness or deterioration. In addition, erroneous data generated by faulty sensor nodes must be protected from entering the network for effective bandwidth and energy utilization.

The rest of this chapter is organized as follows. A brief description of fault, error and failure is presented in Section 1.1. Section 1.2 gives a brief description of system-level fault diagnosis. Definitions and terminologies used are presented

in Section 1.3. In Section 1.4, different sources of faults are discussed. Section 1.5 presents the factors influencing fault diagnosis. Section 1.6 presents the motivation of the proposed works. The problem statement is presented in Section 1.7 and finally, the organization of the thesis is presented in Section 1.8.

1.1 Faults, Errors and Failures

A fault refers to an abnormal physical state of a sensor node which may have several causes such as humidity, temperature, power surge, electromagnetic radiations, design or installation errors, age, etc. Unless ground truth is known or given by something with high confidence, the term fault can only refer to a deviation from the expected model of the phenomenon [17]. When the fault affects a sensor node it produces an erroneous result. The presence of a fault does not ensure that an error will occur. The reverse, however, is true [18]. When the errors make a sensor node unable to perform its routine task, it results in a failure. A failure of the sensor node occurs when the behavior of the node deviates from the system specification.

The modeling of faults in a sensor node can be approached at different levels of abstraction. It can range from hardware or software level to system level or node level. Failures at hardware or software levels may result as errors at the system level and make the sensor node to behave abnormally. Generally, the hardware or software level of fault modeling is most generic, but the system level of fault modeling is easier to analyze as they consider the behavior of the faults [19].

Faults are classified based on behavior of failed components, persistence of faults, or on the underlying cause [20, 21]. Based on how a failed sensor node behaves, faults can be classified as hard or soft faults. A sensor node exhibits hard faults is unable to communicate with other sensor nodes in the network. A sensor node exhibits soft faults continues to operate with altered behaviors. Based on persistence, faults can be classified as permanent, intermittent, or transient. Permanent or hard faults are software or hardware faults that always produce errors when they are fully exercised [1]. Temporary faults can be distinguished into external faults (transient) and internal faults (intermittent). The former are soft

faults that are caused by events, which come from a sensor node's environment and do not imply that the sensor node is faulty. These faults are hard to be traced to a defect since normally their adverse effects rapidly disappear [22]. A particularly problematic type of transient fault is the intermittent fault that by its nature, usually exhibits a relatively high occurrence rate after its first appearance and, eventually, tends to become permanent [21, 22]. An intermittent fault originates from inside the system when software or hardware is faulty. A different mode of classifying faults is presented in [1], where faults are classified as: crash, omission, timing, incorrect computation, fail-stop fault and Byzantine. Crash faults are hard faults, and all the others can be considered soft faults. Fig. 1.2 illustrates the classification of fault types. Knowledge of all possible fault classes allows a protocol designer to design a generalized diagnosis protocol.

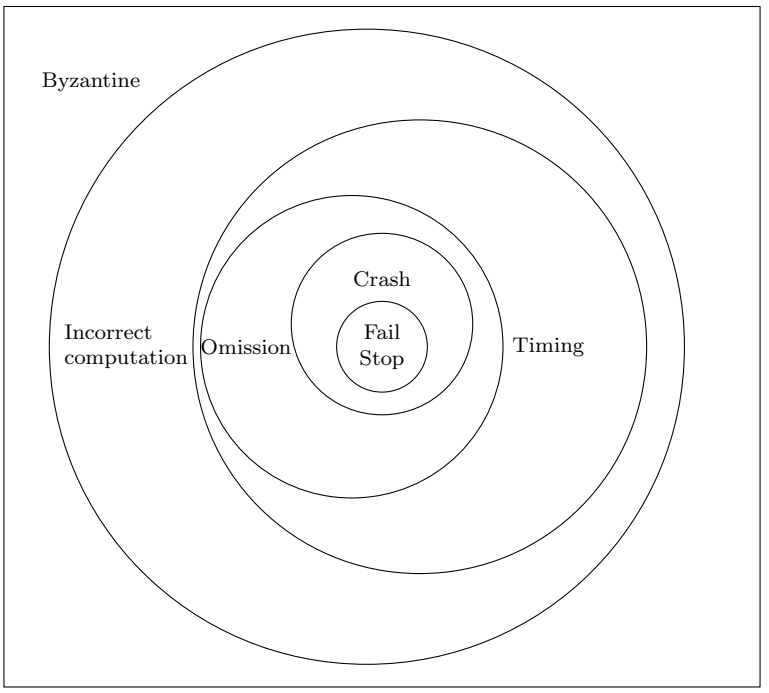


Figure 1.2: An ordered fault classification [1].

- *Crash fault*: The fault that occurs if the battery is completely depleted, the transceiver is faulty or the node is completely damaged. A crash faulty sensor node loses its internal state and cannot participate in network activities. This is a natural permanent fault [23] that are caused by natural phenomena without human participation.

- Omission fault: A sensor node that does not respond to the sink node on time, fails to send a required message on time or fails to relay the received message of its neighbor is exhibiting an omission fault. This may be either a malicious fault [23] that is introduced by a human with the malicious objective or a natural fault. This fault can be either permanent, intermittent or transient in nature.

- *Timing fault*: A timing fault causes the sensor node to respond with the expected value but either too soon, or too late. An overloaded sensor node (e.g., cluster head) which produces correct values, but with an excessive delay suffers from a timing failure. This failure can occur in a WSN which imposes timing constraints on computations. Like an omission fault, this may be a natural or human-made fault. This fault can be either permanent or transient in nature.

- *Incorrect computation fault*: The fault that occurs when a sensor node fails to send the true measurement even though the sensing element of the sensor node perceived the true data. Like omission and timing fault this may be a natural or human-made fault. This fault can be either permanent, intermittent or transient in nature.

- *Fail-stop fault:* The fault that occurs when a sensor node ceases operation due to depletion of battery and alerts its one-hop neighbors of this fault. This may be a natural or human-made permanent fault.

- *Byzantine fault*: The previous failure classes have specified how a sensor node can be considered to fail in the different domain. It is possible for a sensor node to fail in all the domains in a manner, which is not covered by one of the previous classes. A failed sensor node which produces such an output will be said to be exhibiting an arbitrary failure or Byzantine failure.

The most basic fault classes such as crash, fail-stop, omission, and timing failures, are problems that occur and detected in the time domain [1]. The fault classes like incorrect computation and Byzantine faults occur in data domains.

Crash faults are hard faults, and all others can be considered as soft faults [20]. These faults may appear continuously or intermittently.

1.2 System-Level Fault Diagnosis

System-level diagnosis is a technique for fault tolerance that strives to identify the faulty sensor nodes in a WSN. This is done by deduction, based on information in the form of results of tests applied to the sensor nodes [24]. Once the faulty sensor nodes have been identified, the system is able to isolate them, ignore their output such that the lifetime of the WSN [25] can be maintained in the long run.

System-level fault diagnosis was introduced by Preparata, Metze and Chien in 1967 [26], as a technique aimed at diagnosing faults in wired interconnected networks composed by a number of processing elements. The so called PMC model is based upon the outcomes of reciprocal tests performed by the units themselves. In this model, the test requires a bidirectional interconnection between units, i.e., the tester and the tested units must be adjacent. The testing unit provides a test sequence as input to the tested unit, which in turn executes a test program on the input sequence and returns the result to the testing unit. The testing unit generates the test outcome by comparing the actual and the expected results. If they agree, the outcome is 0, otherwise it is 1. Many variants of PMC model have been proposed and are well presented in the survey [1]. The realistic variant of PMC model is based on comparisons rather than explicit tests [27]. The first comparison-based diagnosis model for wired interconnected networks was proposed by Malek [28]. This model assumed that in a system with n processing elements, it is possible to compare the outputs produced by task executions from some or every pair of processing elements. A comparison that results in a mismatch indicates that one or both units are faulty. However, these previously developed diagnosis models are tailored for wired interconnected networks and hence not well suited to wireless networks since observability is the major issue.

System level diagnosis in WSNs takes advantage of the shared nature of communication medium. In addition, this exploits the fact that sensor faults are likely to be stochastically unrelated, while sensor measurements are likely to

be spatially correlated [29]. The major issues involved in system-level diagnosis of WSNs are characterization problem, diagnosability problem, diagnosis problem and diagnosis overhead problem.

Characterization Problem

The characterization problem is to find the necessary and sufficient conditions to achieve the desired diagnosability in a network. Preparata et al. [26] showed that the necessary conditions for t-diagnosability are that $n \geq 2t + 1$ and each node must be tested at least by other t distinct nodes. Hakimi and Amin [30] showed that the conditions given by Preparata *et al.* are sufficient for t-diagnosability for the special case in which no two processors test each other. They also presented a general characterization of t-diagnosable systems. Much efforts have also been made in order to find a general characterization for wired interconnected networks. However, this characterization may not be applicable to WSNs as observability and resources like energy and bandwidth are the major issues. The current hypothesis is that, to achieve high diagnosability in a WSN a node with large one-hop neighbor size is needed [31–36].

Diagnosability Problem

The diagnosability problem is to determine the number of faulty sensor nodes in WSNs those can be unambiguously identified.

Diagnosis Problem

The diagnosis problem is to identify the correct set of faulty sensor nodes in WSNs. The diagnosis problem is scaled by two standard performance metrics namely detection accuracy (DA) and false alarm rate (FAR) [31, 33, 34].

Diagnosis Overhead Problem

The diagnosis overhead problem characterizes the message, time and energy complexity in diagnosing the WSN [20, 37, 38].

1.3 Definitions and Terminologies

A monitoring system that detects faulty sensor nodes and diagnoses their location in the WSN is called a fault diagnosis system [39]. A fault must be promptly diagnosed even at its early stage to prevent any serious consequences. A fault diagnosis system commonly carries out following tasks:

- Fault detection: to make a binary decision - either a sensor node deviates from its normal behavior.
- Fault diagnosis: to locate all faulty sensor nodes in a WSN such that each sensor node will have a global view of the WSN.
- Fault identification: to estimate the severity and type of fault.

The relative importance of these tasks is absolutely subjective. However, fault detection is vital for any practical system, and isolation is almost equally important. Fault identification may not be crucial if reconfiguration action is not demanded. Hence fault diagnosis is very often believed as fault detection and isolation.

The following terminologies are used in connection to fault diagnosis in WSNs:

- *Correctness.* The diagnosis is said to be correct if there are no fault-free sensor nodes mistakenly diagnosed as faulty.
- *Completeness.* The diagnosis is said to be complete if all faulty sensor nodes are correctly identified.
- *Consistency.* All sensor nodes agree on the same set of faulty sensor nodes at each diagnostic round.
- *Latency.* Time elapsed since appearance of fault to isolation of the sensor node.
- *Communication complexity.* Total number of diagnostic messages exchanged in the WSN to ensure correct and complete diagnosis.

- *Detection accuracy.* It is defined as the ratio of the number of faulty sensor nodes detected to the actual number of faulty sensor nodes in the WSN.

- *False alarm rate.* It is the ratio of the number of fault-free sensor nodes diagnosed as faulty to the actual number of fault-free sensor nodes.

1.4 Sources of Faults

Data aggregation and delivery in WSNs are inherently faulty and unpredictable [40,41]. The key sources of failure are calibration error, malfunctioning hardware, hostile environment, low battery and link failure.

Though the calibration during deployment is performed, sensors throughout their deployed lifetime may drift. This in turn lowers the accuracy of sensor measurements. Three different types of calibration errors are reported in [17] namely offset faults (sensor measurements offset from the ground truth by a constant amount), gain faults (the rate of change of the measured data does not match with expectations over an extended period of time), and drift faults (performance may drift away from the original calibration formulas). A falling battery voltage will lead to calibration issues and cause the sensor to drift. Sensors with calibration error are treated as permanent faulty.

Sensor nodes may fail due to hardware problems such as poor connections or malfunctioning sensors or other embedded components. One of the prime causes of hardware faults are weather or environment conditions. As reported in [42], water contact with temperature and humidity sensors leads to a short circuit path between the power terminals which in turn cause for unusually high or low sensor readings. Electrical malfunctions may not be the only cause of hardware failure. For instance, the ion-selective electrode sensors used in soil deployments or sensors exposed to high radiation area are often prone to failures [2,43]. Such type of faults may appear continuously or intermittently.

Noise is common and expected in sensor environment, which creates random errors in sensor reading. Sensor reading is subject to several sources of noise such as noise from external sources (electromagnetic interference, atmospheric perturbation, etc.), hardware noise (white noise, low batteries, etc.) [44]. High

environmental noise influence sensing components and thus sensor nodes may capture and communicate incorrect readings where these readings can even occur outside of the feasible environmental range. Unusually high noise may be due to a hardware failure or low batteries. Faults of such a type appear intermittently and most often transient in nature.

Residual energy left in the battery relative to the minimum operating power required for sensor operation is a crucial measure of sensor health [2, 42]. Low battery levels are not only an indication of remaining lifetime of a sensor node as it can also influence sensor readings from different perspectives and cause unreliable or faulty data. Ramanathan *et al.* [45] have experimentally shown that old battery can result in significantly noise data. Their experimental results show that the standard deviation of samples within a noise window increases more than three times when used with a lower voltage battery. The fault due to law battery is continuous in nature and is treated as permanent fault.

Unlike wireless local-area networks, the path between the source and the destination in WSNs normally contains multiple wireless links (hops). The wireless links between sensor nodes are susceptible to wireless channel fading, which causes channel errors or link failure. In addition, links may fail when permanently or temporarily blocked by an external object or environmental perturbation. Such faults are always transient in nature.

1.5 Challenges of Fault Diagnosis in WSNs

The context of WSNs and the nature of sensor data make design of an efficient fault diagnosis technique more challenging. For the following reasons, conventional fault diagnosis techniques devised for wired interconnected networks [28, 46–50] might not be suitable for WSNs.

- *Resource constraints.* Limited processor bandwidth, small memory, and limited energy source are the arguable constraints in WSNs. Since the message exchange is the only means of fault diagnosis and the energy consumed by the WSN is proportional to the amount of traffic generated in diagnosing the WSN [7], the diagnosis scheme must be lightweight and

impose a negligible extra communication cost in the WSN. The diagnostic messages are advocated to be sent as the output of the routine tasks of a WSN. Accordingly, a challenge for fault diagnosis in WSNs is how to minimize the energy overhead while keeping high detection accuracy and low false alarm rate.

- *Random deployment.* Sensor nodes can be deployed by dropping from a plane, throwing by a catapult, placing in factories, and placing one by one either by a human or a robot [7]. A sparse deployment of sensor nodes is expected in underwater and volcanic data collection contrary to a dense deployment of sensor nodes in a terrestrial WSNs. Fault-free sensor nodes may be wrongly diagnosed as faulty in a threshold-based diagnosis scheme [31, 33, 34, 51] if such schemes are applied to a sparse network or a randomly deployed WSNs having sparse areas.

- *Dynamic network topology.* In this scenario, sensor node densities show large spatio-temporal variations. Dissemination of diagnostic information in such dynamic networks is extremely challenging since network connectivity is a big issue. The ability of diagnosing faults decreases under this scenario, meaning that mobility significantly reduces the quality of the diagnosis returned by a diagnosis protocol [37].

- *Attenuation and Signal loss.* The multi-hop communication in WSNs suffers from channel fading. In addition, applications like underwater WSNs, communications are established through transmission of acoustic waves [3]. In such applications, issues like limited bandwidth, long propagation delay, and signal fading make fault diagnosis more challenging.

1.6 Motivation

The fault diagnosis has been recognized by researchers as an important problem in the wired interconnected networks since the late 1960's. Although the basic principles of fault diagnosis are well understood, its application to specific domains, especially in WSNs is not well studied. However, the recent breakthroughs of

the neighbor coordination-based diagnosis in WSNs have opened enough research scopes.

Large-scale deployment of low-cost sensor nodes in uncontrolled or hostile environments is the inherent property of WSNs. It is common for the sensor nodes to become faulty and unreliable. The normal operation of a WSN suffers from faulty data since it decreases the judgment accuracy of the base station, it increases the traffic in the WSNs, and it wastes much limited energy [32]. System level diagnosis appears to be a viable solution to these problems. System level diagnosis serves as a tool that enhances data reliability, event reporting, and effective bandwidth utilization of the network. In particular, it helps in increasing the network lifetime and reconfiguring the network for better data delivery. System level fault diagnosis provides a list containing all possible faulty sensor nodes. With such a list, further recovery processes become possible, like correcting faulty readings, replacing malfunctioning sensor nodes with good ones and isolating faulty sensor nodes from a WSN that has sufficient redundancy.

Most of the classical fault diagnosis techniques found in literature do not send diagnostic messages as the output of the routine tasks of a WSN. The correct and complete diagnosis of the WSN is affected by sparse areas resulting from random deployment of sensor nodes. With intermittent faults, the diagnosis is more complicated as an intermittent faulty node may pass a test and detected as fault-free. Therefore, several test sessions may be necessary to identify the faulty nodes. On the other hand, effect of transient faults rapidly disappears. If they do not occur too frequently, removing the fault-free nodes with transient faults will affect the available resources. Isolating the sensor nodes that have been hit by transient faults is particularly not worthwhile in the operation of unattended systems like WSNs. The ability of diagnosing faults decreases if the nodes are allowed to move randomly. These issues motivate the need to design fault diagnosis algorithms to address the aforementioned problems.

1.7 Problem Statement

Motivated by the need of fault diagnosis in wireless sensor networks, the shortcomings of the present fault diagnosis techniques and keeping the research directions in view, it has been realized that there exists enough scope to improve the diagnosis performance. In this thesis, the proposed diagnosis algorithms reduce the diagnosis overhead while maintaining high detection accuracy, low false alarm rate, low diagnosis latency and low communication and energy overhead. In particular, the objectives are as follows:

1. To design and evaluate an online lightweight cluster based distributed fault diagnosis algorithm to diagnose hard and soft faults in WSNs. To devise an optimal threshold for effective fault detection in sparse networks.

2. To design and evaluate an online lightweight cluster based distributed intermittent fault diagnosis algorithm to diagnose intermittent faults in WSNs. To tune the detection parameters like inter-test interval and number of test repetitions required to diagnose the intermittent faults by modeling this problem as a multiobjective optimization problem.

3. To design and evaluate an online lightweight cluster based distributed fault diagnosis and discrimination algorithm to diagnose intermittent faults in WSNs. To device a count-and-threshold mechanism to discriminate transient from intermittent or permanent faults in WSNs.

4. To design and evaluate an online lightweight mobility aware cluster based distributed fault diagnosis algorithm to diagnose hard and soft faults in WSNs. To develop and employ mobility and energy aware clustering for fault diagnosis in dynamic environment.

5. To demonstrate the efficacy of the proposed algorithms by using the standard performance parameters like detection accuracy, false alarm rate, diagnosis latency, message overhead, network lifetime, and packet delivery ratio.

6. To validate the proposed diagnosis algorithms using Castalia-2.3b [52], a state-of-art WSN simulator based on the OMNET++ [53] platform.

1.8 Organization of The Thesis

The thesis is organized as follows—

Chapter 1: Introduction

This chapter explains the need and challenges of fault diagnosis in WSNs.

Chapter 2: Related Research

This chapter integrates the key research efforts that are available in this field. Specifically, this chapter describes the shortcomings of existing state-of-art diagnosis techniques.

Chapter 3: Hard and Soft Fault Diagnosis in WSNs

This chapter introduces an online cluster based fault diagnosis algorithm (CDFD). CDFD is shown to be energy efficient as it works in conjunction with the normal network activities and requires minimum additional diagnostic messages to be exchanged.

Chapter 4: Intermittent Fault Diagnosis in WSNs

This chapter introduces an online cluster based intermittent fault diagnosis algorithm (CDIFD). The intermittent fault detection in WSNs is formulated as a multiobjective optimization problem.

Chapter 5: Transient Fault Diagnosis in WSNs

This chapter presents an online cluster based fault diagnosis and discrimination algorithm (CDFDD). A count-and-threshold mechanism is used to discriminate transient from intermittent faults.

Chapter 6: Hard and Soft Fault Diagnosis in WSNs with Mobile Sensor Nodes

This chapter introduces mobility aware cluster-based distributed fault diagnosis (MCDFD) algorithm. Specifically, this chapter investigates mobility factor in diagnosis modeling and analysis.

Chapter 7: Conclusions

This chapter provides the concluding remarks with a stress on achievements and limitations of the proposed schemes. The scopes for further research are outlined at the end.

The contributions made in each chapter are discussed in the sequel, which include proposed schemes, their simulation results, and comparative analysis.

Chapter 2

Related Research

In this chapter, we present the state-of-art system-level diagnosis algorithms developed for wireless sensor networks (WSNs). The existing fault diagnosis algorithms for WSNs may be broadly classified into two primary types: centralized and distributed approaches [54].

In centralized approaches, a geographically or logically centralized sensor node with high computational power, larger memory size and uninterrupted energy sources undertakes the responsibility for fault detection and diagnosis of the overall WSN. The central node periodically sends diagnostic queries into the WSN to obtain the state of the individual nodes in the WSN. After analyzing the diagnostic response messages it takes a decision about failed or suspicious nodes.

The centralized approach is efficient and accurate in certain ways but adopting such approaches may not be advocated for large-scale WSNs. The reason is that it is very expensive for the base station or the sink node to accumulate information from every sensor node and identify them in a centralized manner. In addition, this leads to rapid energy depletion in certain regions of the WSN, especially the nodes closer to the base station. The detection latency is expected to be more due to the multi-hop communication in WSNs and may not be adoptable in applications with short mission time. In summary, localized and distributed generic approaches are highly preferred in WSNs as the implementation of centralized approach would place a bottleneck on performance, reduce availability, and impair expandability. For these reasons, this chapter particularly focuses on localized and distributed approaches. In this chapter, we have presented a technique-based taxonomy for

fault diagnosis in WSNs, where the fault diagnosis techniques are classified based on the nature of a test, correlation between sensor readings and characteristics of sensor nodes and the WSN.

The rest of the chapter is organized as follows. Section 2.1 presents the technique-based taxonomy framework. Section 2.2 presents variants of the distributed approaches namely test based approach, neighbor coordination approach, hierarchal approach, node-self detection approach, cluster-based approach, and probabilistic approach. Finally, this chapter is summarized in Section 2.3.

2.1 Technique-based Taxonomy Framework

Since WSNs become popular in scientific research, many fault detection and diagnosis techniques specifically developed for WSNs have emerged. As illustrated in Figure 2.1, fault diagnosis techniques for WSNs can be broadly categorized into centralized and distributed approaches. In centralized approaches, often it is assumed that a supervising arbiter (sink node or base station) is available to analyze the diagnostic messages and disseminate diagnostic information. The implementation of such an approach would place a bottleneck on the network lifetime. For these reasons, distributed diagnosis has been introduced and studied. In distributed approaches, each sensor node executes the fault detection algorithm and generates a fault local-view. The fault local-view is the view of a sensor node regarding the fault states of its one-hop neighbors. This local-view is then disseminated in the network such that each fault-free sensor node correctly diagnoses the state of all the sensor nodes in the WSN.

Application of test and message passing is the only means to detect faults in distributed approaches. In this section, we identify and discuss several important aspects to further categorize the distributed approach.

Correlation

The faults in a WSN can be detected by exploring temporal, spatial, or spatio-temporal correlations between sensor readings. Sensor data are correlated

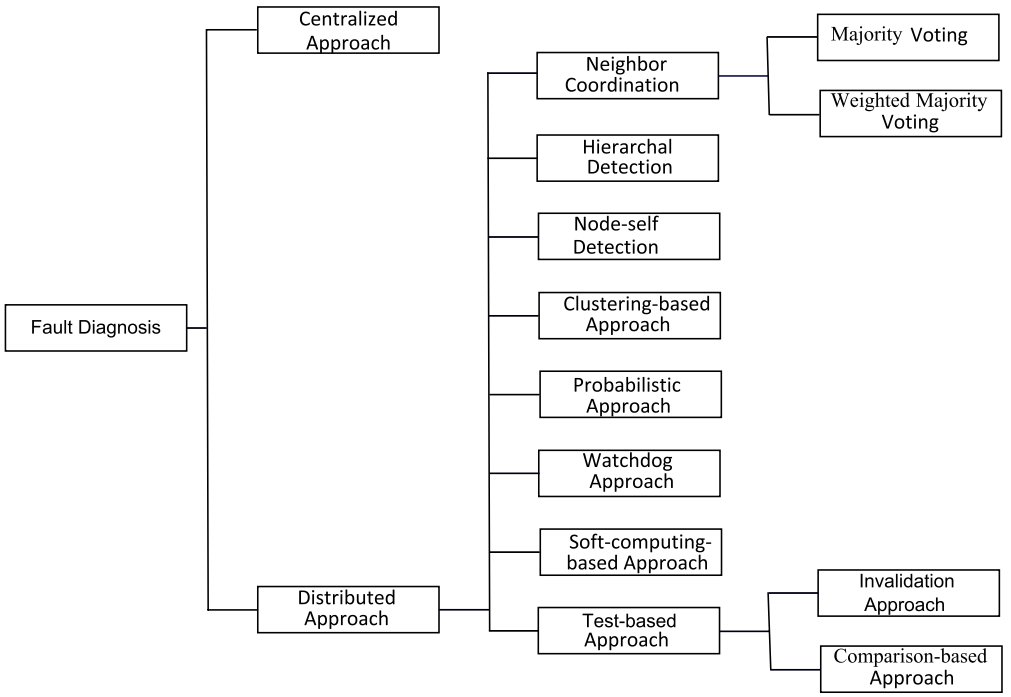


Figure 2.1: Taxonomy framework for fault diagnosis techniques.

in both time and space. Sensor readings are temporally correlated since the readings observed at one time instant are related to the readings observed at the previous time instants. Similarly, sensor readings are spatially correlated since the readings from sensor nodes geographically close to each other are expected to be identical. The faults in a WSN can be detected by exploring temporal, spatial, or spatio-temporal correlations between sensor readings. Neighbor coordination approaches explore these correlations to detect faults. The diagnosis efficiency of these algorithms is proportional to the average node degree of the WSN.

The Nature of Tests

Exchange of test messages is a good alternative to detect faulty nodes in a WSN. A sensor node v_i tests the sensor node v_j by sending a test message and comparing the resulting outputs with some set of correct responses. Based on the nature of test and comparison, fault diagnosis techniques can be classified. A group of sensor nodes may test a sensor node (invalidation approach) to take a decision. On the other hand, the same test task may be assigned to sensor nodes, and decision can be taken based on agreement or disagreement among sensor nodes on the obtained

results (comparison-based approach).

Testing may be performed by each sensor node on itself (self-test) by hardware or software checkers, watchdog timers, or error-detecting codes. Thus, faults can be detected using minimal network resources. In this approach, all free time at sensor node v_i may be utilized to test itself with a predefined test or a test provided by its one-hop neighbor v_j.

Communication Cost

In many diagnosis approaches, the communication and energy overhead is too high. Dissemination of local decision at each node contributes more to this overhead. Cluster-based and hierarchal approaches address this problem. In a cluster-based approach, the diagnosis algorithm works in conjunction with the underlying clustering algorithm. In this approach, the deployment area is divided into a number of clusters. Each cluster is headed with a manager node or cluster head. The head node tests its cluster members and constructs a cluster level local-view or each cluster member constructs their local-view using neighbor coordination or test-based approaches and conveys it to the respective cluster heads. A hierarchal structure of cluster heads can be used to disseminate the cluster level local-views. In hierarchal approaches, the parent nodes in the hierarchal structure tests for faults in its descendant. The same hierarchal structure is used to disseminate the decision made at each node.

Characteristics of Sensor Node and Network

Apart from the temporal, spatial and spatio-temporal correlations between sensor reading characteristics like node dynamics and node degree can be used to predict faults in WSNs. Nodes can estimate its true reading using these characteristics and soft-computing-based approaches. The node interconnection can be explored to detect faults.

Based on the aforementioned aspects, the distributed approach can be categorized into neighbor coordination, hierarchal, node-self detection, watchdog, test-based, clustering-based, soft-computing-based and probability-based approaches. Neighbor coordination approach can be further categorized into

majority-voting and weighted-majority-voting approach. In addition, test-based approach can be further categorized into invalidation and comparison-based approach.

2.2 Distributed Approaches

Different practical applications may require the fault diagnosis to be conducted in a real-time mode with a low latency, low message overhead and high throughput. Therefore, the development of diagnosis approaches should aim to address these issues in addition to the aforementioned limitations of centralized approaches. Distributed approaches address these issues and limitations. The working sensor nodes perform their own independent diagnosis of the WSN. In these approaches, every sensor node decides independently the state of the WSN. Here, a sensor node makes decisions at certain levels by monitoring behavior locally. The more decisions a sensor node can make, the less information (i.e., number of messages) must be delivered to the central node. As a result, these approaches conserve the node energy and consequently, prolong the network lifetime [54]. This allows the diagnostic framework to scale easily to much larger and denser WSNs.

2.2.1 Test-based Approaches

In test-based approaches, tasks are assigned to sensor nodes, and the test results are the basis for identifying the faulty sensor nodes. Based on the type of test this approach is further categorized into invalidation and comparison-based approaches.

Invalidation Approaches

In these approaches, every sensor node tests a set of sensor nodes and passes the test result to other sensor nodes. Based on these results a consensus can be made regarding the correct fault set. In 1967 Preparata, Metze, and Chien (PMC) introduced system-level diagnosis for wired interconnected networks with their well-known PMC model [26]. In their model a processing element tests other processing elements and that the results are used to find the state of the system.

However, test results may not be reliable if the testing processing element is faulty. In order to diagnose t processing elements of a n processing element system, n must be greater than or equal to $2t + 1$, and a processing element must be tested by at least t other processing elements. Later, Hakimi and Amin [30] characterize the PMC model stating that these two conditions are necessary and sufficient for t-diagnosability provided there are no reciprocal tests, i.e., no two units test each other. They suggest a third diagnosability condition for the scenario where the processing elements test each other. They argue that for a digraph G constitutes of n processing elements; if $K(G) \geq t$, then the system is t-diagnosable, where $K(G)$ is the connectivity of G. Variations of this PMC or invalidation approach are applied in WSNs to diagnose faults.

Chessa and Santi [38] propose a fault diagnosis algorithm namely WSNDiag. In this approach, in response to an explicit request of an external operator a unique fault-free sensor node called the initiator initiates the detection process. Two types of messages are exchanged during its execution: *I'm alive* (IMA) messages, and diagnostic messages. A tree spanning all fault-free sensor nodes is constructed during the propagation of IMA messages. After a timeout time T_{out}, sensor nodes that did not reply with their IMA message are diagnosed as faulty. Once a sensor node v_i has its local diagnostic view (i.e., the state of its one-hop neighbors), it waits for the local-views of its children in the spanning tree. Once these local-views have been received, a sensor node updates its view and then selectively sends this updated view to its parent in the spanning tree. Once the initiator receives local-views from all of its children in the spanning tree, it generates the global view by combining these local-views. The global view is then disseminated downward in the spanning tree using a broadcast protocol in order to ensure that each sensor node will have a global view.

Weber *et al.* [55] considers the problem of determining a test strategy of the sensors in a WSN in order to ensure a desired level of diagnosability of the system. They propose a strategy of mutual tests among the sensor nodes in a region where t numbers of faulty sensor nodes are present such that the system graph representing the region of the WSN is t-diagnosable. Thus, this approach depends on the node degree, i.e., depends on WSN topology. Since the diagnosability of a diagnostic

graph depends on whether the graph defines reciprocal tests among units or not, they discuss two strategies namely testing strategy without reciprocal tests and testing strategy with reciprocal tests.

Weber *et al.* present a diagnosis approach namely energy-efficient test assignment (EETA) without reciprocal tests [56] which is based on their previous work [55]. This approach presents a heuristic that chooses the set of sensors to be involved in the tests in order to meet the conditions presented in PMC model.

Comparison Approaches

Chessa *et al.* [37] present two implementations of the comparison-based diagnosis model, which detects both hard and soft faults under the hypothesis of fixed and time-varying topology in ad-hoc networks. In this approach, a fault-free unit v_i (the testing unit) tests its neighbors sending them a test request message $m = (v_i, q, Test_q)$ and waiting for their responses where $Test_q$ is the q^{th} test task. At the same time, it generates the expected result $R_{v_i,q}$. Upon receiving m a node $v_j \in N(v_i)$ generates the result $R_{v_j,q}$ for $Test_q$ and sends response message $m' = (v_i, q, R_{v_j,q})$ at time T', with $T_0 < T' < T_0 + T_{out}$. $N(v_i)$ is the neighbor set of a node v_i. The timeout (T_{out}) is chosen in such a way that all the fault-free one-hop neighbors which do not migrate out of its transmitting range are guaranteed to respond to the test request within that time. As the responses are received, the generated test result is compared with the received test result, and the nodes are detected based on the comparison rule.

The diagnosis techniques proposed by Elhadef *et al.* namely Adaptive-DSDP [57], Mobile-DSDP [57] and Dynamic-DSDP [20] are based on Chessa and Santi's model [37] and WSNDiag [38]. Similar to Chessa and Santi's model and WSNDiag, these approaches can diagnose at most $K-1$ nodes where K is the connectivity of the network. Unlike Chessa and Santi's model, Adaptive-DSDP doses not include the test task when it replies to a test request. A similar test task is executed in each node contrary to different test tasks as in Chessa and Santi's model. In order to test their neighbors, nodes send test tasks either periodically or when abnormal behavior is detected. Once any of the other nodes receive a test task or overhear a response message, it initiates its own testing phase by generating its own test

message. Once a node collects responses of all its one-hop neighbors, it determines its local diagnostic view by comparing their outputs for the identical test task.

Adaptive-DSDP uses a spanning tree to disseminate the local-views where the dissemination starts at the leaf nodes. On the other hand, Chessa and Santi's model uses a flooding-based dissemination strategy. Mobile-DSDP follows Chessa and Santi's model by including the test task with the response message. The rational are that even if the node v_i can no longer reach its old neighbor v_j (the tester), then any other node in its new neighborhood will be able to diagnose its status given that it has provided both the test task and its output for that task. Similar to Chessa and Santi's mode, it uses the flooding-based dissemination to disseminate local diagnostics. In this approach, two timers are used. The first timer is set to T_{out} which is used to detect stable one-hop neighbors. The second timer namely $T_{DiagnosisSession}$ is used to detect nodes those remain undiagnosed due to mobility. In Chessa and Santi's model, Adaptive-DSDP and Mobile-DSDP, every node should reply to any test request it receives. However, in Dynamic-DSDP, each node is required to respond to exactly K test requests. This is feasible since it deals with $K-1$ diagnosable networks. Dynamic-DSDP uses a spanning tree in order to disseminate the local diagnostic views gathered separately by the nodes.

In invalidation approach, in order a system to be t-diagnosable a node must be tested by t number of neighbors. This in turn increases the message overhead. In both invalidation and comparison-based approaches, the applied test is not sufficient enough to check the state of the sensing elements. The reason is that these approaches rely only on the test response generated by the testee sensor node. The test response does not carry any information about the sensing elements.

2.2.2 Neighbor Coordination Approaches

Unlike test-based approaches, a node takes a decision about whether or not to disregard its own sensor reading based on the sensor readings from its one-hop neighbors or based on weights like physical distances from the event, trustworthiness and their measurements, etc. Based on these attributes neighbor coordination approach is further categorized into majority-voting and weighted-

majority-voting approach. In neighbor coordination approaches, sensor nodes co-ordinate with their neighbors (usually one-hop neighbors) to detect faulty sensor nodes before conferring with the central node. Therefore, this design reduces the number of messages and subsequently, conserve sensor node energy.

Majority-voting approaches

This approach exploits the fact that the faulty measurements are uncorrelated while the normal measurements are spatially correlated. This means; readings from faulty sensors are geographically independent while readings from sensors in close proximity are spatially correlated [29]. For example, let v_i be a neighbor of v_j, x_i and x_j are the sensor readings of v_i and v_j respectively. Sensor reading x_i is similar to x_j when $|x_i - x_j| < \delta$ where δ is application dependent. As an illustration, in bolt loosening monitoring, a sensor node and its neighbors are expected to have similar voltage. Similarly, in the case of temperature, a sensor node and its neighbors are expected to have similar temperature reading. Hence δ is expected to be as a small number. The fundamental principle of this approach is to compare a sensor node v_i's measurement with $v_j \in N(v_i)$ and find $Result_{ij} \in \{0, 1\}$. As shown in Figure 2.2, $Result_{ij} = 0$ if $|x_i - x_j| < \delta$. Otherwise, $Result_{ij} = 1$. This approach estimates the fault state of v_i by comparing the number of 0s with a predefined threshold.

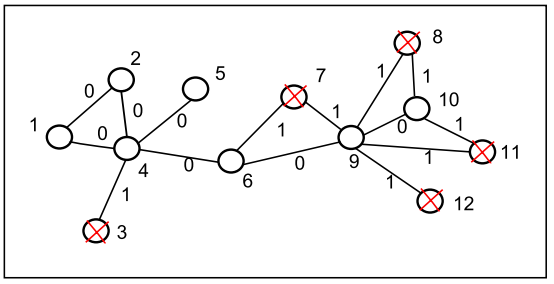


Figure 2.2: Illustration of comparison result. Crossed sensor nodes are faulty.

Chen *et al.* [31] propose a localized distributed fault detection (DFD) algorithm to identify the faulty sensors. It uses local comparisons with a modified majority voting, where each sensor node makes a decision based on comparisons between its own sensor reading (such as temperature) and sensor readings of one-hop neighbors. DFD algorithm consists of four rounds of tests. In the first round,

a test result $Result_{ij} \in \{0, 1\}$ is generated by sensor v_i based on its neighbor v_j's measurement using two variables namely $m_{ij}^{T_l}$ and $\Delta m_{ij}^{\Delta T_l}$, and two predefined threshold values Φ_1 and Φ_2. The measurement difference between v_i and v_j from time T_l to T_{l+1} is defined as

$$\Delta m_{ij}^{\Delta T_l} = m_{ij}^{T_{l+1}} - m_{ij}^{T_l} = (x_i^{T_{l+1}} - x_j^{T_{l+1}}) - (x_i^{T_l} - x_j^{T_l})$$

where $x_i^{T_l}$ is the reading of v_i at time T_l. In DFD algorithm, for any node $v_j \in N(v_i)$, the node v_i first set $Result_{ij}$ to 0. This algorithm next calculates $m_{ij}^{T_l}$. If $|m_{ij}^{T_l}| > \Phi_1$ then it calculates $\Delta m_{ij}^{\Delta T_l}$. The comparison test result $Result_{ij}$ is set to 1 if $|\Delta m_{ij}^{\Delta T_l}| > \Phi_2$. If $Result_{ij}$ is 0, most likely either both v_i and v_j are good or both are faulty. Otherwise, if $Result_{ij}$ is 1, v_i and v_j are most likely in different status. In this approach, for any sensor node v_i, its test results with each sensor node in the neighbor set $N(v_i)$ is obtained. If there are more than $\lceil |N(v_i)|/2 \rceil$ sensor nodes whose comparison test results are 1 in $N(v_i)$, then initial detection status (i.e., tendency value $Tend_i$) of sensor node v_i is possibly faulty (LT), otherwise, it may be possibly normal (LG), i.e.,

$$Tend_i = \begin{cases} \text{LT} & \text{if } \sum\limits_{v_j \in N(v_i)} Result_{ij} \geq \lceil |N(v_i)|/2 \rceil \\ \text{LG} & \text{otherwise} \end{cases} \tag{2.1}$$

where $|N(v_i)|$ represents the number of one-hop neighbors of v_i. Each sensor node sends its tendency value to all its neighbors. When the initial detection status of all nodes in the WSN is obtained, in the second round of test of DFD algorithm, the number of LG nodes whose test result with v_i is 1 is subtracted from the number of LG nodes whose test result with v_i is 0. If the result is greater than or equals to $\lceil |N(v_i)|/2 \rceil$, then v_i is detected as fault-free. That is $\forall v_j \in N(v_i)$ and $Tend_j = LG$, $\sum(1 - Result_{ij}) - \sum Result_{ij} = \sum(1 - 2Result_{ij})$ must be greater or equal to $\lceil |N(v_i)|/2 \rceil$ to detect v_i as fault-free. This can be defined as

$$v_i = \begin{cases} \text{fault-free(GD)} & \text{if } \sum\limits_{v_j \in N(v_i), Tend_j = LG} (1 - 2Result_{ij}) \geq \lceil |N(v_i)|/2 \rceil \\ \text{Undetermined} & \text{otherwise} \end{cases} \tag{2.2}$$

A sensor node v_i that has failed to pass the threshold test of equation (2.2) is marked as *undetermined* and follows a third round of test. All the *undetermined*

nodes repeat to check for $\log n$ times in the best case ($\sqrt{n}$ in average case and n times in the worst case) if one of its neighbors is determined to be fault-free. If such a sensor node exists and $Result_{ij} = 0(1)$ then v_i detected as fault-free (faulty).

If still ambiguity occurs, in fourth round of test, the sensor's own tendency value determines its status. For instance, if the status of $v_j, v_k \in N(v_i)$ are determined as fault free (i.e., $Tend_j = Tend_k = GD$), v_i is marked as *undetermined* and $Result_{ji} \neq Result_{ki}$ then v_i will be detected as fault-free (faulty) if $Tend_i = LG(FT)$.

Jiang [34] claims an improvement over the DFD algorithm by introducing an improved distributed fault detection scheme (improved-DFD). In this approach a node v_i first set $Result_{ij}$ to 0 for any node $v_j \in N(v_i)$. Improved-DFD algorithm then calculates $m_{ij}^{T_l}$ and if $|m_{ij}^{T_l}| > \Phi_1$ then comparison test result $Result_{ij}$ is set to 1. If $|m_{ij}^{T_l}| \leq \Phi_1$ then it calculates $\Delta m_{ij}^{\Delta T_l}$. The comparison test result $Result_{ij}$ is set to 1 if $|\Delta m_{ij}^{\Delta T_l}| > \Phi_2$. This algorithm then follows equation (2.1) to determine the initial detection status (i.e., LG or LT) of the nodes. In this approach, for any sensor node v_i and the sensor nodes in $N(v_i)$ whose initial detection status is LG, if the sensor node whose test result with v_i is 0 is not less than the nodes whose test result is 1, then the status of v_i is GD. Otherwise, the status of v_i is FT. Alternatively this can be explained as

$$v_i = \begin{cases} \text{GD} & \text{if } \sum_{j \in N(v_i), Tend_j = LG} Result_{ij} < \lceil |N(v_i)_{Tend_j = LG}|/2 \rceil \\ \text{FT} & \text{otherwise} \end{cases} \tag{2.3}$$

If there are no neighbor nodes of v_i whose initial detection status is LG, and if the initial detection status $Tend_i$ of v_i is LG, then improved-DFD sets the status of v_i to GD, otherwise to FT.

Lee and Choi [33] approached WSN fault detection problems where time redundancy is used to tolerate transient faults in sensing and communication. A sliding window is employed to eliminate delay involved in the time redundancy. A label x_{ij} is associated with $(v_i, v_j) \in E$ and is set to 0 if $|x_i - x_j| < \delta$. Otherwise, $x_{ij} = 1$. Here E is the set of communication edges. To cope with transient faults,

x_{ij} is obtained at regular intervals for ϱ number of times, where ϱ is a small positive integer. A node v_i represents these comparison results for all $v_j \in N(v_i)$ in a $|N(v_i)| \times \varrho$ matrix $M = [x_{ij}^k]$, where $k = 1, 2, \cdots, \varrho$. The detection algorithm uses two threshold values such as Φ_3 and Φ_4. If $\sum_{k=1}^{\varrho} x_{ij}^k \leq (\varrho - \Phi_4)$, the status variable $Result_{ij}$ is set to 0. The number of 0s in $Result_{ij}$ is denoted by $|Result_i|$. If $|Result_i| \geq \Phi_3$, fault-state of v_i is set to 0 (fault-free) and this decision is broadcasted. A fault-free node v_i failed to pass the threshold test is later diagnosed as fault-free through a fault-free neighbor v_k. For relatively high fault rates, both DA and FAR increase with Φ_3.

Choi *et al.* [35] present an adaptive fault detection scheme that closely follows [33]. They have suggested time redundancy to tolerate transient faults. In their approach, the diagnosis parameters such as effective node degree d_t and decision threshold Φ_4 are dynamically updated. Hsin *et al.* [58] suggest a two-phase neighbor coordination scheme. In the first phase, a sensor node waits for its neighbors to update information regarding the faulty nodes. In the second phase, it consults with its neighbors to reach a more accurate decision. In this approach, two timers are maintained for monitoring a sensor node v_i, with values $C1$ and $C2$ respectively. If a sensor node $v_j \in N(v_i)$ does not receive any packets from v_i before $C1(v_i)$ expires, v_j activates the second timer $C2(v_i)$. During the second timer period, v_j will query the common neighbors regarding the status of v_i and take a decision accordingly.

Miao *et al.* [36] present a failure detection scheme namely Agnostic Diagnosis (AD). This approach is motivated by the fact that the system metrics (e.g., radio-on time, number of packets transmitted) of sensors usually exhibit certain correlation patterns. This approach collects 22 types of metrics from each sensor node that are classified into four categories. AD exploits the correlations between metrics of each sensor using a correlation graph that describes the status of the sensor node. By mining through the periodically updated correlation graphs, abnormal correlations are detected.

Krishnamachari and Iyengar [51] explicitly consider measurement faults and develop a distributed and localized Bayesian algorithm for detecting and correcting such faults. They propose three different detection schemes namely the randomize

decision scheme, the threshold decision scheme, and the optimal threshold decision scheme. The real situation at a sensor node is modeled by a binary variable GT_i. $GT_i = 0$ if the ground truth is that the sensor node is in a normal region and $GT_i = 1$ if the ground truth is that the sensor node is in an event region. The real output of the sensor node v_i set to zero (i.e., $S_i = 0$) if the sensor measurement indicates a normal value and set to one (i.e., $S_i = 1$) if it measures an unusual value. There are thus four possible scenarios: $S_i = 0$; $G_i = 0$ (sensor correctly reports a normal reading), $S_i = 0$; $G_i = 1$ (sensor faultily reports a normal reading), $S_i = 1$; $G_i = 1$ (sensor correctly reports an unusual/event reading), and $S_i = 1$; $GT_i = 0$ (sensor faultily reports an unusual reading). A sensor node makes a decision about whether or not to disregard its own sensor reading S_i in the face of the evidence $E_i(a, k)$ from its neighbors. $E_i(a, k)$ is defined such that k of N one-hop neighbors report the same binary reading a as sensor node v_i.

This work is further extended by Luo *et al.* [59]. They consider both measurement errors and sensor faults in the detection task. Under a given detection error bound, minimum neighbors are selected to minimize the communication overhead. Both Bayesian and Neyman-Pearson detection methods are presented. Their approach did not explicitly attempt to detect faulty sensors, instead their schemes improve the event detection accuracy in the presence of faulty sensors.

Yim and Choi [60] propose an adaptive fault-tolerant event detection scheme. This approach employs a filter for tolerating transient faults. The threshold for event detection is dynamically adjusted depending on the fault status of sensor nodes. Confidence levels are used to manage the status of sensor nodes. The confidence levels are updated each time a fault or event is detected.

The majority-voting techniques have the potential to enhance the detection performance from both detection accuracy and false alarm rate perspectives. The performances of these techniques are worst affected by low average node degree. However, researchers argue a better performance due to the expected high average node degree in WSNs. This hypothesis may not be always correct. The primary reason for not achieving an extremely good performance for a low average node degree is that the fault-free sensor nodes are unlikely to pass the threshold test.

Therefore, a better majority-voting scheme should be formulated by finding an optimal threshold such that it will outperform in sparse WSNs or WSN with sparse areas.

Weighted-voting approaches

Unlike majority-voting approaches, the weighted-voting approaches use properties such as physical distances from the event, trustworthiness, measurements, etc. as weight. These weights are used to take decision regarding the state of a sensor node.

Xiao *et al.* [61] present an in-network voting scheme that determines faulty sensor readings in WSN by considering both the correlation of readings between sensor nodes and the trustworthiness of a sensor node. Each sensor node is associated with a *SensorRank* which is used in voting. *SensorRank* represents the trustworthiness of sensor nodes. In this approach, if a sensor has a large number of neighbors with correlated readings, its vote deserves more weight and a sensor node with a lot of trustworthy neighbors is also trustworthy. This trust voting algorithm consists of two phases such as the self-diagnosis phase and the neighbors diagnosis phase.

Guo *et al.* [62] propose a detection scheme namely FIND that detects sensor nodes with data faults. It ranks the sensor nodes based on their measurements as well as their physical distances from the event. The authors have shown experimentally that the sensor measurements monotonically change as the distance becomes further from the event. A sensor node is detected faulty if there is a significant mismatch between the sensor data rank, and its readings violate the distance monotonicity significantly.

The weighted-voting approaches inherit the advantages of majority-voting approach. However, the computational complexities of these approaches are more. Similar to majority-voting approaches, the weighted-majority-voting approaches show poor performance in sparse WSNs or WSN with sparse areas.

2.2.3 Hierarchal Detection Approaches

The basic idea of hierarchal detection is to first construct a spanning tree rooted at the sink node or base station that spans all the fault-free nodes in the network and next to decide on faults at each level of the tree. The spanning tree is used to disseminate the decision taken at each node in the WSN such that each node in the WSN correctly diagnoses the fault states of all nodes in the WSN.

Rost and Balakrishnan [63] propose a detection algorithm namely Memento. In this approach, the sensor nodes in the WSN cooperatively monitor each other to implement distributed sensor node failure detection. The failure of any sensor node is monitored by a number of other sensor nodes in its vicinity. Memento requires two components for fault detection such as heartbeats and a failure detector. Each sensor node periodically sends heartbeat messages. A failure detector running on a different sensor node detects a sensor node as faulty if a certain amount of time expires since the receipt of that sensor node's last heartbeat. In this approach, a failure detector generates a liveness bitmap. This bitmap summarizes detector's current belief in the liveness of neighbors. The child sensor nodes send their bit patterns to their parent sensor nodes. The parent performs an aggregation (bitwise OR) operation on the results of the child sensor nodes together with its own results and forwards it to the tree ancestor. The sink can then compare the list of fault-free sensor nodes with the roster of deployment to determine which of the sensor nodes have failed.

Gheorghe *et al.* [64] suggest an adaptive trust management protocol (ATMP). This protocol uses cooperative trust management and has a hierarchical view over the WSN. The protocol operates in three phases such as the setup, learning and an exchange phase. In the setup phase, a spanning tree is constructed. In the learning phase, the local penalty value is modified based on the fault detection techniques. In the exchange phase, sensor nodes exchange reputation values, recompute them and determine trust. Reputation can be defined as an expectation about an individual's behavior based on information about or observations of its past behavior [65]. To perform error detection, leaf sensor nodes transmit the sensor measurements. Every other sensor node within the spanning tree waits

to receive sensor measurements from children sensor nodes for a predefined time. After the waiting period, the sensor nodes are grouped in clusters based on the location of each source of data. Each cluster of nodes is represented by a list of measurement values generated by member nodes, and each list of nodes is sorted in an ascending manner. For each list of values, the median value is computed. For each list, the values are compared with the median value. If the difference between the considered value and the median is greater than a constant deviation, the value will be considered erroneous.

These approaches suffer from relatively high detection latency. This is because the diagnosis process is either started by the sink node or the leaf nodes. In addition, similar to centralized approach this approach leads to rapid energy depletion in certain regions of the WSN, especially the sensor nodes closer to the sink. This may lead to the hot spot problem [4].

2.2.4 Node Self-detection Approaches

In these approaches, the sensor node architecture is self-competent of detecting its own status. This is achieved by including additional hardware to the sensor node architecture.

Harte *et al.* [66] propose a self-detection architecture to monitor faults in components of a sensor node. Both hardware and software interfaces are used. The hardware interface consists of a number of miniature accelerometers mounted on a flexible printed circuit board. This acts as a sensing layer around a sensor node to detect the orientation and impact on the sensor node. It also introduces some redundancy into the design to cope with damaged accelerometers. In order to sample sensor nodes' reading, this design adopts several software components (e.g., ADCC, TimerC) from TinyOS operating system.

Koushanfar *et al.* [16] propose self-detection of sensor nodes in WSNs. This approach observes the binary outputs of its sensors by comparing with the predefined fault models. Faults caused by battery exhaustion can be estimated when the hardware is competent to measure the current battery voltage [67, 68]. A detection algorithm can determine an estimation of the time to failure of the battery by analyzing the battery discharge curve, and the current discharge rate.

Wireless sensor node architecture is expected to be simple and energy efficient. Node self-detection approach needs extra hardware which in turn increase the hardware complexity, weight and energy requirement.

2.2.5 Cluster-based Approaches

The cluster-based approaches create a virtual communication backbone to group sensor nodes and split the overall WSN into different groups (e.g., clusters). Fault detection is normally distributed and executed in each individual group. Usually, the leader node of a cluster (e.g., the cluster head) executes fault detection in its group using a centralized or distributed approach.

Gupta *et al.* [69] assume a fail silent model where any erroneous behavior does not affect the healthy components. In their fault tolerant clustering scheme, failure detection of the cluster head is investigated. This approach adopts a method of periodic status updates through inter-cluster communication. Along with the sensed data, sensor nodes provide their energy status to the cluster head. Once the sensed data and energy status of affiliated member sensor nodes are obtained, a cluster head constructs a *Status* containing information about the sensor nodes in its cluster, and the status of the cluster head itself. In the *Status Update* slot, the statuses of cluster heads are exchanged. At the end of detection phase, a cluster head CH_i believes that CH_j is faulty if it does not receive the update from CH_j. Since the updates can be missed due to link failures between two sensor nodes, before taking any decision, CH_i consults the consensus derived by all cluster heads.

Jaikaeo *et al.* [70] propose the sensor information networking architecture (SINA). Their approach consists of mechanisms for hierarchical clustering, attribute-based naming, and querying and tasking supports. The manager node issues a script programmed in the SQTL language to all cluster heads to diagnose the sensor nodes. Upon receiving this script, cluster heads trigger their associated members for temperature readings. Cluster heads then compare the difference between each reading and the average reading of all the associated members with a predefined threshold. Member sensor nodes those failed to pass the threshold test are identified as faulty sensor nodes.

Tai *et al.* [71] propose a heartbeat-style failure detection service for the middle-ware implementation. This approach exploits the inherent message redundancy of ad-hoc networks. This approach is coupled with cluster-based communication architecture. The fault diagnosis is achieved by exchanging three types of messages namely *Heartbeat message*, *Digest message* and *Health-status-update message*.

Ossama *et al.* [72] suggest an approach in which a cluster head periodically broadcasts a heartbeat message to inform its members that it is still functional. Upon not receiving any heartbeats from its cluster head, a member sensor node detects that its cluster head is faulty. In a WSN if member sensor nodes go through a duty cycle, they cannot hear periodic cluster head heartbeats. This approach addresses this issue where a member sensor node can solicit a heartbeat from its cluster head after sending a certain number of messages. Cluster heads detect neighboring cluster head failures using routing updates.

Wang *et al.* [73] propose an agreement-based fault detection mechanism for detecting cluster-head failures in clustered underwater sensor networks. Periodically, it performs a distributed detection process at each cluster member. This requires each cluster member in a cluster to maintain a status vector, in which each bit corresponds to a cluster member and is initialized to zero. A bit in the vector is set to one once its corresponding cluster member detects that the cluster head has failed. If all elements of the status vector of a cluster member become one, an agreement is reached and the cluster member takes a decision.

Venkataraman *et al.* [74] propose an approach in which the sensor nodes detect the energy failures in their respective clusters. In this approach, every sensor node has a record of its balance energy. The sensor nodes in each cluster embed their current energy status in the *hello message* and send to their first hop members, including their parent. The *hello message* consists of the location, energy and ID of the sensor node. A sensor node sends the failure report message to its parent and children when its energy level drops below a threshold value. The threshold value is the energy required to transmit D number of messages across a distance equal to the transmission range. D is the maximum number of one-hop sensor nodes selected during clustering.

Asim *et al.* [75] suggest a cellular-based approach where the cell manager (cluster head) and gateway nodes coordinate with each other to detect faults. In this approach, the cell manager sends *get messages* periodically to the associated member sensor nodes and gateway node and in return they send their *updates*. An *update* includes sensor node ID and energy level. Upon not receiving *update* from any sensor node, it sends an instant message to the sensor node and acquires its status. If a cell manager does not receive the acknowledgment in bounded time, it declares the sensor node as faulty and conveys this decision to other nodes in the WSN. Wei *et al.* [76] suggest a cluster-based real-time fault diagnosis aggregation algorithm (CRFDA). It closely follows [37] where the diagnostic tasks are assigned to the cluster members by the affiliating cluster head. The cluster head takes a decision by comparing the test results sent by its member sensor nodes.

Kazi [77] proposes an asynchronous failed sensor node detection (AFSD) method. In this approach, separate detection protocols are assigned to cluster heads and cluster members. A numeric counter variable called failure counter is used to track the received and sent data packets between active sensor nodes. AFSD modifies the failure counter such that for a fault-free sensor node, the value of the counter is bounded and tends to zero. For a failed sensor node, the value of this counter is unbounded and tends to infinity and eventually will cross a predefined threshold.

2.2.6 Soft-computing-based Approaches

The soft-computing-based approaches use the characteristics of sensor nodes and the WSN to detect faults in the sensor nodes.

Zhang Ji *et al.* [78] exploit the redundant or complementary information of multi-sensor in space or time to detect and isolate the faulty sensor nodes in WSN. The authors present a structure of three-layers to detect and isolate multiple faults. The first layer is a state recognition network. It is composed of some modularity radial basis function neural network (RBFNN). The belief assignment of a sensor state is obtained by RBFNN with two-input and one-output. The two inputs are the data provided by sensor v_i and v_j. The output is $m_{ij}(\{OK_i, OK_j\})$. Here, $m_{ij}(OK)$ means both v_i and v_j are fault-free. Each trained RBFNN is used as

one model. The second layer is merging of the different frames of discernment. These frames of discernment are merged to a common frame of discernment by refinement operation. The third layer is evidence fusion and state decision.

Jabbari *et al.* [79] present the fault detection and isolation technique based on artificial neural network (ANN). This approach follows two phases namely residual generation and residual verification phases. Two separate ANN algorithms are considered for these phases respectively. This approach compares the measured data with network prediction and generates fault residuals. All the residuals are evaluated and analyzed. A residual is a signal that is used as a fault detector. Normally, the residual is considered to be zero (or small in a realistic case where the process is subjected to noise and the model is uncertain) in the fault-free case and deviate significantly from zero when a fault occurs. For generating residuals, it considers generalized regression neural network architecture data approximation. In this phase, measurement residuals are generated by comparing measured data with network prediction. In second phase, it uses a probabilistic neural network (PNN) for analyzing probable fault/failure conditions and fault/failure classification.

Azzam and Rastko [80] introduced a neural network modeling approach for sensor node identification and fault detection in WSNs. The recurrent neural networks (RRNs) have the ability to capture and model the dynamic properties of nonlinear systems. In this approach, RRNs are used to model the sensor node, the node's dynamics, and interconnecting with other WSN nodes. The RRN nodes have their own dynamics with interconnecting weights between the nodes similar to WSNs, and each sensor node has its own dynamics. The dynamic RRNs consist of a set of dynamic nodes that provide internal feedback to their own inputs. This is used to simulate a network of sensors. This approach assumes that there is one sensor per sensor node where the sensor nodes are viewed as small dynamic systems with memory-like features. The introduced ad-hoc RRN is analogous to WSN systems with confidence factors ($0 < CF_{ij} < 1$) between sensor nodes v_i and v_j. The confidence factor depends on the signal strength and the data quality in communication links between the nodes. The overall modeling process is divided into two phases such as the learning phase and the production phase.

In the learning phase, the neural network adjusts its weights that correspond to the healthy and N faulty models. The production phase compares the current output of the sensor node with the output of the neural network. The difference between these two signals is the basis to detect a sensor's health status. Barron *et al.* [81] implement this approach on Moteiv's Tmote Sky platform with TinyOS operating system.

In these approaches, the computational complexity is high, which may lead to more energy overheads.

2.2.7 Watchdog Approaches

The simplest case of monitoring the network for faults is the watchdog mechanism. The basic principle of the watchdog is to monitor whether a node's one-hop neighbor forwards the packets just sent by overhearing. If its one-hop neighbor fails in forwarding within a certain period, the neighbor is viewed as misbehaving node. When the misbehaving rate exceeds a predefined threshold, the source is notified and the following packets are forwarded along other routes.

Marti *et al.* [82] propose an intrusion-detection system for wireless ad-hoc networks. Their approach focuses on intrusion prevention methods by introducing two overlays to the dynamic source routing (DSR) algorithm. The proposed system consists of two tools to detect and mitigate abnormal routing behavior. The Watchdog tool identifies the misbehaving nodes. The *Pathrater* aids the routing protocol in avoiding the misbehaving nodes. When a node forwards a packet, the node's watchdog verifies that the next node in the path also forwards the packet. The watchdog does this by listening promiscuously to the next node's transmissions. If the next node does not forward the packet, then it is misbehaving.

Marti's approach fails to detect misbehaving nodes in a number of scenarios. For example, nodes with malicious intent might falsely report other nodes as misbehaving. A more sophisticated attack that this model cannot detect occurs when multiple nodes collude to bring the network down. Patcha and Mishra [83] have extended Marti's approach. In their approach, collaborating groups of malicious nodes were considered. This approach classifies the nodes in the network into trusted and ordinary nodes. The first few nodes that form a network

are trusted nodes. The watchdog nodes are selected from among the set of trusted nodes for a given period of time depending on node energy, node storage capacity and node computing power. Nodes are considered to be trustworthy only after they show good behavior over a considerable period of time.

Current watchdog techniques can only judge the behavior of its one-hop neighbors. These approaches can make the judgment of its last-hop if the data flows in both forward and reverse direction. However, in practice, most of the traffic is approaching to sink. The amount of reverse traffic is very less.

2.3 Summary

It has been observed from the literature study that quite a good number of fault diagnosis schemes have been proposed till date. However, the existing schemes are expensive from communication, energy, and time perspectives. The shortcomings of present fault diagnosis techniques are as follows.

- In the majority of existing works diagnostic messages are not sent as the output of the routine tasks of a WSN. This increases energy and communication overhead.
- Little work has been done on diagnosing intermittent and transient faults.
- Most of the techniques do not address a mechanism to discriminate transient from an intermittent fault in WSNs.
- Most of the techniques assume that sensor nodes are static and do not consider node mobility.

Chapter 3

Hard and Soft Fault Diagnosis in WSNs

3.1 Introduction

The wireless sensor networks (WSNs) continue to gain the importance due to various applications such as battlefield surveillance, environmental monitoring, intruder detection systems, scientific data collection, intelligent infrastructure monitoring, underwater monitoring, health and medical monitoring, habitat monitoring, industrial monitoring, and ship detection [6–13]. These applications need the sensor nodes to form a network by deploying them in hostile and human inaccessible environment. It is common for the sensor nodes to become hard or soft faulty due to various reasons such as an exposer to unfriendly environment, calibration error, depletion of battery, age, etc. A hard faulty sensor node does not respond, and a soft faulty sensor node continues to operate with altered behavior.

Unlike wireless local-area networks, the path between the source and the destination in WSNs normally contains multiple wireless links (hops). The wireless links between nodes are susceptible to wireless channel fading, which causes channel errors. As the presence of hard and soft faulty nodes impact the network lifetime, design and evaluating the diagnosis algorithm that can diagnose the hard and soft faulty nodes in presence of channel impairment is the main objective of this chapter.

In order to efficiently diagnose the sensor nodes, the WSN is partitioned into non-overlapping clusters. This will not only improve the diagnosis efficiency but

also reduces the message and time complexity. The existing clustering techniques are intended to address the problems such as a prolonged network lifetime. Most of the state-of-art clustering approaches do not consider fault diagnosis as an integral part. Similarly, most of the state-of-art diagnosis techniques fail in exploring the advantages of clustering approaches over a non-cluster-based approach.

The proposed cluster-based distributed fault diagnosis (CDFD) algorithm works in conjunction with the underlying clustering protocol and exploits the spatially correlated sensor measurements to diagnose the WSN. A decision about the fault state of a node is taken based on the number of one-hop neighbors agree with its sensor measurement. This is based on the consensus on the number of one-hop neighbors of a sensor node to declare the sensor node fault-free. We derive an optimal value for this threshold which in turn makes our algorithm less sensitive to the average node degree for a wide range of fault rates.

The rest of the chapter is organized as follows. Section 3.2 presents the system model. The description, analysis and implementation are presented in Section 3.3. Simulation results are presented in Section 3.4 and finally, summary is given in Section 3.5.

3.2 System Model

3.2.1 Notations

The list of the notations used in this chapter and their meanings are shown in Table 3.1.

3.2.2 Assumptions

The following assumptions are considered for the algorithm CDFD.

- A base station (i.e., sink node) located outside the sensing field. Sensors and the base station are all stationary after deployment.
- Sensors are homogeneous and have the same capabilities.
- Each sensor node is assigned a unique identifier (ID).

Table 3.1: Notations.

n	Number of sensor nodes.
n_c	Number of cluster heads.
v_i	Sensor node.
F_{state_i}	Fault state of v_i.
x_i	Sensor reading at node v_i.
$N(v_i)$	One-hop neighbor set of v_i.
N_x	Number of one-hop neighbors report similar reading x.
R_{tx}	Transmission range.
E_{Tx}	The energy spent in transmitting one bit.
E_{Rx}	The energy spent in receiving one bit.
E_{DA}	The energy spent in aggregation of both routine and diagnostic data.
E_{elec}	The per bit electronics energy.
α	Path loss exponent.
M_c	Matching criteria.
d_a	Average node degree.
p	Fault probability.
p_{cerr}	The average bit error probability of the channel.
T_{out}	Timeout timer.
T_{slot}	The duration of each TDMA time slot.
T_p	Upper bound on the time needed to propagate a message between cluster heads.
T_{sink}	Upper bound time to propagate a message from the sink node and the farthest node from sink node.
δ	Application specific constant.
θ	Optimal threshold.
D_{ST}	Depth of the spanning tree.

- Each sensor node can estimate its channel error probability.
- Each sensor node is capable of transmitting at variable power levels depending on the distance to the receiver. An example of such sensor nodes is MICA Motes which use the MSP430 [84, 85] series micro controller and can be programmed to 31 different power levels.
- Links are symmetric, i.e., the data speed or quantity is the same in both directions, averaged over time.

3.2.3 Network Model

The WSN consists of n sensor nodes v_1, v_2, v_3,$\cdots$, v_n. The sensor nodes are uniformly distributed in the network which form a random network topology. Sensor nodes are considered as neighboring sensors if they are within the transmission range (R_{tx}) of each other. Every sensor node v_i maintains a neighbor table $N(v_i)$. All sensor nodes in the WSN are identical and are arranged into non-overlapping clusters. Nodes are organized into one-hop clusters where every node is aware of its cluster head. The intra-cluster communication is accomplished using the TDMA-MAC protocol. The inter-cluster communication is accomplished using the CSMA-MAC protocol. Each sensor node periodically produces sensor measurements such as temperature as it monitors its vicinity.

3.2.4 Fault Model

The proposed algorithm considers both hard and soft faults in sensor nodes. A hard faulty node is unable to communicate with the other sensor nodes in the WSN, whereas a soft faulty sensor node continues to operate and generates erroneous results. A sensor's reading is said to be erroneous if it is significantly different from those of its one-hop neighbors. A sensor node is subjected to hard faults due to the faulty transceiver, depleted battery, and damaged node. A sensor node is subjected to soft faults if at least one of the functional blocks (Figure 1.1) is malfunctioning. The sensor fault probability p is defined as

$$p = P(S = \neg x | A = x) \tag{3.1}$$

where the real temperature reading obtained by the sensor node is represented by variable S and the actual ambient temperature is represented by variable A. x is any value of S and A and $\neg x$ is any value not equals to x. The faulty measurements are uncorrelated. The normal measurements are spatially correlated. This means; readings from faulty sensors are geographically independent while readings from sensors in close proximity are spatially correlated.

3.2.5 Channel Model

The model used for a channel is a two-state Gilbert-Elliott channel (two-state state Markov channel model) with two states: G (good) state and B (bad) state [86, 87]. This model describes errors on the bit level. In the good state, the bits are received incorrectly with probability P_{good} and in the bad state, the bits are received incorrectly with probability P_{bad}. For this model it is assumed that $P_{good} \ll P_{bad}$. The transition probability $T_{GB} = P(G \rightarrow B)$ and $T_{BG} = P(B \rightarrow G)$ will be small and the probability remaining in G and B is large. The steady-state probability of a channel being in the bad state is $P_B = T_{GB}/(T_{GB} + T_{BG})$. Thus, the average bit error probability of the channel is $P_{cerr} = P_{bad}P_B + P_{good}(1 - P_B)$. For the simulations, this work uses this model that independently generates error patterns for all channels between nodes.

3.2.6 Energy Consumption Model

Similar to [88], we assume a simple model for the radio hardware energy dissipation. The transmitter dissipates energy to run the radio electronics and the power amplifier. The receiver dissipates energy to run the radio electronics. Both the free space (fs) (q^2 power loss) and the multi-path (mp) fading (q^4 power loss) channel models are used, depending on the distance between the transmitter and receiver. The threshold q_0 for practical systems using low gain antennas is typically chosen to be 1 meter in indoor environments and 100 meters in outdoor environments [89]. The energy spent for transmission of a r-bit packet over distance q is:

$$E_{Tx}(r, q) = rE_{elec} + r\epsilon q^{\alpha} = \begin{cases} rE_{elec} + r\epsilon_{fs}q^2 & q < q_0 \\ rE_{elec} + r\epsilon_{mp}q^4 & q \geq q_0 \end{cases} \tag{3.2}$$

The electronics energy, E_{elec}, depends on factors such as the digital coding, and modulation. The amplifier energy, $\epsilon_{fs}q^2$ or $\epsilon_{amp}q^4$, depends on the transmission distance and the acceptable bit-error rate. To receive this message, the radio expends energy:

$$E_{Rx}(r) = rE_{elec} \tag{3.3}$$

Cluster head consumes $E_{DA}(nJ/bit/signal)$ amount of energy for both routine and diagnostic data aggregation.

3.2.7 Diagnostic Model

Each sensor node periodically senses temperature measurements and broadcast to one-hop neighboring nodes. The sensor measurements are spatially correlated, i.e., for each fault-free sensor node, its neighboring fault-free sensor nodes have broadcasted similar sensor reading in their allotted TDMA time slots. Due to the shared nature of communication in wireless networks, a node v_i receives the sensor measurements of its one-hop neighbors. Since TDMA-MAC protocol is used for intra-cluster communication, v_i will receive these sensor measurements at different times. A sensor performs a self-test on the received identical temperature measurements from one-hop neighbors and a derived optimum threshold. Fault-free nodes fail to pass the threshold test later been diagnosed as fault-free through the fault-free neighbor(s). The sensor measurement x_i is identical to $x_j \in N(v_i)$ if $|x_i - x_j| < \delta$. In other words, $x_i \approx x_j$ if $|x_i - x_j| < \delta$. Based on this a matching criteria $M_{c_{ij}}$ can be defined as

$$M_{c_{ij}} = \begin{cases} 1 & \text{if } |x_i - x_j| < \delta \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

where $\delta = C \times f(T_{diff}(x_i, x_j))$. $T_{diff}(x_i, x_j)$ is the time difference between the time v_i takes its own measurement and receives sensor measurement of $v_j \in N(v_i)$ and C is application dependent constant.

3.3 The Proposed CDFD Algorithm

3.3.1 Description of the Algorithm

The CDFD algorithm consists of three phases namely clustering phase, fault detection phase and dissemination phase. In the clustering phase, the WSN is partitioned into different non-overlapping clusters. In the fault detection phase, each cluster head accumulates the fault states of their affiliated member sensor nodes. In the dissemination phase, the cluster level local view is disseminated in

the WSN such that each sensor node correctly diagnoses the state of each sensor node in the WSN.

Clustering Phase

Numerous previous studies have focused on WSN clustering [5]. However, we choose the approach namely unequal cluster-based routing (UCR) protocol suggested by Chen *et al.* [90] because it mitigates the hot spot problem [4]. The operation of UCR protocol is divided into rounds. The cluster head status is rotated among sensors in each round to distribute the energy consumption across the WSN.

The clustering phase further consists of two phases such as cluster head selection phase and setup phase. In the cluster head selection phase, the energy-efficient unequal clustering (EEUC) algorithm selects cluster heads based on the residual energy of tentative cluster heads. The EECU algorithm produces clusters of unequal sizes to address the hot spot problem. Clusters closer to the base station have smaller cluster sizes, and in turn consume less energy during the intra-cluster data processing. The size of cluster increases with an increase in distance from the base station or the sink node. In this phase, several tentative cluster heads are randomly selected to compete for final cluster heads with the same predefined probability. Nodes those fail to be tentative heads keep sleeping until the cluster head selection stage ends. Each tentative cluster head v_i has a competition range R_i. Different competition ranges are used to produce clusters of unequal sizes. R_0 is the predefined maximum competition range. The minimum competition range is set to $(1-c)R_0$, where c is a constant coefficient ($0 \leq c \leq 1$). Only one final cluster head is allowed in each competition range. The tentative cluster head sensor node v_i's competition range R_i can be expressed as a linear function of its distance to the base station [90]:

$$R_i = \left(1 - c\frac{D_{max} - D(v_i, BS)}{D_{max} - D_{min}}\right) R_0 \tag{3.5}$$

where D_{max} and D_{min} denote the maximum and minimum distance between network boundary and the base station. For instance, D_{max} for a WSN shown in Figure 3.1 is $\sqrt{(l + D_{min})^2 + (b/2)^2}$. $D(v_i, BS)$ denotes the distance between

v_i and the base station.

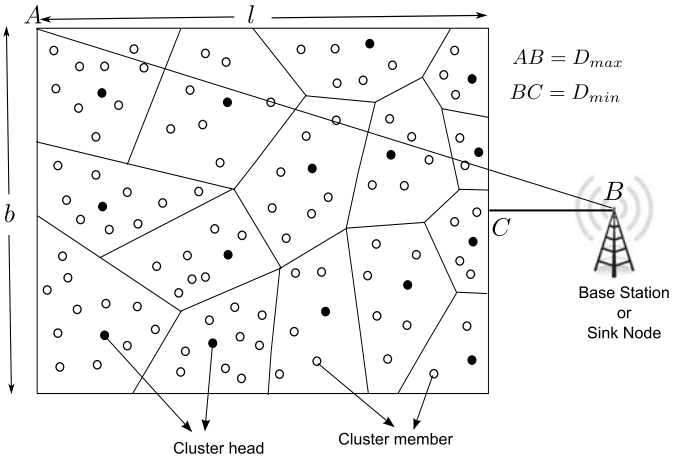


Figure 3.1: An overview of D_{max} and D_{min}.

In the cluster head selection process, each tentative cluster head nodes broadcast COMPETE_HEAD_MSG by setting transmission radius to R_0. The COMPETE_HEAD_MSG contains tentative cluster head node's competition radius and residual energy. Upon receiving COMPETE_HEAD_MSG, each tentative cluster head sensor node constructs a set (S_{CH}) of its adjacent tentative cluster heads. Tentative head sensor node v_j is an adjacent node of v_i if v_i is in v_j's competition diameter. A tentative cluster head sensor node v_i decides to become a final cluster head based on the residual energy of the nodes in $v_i.S_{CH}$. Once v_i finds that its residual energy is more than all the nodes in $v_i.S_{CH}$, it broadcasts the FINAL_HEAD_MSG to inform its adjacent tentative cluster heads. Tentative cluster head sensor node $v_j \in v_i.S_{CH}$ quit the competition immediately after receiving this FINAL_HEAD_MSG, and inform all nodes in its $v_j.S_{CH}$ by broadcasting a QUIT_ELECTION_ MSG.

In the setup phase, sleeping nodes wake up and each cluster head broadcasts an advertise message. Each non-cluster head node chooses its closest cluster head with the largest received signal strength and then informs the cluster head by sending a JOIN_CLUSTER_MSG. Each cluster head sets up a TDMA schedule and transmits it to the affiliated member nodes. After the TDMA schedule is known to all nodes in the cluster, the setup phase is completed, and the steady-state operation (data transmission) begins. To reduce inter-cluster interference, nodes

in each cluster communicate by using the direct-sequence spread spectrum (DSSS). Each cluster uses a unique spreading code. All the nodes in the cluster transmit their data to the cluster head using this spreading code.

In the setup phase, the greedy geographic routing protocol constructs a cluster head backbone rooted at the base station as shown in Figure 3.2. In this approach, distance between each pair of cluster heads can be calculated approximately according to the received signal strength. Each sensor node computes the approximate distance to the base station based on the received signal strength of a beacon signal broadcasted by the base station during initial deployment. If a node's distance to the base station is smaller than a threshold (TD_MAX), it transmits its data to the base station directly. Otherwise, it finds a relay node which can forward its data to the base station. The relay node is chosen based on the energy cost of the relay path.

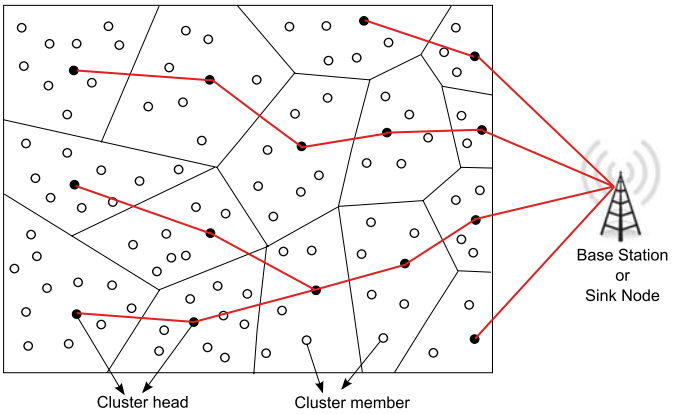


Figure 3.2: An overview of clusters and spanning tree.

Detection Phase

Each node broadcasts their sensor measurements in their allotted TDMA time slots (Algorithm 1). Each node sets a timeout timer T_{out} to detect hard faults. A node v_i will receive these sensor measurements at different times. Node v_i executes Algorithm 2 to form a set of neighbors $\{N_x\} \subseteq N(v_i)$ those have reported similar sensor measurement (x). Algorithm 2 uses two counters such as $count_1$ and $count_2$ to determine the value of x. Upon receiving the sensor measurements, v_i buffers the sensor measurements and marks their time of reception. Next, v_i

compares this currently received measurement with the sensor measurements (if any) those have already been received by v_i from other one-hop neighbors. If it finds a match, the $count_1$ is incremented. The $count_1$ defines the number matches between sensor measurements received earlier, and the currently received sensor measurement. The $count_1$ is then compared with $count_2$. If $count_1 > count_2$ then $count_2$ is assigned with value of $count_1$. This ensures a value of x with which highest number of one-hop neighbors of v_i agrees. The node v_i next compares its own reading x_i with x. If $x_i \approx x$ and $|\{N_x\}| \geq \theta$ then v_i is detected as fault-free where $|\{N_x\}|$ equals to $count_2$. Otherwise, v_i is detected as possibly soft-faulty. Alternatively, this can be defined as

$$v_i = \begin{cases} \text{Possibly soft-faulty} & \text{if } (x_i \approx x \text{ and } |\{N_x\}| < \theta) \text{ or } (x_i \not\approx x \text{ and } |\{N_x\}| \geq \theta \\ \text{Fault-free} & \text{otherwise.} \end{cases} \tag{3.6}$$

The optimal value for θ is $0.5(|N(v_i)| - 1)$, which is derived in Section 3.3.2 where $|N(v_i)|$ is the number of neighbors of v_i. This decision is then broadcasted. The probability that a node v_i is detected as possibly soft faulty is given by

$$p_{ps} = \sum_{\hat{l}=0}^{\theta-1} \binom{|N(v_i)|}{\hat{l}} (1-p)^{\hat{l}} p^{(|N(v_i)|-\hat{l})} \tag{3.7}$$

A node v_i identified as possibly soft-faulty first checks for a node $v_k \in N(v_i)$ such that the k^{th} entry in its fault table is fault-free, i.e., v_i believes that v_k is fault-free. If such v_k exists, $M_{c_{ij}} = 1$ and $v_k \in \{N_x\}$ then v_i is detected as fault-free or else faulty.

The detection algorithm uses a timeout mechanism to detect hard faulty nodes. Node v_i declares node $v_j \in N(v_i)$ as possibly hard-faulty (initial detection status), if v_i does not receive the sensor reading from v_j before T_{out}. T_{out} should be chosen carefully so that all the fault-free nodes $v_j \in N(V_i)$ connected by fault-free channels $Channel_{ij}$ must report node v_i before T_{out}. The node v_j cannot report to v_i due to at least one of the following reasons: the transceiver of v_j is faulty, the communication channel $Channel_{ij}$ is faulty, battery is drained and the node is completely damaged. For faulty communication channel v_i will mark v_j as hard

Algorithm 1 CDFD

1: // Clustering phase
2: Construct clusters using EECU algorithm [90].
3: Construct the cluster head backbone (spanning tree) using greedy geographic routing protocol [90].
4: // Detection phase
5: Broadcast the sensor reading x_i in its TDMA time slot.
6: set timer T_{out}.
7: Determine $\{N_x\}$, the set of one-hop neighbors report similar sensor measurement x using Algorithm 2.
8: **if** $T_{out} = true$ **then**
9: Declare unreported nodes as possibly hard faulty.
10: **end if**
11: **if** ($x_i \approx x$ and $|\{N_x\}| < \theta$) or ($x_i \not\approx x$ and $|\{N_x\}| \geq \theta$) **then**
12: $F_{state_i} \leftarrow$ Possibly soft faulty.
13: **else**
14: $F_{state_i} \leftarrow$ Fault-free.
15: **end if**
16: Broadcast the F_{state_i}.
17: Node identified as possibly soft faulty checks for a node $v_k \in N(v_i)$ such that F_{state_k} is fault-free. If such v_k exists, $M_{c_{ij}} = 1$ and $v_k \in \{N_x\}$ then set F_{state_i} as fault-free or else faulty. Broadcast the fault table FT_i.
18: If v_i is cluster head, it constructs a cluster level local view and takes a decision on possibly hard faulty nodes by comparing the fault tables of member nodes.
19: // Dissemination phase
20: **if** v_i is cluster head and has no tree descendants **then**
21: Append the cluster level local diagnostic view to its data packet and send to the tree ancestor.
22: **else if** v_i is cluster head and has tree descendants **then**
23: Wait until all diagnostic views of its tree descendants are received.
24: Take decision on possibly hard faulty nodes.
25: Combine the received and its own diagnostic views, append this combined view to its data packet and send to the tree ancestor.
26: **end if**
27: Upon receiving all the diagnostic local views, the base station constructs the diagnostic global view and broadcast it along with the synchronization message.

faulty, which may not be always correct. Final decision regarding v_j (hard faulty or fault-free) is taken during the dissemination stage.

The node v_i constructs a fault table FT_i that contains its own correct fault state and the IDs of nodes those have detected as possibly hard faulty by it. At this stage, each cluster head has a local view that reflects its view about the

Algorithm 2 Determination of x

1: // Node v_i receives sensor measurements from all $v_j \in N(v_i)$ in different time, where the time of reception depends on the TDMA time schedule of these neighbors.
2: Initialize an empty array of size $|N(v_i)|$ and set a counter $count_2 = 1$.
3: **for** $j = 1$ to $|N(v_i)|$ **do**
4: Upon receiving sensor measurement from $v_j \in N(v_i)$, store the sensor measurement in $array(j)$.
5: **if** $j \geq 2$ **then**
6: Initialize a variable $data = 0$ and counter $count_1 = 1$.
7: **for** J=j-1 to 1 **do**
8: **if** $array(j) \approx array(J)$ **then**
9: Increment the counter, i.e., $count_1 = count_1 + 1$.
10: $data = data + array(J)$.
11: **end if**
12: **end for**
13: **if** $count_2 < count_1$ **then**
14: $count_2 \leftarrow count_1$.
15: $x \leftarrow \frac{data+array(j)}{count_2}$.
16: **end if**
17: **else**
18: $x \leftarrow array(j)$.
19: **end if**
20: **end for**

state of its affiliated member nodes as well as the non-affiliated nodes in one-hop distance. We call this as cluster level local view.

Dissemination Phase

The cluster level diagnostic views are disseminated using the spanning tree of cluster heads constructed by the greedy geographic routing protocol. The leaf cluster heads start this dissemination by appending its cluster level local diagnostic view to its data packet. A cluster head that has received data packets from tree descendants first compares cluster level local diagnostics of their decedent and takes a decision about hard faults. The final decision regarding a node detected as hard faulty is based on a consensus made at each level. For example, if a node is detected as hard faulty by a cluster head CH_i but CH_j believes that the node is fault-free, then cluster heads in the upper level of the spanning tree detect the node as fault-free. Second, it appends its cluster level local view and the data

packet of its tree descendants to its own packet and sends them up to the root. At the end of local dissemination, the base station generates the global view and broadcasts it along with the synchronization message. This ensures that each fault-free node correctly diagnose the state of all the sensor nodes in the WSN.

Implementation

In this section, we discuss the design details for practical deployment of CDFD algorithm. As shown in Figure 3.3 CDFD algorithm includes six time triggers: (1) cluster head selection triggers (T1), (2) cluster set-up triggers (T2), (3) intra-cluster communication for routine data triggers (T3), (4) intra-cluster communication for exchanging fault state information triggers (T4), (5) intra-cluster communication for communicating the correct decisions about nodes detected as possibly soft-faulty triggers (T5), inter-cluster communication and local dissemination trigger (T6), and global dissemination and network synchronization triggers (T7).

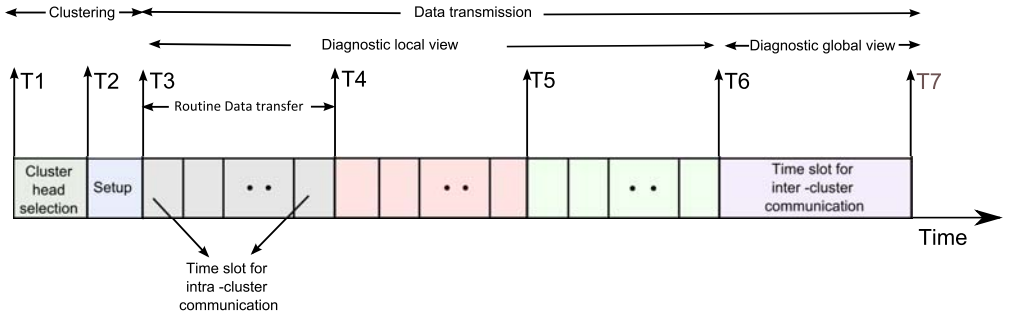


Figure 3.3: Time line showing UCR and CDFD algorithm operation.

.

First, we describe the MAC mechanism together with the duty cycles schedule in various phases of the CDFD. T1 triggers the cluster formation phase. At T2, the non-cluster head nodes wake up. The cluster heads transmit the advertise message for cluster formation and the control messages to construct the cluster head backbone using CSMA. To reduce inter-cluster interference, each cluster in UCR protocol communicates using the direct-sequence spread spectrum (DSSS). UCR protocol uses transmitter-based code assignment [91], where all transmitters within the cluster use the same spreading code. Once the cluster head backbone is

constructed, and the base station is aware of the IDs and cluster head declaration times, the base station assigns unique spreading codes to cluster heads from a predefined list. The codes are assigned on a first-in, first-served basis, starting with the first cluster head to announce its position, followed by subsequent cluster heads. Each cluster head transmits TDMA schedule and their unique spreading code messages using CSMA. The number of TDMA time slots in each cluster depends on the number of member nodes in the cluster. Neighboring cluster heads share their spreading codes such that a node, irrespective of its affiliation, can de-spread the messages received from all its one-hop neighbors. This ensures a node to obtain the sensor measurements of all its neighbors for a diagnosis purpose. T3 triggers intra-cluster routine data transmission and the detection phase. Member nodes turn off their transmitter at all times except during their transmit time. However, the receiver is on to receive diagnostic data. During T4 to T5 nodes broadcast the local decisions in their time slot. The decision is stored in the local fault table. During T5 to T6 nodes with possibly soft-faulty status take final decision and broadcast the updated decision in their time slot. Cluster head turns off their radio once their TDMA time slots run out. At T6, all cluster heads wake up, and the inter-cluster communication is triggered. Cluster heads transmit control messages and data packets using CSMA. The local diagnostics and the routine data are aggregated through the spanning tree up to the sink. Thus, at T7, the sink has the global fault state view of the WSN. Synchronization is important for the operation of UCR and CDFD. This work assumes that all sensor nodes are synchronized and start CDFD phase at the same time. This could be achieved, for example, by having the sink periodically broadcast synchronization pulses. In this work at T7, the sink node broadcasts the synchronization message along with the global view such that all nodes in the WSN will receive this message.

Since the diagnosis operation is integrated to clustering, there is a cost in terms of energy and time to cluster as well as diagnose the WSN. If the clustering and diagnosing overhead are incomparable to the application packet load, clustering and diagnosing can be triggered every data-gathering round. For continuous monitoring applications where all nodes are continuously sending reports, however, frequent clustering and diagnosing the network will lead to instability, delayed

response and faster energy depletion. In addition, the mean time between failures (MTBF) is expected to be much longer, thus, frequent diagnosis of the network may not be cost effective. Therefore, there is always a trade-off in determining time duration between two diagnosis rounds, and it is application specific.

3.3.2 Analysis of The Algorithm

Determination of T_{out}

In scenarios where the node density is not homogeneous, if this timeout is short, some sensor nodes with high neighbor density can be working on the fault diagnosis while some sensor nodes with low neighbor density may be detected incorrectly as faulty. If the timeout is longer and most of the nodes have high neighbor density, the diagnosis may not be efficient in terms of time. Thus, determination of a proper value for T_{out} is required, which is determined by considering various delay components from WSN perspective. We first discuss the delays introduced at each layer of WSN architecture. When a node decides to transmit a packet, it is scheduled as a task in a sensor node. The total time spent in constructing the packet at the application layer and then passing to MAC layer is denoted as T_{AM}. This delay depends on the underlying operating system. At MAC layer, the diagnostic packet waits until it can access the channel. This delay (T_{MAC}) is specific to wireless networks, which is critical and depends on the MAC protocol employed by the sensor node. The delay in transmitting a packet bit by bit at the physical layer over the wireless link is mainly deterministic in nature and can be estimated using the packet size and the radio speed. The propagation delay (T_{pro}) is the actual time taken by the packet to traverse the wireless link from the sender node to the receiver node which is negligible as compared to other sources of delay. The reception delay (T_{res}) refers to the time taken in receiving the bits and passing them to the MAC layer. This delay is mainly deterministic in nature. The MAC layer then passes the received packet to the application layer where it is decoded. The total time spent in passing the received packet from MAC layer to the application layer (T_{MA}) depends on the underlying operating system.

Lemma 1 *The upper bound on the time such that a fault-free node will receive the sensor measurements of all its neighbors is* $T_{out} = T_{AM} + T_{pro} + T_{res} + T_{MA} + N_c T_{slot}$

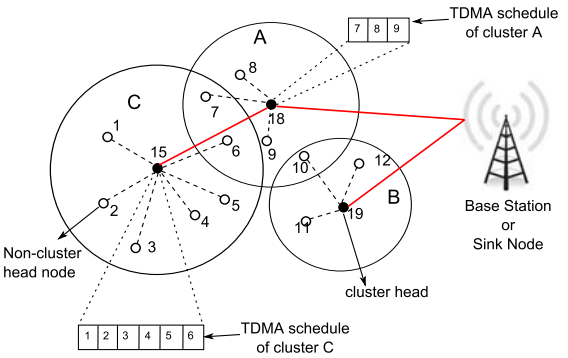


Figure 3.4: Analysis of T_{out} (A,B, and C are clusters).

.

where N_c is the number of member nodes of the largest neighbor cluster and T_{slot} is the duration of each TDMA time slot.

Proof: The time delay introduced in receiving a packet from a neighbor sensor node $v_j \in N(v_i)$ by node v_i is $T_{AM} + T_{MAC} + T_{pro} + T_{res} + T_{MA}$. As TDMA-based MAC protocol is used, T_{MAC} depends on the slot position in the TDMA frame. In addition, the sizes of neighboring clusters are not same. Thus, the delay in receiving sensor measurements by a sensor node from all its neighbors depends on N_c. For instance, as shown in Figure 3.4, node $A1$ will receive data of $C6$ after a delay equals to $6T_{slot}$. Therefore, $T_{MAC} = N_c T_{slot}$. ∎

The proposed algorithm CDFD is analyzed to show the completeness and correctness. The diagnosis is complete if all sensor nodes are identified as faulty or fault-free in a bounded time. A diagnosis is said to be correct if there are no fault-free nodes mistakenly diagnosed as faulty and no faulty nodes mistakenly diagnosed as fault-free. Generally, an incorrect diagnosis is unacceptable because the error information may propagate to the base station or users. Our proposed algorithm minimizes the likelihood of incorrect diagnosis and ensures a complete diagnosis.

Threshold Formulation

In this section, we formulate the threshold θ.

Theorem 1 *The error probability in detecting the fault state of a node is given by* $P_e = f_1 \cdot \left(1 - p - \sum_{l=\theta}^{N} (1-p) f_l + p f_{N-l}\right)$.

Proof: After its deployment in the field, a sensor node can be modeled by two variables such as S and A. S represents the sensor reading and A represents the actual reading. Let $E_i(x, l, N)$ be the set of l sensor nodes out of N one-hop neighbors of a node v_i report the similar sensor reading x. The fault detection estimate (DE) is calculated after obtaining information about the sensor readings of neighboring nodes. The possible values of DE is fault-free (FF) and faulty (F).

The probability that the detection estimate is fault-free, given that l of the N neighboring sensors report the same reading x as node v_i is defined as:

$$P_l = P(DE = FF | S = x, E_i(x, l, N)) \tag{3.8}$$

For a faulty communication channel $Channel_{i,j}$, v_i believes that $v_j \in N(v_i)$ is faulty. In presence of channel faults, let f_l is the probability that l out of N one-hop neighbors of node v_i are fault-free. This probability is determined as

$$\begin{aligned} f_l &= \binom{N}{l} P(S_i = x | A_i = x, Ch = G)^l \\ &\quad \cdot P(S_i = x | A_i = \neg x, Ch = G)^{N-l} \\ &= \binom{N}{l} P(S_i = x | A_i = x, Ch = G)^l \\ &\quad \cdot P(S_i = x | A_i = x, Ch = B)^{N-l} \\ &= \binom{N}{l} (1-p)^l p^{N-l} \end{aligned} \tag{3.9}$$

The possible values for variables S and A are x and $\neg x$ where $\neg x$ defines a value which is not similar to x. Thus, eight possible combinations exist for DE, S and A. The correctness of the proposed algorithm can be analyzed by the conditional probabilities corresponding to these combinations. From these combinations, we can calculate the probability that the algorithm estimates the node is faulty though both the sensed, and actual readings are similar. By using

marginal probability this can be derived as

$$\begin{aligned} P(DE = F|S = x, A = x) &= 1 - P(DE = FF|S = x, A = x) \\ &= 1 - \sum_{l=0}^{N} P(DE = FF, E_i(x, l, N)|S = x, A = x) \\ &= 1 - \sum_{l=0}^{N} P(DE = FF|S = x, A = x, E_i(x, l, N)) \\ &\quad \cdot P(E_i(x, l, N)|S = x, A = x) \\ &= 1 - \sum_{l=0}^{N} P_l \cdot f_l \end{aligned} \tag{3.10}$$

In a similar manner, we can calculate the probability that the algorithm estimates the node is fault-free though the sensor reading does not agree with actual reading.

$$\begin{aligned} P(DE = FF|S = \neg x, A = x) &= \sum_{l=0}^{N} P(DE = FF, E(x, N-l)|S = \neg x, A = x) \\ &= \sum_{l=0}^{N} P(DE = FF|S = \neg x, A = x, E_i(x, N-l, N)) \\ &\quad \cdot P(E_i(x, N-l, N)|S = \neg x, A = x) \\ &= \sum_{l=0}^{N} P(DE = FF|S = \neg x, A = x, E_i(\neg x, l, N)) \\ &\quad \cdot P(E_i(x, N-l, N)|S = \neg x, A = x) \\ &= \sum_{l=0}^{N} P_l \cdot f_{N-l} \end{aligned} \tag{3.11}$$

As discussed earlier, fault-free nodes which failed to pass the threshold test are later diagnosed as fault-free through a fault-free neighbor. The probability that at least one out of N one-hop neighbors is fault-free can be derived from equation (3.9) as

$$f_1 = N(1-p)p^{N-1} \tag{3.12}$$

Equation (3.11) and (3.12) suffice to calculate the probability that the detection algorithm declares a fault-free node as faulty. This probability is given by

$$\begin{aligned} P_{gf} &= P(DE = F, S = x | A = x) \cdot f_1 \\ &= P(DE = F | S = x, A = x) \cdot P(s = x | A = x) \cdot f_1 \\ &= \left(1 - \sum_{l=0}^{N} P_l . f_l\right) \cdot (1 - p) \cdot f_1 \end{aligned} \tag{3.13}$$

In the similar manner, the probability that the detection algorithm declares a faulty node as fault-free can be derived as

$$\begin{aligned} P_{fg} &= P(DE = FF, S = \neg x | A = x) \cdot f_1 \\ &= P(DE = FF | S = \neg x, A = x) \cdot P(s = \neg x | A = x) \cdot f_1 \\ &= \left(\sum_{l=0}^{N} P_l . f_{N-l}\right) \cdot p \cdot f_1 \end{aligned} \tag{3.14}$$

In the proposed algorithm, the detection estimation is fault-free only when $l > \theta$. Thus equation (3.8) can be rewritten as

$$P_l = \begin{cases} 1 & \text{if } l > \theta \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

Thus, the error probability of the proposed algorithm in detecting the status of a node is given by

$$\begin{aligned} P_e &= P_{gf} + P_{fg} \\ &= f_1 \cdot \left(1 - p - \sum_{l=\theta}^{N} (1 - p) f_l - p f_{N-l}\right) \quad \blacksquare \end{aligned} \tag{3.16}$$

Theorem 2 *The optimum value of θ which minimizes the error probability (P_e) is $0.5(N-1)$.*

Proof: Proof of this theorem closely follows a similar proof in [51]. Substituting f_l in equation (3.16), the expression of summand of equation (3.16) can be written as

$$\begin{aligned} &\binom{N}{l}((1-p)^{l+1}p^{N-l} - p^{l+1}(1-p)^{N-l}) \\ &= \binom{N}{l}\left((1-p)^{l+1}p^{l+l}\left(p^{N-2l-1} - (1-p)^{N-2l-1}\right)\right) \end{aligned} \tag{3.17}$$

For $p < 0.5$, equation (3.17) is negative for $N > 2l + 1$, zero for $N = 2l + 1$, and positive for $N < 2l + 1$. Additional terms with negative contributions is produced by decreasing θ one at a time from N while $\theta > 0.5(N-1)$ and positive contributions once $\theta < 0.5(N-1)$. It follows that P_e achieves a minimum when $\theta = 0.5(N-1)$. ∎

Complexity Analysis

The upper bound time complexity is expressed in terms of the following bounds:

- T_p: an upper bound on the time needed to propagate a message between cluster heads.
- T_{sink}: an upper bound on the time needed to propagate a message between the sink node and the farthest node from the sink node.

Lemma 2 *The time complexity of CDFD algorithm is* $O(T_{out} + T_p d_{ST} + T_{sink})$ *where,* d_{ST} *is the depth of the spanning tree.*

Proof: The detection phase requires three rounds of communication each costing T_{out} time. In at most $d_{ST}T_p$, the sink node collects all diagnostic views. Then the sink node broadcasts the global diagnostic view that reaches the farthest node in at most T_{sink} time. It follows that, CDFD algorithm requires at most $3T_{out} + T_p d_{ST} + T_{sink}$ to complete a diagnosis session. ∎

The total number of messages exchanged by nodes executing the algorithm is termed as message complexity of the algorithm.

Lemma 3 *Message complexity of CDFD algorithm is* $O(n)$.

Proof: The diagnosis starts at each node by sending the sensor reading to its neighbors. This does not contribute to the message complexity as it is the routine task of the WSN. Each node then takes a decision about their state and broadcasts their decision costing one message per node, i.e., n messages in the WSN. The nodes failed to pass the threshold test (detected possibly soft-faulty) broadcast their updated state costing one message per node and in the worst case, n messages in the WSN where all faults are soft faults (hard faults are detected without any message exchange), and all nodes are detected as possibly soft faulty. Each cluster

head, excluding the sink, sends the local diagnostic message along their routine message and thus contributing no additional cost. The sink node broadcasts the global diagnostics along with the synchronization costing no additional message overhead. So, the total number messages exchanged explicitly for diagnosis is

$$M_{cost} = 2n = O(n) \quad \blacksquare \tag{3.18}$$

Lemma 4 *The number of bits exchanged to diagnose the WSNk is* $O(n \log_2 n)$.

Proof: In a n-node WSN each node has a unique identifier which can be encoded with $\log_2 n$ bits. The state of each node is identified with a single bit (0: fault-free and 1: faulty). Each node broadcasts its state requires $n(\log_2 n + 1)$ bits to be exchanged in the WSN. The nodes those have failed in the threshold test, broadcast their updated state. The number of bits to be exchanged in WSN for this operation is $n(\log_2 n + 1)$ bits. Local dissemination of diagnostics up to the sink needs $pn_c n(\log_2 n + 1)$ bits to be exchanged where n_c is the number of cluster heads. The sink broadcasts the global view costing $pn(\log_2 n + 1)$ bits. Thus, the total number of bits exchanged is $n(\log_2 n + 1)(pn_c + p + 2) = O(n \log_2 n)$. $\blacksquare$

Lemma 5 *The energy overhead in diagnosing the WSN is* $n(log_2 n + 1)((pn_c + 2)(E_{Tx} + E_{Rx}) + pE_{Rx})$.

Proof: The energy dissipated in exchanging states is $n(\log_2 n + 1)(E_{Tx} + E_{Rx})$. The energy overhead in exchanging correct decision on the possibly soft-faulty nodes is $n(\log_2 n + 1)(E_{Tx} + E_{Rx})$. Dissemination local diagnostics dissipate $pn_c n(\log_2 n + 1)(E_{Tx} + E_{Rx})$ units of energy and global diagnostic message dissipate $pn(\log_2 n + 1)E_{Rx}$ units of energy. Thus, the energy overhead in diagnosing the WSN is $n(log_2 n + 1)((pn_c + 2)(E_{Tx} + E_{Rx}) + pE_{Rx})$. $\blacksquare$

3.4 Simulation Results

The performance of the proposed scheme through simulations is presented in this section. The performance metrics namely diagnosis latency, per node message overhead, energy overhead, DA, FAR, and network lifetime are used to evaluate the performance of CDFD algorithm. This work uses Castalia-2.3b [52], a state-of-art WSN simulator based on the OMNET++ [53] platform. The

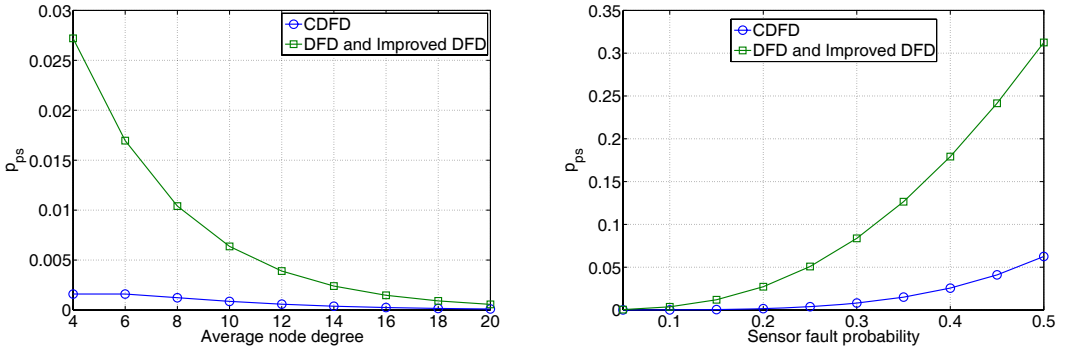
simulation parameters are given in Table 3.2. For these simulations, energy is consumed whenever a sensor transmits or receives data or performs data aggregation. The channel error rate P_{cerr} estimate at each node is 1×10^{-3}. As suggested in [90], UCR protocol parameters are set as $T = 0.2$, $R_0 = 80m$, $c = 0.3$, $TD\_MAX = 200m$, and $k = 2$.

Table 3.2: Simulation Parameters.

Parameter	Value
Number of sensors	1000
Network grid	From (0, 0) to (600, 400)m
Sink	At (700,200)m
Initial energy	1 J
E_{elec}	50 nJ/bit
ϵ_{fs}	$10pJ/bit/m^2$
ϵ_{amp}	$0.0013pJ/bit/m^4$
d_0	$87m$
E_{DA}	5 $nJ/bit/signal$

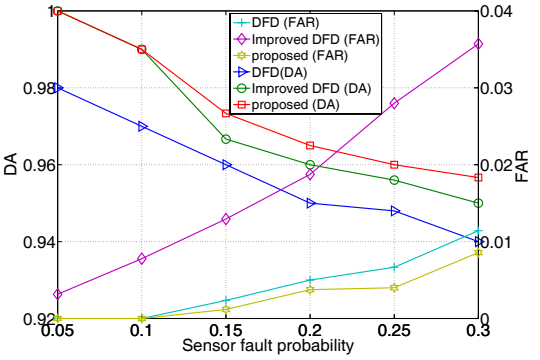
3.4.1 Experiment 1: Efficiency with regard to d_a and p

In this experiment, the performance of the diagnosis algorithm in regard to DA and FAR is evaluated and compared with the state-of-art schemes namely DFD [31] and Improved DFD [34]. In this simulation, sensor nodes are assumed to be faulty with probabilities of 0.05, 0.10, 0.15, 0.20, 0.25, and 0.30 respectively. To show the effectiveness, we consider an equal number of hard and soft faults.

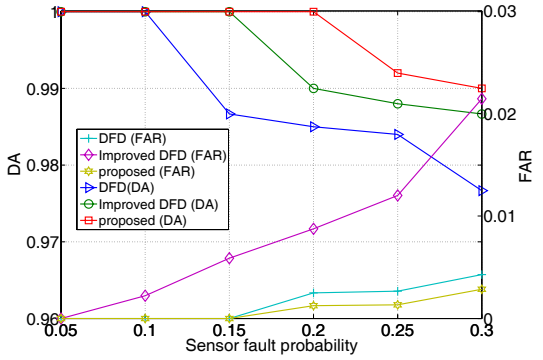


(a) (b)

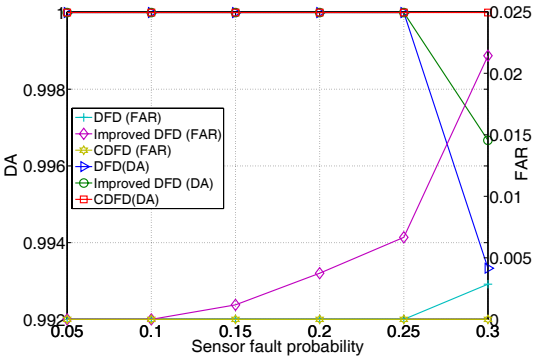
Figure 3.5: Theoretical value of p_{ps}: (a) At varying value of d_a: $p = 0.2$. (b) At varying value of p: $d_a \approx 4$.



(a) $d_a \approx 4$



(b) $d_a \approx 12$

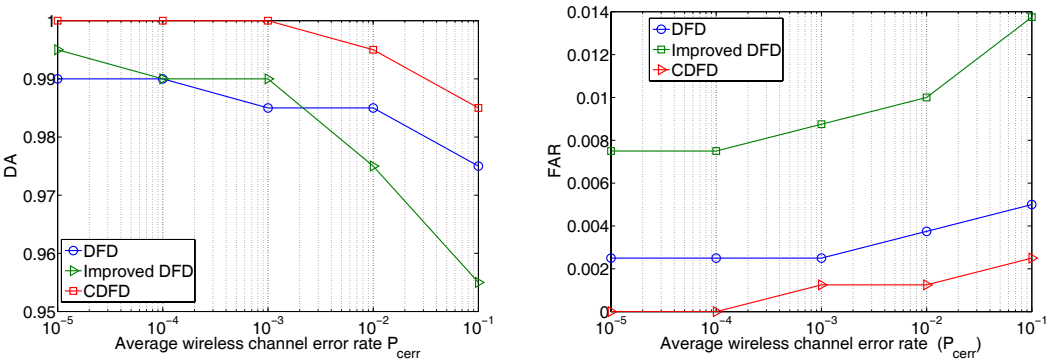


(c) $d_a \approx 20$

Figure 3.6: DA and FAR: $P_{cerr} = 1 \times 10^{-3}$.

In this simulation, the range is tuned to obtain $d_a \approx 4, 12$ and 20. As expected and shown in Figure 3.6, the DA and FAR of CDFD algorithm outperform both DFD and improved-DFD algorithm. The results can be explained as follows. As shown in Figure 3.5, at low average node degree the probability that a node detected as possibly soft faulty in both DFD and improved-DFD algorithm is very high as compare to CDFD algorithm. As discussed in Chapter 2, a fault-free node v_i will be detected as fault-free in the second round of test of DFD algorithm if $\sum_{v_j \in N(v_i), Tend_j = LG} (1 - 2Result_{ij}) \geq \lceil |N(v_i)|/2 \rceil$. There is a less probabilities that a node will pass this threshold test in sparse WSNs or WSNs with sparse areas. Thus, the number of nodes detected as fault-free in this test round is very less. In third round of test, DFD algorithm uses the nodes detected as fault-free (GD) in second round of test to take decision regarding the undetermined nodes. Thus, there is a high probability that the undetermined nodes will remain undetermined after third round of test, i.e., the incorrect decision taken in the second round

of test may not be rectified in subsequent test rounds. In second round of test, the improved-DFD algorithm diagnoses a fault-free node v_i as fault-free if reading of v_i matches with more than $\lceil |N(v_i)|/2 \rceil$ number of LG one-hop neighbor nodes (i.e., one-hop neighbor nodes with status possibly fault-free). The probability of passing this threshold test is very less in sparse WSNs or WSNs with sparse areas. In the third round of test, the LG (LT) nodes failed to pass this threshold test are diagnosed as fault-free (faulty) by the improved-DFD algorithm which may not be always correct. Since the FAR accounts for the number of fault-free nodes wrongly diagnosed as faulty by a diagnosis algorithm, an algorithm showing high FAR will reduce available sensor nodes in the WSN and impacting reliability. As shown in Figure 3.6(a) CDFD algorithm achieves a marginal improvement over DFD algorithm from FAR perspective. However, the number of messages exchanged by DFD algorithm to achieve this low level of FAR is comparably higher than CDFD algorithm.



(a) (b)

Figure 3.7: DA and FAR at varying value of P_{cerr}: $d_a \approx 12$.

3.4.2 Experiment 2: Robustness with regard to channel fault

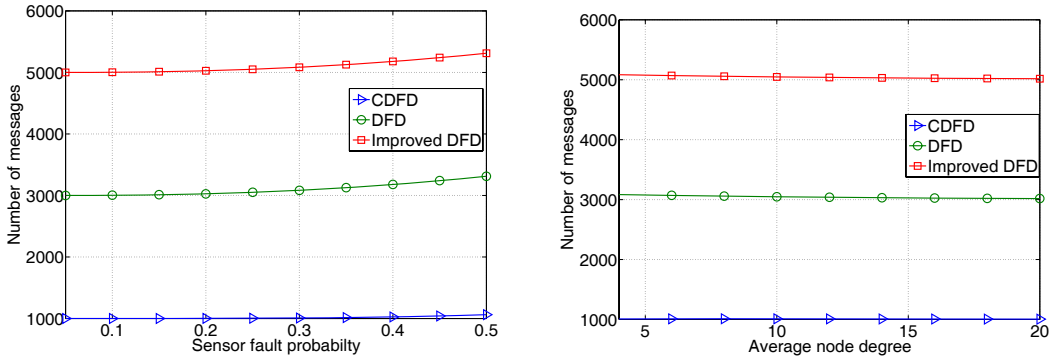
In this experiment, the robustness of the detection algorithm to faults in the communication channel is analyzed by estimating DA and FAR for various channel error probabilities. For simplicity in the simulation P_{good} is taken as 0 and P_{bad} is taken as 1. P_{BG} is fixed to 1/8 and P_{GB} is varied to get different channel error probabilities P_{cerr} [103]. We use the previously generated network with $p = 0.2$

and $d_a = 12$. The channel error rate is increased in steps from 10^{-5} to 10^{-1}. Faults in the communication channel might cause some fault-free nodes to fail in receiving the sensor measurements from its neighbors. This in turn decreases the effective neighbor size of a sensor node and might affect the local decision. However, as discussed in Experiment 1, CDFD algorithm shows better performance even in sparse WSNs. Thus, as expected and shown in Figure 3.7, the detection algorithm effectively tolerates faults in the communication channel. It is observed that the improved-DFD algorithm is worst affected by varying channel error rate. The reason is that in this approach, a node detected as possibly fault-free in first test round will be detected as fault-free only when more than half of its neighbors with initial detection status possibly fault-free agrees with its sensor measurement. This is hard to realize in scenario where the average node degree is affected by the channel error rate. It is observed that DFD algorithm effectively tolerates errors in channel but as discussed earlier and will be shown in experiment 3, it suffers from large message overhead and diagnosis latency.

3.4.3 Experiment 3: Time, message and energy efficiency

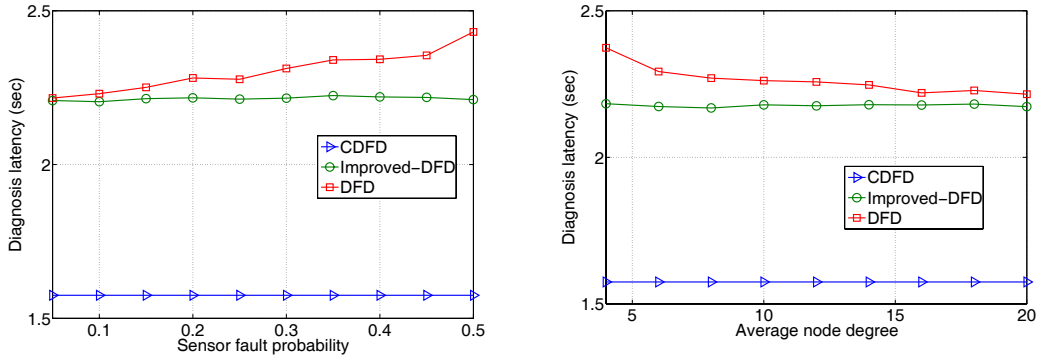
In this experiment, we attempted to illustrate the time, message and energy efficiency of CDFD algorithm with regard to varying average node degree and fault rate. For better analysis, we consider only soft faults as detection of hard faults does not require any message exchange. Both DFD and improved-DFD algorithms only generate the diagnostic local view. Thus, to compare the time, message and energy efficiency, we implement a spanning tree-based dissemination [20] for both of the schemes. As shown in Figure 3.8, the message overhead of CDFD algorithm in diagnosing the WSN is well below that of DFD and improved-DFD algorithm. The reason is that in CDFD algorithm, the diagnostic messages are sent as the output of the routine tasks of the WSN. As expected the message overhead of DFD algorithm is highest among the three schemes. This is because, in the worst case, a node in DFD algorithm needs to exchange four messages to obtain the diagnostic local view.

As expected and shown in Figure 3.9 (a), the diagnosis latency of CDFD and improved-DFD algorithm is not much affected by varying sensor fault probability



(a) (b)

Figure 3.8: Number of messages explicitly exchanged for diagnosis: (a) At varying value of p: $d_a = 12$, $P_{cerr} = 10^{-3}$. (b) At varying value of d_a: $p = 0.2$, $P_{cerr} = 10^{-3}$.

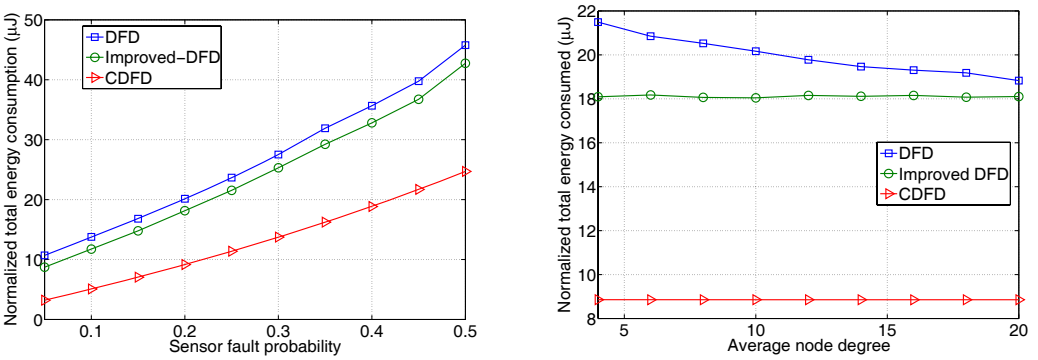


(a) (b)

Figure 3.9: Diagnosis latency: (a) At varying value of p: $d = 12$, $P_{cerr} = 10^{-3}$. (b) At varying value of d_a: $p = 0.2$, $P_{cerr} = 10^{-3}$.

and average node degree. The diagnosis latency of the DFD algorithm increase with an increase in fault rate because this predominately depends on the number of undetermined nodes (i.e., node with status either possibly faulty or possibly fault-free), and this number increases with an increase in fault rate. As reported in Figure 3.9 (b), the diagnosis latency of DFD algorithm decreases with an increase in average node degree for a fixed fault rate. The reason is that for a higher average node degree, most of the nodes meet the threshold test, and undetermined nodes' population is very less. However, diagnosis latency of improved-DFD algorithm and CDFD algorithm remains unaffected by varying node degree.

The comparisons between the normalized total energy dissipation of these three schemes are shown in Figure 3.10. We normalize the total energy consumption



(a) (b)

Figure 3.10: Normalized total energy consumption: (a) At varying value of p: $d = 12$, $P_{cerr} = 10^{-3}$. (b) At varying value of d_a: $p = 0.2$, $P_{cerr} = 10^{-3}$.

with respect to the number of nodes participated in diagnosis. We examine the energy consumption with respect to the varying fault rate and average node degree. It is observed that the difference between DFD and improved DFD algorithm in terms of the normalized total energy consumption is very small. However, CDFD algorithm outperforms both DFD and improved DFD algorithms. The reason is that the energy consumption is directly proportional to the amount of traffic generated. It is observed from Figure 3.8 that the number of messages explicitly exchanged for diagnosis of the WSN is very less than these two schemes. As reported in Figure 3.10(a), the energy consumption increases with a fault rate because for a higher fault rate, the probability of failing the threshold test is more. Thus, more messages need to be exchanged to take a correct decision. It is observed from Figure 3.10(b) that the energy consumption of DFD algorithm decreases for an increase in average node degree. The reason is that for a higher average node degree, the probability of failing to pass the threshold test is less and accordingly, the number of undetermined nodes is less. As expected and observed the energy consumption of improved DFD and CDFD algorithm is not much affected by average node degree.

3.4.4 Experiment 4: Network lifetime

The network lifetime is the measure of the number of data-gathering rounds when the first node dies due to depletion of battery. In this experiment, a node is

considered dead if it has lost 99 percent of its initial energy. For better comparative analysis, we implement DFD and improved-DFD algorithm in conjunction with UCR protocol. Like UCR protocol, we set data packet size to 4000. We first run the simulation without considering any diagnosis technique. Next we implement CDFD, DFD and improved-DFD algorithms in conjunction with UCR protocol. The network lifetime for varying fault rates is shown in Figure 3.11.

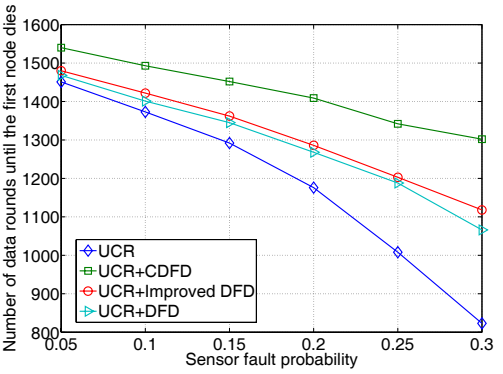


Figure 3.11: The network lifetime: $d_a = 12$, $P_{cerr} = 10^{-3}$.

An improvement in the network lifetime is observed when the detection algorithms work in conjunction with the clustering protocol. As shown, this improvement is remarkable in the scenario where fault rate is high. The reason is that if faulty nodes are allowed to send their data, then relay nodes dissipate energy in forwarding this erroneous data to the sink node. In this approach since the erroneous data generated by the faulty nodes are discarded, wastage of energy in relaying these erroneous data is avoided. This in turn improves the network lifetime. In addition, if the non-cluster-head nodes are unaware of the failure at the head sensor node, they send meaningless data and therefore, waste energy. As expected and shown in Figure 3.11, CDFD algorithm shows better performance compare to both DFD and improved-DFD algorithm. This is because the number of messages explicitly exchanged for diagnosis of the WSN by CDFD algorithm is very less than both DFD and improved-DFD algorithm. In addition, the CDFD algorithm outperforms both DFD and improved-DFD algorithm from FAR perspective. Thus, the number of fault-free nodes diagnosed as faulty and isolated by CDFD algorithm is very less as compare to DFD and improved-DFD algorithm. In other words, under the same fault scenario, the available fault-free

sensor nodes after the execution of DFD and improved-DFD algorithm are less compared with CDFD. Thus, the per-node network load is more when DFD and improve-DFD algorithms are implemented. This in turn cause faster depletion of battery energy and impacting network lifetime. Though the improved-DFD algorithm consumes less energy in diagnosing the WSN, it isolates more fault-free nodes compare to DFD algorithm. Thus, due to the reasons discussed above DFD algorithm performs better than the improved-DFD algorithm from network lifetime perspective.

3.5 Summary

In this chapter, a lightweight cluster based distributed fault diagnosis algorithm has been proposed to diagnose hard and soft faulty nodes in the WSNs. CDFD algorithm is lightweight since diagnostic messages are sent as the output of the routine tasks of the WSN. An optimal threshold for fault detection is derived which in turn improves the performance in regard to detection accuracy and false alarm rate. A high level (> 0.95) of DA is achieved while keeping the FAR low (< 0.01) for sparse networks. The message complexity of CDFD algorithm is $O(n)$ and the number of bits exchanged to diagnose the WSN are $O(n \log_2 n)$.

Chapter 4

Intermittent Fault Diagnosis in WSNs

4.1 Introduction

Experimental studies have shown that more than 80% of the faults that occur in real systems such as WSNs are intermittent faults [92, 93]. An intermittent fault originates from inside the system when software or hardware is faulty. By its nature, an intermittent fault will not occur consistently, which makes its diagnosis a probabilistic event over time [1]. Since the effect of a fault is not always present, detection of an intermittent fault requires repetitive testing at a discrete time kT $(k = 1, 2, \cdots)$ in contrast to single test for detection of permanent faults. Intuitively this implies that to detect an intermittent fault the issues like number of test repetitions required, and inter-test interval (T) are crucial. If T is too large, then probability that the fault appears after k^{th} test and disappears before $(k+1)^{th}$ test increases and thus detection accuracy decreases. Diagnostic latency is expected to be more for larger value of T which might not be acceptable for short mission time applications. Improvement in both detection accuracy and latency can be achieved with smaller value of T. However, if T is too small, then frequent exchange of sensor measurements is required as message exchange is the only means to detect faults. This in turn increases the energy overhead.

These issues motivate to find a trade-off between detection accuracy, detection latency and energy overhead by properly tuning the detection parameter like inter-test interval (T). Finding a good trade-off can be formulated in several

possible ways, and with emphasis on various aspects of the final output expected. Thus, there may not exist a single optimal solution rather a whole set of possible solutions of similar quality. This motivated us to use Multiobjective Optimization algorithms that deal with such simultaneous optimization of multiple, possibly conflicting, objective functions. This chapter introduces Two-lbests based multi-objective particle swarm optimization (2LB-MOPSO) [94] algorithm as a tool in finding trade-offs accounting for the relative importance of detection accuracy, latency of isolation of faulty nodes and energy overhead. A fuzzy-based mechanism is employed to extract the best trade-off solution from the Pareto optimal solutions provided by the 2LB-MOPSO algorithm [95].

The proposed cluster based distributed intermittent fault diagnosis (CDIFD) algorithm is executed at each data-gathering phase. Similar to the algorithm CDFD, the algorithm CDIFD exploits the spatially correlated sensor measurements. CDIFD detects a node as fault-free if the number of matches between its sensor reading and that of one-hop neighbors exceeds a predefined threshold value. The system model is presented in Section 4.2. The description, analysis and implementation details of the diagnosis algorithm are investigated in Section 4.3. The multiobjective optimization problem is discussed in Section 4.4. Simulation results are presented in Section 4.5 and finally, this chapter is summarized in Section 4.6.

4.2 System Model

4.2.1 Notations

The list of the notations used in this chapter and their meanings are shown in Table 4.1.

4.2.2 Assumptions

In addition to the assumptions made in Chapter 3, the following assumptions are considered for the CDIFD algorithm.

1. The sensor nodes are subjected to either permanent or intermittent faults.

Table 4.1: Notations.

n	Number of sensor nodes.
v_i	Sensor node.
F_{state_i}	Fault state of v_i.
x_i	Sensor reading at node v_i.
S_{x_i}	Standard deviation of I successive sensor measurements of v_i.
$N(v_i)$	One-hop neighbor set of v_i.
$N_{S_{x_i}}$	Number of one-hop neighbors report similar standard deviation set $\{S_x\}$.
T	Inter-test interval.
p	Fault probability.
P_e	Probability that a error appears and is not detected by a test.
P_{cerr}	Channel error probability.
T_{out}	Timeout timer.
k_{max}	Maximum number of test repetitions.
FAD	Fault appearance duration.
FDD	Fault disappearance duration.
δ_1	Application specific constant.
θ	Optimal threshold.
θ_1	Detection error threshold.
λ_k	Failure rate of Weibull distributed FDD.
β	Shape parameter of Weibull distribution.
μ	Failure rate of exponentially distributed FAD.
γ	Failure rate of fault-free nodes (exponentially distributed).
NP	Population size.
Q	Number of solutions in non dominated set.
d_a	Average node degree.

2. Faults are detected only through tests based on comparisons of sensor reading between neighboring nodes where tests are scheduled at the periodic time kT $(k = 1, 2, \cdots)$ for a fixed T.

3. Test is not perfect, i.e., a fault appears and is detected by the test with probability $1 - P_e$ and not detected with probability P_e.

4.2.3 Network, Channel, and Energy Model

In this chapter, we consider the network model, channel model, and energy model the same as specified in Chapter 3. The WSN consists of n sensor nodes where all sensor nodes are arranged into non-overlapping clusters. Gilbert-Elliott channel model is considered [86, 87]. Similar to [88], this chapter assumes a simple energy

model for the radio hardware energy dissipation.

4.2.4 Fault Model

The proposed model considers both hard and soft faults. If a node is hard faulty, the sensor node is unable to communicate. A soft faulty node continues to operate and communicate with altered behavior. Both the hard and soft faults may appear continuously or intermittently. The model based on the two-state Markov chain of the reference [96]. The sensor fault probability p is defined as

$$p = P(S = \neg x | A = x) \tag{4.1}$$

where the real temperature reading obtained by the sensor node is represented by variable S and the actual ambient temperature is represented by variable A. x is any value of S and A and $\neg x$ is any value not equals to x.

4.2.5 Diagnostic Model

Each sensor node produces temperature measurements at the discrete time kT. The interval between two successive diagnosis rounds is sampled such that each sample duration is $I \times T$, where I is an integer. At each sample interval, each node calculates and stores the standard deviation of these I readings. Each node broadcasts these standard deviations along with the routine sensed data in their allotted TDMA time slots. A sensor node performs a self-test and is declared as fault-free if the reading matches with received identical sensor measurement x and the number of matches between its standard deviations of temperature. Fault-free nodes fail to pass the threshold test later been diagnosed as fault-free through a fault-free neighbor. We redefine the matching criteria discussed in Chapter 3 to detect the intermittent fault as

$$M'_{c_{ik}} = \begin{cases} S_{x_i} \approx S_{x_k}(1) & \text{if } |S_{x_i} - S_{x_k}| < \delta_1 \\ S_{x_i} \not\approx S_{x_k}(0) & \text{otherwise} \end{cases} \tag{4.2}$$

where S_{x_i} is the standard deviation of I successive sensor measurements of sensor node v_i and $\delta_1 = C \times f(T_{diff}(S_{x_i}, S_{x_k}))$. $T_{diff}(S_{x_i}, S_{x_k})$ is the time

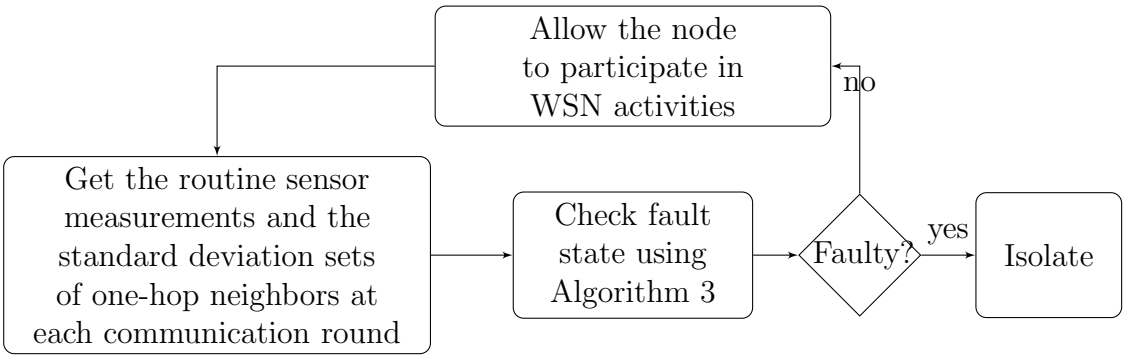


Figure 4.1: Flow diagram to detect an intermittent fault.

difference between the time v_i takes its own I^{th} measurement and v_i receives sensor measurement of $v_k \in N(v_i)$ and C is a constant.

4.3 The Proposed CDIFD Algorithm

4.3.1 Description of the Algorithm

Similar to the algorithm CDFD, the algorithm CDIFD consists of three phases namely the clustering phase, the fault detection phase and the dissemination phase. The clustering phase and dissemination phase are similar to CDFD algorithm proposed in Chapter 3. However, the detection phase of CDFD algorithm is modified to detect intermittent faults. In the clustering phase, the WSN is partitioned into different non-overlapping clusters. In the fault detection phase, each cluster head accumulates the fault states of their affiliated member sensor nodes. In the dissemination phase, the cluster level local view is disseminated in the WSN such that each sensor node correctly diagnoses the state of each sensor node in the WSN.

Detection Phase

To test for permanent faults, any particular test need only be applied once. This is because permanent faults are software or hardware faults that always produce errors when they are fully exercised. In contrast, the only approach to test for intermittent faults is through repeated application of tests. To detect intermittent faults, each node produces sensor readings at discrete times kT $(k = 1, 2, 3, ...)$ and stores in its local memory. It calculates the standard deviation of I successive

Algorithm 3 CDIFD (Detection Phase)

1: Each node regularly records its sensor measurement at discrete time kT.
2: // The inter data gathering interval is sampled with sampling interval IT. The algorithm is executed at each data gathering phase.
3: Calculate the standard deviations of successive sensor measurements in each sample interval and generate a standard deviation set $\{S_{x_i}\}$.
4: Broadcast x_i and $\{S_{x_i}\}$ in its TDMA time slot.
5: Set timer T_{out}.
6: Determine $\{N_{S_{x_i}}\}$, the set of one-hop neighbors report similar sensor measurement x and identical standard deviation set $\{S_x\}$.
7: **if** $T_{out} = true$ **then**
8: Declare unreported nodes as possibly hard faulty.
9: **end if**
10: **if** ($x_i \approx x$ and $\{S_{x_i}\} \approx \{S_x\}$ and $|\{N_{S_{x_i}}\}| < \theta$) or ($x_i \not\approx x$ and $\{S_{x_i}\} \not\approx \{S_x\}$ and $|\{N_{S_{x_i}}\}| \geq \theta$) **then**
11: $F_{state_i} \leftarrow$ Possibly soft faulty.
12: **else**
13: $F_{state_i} \leftarrow$ Fault-free.
14: **end if**
15: Broadcast the F_{state_i}.
16: Node identified as possibly soft faulty checks for a node $v_k \in N(v_i)$ such that F_{state_k} is fault-free. If such v_k exists, $\{M_{c_{ik}}\}$ equals to 1, all elements of $\{M'_{c_{ik}}\}$ equals to 1 and $v_k \in \{N_{S_{x_i}}\}$ then set F_{state_i} as fault-free or else faulty. Broadcast the fault table FT_i.
17: If v_i is cluster head, it constructs a cluster level local view and takes a decision on possibly hard faulty nodes by comparing the fault tables of member nodes.

sensor measurements, where I is an integer. Each node v_i broadcasts the routine sensed data and the set of standard deviations $\{s_{x_i}\}$ in its allotted TDMA time slot. For instance, if the duration between two data-gathering round is $50T$ and $I = 10$, then $|\{s_{x_i}\}| = 5$. Upon receiving the routine sensor measurements and the standard deviation sets, v_i buffers these routine sensor measurements and standard deviation sets and marks their times of reception. Node v_i next forms a set of neighbors $\{N_{S_{x_i}}\} \subseteq N(v_i)$ those have reported identical sensor measurements and identical standard deviation sets, say x and $\{S_x\}$ respectively. In this approach, $\forall v_j \in N(v_i)$, two standard deviation sets are similar if each element in the set $\{s_{x_i}\}$ is similar to the corresponding elements in the set $\{s_{x_j}\}$. The decision regarding

the fault state of a node is taken as follows—

$$v_i = \begin{cases} \text{Fault-free} & \text{if } x_i \approx x,\ \{s_{x_i}\} \approx \{S_x\} \text{ and } |\{N_{S_{x_i}}\}| \geq \theta \\ \text{Possibly soft-faulty} & \text{otherwise.} \end{cases} \tag{4.3}$$

A description of the detection phase of the algorithm CDIFD is presented in Algorithm 3. The operation of the algorithm is described by the flow diagram in Figure 4.1. The conditional block labeled "Faulty?" represents the snapshot view of the current diagnostic round. The algorithm loops as long as no errors from a node are detected. The node is isolated when a fault is observed.

4.3.2 Analysis of the Algorithm

Once an intermittent fault is activated in a sensor node, faults are observable for a duration called fault appearance duration (FAD) before they disappear. Eventually, errors will reappear after fault disappearance duration (FDD) either because of permanent faults or correlated intermittent faults. This is depicted in Figure 4.2. The behavior of intermittent faults can be characterized by measuring or estimating the probabilities of error disappearance and reappearance in discrete time kT.

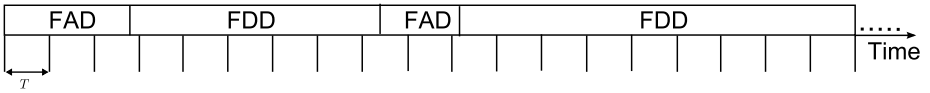


Figure 4.2: Appearance and disappearance of a fault.

The state of a sensor node is modeled as four-state Markov model. Figure 4.3 depicts this model where the transition probabilities between different states of the sensor node are shown. According to the proposed model, the node can be in

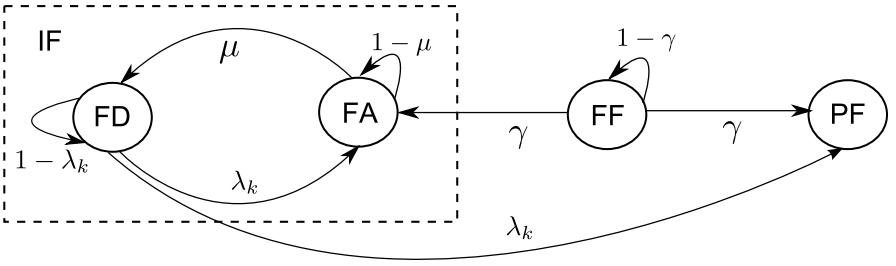


Figure 4.3: Analytical model.

either one of the four states — fault-free (FF), permanent faulty (PF), intermittent faulty and fault is active (FA), and intermittent faulty but the fault is inactive (FD). The sensor node in FF state can make a transition to either PF state or FA state with a rate γ. From FD state, it can go either to PF state or to FA state or stay in FD state. In order to analyze an intermittent fault in more details we focus on FA and FD states, which can be visualized as a two-state Markov model. The state FA (1) corresponds to fault exits and appears and state FD (0) corresponds to fault exits but does not appear. The probabilities for going from one state at time kT to either state FA or FD at time $(k+1)T$ depends on FDD and FAD respectively. The FDD for intermittent faults in a sensor node is system and deployment specific, thus, unpredictable in most practical scenarios. Intermittent faults usually exhibit a relatively high occurrence rate after its first appearance and eventually tend to become permanent. Therefore, as suggested in [21] a Weibull distribution is considered for FDD with shape parameter $\beta > 1$ and failure rate λ_k. An exponential distribution is considered for FAD with a constant failure rate $\mu = (1/\text{mean time in FA state})$ [21,96]. A similar distribution is considered for time to failure of a fault-free node with the constant failure rate $\gamma = (1/\text{mean time in the fault-free state})$. In practice $\mu \gg \lambda_k \gg \gamma$.

In order to devise such a model, let $\{F_j\}$ is the state space where F_0 denotes that node is fault-free and F_1 denotes that node is intermittent faulty. Let $\{t_k\}$ is the test pattern where t_k is the k^{th} test performed by the sensor node by using Algorithm 3 at time kT $(k = 1, 2, \cdots)$. The outcome of the k^{th} test is 0 if the node is either fault-free or node is intermittent faulty but the fault does not appear during the test. Since effect of a fault is not always present, deriving an optimal test pattern which can certainly detect the intermittent fault is hard to realize. In order to get a near optimal test pattern, we consider an inequality where the probability that an intermittent fault exists and is not detected must be smaller than the error threshold θ_1. Using the fact that the network is sampled with sampling period T, the following inequality is obtained

$$P(F_1|t_k = 0) \leq \theta_1 \tag{4.4}$$

For $k = 1$, using Baye's rule we can write

$$\frac{P(t_1 = 0|F_1) \cdot P(F_1)}{P(t_1 = 0|F_0) \cdot P(F_0) + P(t_1 = 0|F_1) \cdot P(F_1)} \leq \theta_1 \tag{4.5}$$

For k_{max} number of tests the above equation can be rewritten as

$$\frac{\prod_{k=1}^{k_{max}} P(t_k = 0|F_1) \cdot p}{(1-p) + \prod_{k=1}^{k_{max}} P(t_k = 0|F_1) \cdot p} \leq \theta_1 \tag{4.6}$$

The term $\prod_{k=1}^{k_{max}} P(t_k = 0|F_1)$ of (4.6) defines the probability that the fault remains inactive at time instants kT, where $k = 1, 2, \cdots, k_{max}$. Thus, the inequality can be rewritten as

$$\frac{\prod_{k=1}^{k_{max}} P_{00}(kT) \cdot p}{(1-p) + \prod_{k=1}^{k_{max}} P_{00}(kT) \cdot p} \leq \theta_1 \tag{4.7}$$

where $P_{00}(kT)$ is called the state transition probability, which is the conditional probability that the sensor node will be in the state FD (0) at time kT immediately after the next transition, given that it was in the state FD (0) at time $(k-1)T$. This probability is [96, 97]

$$P_{00}(kT) = \frac{\mu}{\mu + \lambda_k} + \frac{\lambda_k}{\mu + \lambda_k} e^{-(\lambda_k + \mu)T} \tag{4.8}$$

Equation (4.7) is derived under perfect test condition, i.e., a fault is always detected by a test when it occurs. Since we adopt neighbor coordination as a test to detect faults, thus, a fault is detected by a test with probability $1 - P_e$ and is not detected with probability P_e. The probability P_e is (3.16)

$$P_e = f_1 \cdot \left(1 - p - \sum_{l=0.5(N-1)}^{N} (1-p) f_l + p f_{N-l}\right)$$

where f_l is the probability that l out of N 1-hop neighbors of a node are fault-free. For imperfect test condition, equation (4.7) can be rewritten as

$$\frac{\prod_{k=1}^{k_{max}} P_{00}(kT) \cdot p \cdot (1 - P_e)}{(1-p) + \prod_{k=1}^{k_{max}} P_{00}(kT) \cdot p \cdot (1 - P_e)} \leq \theta_1 \tag{4.9}$$

As comprehended from (4.9), a better trade-off between detection accuracy, detection latency and energy overhead can be achieved by properly tuning the detection parameters such as k_{max} and T.

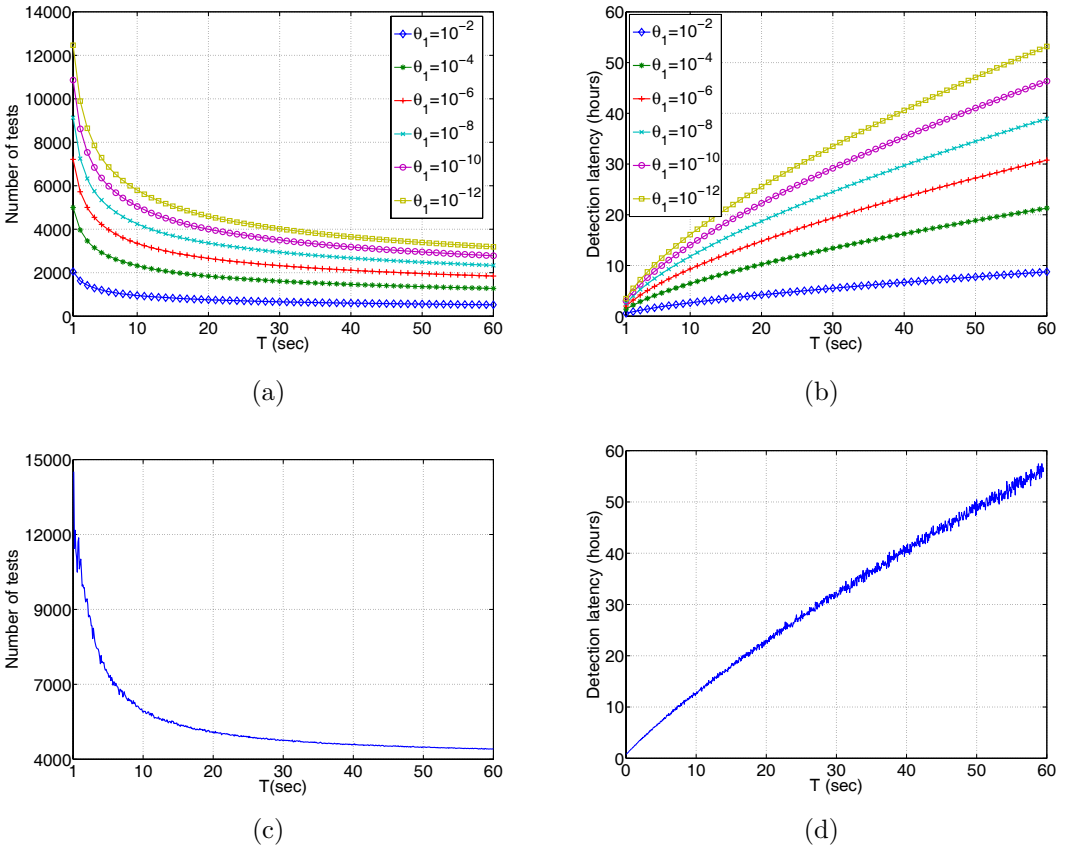


Figure 4.4: Impact of design parameters.

Impact of Design Parameters on fault Detection

The modeling framework discussed in earlier sections allow us to highlight detection accuracy, detection latency and energy overhead trade-offs in detecting an intermittent fault. To evaluate the impact of the design parameters on these trade-offs we have first used (4.9) to find out the number of tests required to detect faults and the detection latency at varying values of T and θ_1. These theoretical results are shown in Figures 4.4(a) and 4.4(b) respectively. Second, we have conducted a simulation on a simple network to find the impact of these design parameters. This simple network we considered has one intermittent faulty node surrounded by four fault-free one-hop neighbors. For simulation, the mean value

of FAD is considered $50ms$ where FAD is exponentially distributed. The FDD is assumed to follow a Weibull distribution with increasing failure rate ($\beta = 1.5$) and expected value of 1 *hour*. We run the experiment until the fault is detected, and the results are shown in Figure 4.4(c) and 4.4(d). As discussed earlier and shown in Figures 4.4(a) and 4.4(c), the number of tests required and thus the number of messages exchanged to detect the intermittent faults decreases for an increase in T. Figures 4.4(b) and 4.4(d) show the latency in detecting the intermittent fault. It is observed that the latency tends to increase with T. As comprehended from Figures 4.4(a) and 4.4(b), better detection accuracy, i.e., extremely small value for θ_1 can be achieved at the cost of the number of messages to be exchanged and detection latency.

Calculation of Objective Functions

From the above discussions, it can be concluded that the objectives are conflicting. These two conflicting objectives are; 1) to minimize the detection latency and 2) to minimize energy overhead (energy overhead is proportional to number of tests), while satisfying detection error constraint. This problem is formulated, mathematically, in this section.

Energy Overhead: The number of messages exchanged to detect intermittent faults is significant in WSNs as the energy consumed by a sensor node is directly proportional to the amount of traffic it generates or receives. Thus, a reduction in the number of tests required (k_{max}) to detect an intermittent faulty node will in turn significantly decrease the energy overhead. From Lemma 5, the normalized total energy dissipation can be expressed as

$$F_1 = k_{max}(\log_2 n + 1)(E_{Tx} + E_{Rx}) \tag{4.10}$$

We normalize the total energy consumption with respect to the number of nodes participated in diagnosis.

Detection Latency: Detection latency is the time elapsed between the first occurrence of the fault, and the fault detected. Thus, the detection latency is a function of k_{max} and T. As discussed earlier and shown in Figure 4.4(b) and 4.4(d), detection latency increases with T and might be undesirable for critical

applications with short mission time. The detection latency can be expressed as

$$F_2 = k_{max} \cdot T \tag{4.11}$$

Constraint Function: There is mainly one constraint corresponding detection error that should be satisfied, which is given as

$$\frac{\prod_{k=1}^{k_{max}} P_{00}(kT) \cdot p \cdot (1 - P_e)}{(1-p) + \prod_{k=1}^{k_{max}} P_{00}(kT) \cdot p \cdot (1 - P_e)} \leq \theta_1 \tag{4.12}$$

Implementation

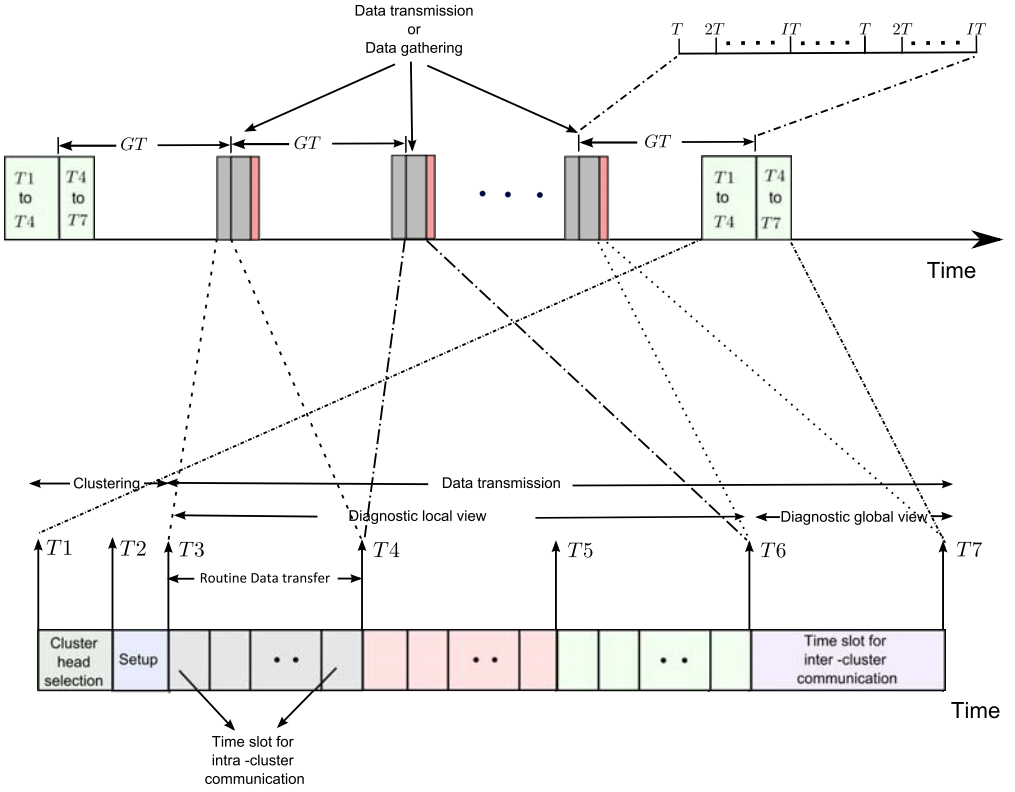


Figure 4.5: Time line showing intermittent fault detection.

In this section, we discuss the design details for practical deployment of the intermittent fault diagnosis algorithm. Each sensor node in the WSN is scheduled to take sensor measurement at the discrete time kT. As shown in Figure 4.5, the data-gathering stage is scheduled at GT where G is an integer and is application specific. For instance, applications with short mission time need the data to be

gathered more frequently in contrast to applications, where frequency of data gathering is less. For applications with long mission time, GT is large. Thus, to detect intermittent faults, G/T number of sensor measurements needs to be broadcasted by each node. This in turn make the packet to grow with G. Since energy consumed by a sensor node is directly proportional to the number of bits it transmits or receives, the energy overhead will be more for large value of G and may not be practically implementable. To address this issue, we suggest sampling the interval GT where each sample constitutes of I consecutive senor measurements (Figure 4.5). The standard deviation of these I sensor measurements correspond to each sample interval are calculated and broadcasted along with the routine data at its defined slot during $T3$ to $T4$ of each data-gathering phase. This in turn reduces the packet size and makes the algorithm energy efficient. Use of standard deviation instead of individual measurements does not affect the detection performance since the rate of change in sensor measurements of a fault-free sensor over time is very less. In addition, a sensor often reports unusually high or low sensor measurement during FAD. Thus, the standard deviation of sensor measurements of a sample interval with at least one incorrect measurement will be distinguished from the corresponding standard deviations of one-hop neighbors with all true measurements.

During T4 to T5 of each data-gathering phase, nodes broadcast the local decisions in their time slot. The decisions of one-hop neighbors are stored in the local fault table. At T5 nodes with possibly soft-faulty status take a final decision using this fault table and broadcast its updated decision in its time slot during T5 to T6 of each data-gathering stage. Cluster head turns off their radio once their TDMA time slots run out. At T6, all cluster heads wake up, and the inter-cluster communication is triggered. Cluster heads transmit control messages and data packets using CSMA. The local diagnostics and the routine data are aggregated through the spanning tree up to the sink. Thus, at T7, the sink has the global fault state view of the WSN. The sink broadcasts this global view.

4.4 Multiobjective Optimization Problem

Multiobjective optimization is the process of simultaneously optimizing two or more conflicting objectives subject to certain constraints. Since many conflicting objectives to be optimized simultaneously, there is a set of possible solutions of equivalent quality. Most real-world problems employ the optimization of several objectives, which are often conflicting in nature. A multiobjective optimization problem with M conflicting objectives can be defined as in [98]:
Maximize/minimize

$$\begin{aligned} y &= f(x) \\ &= (f_1(x), f_2(x), ..., f_M(x)), x \in [X_{min}, X_{max}] \end{aligned}$$

subject to:

$$g_j(x) \leq 0, j = 1, ..., J$$

$$h_k(x) = 0, k = 1, ..., K$$

where x and y are the decision vector and the objective vector respectively. Different from the single objective optimization, there are two spaces to be considered. One is the decision space denoted as x and the other is the objective space denoted as y.

Definition 1 *Let w_i and w_j are two solutions to a multiobjective problem. w_i dominates w_j if w_i performs at least as good as w_j with respect to all the objectives and performs strictly better than w_j in at least one objective [94].*

Definition 2 *Among a set of solutions W, the non-dominated set of solutions W' are those that are not dominated by any member of the set W [94].*

Definition 3 *When the set W is the entire feasible search space, the resulting non-dominated set W' is called the Pareto-optimal solution set [94].*

4.4.1 Finding Pareto Optimal Solution

Evolutionary algorithms are correctly fitted to multiobjective optimization problems as they are essentially based on biological processes, which is inherently multiobjective. An extensive survey on multiobjective evolutionary algorithms are well presented in [99]. Central to these articles, considering superior performance

for solving multi objective problems, the 2LB-MOPSO [94] and NSGA-II [100] algorithms have been used in this study.

In NSGA-II, initially a random population of size H, which is sorted based on the non-domination, is created. This population subsequently undergoes selection, crossover and mutation processes to produce an offspring population of size H. A combined population of size $2H$ is formed from the parent and offspring population. Next, the population is sorted according to the non-domination relation. This in turn classifies the complete population into several non-dominated fronts based on the values of the objective functions. Until each member of the population falls into one front, the other fronts are determined. The new parent population is generated by adding the solutions from the first front. Several non-dominated fronts are discarded as the population size is predefined. The required numbers of members for the new population are selected using a new parameter called crowding distance. The crowding distance describes how close an individual is to its neighbors.

Similar to GA, the PSO algorithm has been successfully extended to multiobjective optimization problems. Different from other variants of MOPSO algorithms, the 2LB-MOPSO algorithm uses two local bests instead of one personal best and one global best to lead each particle. The two local bests are selected to be close to each other in order to enhance the local search ability of the algorithm. Compared to the other variants of MOPSO algorithms, the 2LB-MOPSO algorithm shows great advantages in maintaining a good diversity of the solutions, convergence speed and fine-searching ability.

In 2LB-MOPSO algorithm, NP number of particles are randomly and uniformly initialized in the D-dimensional search space. Next, the fitness values of all particles are evaluated and all current positions set to be $A(0)$, the external archive. An external archive is commonly used to store the non-dominated solutions obtained in the search process. The size of the archive is usually restricted to a pre-specified size which is normally the same as the finally required approximation solution set size. In 2LB-MOPSO algorithm, the initialized archive includes all initialized solutions at iteration 1. In order to select the first *lbests* for a particle from the $A(0)$, an objective is first randomly selected followed by a

random selection of a non empty bin of the chosen objective. Within this bin, the archived member with the lowest front number and among these with the highest crowding distance is selected as the first *lbests*. The second *lbests* is selected from a neighboring non empty bin with the lowest front number and the smallest Euclidean distance in the parameter space to the first *lbests*. As velocity of each particle is adjusted by the two *lbests* from two neighboring bins, the flight of each particle will be in the direction between the positions of two *lbests* and oriented to improve upon the current non-dominated solutions.

Since the first *lbests* of every particle is chosen randomly, every particle should not be assigned with a new pair of *lbests* which come from the different pair of bins in every iteration. This is because the flight of each particle will be almost random in this case. Therefore, after assigning a pair of *lbests* to a particle, the number of iterations the particle fails to contribute a solution to the archive A(t) is counted. The particle is reassigned with another pair of *lbests* when the count exceeds a pre-specified threshold. When the count is less than or equal to the pre-specified threshold during the iterative optimization stage, two *lbests* are chosen from the same assignment of the objective and the bin as used in the last iteration. The particle will accelerate potentially in a direction between the two *lbests* and hence may explore the region of the two *lbests*.

The velocity and position of each particle are updated. If any dimension exceeds the search space, then they are reset to the corresponding bound value. Next, the fitness value of a particle is evaluated. In every iteration, all new positions $Q(t)$ generated in iteration t is combined with the members in the archive $A(t)$ to obtain the mixed-temporary external archive. The sorted archive $R(t)$ is obtained by applying the non-domination sorting to this mixed-temporary archive. During this process, all the sorted solutions retain two indicators, namely, the front rank and crowding distance value. The sorted solution with the lowest front rank is first included in the archive $A(t+1)$. When the size of the archive equals to the permitted maximum size of the archive, the crowding distance is applied to select the required number of members to be included in $A(t+1)$ from the lowest front that still remains unselected in the archive $R(t)$.

4.4.2 Performance Metrics and Best Trade-off Solution

All the existing multiobjective optimization algorithms aim to find solutions as close as possible to the Pareto optimal front and as diverse as possible in the non-dominated front. Different performance metrics to measure these two objectives have been suggested in the literature. Since the true Pareto-optimal front for the proposed application is unknown, for performance analysis, we consider coverage of the Pareto front [101], and spacing of the Pareto front [102]. The first metric measures the convergence of the Pareto front, while the second metric measures the distribution of solutions along the Pareto front.

Coverage of The Pareto Front

Let A and B are two Pareto-optimal sets. This metric, measures the relative spread of solutions between two non-dominated sets. The function H maps the ordered pair (A, B) to the interval [0, 1] and is given by

$$H(A,B) = \frac{|\{b \in B | \exists a \in A : a \succeq b\}|}{|B|} \tag{4.13}$$

where $|B|$ represents the number of solutions in the set B, and $a \succeq b$ implies that solution a weakly dominates solution b. The value $H(A, B) = 1$ implies that all decision vectors in B are weakly dominated by A. In contrary, $H(A, B) = 0$, implies that none of the points in B are weakly dominated by A. If $H(A, B) > H(B, A)$, then the set A has better solutions than the set B.

Spacing

Schott [102] introduced a metric namely *Spacing* that measures the distribution of the solutions over the non-dominated front. *Spacing* between solutions is computed as

$$Spacing = \sqrt{\frac{1}{Q-1}\sum_{i=1}^{Q}\left(\frac{Y_i}{\bar{Y}} - 1\right)^2} \tag{4.14}$$

where

$$Y_i = min_j \sum_{m=1}^{M} |F_m^i - F_m^j| \quad \text{for } j = 1, \cdots, Q \text{ and } i \neq j. \tag{4.15}$$

Q is the number of solutions in the non-dominated set, M is the total number of objectives to be optimized and $\bar{Y}$ is the mean of all the Y_i. The nearer the

value of *Spacing* to zero, the more uniformly distributed the solutions found over the Pareto optimal front.

Fuzzy Decision Making

Upon obtaining a set of Pareto optimal solutions, we need to find a best optimum trade-off. As suggested in [95], the fuzzy membership functions that represent the goals of each objective function are used. The fuzzy sets are defined by these membership functions. These functions represent the degree of membership in certain fuzzy sets using values from 0 to 1. The membership functions for both objectives are defined as

$$\mu_i = \begin{cases} 1 & F_i \leq F_i^{min} \\ \frac{F_i^{max} - F_i}{F_i^{max} - F_i^{min}} & F_i^{min} < F_i < F_i^{max} \\ 0 & F_i \geq F_i^{max} \end{cases} \tag{4.16}$$

where F_i^{min} and F_i^{max} are the minimum and maximum values from non-dominated solutions of each objective function, respectively. For each non-dominated solution, the normalized membership function, can be calculated as

$$\mu^r = \frac{\sum_{i=1}^{2} \mu_i^r}{\sum_{r=1}^{R} \sum_{i=1}^{2} \mu_i^r} \tag{4.17}$$

where, R is the number of non-dominated solutions. The solution that attains the maximum membership μ^r in the fuzzy set can be chosen as the best solution.

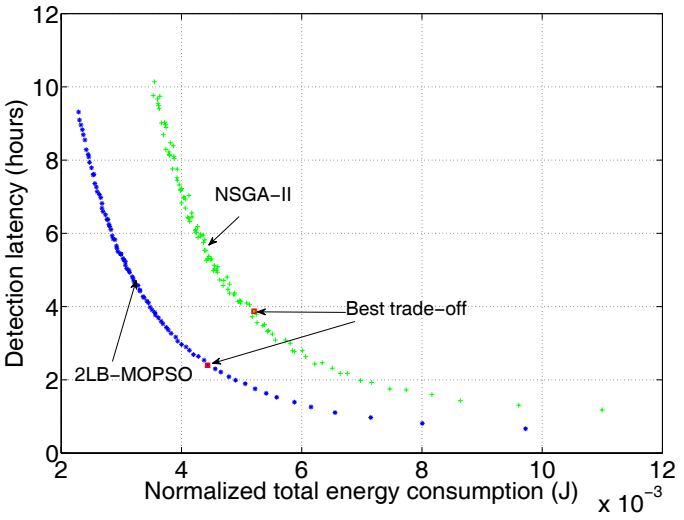
$$\text{Best solution} = max\{\mu^r : \ r = 1, ..., R\}$$

4.5 Simulation Experiments

4.5.1 Experiment 1: Tuning of detection parameters

This section is primarily meant to study how the design parameters namely k_{max} and T affect detection of intermittent faults in terms of two important figures of merit: the detection latency and energy overhead while maintaining low detection error. The NSGA-II and 2LB-MOPSO algorithm based proposed approach for

tuning of the detection parameters have been implemented in MATLAB. The mean value of FAD is considered 50 ms where FAD is exponentially distributed. The FDD is assumed to follow a Weibull distribution with an increasing failure rate ($\beta = 1.5$) and expected value of 1 $hour$. For 2LB-MOPSO algorithm; the parameters are set as in the [94]: count and number of bins are considered as 5 and 10 respectively, population size $NP = 50$, inertia weight $\omega = 0.729$, $C1 = C2 = 2.05$, $V_{max} = 0.25(X_{max} - X_{min})$. For NSGA-II algorithm (real-coded), we use a population size of 50, crossover probability of 0.9 and mutation probability of 0.1. As suggested in [100], the distribution indexes for crossover, and mutation operators are set as $\eta_c = 20$ and $\eta_m = 20$. The decision variables are initialized with uniformly distributed pseudorandom numbers that take the range of these variables, i.e., $T = \text{rand}\ [T_{min}, T_{max}]$ and $k = \text{rand}\ [k_{min}, k_{max}]$. We consider $T_{min} = 1000\ ms$, $T_{max} = 60000\ ms$, $k_{min} = 1$ and $k_{max} = 15000$, and $\theta_1 = 10^{-20}$. The maximum function evaluations are set as 15000.



(a)

Figure 4.6: Trade-off curve.

Performance Analysis

In order to evaluate the performance, 20 independent runs were conducted for both NSGA-II and 2LB-MOPSO algorithms. To illustrate the difference between the Pareto fronts obtained with 2LB-MOPSO algorithm and NSGA-II, the Pareto

fronts obtained with one of the twenty runs of 2LB-MOPSO algorithm and NSGA-II are plotted in Figure 4.6. Here, we consider the normalized total energy which is the ratio between the total energy and the number of nodes participated in the detection. The quality of the Pareto-optimal solutions obtained with NSGA-II and 2LB-MOPSO algorithm is measured by the two aforementioned performance metrics. The best, worst, mean, median and standard deviation of the two performance metrics is presented in Table 4.2. The best average result with respect to each metric is shown in a bold font.

Table 4.2: Results of different performance metrics for 2LB-MOPSO and NSGA-II.

	2LB-MOPSO	NSGA-II
Coverage		
Best	0.9886	0.3126
Worst	0.7835	0.0192
Average	**0.9133**	0.2013
Median	0.8361	0.0133
Standard deviation	0.0821	0.1352
Spacing		
Best	0.2096	0.3862
Worst	0.3932	0.6696
Average	**0.3206**	0.5182
Median	0.3182	0.5021
Standard deviation	0.0749	0.1204

In Table 4.2, the value for $Coverage = 0.9133$ implies that 91.33% of the Pareto-optimal solutions obtained with NSGA-II are weakly dominated by the solutions obtained with 2LB-MOPSO algorithm. Likewise, the value for $Coverage = 0.2013$ means that only 20.13% of the solutions obtained with 2LB-MOPSO algorithm are weakly dominated by those with NSGA-II. In addition, the standard deviation of 2LB-MOPSO algorithm with respect to *Coverage* implies that the performance of 2LB-MOPSO algorithm is more stable. The distributions of the Pareto-optimal solutions over the non-dominated front obtained with 2LB-MOPSO algorithm and NSGA-II are evaluated with metric *Spacing*. Since a lower value of *Spacing* implies uniform spread of solutions, as shown in Table 4.2 for our application, 2LB-MOPSO algorithm outperforms

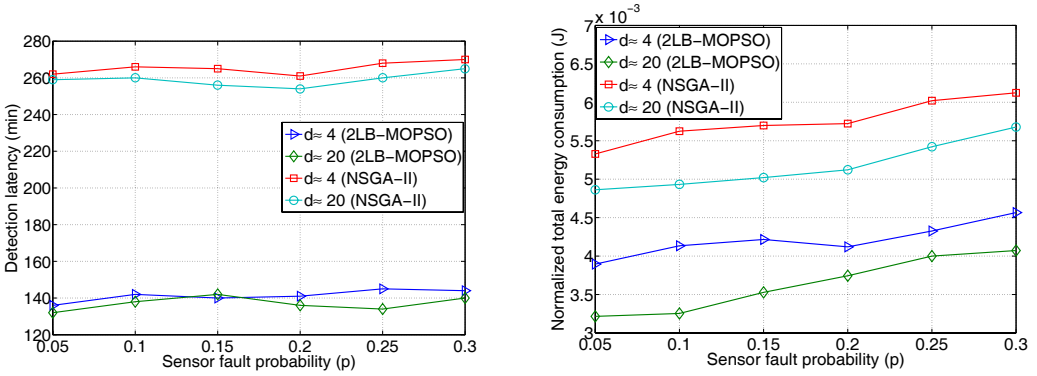
NSGA-II. The best trade-off solution is obtained on both the solutions by using the aforementioned fuzzy logic-based mechanism and is shown in Figure 4.6. As depicted in Figure 4.6, the best trade-off solution using 2LB-MOPSO algorithm-based approach (0.0044 $J, 150.7867$ $minutes$) is obtained for $T = 8600$ ms and $k_{max} = 1052$. Similarly, the best trade-off solution using NSGA-II based approach (0.0057 $J, 271.6728$ $minutes$) is obtained for $T = 12978$ ms and $k_{max} = 1256$.

4.5.2 Experiment 2: Time and energy efficiency

In order to further validate the obtained detection parameter T and measure its effectiveness, we chose to conduct an extensive set of simulations using Castalia-2.3b [52], a state-of-art WSN simulator based on the OMNET++ [53] platform. For the simulation purpose, a communication scenario has been generated with simulation parameters as summarized in Table 3.2, where nodes were uniformly distributed. In this experiment, we use the tuned detection parameters obtained using both 2LB-MOPSO algorithm and NSGA-II. Each sensor node senses data at every $T = 8600$ ms interval for 2LB-MOPSO based implementation and at every $T = 12978$ ms for NSGA-II based implementation and stores in its local memory. The values for G and I are set to 50 and 10 respectively. However, different values for G and I can be used depending on applications, and the type of sensors used. In this experiment, we assume temperature sensors. The channel error probability estimate at each node is 1×10^{-3}.

As discussed earlier both FAD and FDD are system specific and depend on multiple factors. Thus, to simulate the real fault scenario FAD follows a Weibull distribution with expected value ranging from 1 $minute$ to 10 $hours$ and FAD follows an exponential distribution with expected value ranging from 5 ms to 50 ms. All the intermittent faults are activated randomly before first tests, i.e., before 8600 ms for 2LB-MOPSO based implementation and before 12978 ms for NSGA-II based implementation. In this simulation, sensor nodes are assumed to be faulty with probabilities of 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, respectively. The transmission range is chosen for the WSN to have the desired average node

degree d_a. In this experiment, we attempted to illustrate the detection latency and normalized total energy overhead of the detection algorithm. All results are the average of results obtained on 100 topologies. For better analysis, we consider only intermittent faults. The average detection latency and the average normalized total energy overhead are shown in Figure 4.7(a) and 4.7(b) respectively for varying fault rate and d_a. As shown, both the detection latency and normalized energy overhead are less affected by the number of faults. The reason is that the detection of intermittent faults depends only on T and the detection latency depends on the number of test repetitions executed to detect the fault. The normalized total energy overhead depends purely on the number of messages exchanged to detect the fault. As discussed earlier more messages need to be exchanged if nodes fail to pass the threshold test. Since only intermittent faults are considered, the number of nodes failed to pass the threshold test is less. This is because the probability that fault appears in all the intermittent faulty neighbors at the time of test is less.



(a) (b)

Figure 4.7: Average detection latency and normalized total energy overhead: $P_{cerr} = 10^{-3}$.

4.5.3 Experiment 3: Efficiency with regard to d_a and p

In this experiment, the performance of the diagnosis algorithm in regard to DA and FAR is evaluated by first considering only intermittent faults and then considering both intermittent and permanent faults. In the later experiment, the number of intermittent and permanent faults are randomly chosen while maintaining the

total number of faults. For performance evaluation, we assume the number of intermittent and permanent faults do not change during the simulation period. Note that this assumption does not mean that the detection algorithm is not adaptive to change in fault type and fault rate. Since in all respect 2LB-MOPSO algorithm outperforms NSGA-II, we consider $T = 8600\ ms$ and $k_{max} = 1052$. To validate the obtained detection parameter T, the experiment was conducted for 21 epochs ($k_{max}/G = 1052/50 \approx 21$).

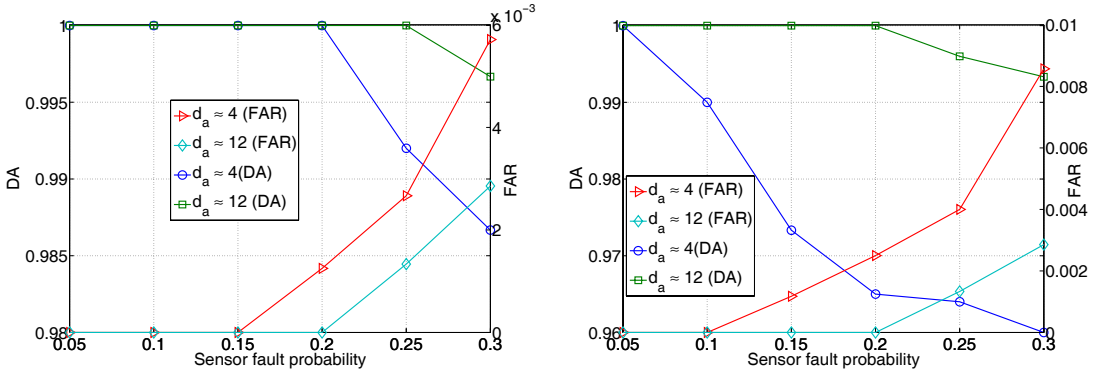
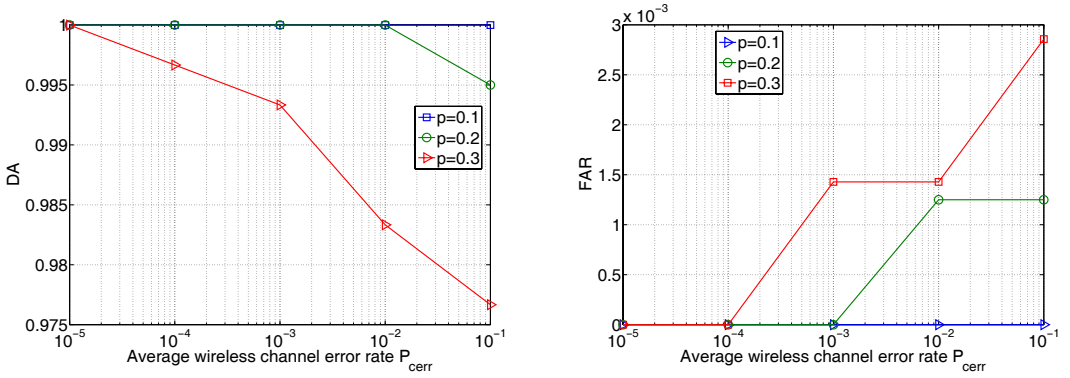


Figure 4.8: DA and FAR with $d_a \approx 4$ and $d_a \approx 12$ for a WSN considering (a) only intermittent faults and (b) both intermittent and permanent faults: $P_{cerr} = 1 \times 10^{-3}$.

Figures 4.8(a) and 4.8(b) show the average detection accuracy and average false alarm rate of the detection algorithm considering only intermittent faults. Interestingly, an improvement in both DA and FAR is observed. The reason of this improvement can be explained as — (i) a faulty node may be detected as fault-free only when the node has more than θ faulty neighbors and shows a match in comparison, and a fault-free node detected as faulty only when the node has more than θ faulty neighbors and in second round of test it does not find a node which is detected as fault-free in first round of test, (ii) for the scenario where all faults are intermittent, the probability of mentioned neighbors at the time of test is less as compare to the scenario where all faults are permanent because the probability that fault appears in all the faulty neighbors at the time of test is less.

4.5.4 Experiment 4: Effect of communication channel faults

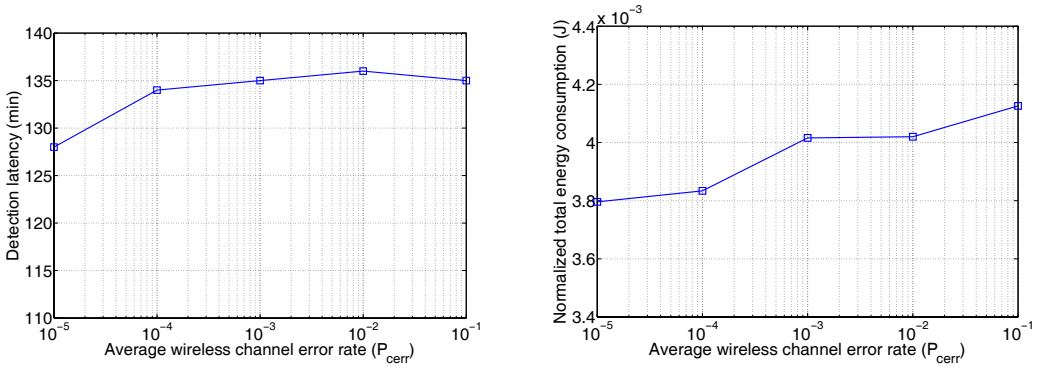
The robustness of the detection algorithm to faults in the communication channel is analyzed by estimating DA and FAR for various P_{cerr}. For better analysis, we consider only intermittent faults. As suggested in [103], for simplicity in the simulation P_{good} is taken as 0 and P_{bad} is taken as 1. P_{BG} is fixed to 1/8 and P_{GB} is varied to get different channel error probabilities P_{cerr}. We have used the previously generated scenario with $d_a = 12$ and the simulation was conducted for 31 epochs. The channel error rate is increased in steps from 10^{-5} to 10^{-1}. Faults in the communication channel might cause some fault-free nodes to fail in receiving the sensor measurements from its neighbors. This in turn decreases the effective neighbor size of a sensor node and might affect the local decision. However, as discussed and shown in Chapter 3, the proposed test algorithm shows better performance even in sparse WSNs. In addition, we consider only intermittent faults and the probability that the fault appears in all the faulty nodes at the time of test is very small. Thus, the effective neighbor size is more while considering only intermittent faults as compared to considering only permanent faults or both. Accordingly, as shown in Figures 4.9(a) and 4.9(b) the DA and FAR is less affected by channel error.



(a) (b)

Figure 4.9: DA and FAR at varying value of P_{cerr} (considering only intermittent faults): $d_a \approx 12$.

The robustness of the detection algorithm to faults in the communication channel is analyzed by estimating detection latency and normalized total energy



(a) (b)

Figure 4.10: Average detection latency and normalized total energy overhead at varying value of P_{cerr}: $d_a = 20$.

overhead in detecting intermittent faults for various P_{cerr}. As expected and shown in Figure 4.10(a), the detection latency is less sensitive to change in channel error rate. The reason is that detection of an intermittent fault only depends on T and number of test repetitions but not on the average node degree. As expected and shown in Figure 4.10(b) the normalized total energy overhead is less affected by varying channel fault rate.

4.6 Summary

The diagnosis of intermittent faults in WSNs is modeled as a biobjective optimization problem. The NSGA-II and 2LB-MOPSO algorithm are used as tools to tune the detection parameters such as T and k_{max}. A fuzzy based mechanism is also used to find out the best compromised solution on the optimal Pareto front. In general, it is observed that 2LB-MOPSO algorithm outperforms NSGA-II for tuning of detection parameters. A high level (> 0.95) of DA is achieved while keeping the FAR low (< 0.01) for sparse WSNs. The proposed CDIFD algorithm effectively tolerates faults in communication channels. The CDIFD algorithm is energy efficient since the normalized total energy consumption is significantly less.

Chapter 5

Transient Fault Diagnosis in WSNs

5.1 Introduction

The occurrence of transient faults in sensor nodes of a WSN affects the normal operation of the network. The sensor nodes in WSNs are subjected to transient faults due to external interventions such as electromagnetic radiations, noise, etc. If a sensor node suffers from transient faults, the sensor node will not perform its desirable operation for a small duration of time. After the fault disappears, the fault may reappear after a long normal operational time. As a first step to solve this problem of diagnosing transient faults, it is necessary to discriminate the transient from intermittent or permanent faults. This is because a sensor node with transient faults does not necessarily imply that the sensor node should be isolated although the unstable environment might warrant a temporary shutdown [1]. A discrimination between transient and intermittent or permanent faults addresses the following key problems that are more likely in a WSN—

- Effective energy utilization. Isolation of sensor nodes with transient faults will reduce available sensor nodes in the WSN. This increases the network workload of each sensor node and in turn leads to faster depletion of sensor node battery energy and impacting network lifetime.

- Network coverage and connectivity. Isolation of fault-free nodes with transient faults will reduce the available sensor nodes in the WSN thus impacting network coverage and connectivity.

These issues motivate the need to design an efficient fault detection and discrimination algorithm suitable for WSNs. The proposed cluster based distributed fault diagnosis and discrimination (CDFDD) algorithm not only detects the faulty sensor nodes but also discriminates the transient from intermittent or permanent faults.

The rest of the chapter is organized as follows. The system model is discussed in Section 5.2. The description, analysis and implementation of the fault discrimination algorithm are discussed in Section 5.3. Simulation results are presented in Section 5.4 and finally, chapter summery is given in Section 5.5.

5.2 System Model

5.2.1 Notations

The list of the notations used in this chapter and their meanings are shown in Table 5.1.

Table 5.1: Notations.

n	Number of sensor nodes.
v_i	Sensor node.
F_{state_i}	Fault state of v_i.
S_{x_i}	Standard deviation of I successive sensor measurements of v_i.
$N(v_i)$	One-hop neighbor set of v_i.
T	Inter-test interval
r	Reward counter.
z	Penalty counter.
ξ	Penalty increment.
θ_2	Reward counter threshold.
θ_3	Penalty counter threshold.
p_p	Permanent fault probability.
p_i	Intermittent fault probability.
p_t	Transient fault probability.

5.2.2 Network, Channel, and Energy Model

In this chapter, we consider the network model, channel model, and energy model same as specified in Chapter 3. The WSN consists of n sensor nodes where all

sensor nodes are arranged into non-overlapping clusters. Gilbert-Elliott channel model is considered [86, 87]. Similar to [88], this chapter assumes a simple energy model for the radio hardware energy dissipation. This chapter follows the assumptions of Chapter 3 and 4 such as a static network, homogeneous sensor nodes, unique ID for each sensor node, variable transmission power level, symmetric links, and synchronized networks. The assumptions made in this chapter are similar to Chapter 5.

5.2.3 Fault Model

The proposed model considers both hard and soft faults in sensor nodes. If a node is hard faulty, the sensor node is unable to communicate. A soft faulty node continues to operate and communicate with altered behavior. Both the hard and soft faults may appear continuously or intermittently. A sensor node that gives an erroneous reading is not always treated as faulty. A sensor node exhibits consistent faulty behavior are detected as faulty.

5.2.4 Diagnostic Model

Each sensor node produces temperature measurements at the discrete time kT. The interval between two successive diagnosis rounds is sampled such that each sample duration is $I \times T$, where I is an integer. At each sample interval, each node calculates and stores the standard deviation of these I readings. Each node broadcasts these standard deviations along with the routine sensed data in their allotted TDMA time slots. A sensor node performs a self-test on the received identical sensor measurements, identical standard deviations of temperature measurements from one-hop neighbors and the derived optimum threshold. Fault-free nodes fail to pass the threshold test later been diagnosed as fault-free through a fault-free neighbor. A node detected as faulty enters the observation stage. In the observation stage, the health of the node is periodically monitored with interval T. The number of times the node pass (fail) the test is compared with a threshold namely reward (penalty) threshold, and a decision is taken. We use the matching criteria discussed in Chapter 4.

5.3 The Proposed CDFDD Algorithm

5.3.1 Description of the Algorithm

The fault discrimination algorithm consists of two phases namely fault detection phase and observation phase. The algorithm is executed at the discrete time kT where $k = 1, 2, \cdots$ and T is the inter-test interval. A sensor node detected as faulty in the detection phase undergoes the observation stage before being isolated from the WSN. A formal description of the algorithm is presented in Algorithm 4.

The observation phase decides whether to isolate the node from the WSN (intermittent or permanent faulty) or to reintegrate the node to the WSN (fault-free with transient faults). To discriminate transient from intermittent faults, this algorithm follows the count and threshold mechanism proposed by Serafini *et al.* in [104] which was primarily designed for wired interconnected networks. Similar to [104], our approach uses two counters namely reward (r) and penalty (z) counter to discriminate fault types with low latency and low energy overhead. Unlike [104] we first tune the inter-test interval T to detect the presence of faults with minimum test repetition. Second, we adopt the earlier discussed two-state Markov chain to model fault appearance and disappearance. We consider the time a node spends in the fault disappearance state to tune the detection parameters. We consider the following detection parameters:

- *Inter-test interval (T).* The time interval of two consecutive sensor measurements.
- *Reward counter threshold (θ_2).* The number of diagnostic rounds a node under observation shows expected behavior, after which a node is reintegrated to the WSN.
- *Penalty counter threshold (θ_3).* The number of correlated diagnostic rounds, after which a node gets isolated.
- *Adaptive penalty increments.* The penalties assigned after a fault is detected.

In the observation phase, the node under observation first initializes the penalty counter to one and the reward counter to zero. If a fault appears and is detected

Algorithm 4 CDFDD

Each node regularly records its sensor measurement at at discrete time kT.
Initialize $z = 0$ and $r = 0$.
// The inter data gathering interval is sampled with sampling interval IT. The algorithm is executed at each data gathering phase.
// Detection phase.
Execute detction phase of algorithm CDIFD.
if F_{state_i} is faulty and fault is detected for first time, i.e., $z = 0$ **then**
    The node enters to observation stage.
    Set $z = 1$.
end if
// Observation stage.
repeat
    if F_{state_i} is faulty **then**
        Reset reward counter, i.e., $r = 0$.
        if $\text{FDD}_i \leq \text{FDD}_{i-1}$ **then**
            Increment the penalty counter by ξ, i.e., $z = z + \xi$.
        else
            $z = z + 1$.
        end if
    else
        Increment the reward counter, i.e., $r = r + 1$.
    end if
    if $r > \theta_2$ and $z < \theta_3$ **then**
        Node is reintegrated. Set $z = 0$ and $r = 0$.
    else
        Node is isolated.
    end if
until Node is isolated or reintegrated

by the k^{th} test at time kT, it first reset the reward counter to zero. Second, it checks the present fault disappearance duration (FDD_i) with the preceding fault disappearance duration FDD_{i-1}. If $\text{FDD}_i \leq \text{FDD}_{i-1}$, the penalty counter is incremented by a factor equal to ξ. If $\text{FDD}_i > \text{FDD}_{i-1}$, then the penalty counter is incremented by a factor equal to one. This is because, after their first appearance, intermittent faults usually exhibit a relatively fast occurrence rate. This adaptive penalty increment ensures a faster isolation of intermittent faults and in turn decreases the detection latency. If the penalty counter exceeds its threshold (θ_3), the node is isolated from the WSN. Similarly, if the reward counter exceeds its threshold (θ_2), the node is reintegrated to the WSN.

5.3.2 Analysis of the Algorithm

In this section we analyze time complexity, message complexity and the implementation issues.

Complexity Analysis

Lemma 6 *The latency in isolating a faulty node is* $\sum_{i=1}^{\theta_3} k_{max_i} T$.

Proof: After the first appearance of the fault, the maximum number of test repetitions required to satisfy a minimum detection error is k_{max} (4.9). Since Weibull distribution is assumed for FDD of an intermittent faulty node, k_{max} changes after each fault appearance. A node will be isolated when the number of times the fault appears and is detected by the algorithm is greater than or equals to the threshold θ_3. Thus the latency in isolating a node that enters the observation stage is $\sum_{i=1}^{\theta_3} k_{max_i} T$. ∎

Lemma 7 *The number of messages exchanged to isolate a faulty node in observation stage is* $2n \sum_{i=1}^{\theta_3} k_{max_i}$.

Proof: From Lemma 3 it is evident that a single test execution requires exchange of two messages by each node explicitly for a diagnosis purpose. From Lemma 6, the number of test repetitions required to isolate a faulty node in the observation stage is $\sum_{i=1}^{\theta_3} k_{max_i}$. Thus, the number of messages needs to be exchanged to isolate faulty nodes in the observation stage is $2n \sum_{i=1}^{\theta_3} k_{max_i}$. ∎

Implementation

In this section, we discuss the design details for practical deployment of the proposed fault discrimination algorithm. The time line of the proposed approach follows the time line shown in Figure 4.5. As discussed in Chapter 4, each sensor node in the WSN is scheduled to take sensor measurement at the discrete time kT with $T = 8600\ ms$. The data-gathering stage is scheduled at GT where G is an integer. The interval GT is sampled, where each sample constitutes of I consecutive sensor measurements. The standard deviation of these I sensor

measurements correspond to each sample interval are calculated and broadcasted along with the routine data at its defined time slot. Each node takes the decision by comparing the corresponding standard deviations of one-hop neighbors. If a fault is detected, the node enters to the observation phase. The corresponding cluster head reschedules its TDMA schedule by excluding the node. The detected node initializes the penalty and reward counter to one and zero respectively. If the penalty counter reaches θ_3 the node is isolated. If the reward counter reaches θ_2 the node is reintegrated and joins a suitable cluster head.

5.4 Simulation Experiments

In order to measure the effectiveness of the proposed discrimination algorithm, we chose to conduct an extensive set of simulations using Castalia-2.3b [52], a state-of-art WSN simulator based on the OMNET++ [53] platform. For the simulation purpose, a communication scenario has been generated with simulation parameters as summarized in Table 3.2, where nodes were randomly distributed. Each sensor node senses data at every $T = 8600\ ms$ interval and stores in its local memory. The values for G and I are set to 50 and 10 respectively. However, different values for G and I can be used depending on application, and the type of sensors used. Every node exchanges data at an epoch with interval $G = 8600 \times 50\ ms$. In this experiment, we assume temperature sensors. The channel error probability estimate at each node is 1×10^{-3}.

5.4.1 Experiment 1: Tuning of Detection Parameters

There are several design parameters in the proposed approach, namely T, z, r, and ξ. In this experiment, we tune these parameters with regard to the accuracy, coverage, number of test repetitions and detection latency.

- *Accuracy* is the probability that a fault-free node with transient fault in the error-free state entering the observation phase is not isolated [104].
- *Coverage* is the probability that an intermittent faulty node in the error-free state entering the observation phase is isolated [104].

- *Number of test repetitions* is the measure of the number of times the test repeated to discriminate transient from intermittent or permanent faults.
- *detection Latency* is the time interval between a node detected faulty and its isolation.

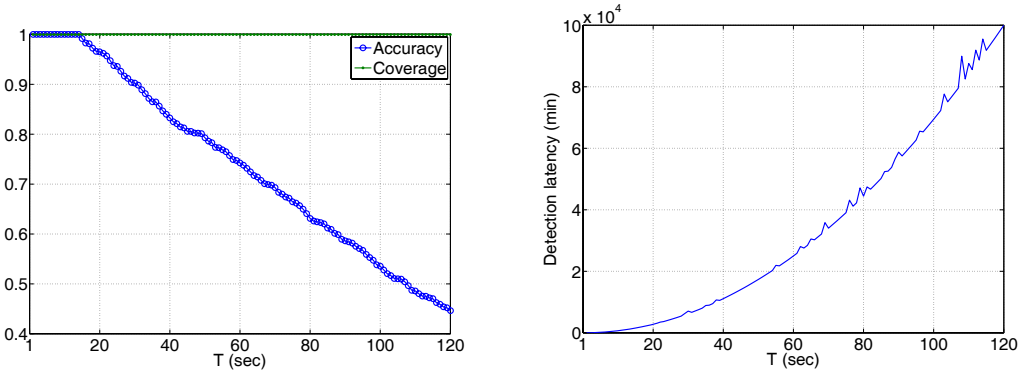
In this experiment, we have deployed 100 faulty nodes ($p = 0.1$) randomly. Each faulty node can exhibit the permanent, intermittent and transient faults. While conducting sensitivity analysis on each design parameter, we fix the others to the nominal values as summarized in Table 5.2. The transmission range of each node is chosen to have $d_a \approx 20$. This ensures that a fault is detected by a test (execution of the fault detection algorithm) if it appears at the time of test. In addition larger value of d_a ensures low FAR. However, this restriction in node degree is relaxed in subsequent experiments to observe the performance of the proposed approach in sparse WSNs.

Table 5.2: Design and system parameters and their nominal values.

Parameter	Description	Nominal value
θ_2	Reward threshold.	10^4
θ_3	Penalty threshold.	5
ξ	Penalty increment.	2
T	Inter-test interval.	8600 ms
FAD	Continuous distribution of fault appearance duration.	exponential
E[FAD]	Expected fault appearance duration.	$5ms$
FDD_u	Continuous distribution of fault disappearance duration of intermittent faulty node.	Weibull($\alpha = 1.4$)
$E[FDD]_u$	Expected fault disappearance duration of intermittent faulty node.	$1h$
FDD_h	Continuous distribution of fault disappearance duration of fault-free node with transient fault.	Exponential
$E[FDD]_h$	Expected fault disappearance duration of fault-free node with transient fault.	$100h$

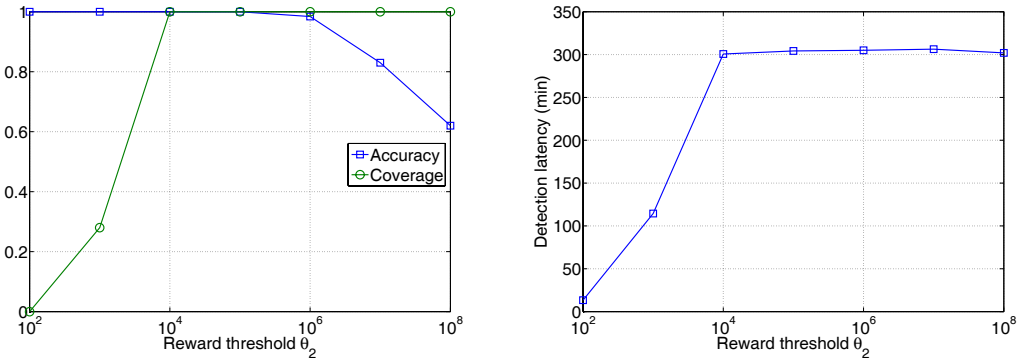
Figure 5.1(a) shows the average accuracy and coverage at varying values of T. This result confirms that T has a strong impact on average accuracy. It is observed that the average accuracy falls after $T = 9.4$ *sec*. This is because when T is excessively long, an excessively long time is required to reach the reward

threshold. For instance, this time for $T = 20$ *sec* and $\theta_2 = 10^4$ is 55.56 *hours*. The mentioned period of correct operation is too long and increases with T. Thus, the occurrence of subsequent transient faults will be viewed as correlated intermittent faults and the node will be isolated. It is observed that the average coverage remains unaffected by change in T. However, as shown in Figure 5.1(b), the average latency of isolation increases with T.



(a) (b)

Figure 5.1: (a) Accuracy and coverage at varying value of T. (b) Detection latency at varying value of T.

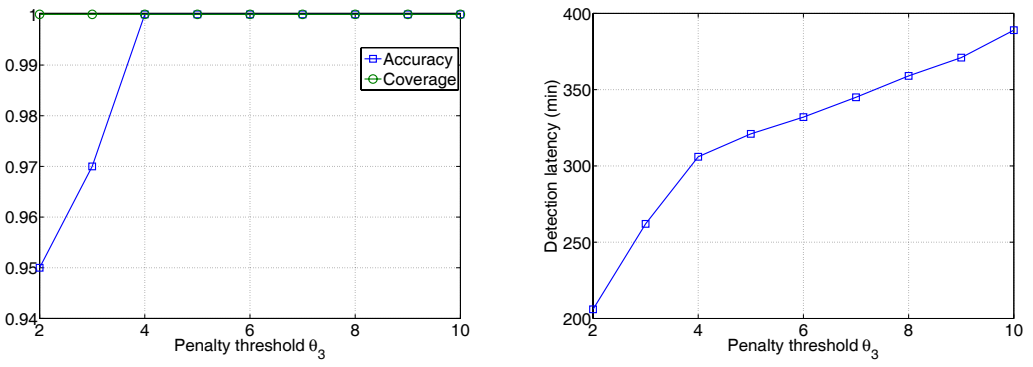


(a) (b)

Figure 5.2: (a) Accuracy and coverage at varying value of θ_2. (b) Detection latency at varying value of θ_2.

The impact of the reward threshold θ_2 on the average accuracy and coverage is shown in Figure 5.2(a). In the proposed fault discrimination algorithm a node is isolated if it fails before reaching the reward threshold θ_2. If θ_2 is too large, then a fault-free node with transient faults enters the observation stage may be isolated

causing poor accuracy. If θ_2 is too small, then intermittent faults will be treated as transient faults and will be reintegrated to the WSN causing poor coverage. This is because the value of θ_2 must be greater than the average number of test receptions required to detect the presence of a fault. Thus, proper tuning of θ_2 is crucial to achieve good discrimination. The best trade-off for the given scenario is observed at $\theta_2 = 10^4$. The average detection latency for varied values of θ_2 is shown in Figure 5.2(b). The average latency of isolation is reported almost unaffected for $\theta_2 \geq 10^4$. This is because 100% coverage is reported for $\theta_2 \geq 10^4$. In addition, detection latency depends only on T and the number of test repetitions required to reach the penalty threshold θ_3. Thus, for $\theta_2 \geq 10^4$ the detection latency is negligibly affected.



(a) (b)

Figure 5.3: (a) Accuracy and coverage at varying value of θ_3. (b) Detection latency at varying value of θ_3.

Figure 5.3(a) shows the coverage and accuracy at varying value of the penalty threshold. As discussed earlier, the penalty counter is incremented by a value ξ if the present FDD is smaller than the preceding FDD. For smaller value of θ_3 the probability of isolation of fault-free nodes with transient faults in the observation state is more as the transient faults are appeared like correlated intermittent faults. As expected and shown in Figure 5.3(a), the average coverage is not affected by varying values of θ_3. As shown in Figure 5.3(b), the average latency of isolation increases with θ_3. This is because the number of test repetitions required to detect the presence of fault increases with θ_3. Since the proposed approach implements an adaptive penalty increment technique and a relatively high fault occurrence

rate are observed in an intermittent faulty node, the average detection latency grows less after $\theta_3 = 5$.

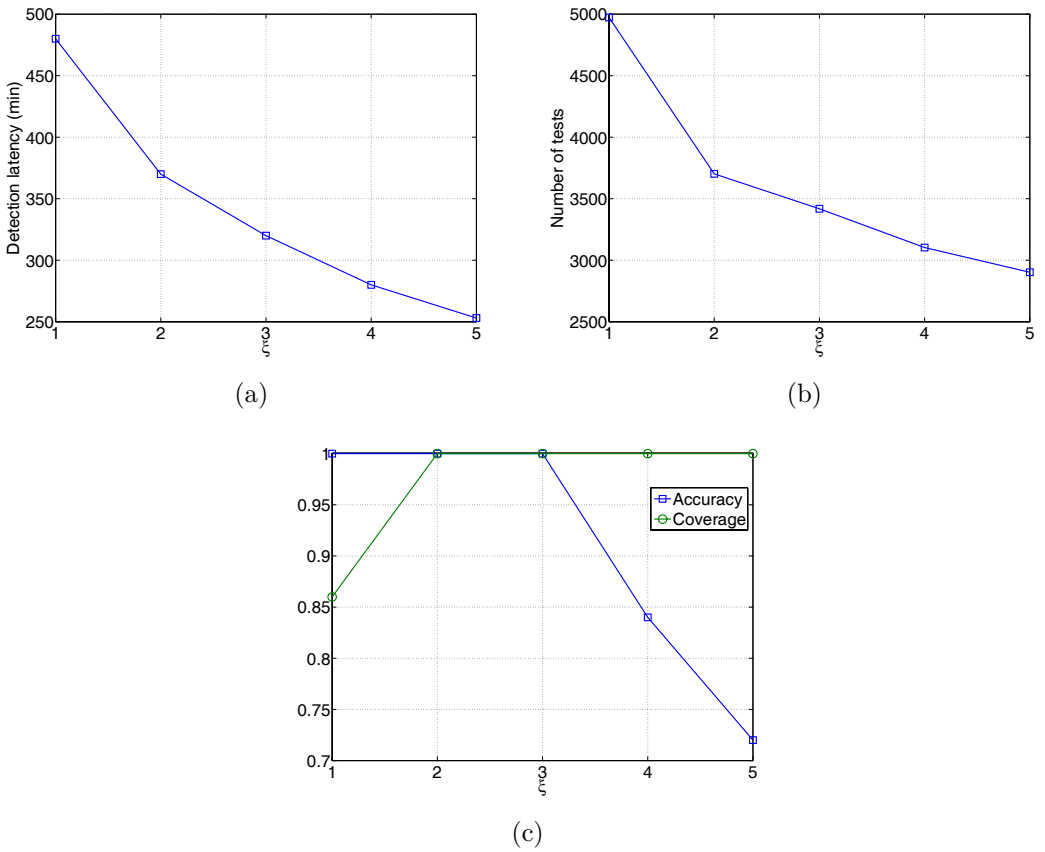


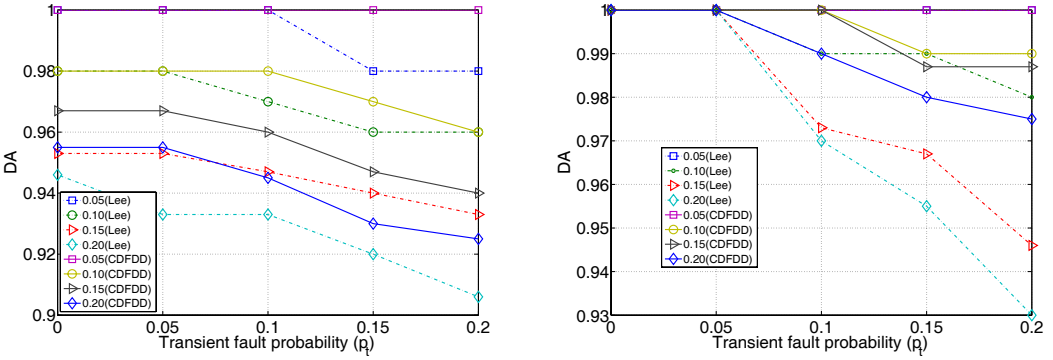
Figure 5.4: (a) Detection latency at varying value of ξ. (b) Average number of tests at varying value of ξ. (c) Accuracy and coverage at varying value of ξ.

Finally, we study the effect of ξ on the average detection latency, the average number of test repetitions, the average coverage, and average accuracy. Figure 5.4(a) illustrates the improvement of the detection latency with ξ. When ξ is greater than 2 the detection latency is lower than that of the circumstance when $\xi = 1$. Similarly Figure 5.4(b) illustrates the improvement of the number of test repetitions required to discriminate transient from intermittent faults. The effect of ξ on average accuracy and coverage is depicted in Figure 5.4(c). A trade-off is observed where both the average accuracy and coverage attain their highest value form $\xi = 2$ to $\xi = 3$. These results suggest the importance of ξ in discriminating transient from intermittent faults.

In summary, for WSNs, a setting of $T = 8600\ ms$, $\theta_2 = 10^4$, $\theta_3 = 5$ and $\xi = 2$ allow to discriminate most of the transient from intermittent faults.

5.4.2 Experiment 2: Robustness with regard to transient faults

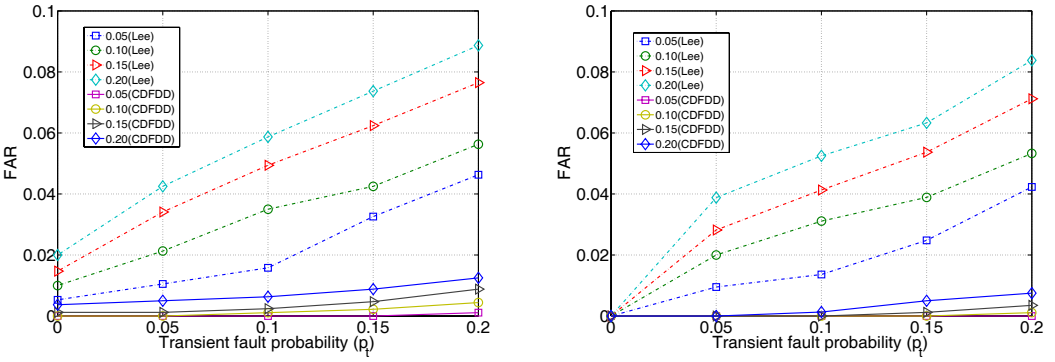
In this experiment, we estimate how well the proposed detection algorithm discriminate transient from intermittent or permanent faults. We compare the performance of the diagnosis algorithm with the sate-of-art diagnosis algorithm proposed by Lee *et al.* in [33]. Similar to [33], we redefine the FAR as follows. Let n_g, n_t and n_f represent the number of good nodes, the number of good nodes with a transient fault and the number of faulty nodes, respectively. Let n_{gf} is the number of nodes wrongly detected as faulty out of the n_g good nodes. Similarly, the number of fault-free nodes with transient faults identified as faulty is denoted by n_{tf}. The FAR is redefined as $\frac{n_{gf}+n_{tf}}{n_g+n_t}$. In this experiment, the impact of transient fault rates (p_t) on DA and FAR have been evaluated for $p = 0.05, 0.10, 0.15$, and 0.20.



(a) $d_a \approx 4$ (b) $d_a \approx 20$

Figure 5.5: DA in presence of transient faults: $P_{cerr} = 10^{-3}$.

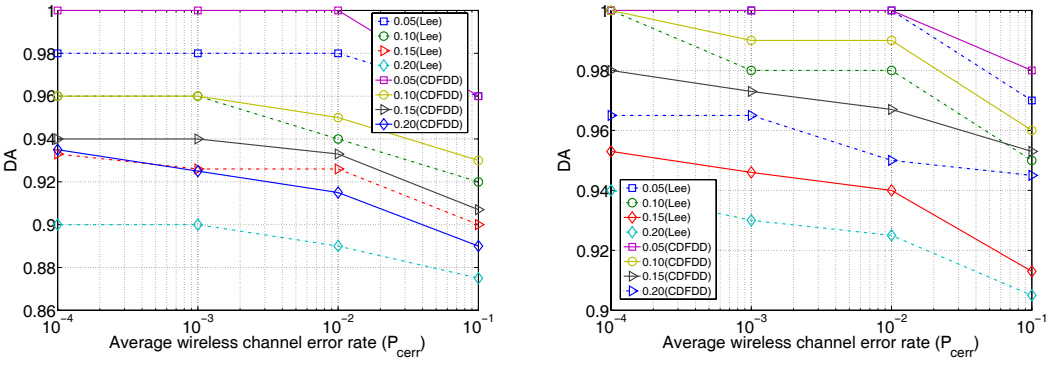
As expected and shown in Figure 5.5, the detection accuracy is less affected by varying rate of transient faults in the WSN. Similar to the proposed detection algorithm, the detection accuracy of the detection algorithm proposed by Lee *et al.* is less sensitive to change in p_t. However, as shown in Figure 5.6, the proposed detection algorithm outperforms Lee *et al.* approach from FAR perspectives. In the proposed approach, proper tuning of detection parameters ensures efficient



(a) $d_a \approx 4$ (b) $d_a \approx 20$

Figure 5.6: FAR in presence of transient faults: $P_{cerr} = 10^{-3}$.

discrimination of transient from intermittent faults. The fault-free nodes with transient faults are effectively reintegrated into the WSN which in turn keeps the FAR low. The two thresholds used in Lee *et al.* scheme are not adequate to discriminate transient from intermittent or permanent faults. Thus, their approach isolates a maximum number of fault-free nodes with transient faults.

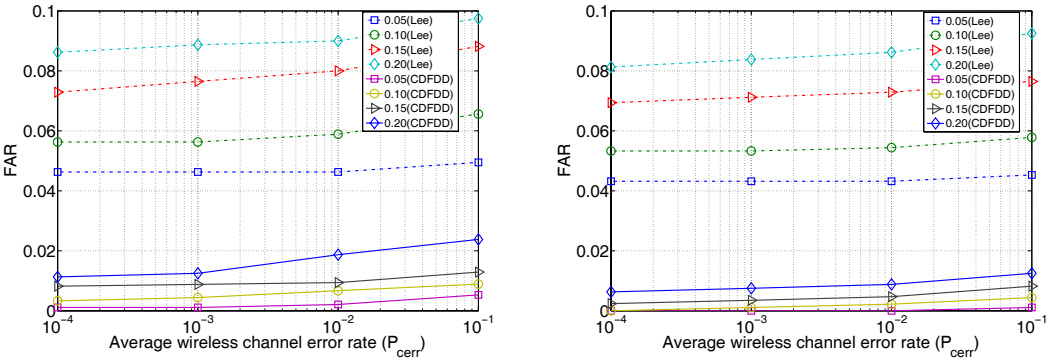


(a) $d_a \approx 4$ (b) $d_a \approx 20$

Figure 5.7: DA at varying value of P_{cerr}: $P_t = 0.2$.

5.4.3 Experiment 3: Robustness with regard to channel faults

In this experiment, the robustness of the detection algorithm to faults in the communication channel is analyzed by estimating DA and FAR for various channel error probabilities. In this experiment, we set $p_t = 0.2$. For simplicity in the simulation P_{good} is taken as 0 and P_{bad} is taken as 1. P_{BG} is fixed to 1/8 and P_{GB}



(a) $d_a \approx 4$ (b) $d_a \approx 20$

Figure 5.8: FAR at varying value of P_{cerr}: $P_t = 0.2$.

is varied to get different channel error probabilities P_{cerr}. The channel error rate is increased in steps from 10^{-4} to 10^{-1}. Faults in the communication channel might cause some fault-free nodes to fail in receiving the sensor measurements from its neighbors. This in turn decreases the effective neighbor size of a sensor node and might affect the local decision. However, the analytical and simulation study in Chapter 3 shows better performance even in sparse WSNs. Thus, as expected and shown in Figures 5.7 and 5.8, the detection algorithm effectively tolerates faults in the communication channel. It is observed that the detection scheme proposed in [33] effectively tolerates faults in the communication channel.

5.4.4 Experiment 4: Network lifetime

In this experiment, we evaluate the energy efficiency of the proposed detection algorithm and compare with Lee *et al.* approach. We consider an example network where all sensor nodes are assumed to be fault-free or fault-free with transient faults. In this simulation, the sensor nodes are assumed to have transient faults with probability 0.05, 0.10, 0.15 and 0.20 respectively. A node is considered dead if it has lost 99 percent of its initial energy. As expected and shown in Figure 5.9 the proposed diagnosis algorithm outperforms Lee's approach. This is because FAR of Lee *et al.* approach is worst affected by the increase in transient fault rates. Thus, their approach isolates fault-free nodes with transient faults. This in turn increases the workload of each node in the WSN, and the nodes depletes energy faster. In contrary, the proposed detection algorithm keeps FAR low and

is less sensitive to change in transient fault rates.

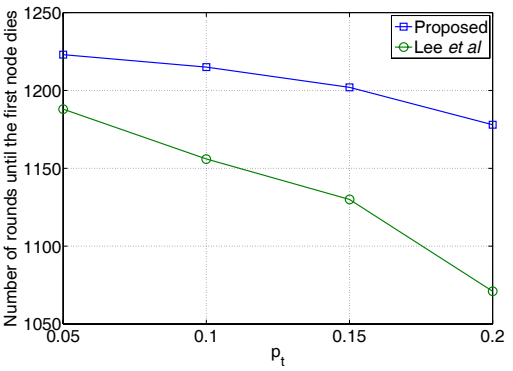


Figure 5.9: Network lifetime.

5.5 Summary

This chapter has presented a simple, distributed fault detection algorithm for WSNs where permanent, intermittent and transient faults have been considered. The detection parameters namely an inter-test interval, reward counter and penalty counter were tuned to effectively discriminate the persistence (permanent, intermittent and transient) of faults in WSNs. An adaptive penalty increment is suggested to reduce the detection latency. In general, it is observed that the performance of the proposed diagnosis algorithm CDFDD outperforms Lee's approach from false alarm rate perspective.

Chapter 6

Hard and Soft Fault Diagnosis in WSNs with Mobile Sensor Nodes

6.1 Introduction

In WSNs the individual sensor nodes are generally assumed to be static. However, some recent applications of WSNs (e.g., in medical care and disaster response) make use of mobile sensor nodes where different nodes often have different mobility patterns. Some nodes are highly mobile, while others are primarily stationary. This causes the network topology to change randomly since sensor nodes are free to move arbitrarily with different speeds. The ability of diagnosing faults decreases under this scenario, meaning that mobility significantly reduces the quality of the diagnosis returned by a diagnosis protocol [37]. This motivates to explore a mobility aware distributed diagnosis algorithm for WSNs. As an effective solution to the mobility problem, we propose a hierarchical architecture. The proposed mobility aware cluster-based distributed fault diagnosis (MCDFD) scheme works in conjunction with this hierarchical architecture.

Although the considerations of node mobility and the avoidance of frequent re-clustering enhance the cluster stability, the present mobility aware clustering algorithms [105–109] have not considered the hot spot problem and presence of faulty nodes in the WSN. Since multi-hop forwarding mode is adopted in inter-cluster communication, this many-to-one traffic pattern results in the hot spot problem. The reason is that the cluster heads closer to the base station need to relay heavier traffic and deplete energy faster than the farthest cluster heads

and die much faster than other nodes in the WSN. Thus, the area near the base station becomes a hot spot. To mitigate the hot spot problem in mobile scenarios, we extend the UCR protocol which is primarily designed for static WSNs. UCR protocol consists of two parts namely energy-efficient unequal clustering (EEUC) algorithm for topology management, and a greedy geographic and energy-aware routing protocol for inter-cluster communication. The proposed mobility aware unequal cluster-based Routing (MAUCR) protocol modifies EEUC and proposes mobility adaptive EEUC (MAEEUC) algorithm where cluster heads are selected based on local information namely the residual energy of neighboring nodes and expected neighbor time $E(T_N)$. The expected neighbor time is the expected duration during which two nodes remain in transmission range of each other. Neighbor time is proportional to nodes' relative velocity to its neighbor: a larger value means higher stability.

The proposed fault detection algorithm is executed just prior to the MAEECC. The nodes detected as faulty are excluded from the competition to become cluster head. The proposed detection algorithm follows CDFD algorithm but does not solely depend on readings of one-hop neighbors. In MCDFD algorithm, a test result set is obtained periodically at each node by executing the proposed test pattern. Based on these test results, faults are detected.

The remaining of the chapter is organized as follows. Section 6.2 presents the system model. Description, analysis and implementation of MCDFD are investigated in Section 6.3. Simulation results are presented in Section 6.4 and finally, summary is given in Section 6.5.

6.2 System Model

6.2.1 Notations

The list of the notations used in this chapter and their meanings are shown in Table 6.1.

Table 6.1: Notations.

n	Number of sensor nodes.
v_i	Sensor node.
$\{TR_i\}$	Test result set of v_i.
$E[T_N]$	Expected neighbor time.
D_{max}	Maximum distance.
τ	Pause time.
F_{state_i}	Fault state of v_i.
$N(v_i)^t$	One-hop neighbor set of v_i at time t.
N_{TR}	Number of one-hop neighbors report similar test result set $\{TR\}$.
E_i	Initial energy of v_i.
$E_{Tx}(t)$	The energy spent in sending number of bits at time t.
$E_{Rx}(t)$	The energy spent in receiving number of bits at time t.
$E_{relay}(v_i, v_j)$	The total energy cost of the path $v_i \rightarrow v_j \rightarrow$ base station.
d_a	Average node degree.
c	Constant coefficient.
w_i	Distance based weight.
T_{slot}	The duration of each TDMA time slot.
p	Fault probability.
p_{cerr}	The average bit error probability of the channel.
T_{out}	Timeout timer.
σ_2	A difference value which is function of velocity.
θ	Optimal threshold.
θ_4	Residual battery lifetime threshold.
$E[T_{avg}]$	Neighbor time threshold.
R_i	Competition range of v_i.
R_0	Maximum competition range.

6.2.2 Network, Fault, Channel, and Energy Model

This chapter follows the assumptions of Chapter 3 and 4 such as homogeneous sensor nodes, unique ID for each sensor node, variable transmission power level, symmetric links, synchronized networks, and imperfect test. However, the assumption on node mobility is relaxed and the nodes are allowed to move randomly. We consider the network model, channel model, and energy model the same as specified in Chapter 3. The WSN consists of n sensor nodes where all sensor nodes are arranged into non-overlapping clusters. Gilbert-Elliott channel model is considered [86, 87]. The assumptions made in this chapter are similar to Chapter 3. However, after deployment only the sink node is assumed to be static

and the sensor nodes are allowed to move randomly.

6.2.3 Mobility Model

To demonstrate the effectiveness of the proposed scheme, this work adopts the flexible node mobility model as suggested in [110], where a node alternates between the moving and the pausing phases. A node moves from its current location to a new location by randomly choosing a direction and speed in which it will travel. The new speed and direction are both chosen from $[u_{max}, u_{min}]$ and $[0, 2\pi]$ respectively. This mobility model allows a node to choose its travel distance, which is a random variable that is uniformly distributed in $[0, D_{max}]$. Upon arriving at the destination; the node pauses for an exponentially distributed random time τ_s before starting another movement.

6.2.4 Diagnostic Model

Each sensor node periodically produces the test result set $\{TR_i\}$ for a common test pattern and broadcast $\{TR_i\}$. Due to the shared nature of communication in wireless networks, a node v_i receives the result sets of its one-hop neighbors. Since TDMA-MAC protocol is used for intra-cluster communication, v_i will receive these sensor measurements at different times. A sensor performs a self-test on the received identical result sets from one-hop neighbors and a derived optimum threshold. Fault-free nodes fail the threshold test later been diagnosed as fault-free through the fault-free neighbor(s). The result set $\{TR_i\}$ is identical to $\{TR_j\}$ if $\{TR_i\} = \{TR_j\}$.

6.3 The Proposed MCDFD Algorithm

6.3.1 Description of the Algorithm

MCDFD algorithm consists of four parts; (i) mobility adaptive fault detection algorithm to generate diagnostic local views, (ii) energy-efficient mobility adaptive unequal clustering algorithm for topology management (MAEEUC), (iii) energy and mobility aware greedy geographic routing protocol for inter-cluster communication, and (iv) dissemination of diagnostic local views. The detection

algorithm generates the diagnostic local view of one-hop neighbors. Nodes detected as faulty do not participate in clustering. The clustering algorithm is a self-organized competition-based algorithm, where cluster heads are selected based on local information (i.e., the residual energy of neighboring nodes and the neighbor time). Similar to UCR protocol, the node's competition range decreases as its distance to the base station decreases. Clusters closer to the base station are expected to have smaller cluster sizes. Cluster heads closer to the base station consumes less energy during the intra-cluster data processing, and thereby preserves some more energy for the inter-cluster communication. The greedy geographic routing protocol constructs a stable cluster head backbone by considering both neighbor time and residual energy. The diagnostic local views are disseminated using the cluster head backbone.

SensorID	Status	Position (x,y,z)	Speed	Direction (d_x, d_y, d_z)	Residual energy	Diagnosis information

Figure 6.1: Frame format for *Hello message*.

Fault Detection Algorithm

In this approach, each node in the WSN broadcasts a *Hello message* periodically to acquire the one-hop neighbor set and the diagnosis local view. We propose a frame format for *Hello message* as shown in Figure 6.1. The *Hello message* carries sender's unique identification number, position information, residual energy and the diagnostic information. Information regarding position, speed and direction is obtained by the GPS system fabricated in each node. The status field is of eight bit length where the first two bits represent the status (00-cluster head, 01-cluster member and 11-under clustering processes), and the last six bits represent the number of nodes affiliated if the status is cluster head. Similar to CDFD algorithm, it follows neighbor coordination approach. However, unlike CDFD algorithm, it uses a predefined test pattern to test all the functional blocks of a sensor node. In CDFD algorithm, the diagnostic local view is obtained by comparing the sensor readings of one-hop neighbors. In contrary, MCDFD algorithm uses the test result sets received from one-hop neighbors to construct the local view. This local view

is shown to be less affected by the node mobility. A detailed description of the robustness with regard to node mobility is left to the Section 6.3.2. As shown in Algorithm 5, a node v_i executes the test pattern and embeds the obtained result set $\{TR_i\}$ to the *Hello message* and then broadcast it. Upon receiving the *Hello messages* of one-hop neighbors, v_i compares its own test result set with that of its one-hop neighbors. Next, it forms a set ($\{N_{TR}\} \subset \{N(v_i)^t\}$) of nodes with the similar test result set $\{TR\}$ where, $\{N(v_i)^t\}$ is the neighbor set of v_i at time t. If $\{TR_i\}$ disagrees with $\{TR\}$ and the cardinality of set $\{N_{TR}\}$ is greater than a threshold (θ) then v_i makes a decision to disregard its reading and is marked as possibly soft faulty. The optimal value for θ is $0.5(N-1)$ where N is the number of nodes from which v_i has received the test result sets. Next, each node broadcasts their decision. The decision contains the node ID, one-bit decision variable (1 if possibly soft faulty and 0 if fault-free), and its own result set. In the second round of test, a node v_i identified as possibly soft-faulty first checks for a node v_k identified as fault-free, i.e., v_i believes that v_k is fault-free. If such v_k exists and $\{TR_i\} = \{TR_k\}$ then v_i is detected as fault-free. Otherwise, faulty. This correct decision is subsequently broadcasted. At this stage, each node has a local view that reflects its view about the state of its one-hop neighbors.

Node ID	Location			Speed	Direction			Test results		
	x	y	z		dx	dy	dz	R_1	...	R_n
1										
2										
3										
....										

Figure 6.2: Neighbor table.

The detection algorithm uses a timeout mechanism to detect hard faulty nodes. The node v_i declares node $v_j \in N(v_i)$ as possibly hard faulty (initial detection status), if v_i does not receive the *Hello message* of v_j before T_{out}. v_j cannot report to v_i due to at least one of the following reasons: 1) the communication subsystem of v_j may be faulty, 2) v_j may be damaged, 3) battery of v_j may be drained out, 4) v_j may be no more in the transmission range of v_i. For the reason 4, v_i will mark v_j as hard faulty, which may not be correct. Final decision regarding v_j (hard faulty or fault-free) is taken by the cluster heads as discussed Chapter 3.

Algorithm 5 MCDFD

1: Embed the result set $\{TR_i\}$ to *Hello message* and broadcast the *Hello message*.
2: Set timer T_{out}.
3: Construct the neighbor table $N(v_i)^t$ using the received *Hello messages* as shown in Figure 6.2.
4: Determine $\{N_{TR}\}$, the set of one-hop neighbors report identical result set $\{TR\}$.
5: **if** $T_{out} = true$ **then**
6: Declare unreported nodes as possibly hard faulty.
7: **end if**
8: **if** ($\{TR_i\} = \{TR\}$ and $|\{N_{TR}\}| < \theta$) or ($\{TR_i\} \neq \{TR\}$ and $|\{N_{TR}\}| \geq \theta$) **then**
9: $F_{state_i} \leftarrow$ Possibly soft faulty.
10: **else**
11: $F_{state_i} \leftarrow$ Fault-free.
12: **end if**
13: Broadcast the F_{state_i}, TR_i, and own node ID.
14: Node identified as possibly soft faulty checks for a node $v_k \in N(v_i)^t$ such that F_{state_k} is fault-free. If such v_k exists, $\{TR_i\} = \{TR_k\}$ and $v_k \in \{N_{TR}\}$ then set F_{state_i} as fault-free or else faulty and broadcast F_{state_i}.
15: Perform clustering of the network.
16: Construct cluster level local view.
17: Construct the cluster head backbone.
18: Disseminate the cluster level local views using cluster head backbone.

The Clustering Algorithm MAEECU

In this section, the clustering algorithm intended to achieve the longest cluster lifetime is proposed. Before proceeding with the presentation of the various steps of the algorithm, the major feature of the algorithm is presented: 1) it produces clusters of unequal sizes to address the hot spot problem where clusters closer to the base station have smaller cluster sizes, 2) a new cluster head does not force an existing valid cluster to reconstruct, 3) the cluster head lifetime lasts until all of its affiliated cluster members have moved away and/or cluster head is detected as faulty and/or the remaining battery lifetime (RBL) drops below certain threshold, 4) a non-cluster-head enters in to re-clustering phase if it has moved away from transmission range of its affiliating cluster head or its affiliating cluster head is detected as faulty, 5) it attempts to maximize the cluster lifetime at cluster construction by choosing the most stable nodes in mobility perspective

to become the cluster heads and 6) the nodes near the base station (i.e., distance to the base station is smaller than a distance threshold) send the forwarding data directly to the base station. The algorithm for each sensor node at the cluster head selecting stage is given in Algorithm 6.

Algorithm 6 MAEECU Algorithm

1: **if** $v_i.status$ =head and $v_i.member$ =NULL
   or $v_i.status$ =head and $v_i.RBL \leq \theta_4$
   or $v_i.status$ =member and $v_i.head$ =NULL
   or $v_i.status$ =member and $v_i.head$ =faulty **then**
2: For all $v_j \in N(v_i)$
3: $V_{CH} \leftarrow \{v_j | v_j.status \neq \text{member},\ v_j \notin \text{fault set},\ D(v_i, v_j) < max(R_i, R_j)\} \cup \{v_i\}$
4: $V'_{CH} \leftarrow \{v_j | v_j \in V_{CH},\ E[T_N]_{ij} \geq E[T_{avg}]\}$
5: $E \leftarrow \{v_j.energy | v_j \in V'_{CH}\}$
6: $v \leftarrow \{v_j | v_j \in V'_{CH},\ v_j.energy = max\langle E \rangle\}$
7: **if** $v_i = v$ **then**
8: $v_i.status \leftarrow$ head
9: **else**
10: $v_i.status \leftarrow$ member
11: $v_i.head \leftarrow v$
12: **end if**
13: **end if**

In EECU algorithm, several tentative cluster heads are randomly selected to compete for final cluster heads. In contrary, in MAEECU algorithm all the sensor nodes are tentative cluster heads and compete for final cluster heads. Each tentative cluster head v_i has a competition range R_i. Like EECU algorithm, different competition ranges are used to produce clusters of unequal sizes. Only one final cluster head is allowed in each competition range. To mitigate the hot spot problem, clusters closer to the base station should be smaller cluster sizes, and more clusters need to be produced closer to the base station. Alternatively, competition range of the tentative cluster heads should decrease with its distance to the base station. Like EECU algorithm, we select R_0 as the predefined maximum competition range. The minimum competition range is set to $(1-c)R_0$, where c is a constant coefficient between 0 and 1. Sensor node v_i's competition range R_i can be expressed as a linear function of its distance to the base station

[90]:

$$R_i = \left(1 - c\frac{D_{max} - D(v_i, BS)}{D_{max} - D_{min}}\right) R_0 \tag{6.1}$$

where D_{max} and D_{min} denote the maximum and minimum distance between network boundaries and the base station. For instance, D_{max} for a network shown in Figure 6.3 is $\sqrt{(l + D_{min})^2 + (b/2)^2}$. $D(v_i, BS)$ denotes the distance between v_i and the base station.

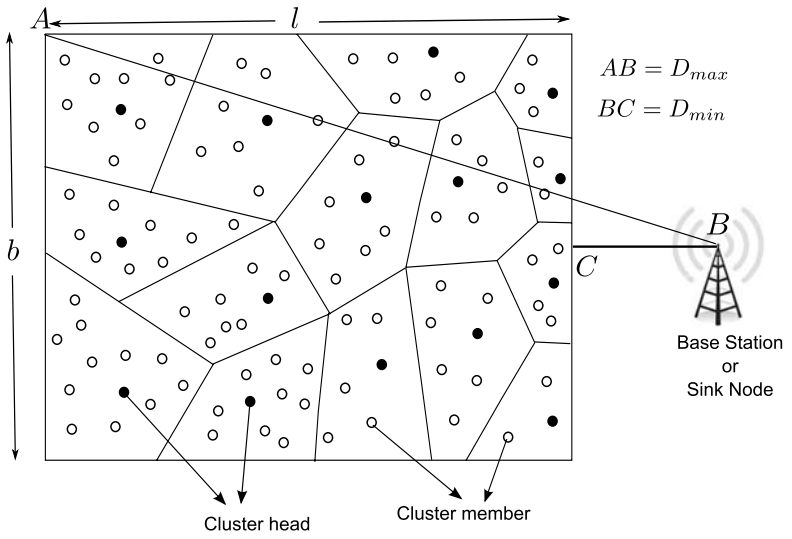


Figure 6.3: An overview of D_{max} and D_{min}.

In this approach, every node periodically checks if it needs re-clustering. A cluster head starts re-clustering if it does not receive *Hello messages* from any of its member nodes and/or the remaining battery lifetime (RBL) dropped below θ_4. A cluster member starts re-clustering if it does not receive *Hello message* from its affiliating cluster head or the affiliating cluster head is detected as faulty. The remaining battery lifetime of the cluster head is the normalized remaining battery energy of the cluster head at moment t_m. In the processing of the WSN, the energy is consumed when the sensor receives or sends a message. Thus, the normalized remaining battery life time can be calculated as

$$RBL(v_i) = \frac{E_i - \sum_{t=1}^{t_m} E_{Tx}(t) + E_{Rx}(t)}{E_i} \tag{6.2}$$

where $E_{Tx}(t)$ and $E_{Rx}(t)$ are propositional to the number of bits it processes and forwards at time t. E_i is the initial energy of node v_i. The value for θ_4 is the

remaining life time (RNL) of the WSN which is dynamically calculated at each round. The remaining lifetime of the WSN can be calculated as [111]

$$RNL = \sum_{i=1}^{n} w_i RE(v_i) \tag{6.3}$$

where w_i is the weight associated with each node based on its distance from the base station and is given by

$$w_i = c\frac{1}{D(v_i, BS)^2} \tag{6.4}$$

Before a node attempts to join a cluster, it must gather a list of its tentative cluster head competitors to determine the most suitable node to cluster with or to act as cluster head. In this approach, the set of tentative cluster heads is determined by using the received *Hello messages*. Node v_i adds itself to this set from which the cluster head will be selected. Tentative head v_j is an *adjacent* node of v_i if v_j is in v_i's competition diameter or v_i is in v_j's competition diameter. Each sensor node constructs a set of eligible *adjacent* tentative nodes ($\{V_{CH}\}$) in line 3 of Algorithm 6. In this process; it excludes any node that has become a cluster member or detected as faulty by the detection algorithm, as it cannot take the cluster head status. In line 4, the algorithm subsequently removes the tentative cluster heads those do not satisfy the expected neighbor time threshold and constructs a set $\{V_{CH}\}' \subseteq \{V_{CH}\}$. The neighbor time threshold $E[T_{avg}]$ is given by

$$E[T_{avg}] = \frac{\sum_{|V_{CH}|} E[T_N]_{ij}}{|V_{CH}|} \tag{6.5}$$

In this approach, a decision to elect the sensor node v_i as cluster head depends on the nodes in $v_i.V_{CH}$ only, i.e., the algorithm is localized. The node $v_j \in \{V_{CH}\}'$ will be selected as optimal cluster head, if it has highest residual energy among the nodes in $\{V_{CH}\}'$. Thus, the tradeoff between selecting a high-stability node and selecting a high-energy node is addressed. If v_i satisfies the aforementioned conditions, then it becomes a cluster head. Otherwise, v_i registers its membership with v_j. At this point, four possibilities may arise: 1) if v_j is an existing cluster head, v_i is registered with v_j, 2) if v_j settles as cluster member, v_j rejects registration of v_i and v_i repeats the clustering algorithm excluding v_j from $\{V_{CH}\}'$

and 3) if v_j is under clustering processes and has not yet made its decision, v_i waits until v_j has made its decision.

Inter-cluster multi-hop routing

Before selecting the next hop node, each cluster head broadcasts a short beacon message across the WSN at a fixed power which consists of its node ID, residual energy, position information, speed, and direction. Distance from the base station and the distance between the cluster heads are calculated from the position information obtained from the GPS system. Similar to UCR protocol we use a threshold TD_MAX in the multi-hop routing protocol. If a node's distance to the base station is smaller than TD_MAX, it transmits its data to the base station directly. Otherwise, it finds a relay node which can forward its data to the base station. The value of TD_MAX is always smaller than the actual maximum transmission range of a sensor node. In this approach, the multi-hop forwarding algorithm considers nodes in the forward direction, i.e., closer to the base station only. Cluster head v_i constructs a neighboring set of cluster heads $\{R_{CH}\}$ from which the most eligible one-hop relay cluster head is selected. This is defined as

$$v_i.R_{CH} = \{v_j | D(v_i, v_j) \leq XR_i, D(v_j, BS) < D(v_i, BS), E[T_N]_{ij} > \theta_5\} \quad (6.6)$$

where X is the minimum integer that lets $v_i.R_{CH}$ to contain at least one neighbor cluster head. For cluster heads, if such an X does not exist, i.e., $v_i.R_{CH} = NULL$, v_i will send its own data together with forwarding data directly to the base station.

The network life time can be extended by either choosing the relay node with more residual energy or by decreasing the energy cost per packet. Similar to UCR protocol, we propose a greedy geographic forwarding algorithm that aims to minimize the energy cost per packet. For simplicity, similar to UCR protocol we assume a free space propagation channel model. Suppose v_i chooses v_j as its relay node. Since a localized algorithm is desirable, presence of a virtual hop between v_j and the base station is assumed. To deliver a l-length packet to the base station, the total energy cost of the path $v_i \rightarrow v_j \rightarrow BS$ is [90]

$$E_{relay}(v_i, v_j) = D^2(v_i, v_j) + D^2(v_j, BS) \quad (6.7)$$

In this approach, v_i first chooses k eligible neighbor nodes from $v_i.R_{CH}$, denoted as the set $V_{eligible}$ [90]:

$$v_i.V_{eligible} = \{v_j | v_j \in v_i.R_{CH}, E_{relay}(v_i, v_j) \text{is the k smallest}\}. \quad (6.8)$$

To reduce inefficiencies of energy consumption, to increase the inter-cluster path life time a tradeoff should be made between residual energy, link cost E_{relay}, and expected neighbor time. In our approach, v_i first calculates the average of residual energies (RE_{avg}) of all $v_j \in v_i.V_{eligible}$. It chooses as its relay node the neighbor in $v_i.V_{eligible}$ that has the biggest neighbor time and has residual energy greater than RE_{avg}.

Dissemination of Local Diagnostics

The local diagnostic views are disseminated using the spanning tree of cluster heads constructed by the MAUCR protocol. The leaf cluster heads start this dissemination by appending its cluster level local diagnostic view to its data packet. A cluster head receives data packets from tree descendants first compares cluster level local diagnostics of their descendants and takes a decision about hard faults. Similar to CDFD algorithm the final decision regarding a node detected as hard faulty is based on a consensus made at each level. At the end of local dissemination, the base station generates the global view and broadcasts it. This ensures that each fault-free node correctly diagnose the state of all the sensor nodes in WSN.

Implementation of MCDFD Algorithm

In this section, we discuss the design details for practical deployment of MCDFD algorithm. As shown in Figure 6.4, MCDFD algorithm includes six time triggers: (1) neighborhood tracking and fault detection (collecting local diagnostic views) trigger (T1), (2) cluster head selection trigger (T2), (3) cluster set-up trigger (T3), (4) intra-cluster communication for routine data triggers (T4), (5) inter-cluster communication and dissemination of local diagnostics trigger (T5), and (6) global dissemination and network synchronization triggers (T6).

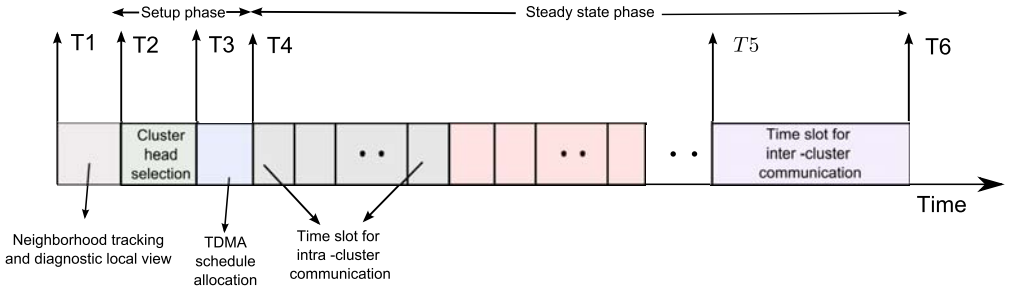


Figure 6.4: Time line showing MAUCR and MCDFD operation.

First, we describe the MAC mechanism together with the duty cycles schedule in various phases of the MCDFD. T1 triggers the neighborhood tracking and fault detection. Each node sends *Hello message* to communicate the state of its mobility to its neighbors in order to assist them in tracking it and make more accurate forwarding decisions. The *Hello message* carries sender's unique identification number, position information, residual energy and the result set. The *Hello messages* are exchanged using the carrier-sense multiple access (CSMA) MAC protocol. Upon receiving the *Hello messages* a node generates its neighbor table as shown in Figure 6.2. Each node exchange their decisions regarding its fault state using the CSMA MAC protocol. Nodes detected as possibly soft faulty takes the final decision by considering the test result of a one-hop neighbor node detected as fault-free. This updated decision is exchanged using the CSMA MAC protocol.

A node checks whether it requires re-clustering, if so, at time $T2$ the clustering process is triggered. After v_i has determined its clustering status, if its status is a cluster member, at $T3$ it sends the join request using CSMA to the cluster head it has decided to join and obtains its TDMA slot. If its status is a cluster head, it obtains the direct-sequence spread spectrum (DSSS) code from the sink node. In this approach, to reduce inter-cluster interference, a transmitter-based code assignment scheme is used. A node communicates with its cluster head and neighbors inside the cluster by using the DSSS. Each cluster is assigned a unique spreading code, which is used by all nodes in the cluster. Spreading codes are assigned to cluster heads on a first-in, first-served basis, starting with the first cluster head to announce its position, followed by subsequent cluster heads. $T4$

triggers intra-cluster data communication. Cluster members turn off their radio at all times except during their transmit time. A cluster head turns off its radio once the cluster's TDMA time slots run out. At $T5$ all cluster heads wake up and transmit control messages and data packets using CSMA.

In each phase, an appropriate time interval should be chosen to let MCDFD algorithm and MAUCR algorithm run correctly. The time interval depends on the network size and wireless channel quality. The waiting time between $T1$ and $T2$ depends on T_{out}, where T_{out} is an upper bound to the time needed to propagate a message and process the received message. The time duration between $T2$ and $T3$ depends on the clustering algorithm which needs several message exchange steps to finish. This can be explained as follows. If v_i satisfies the condition of line 6 of Algorithm 6, its status is finalized in one round of executing this algorithm. If v_i selects v_j, and v_j is determined to be a cluster head, its status is finalized in one round of executing this algorithm. If v_i selects v_j, and v_j is determined to be a cluster member, v_i repeats the algorithm until it finds $v_j \in v_j.R_{CH}$ determined to be cluster head. In the worst case, it ends up with itself and takes $|v_j.R_{CH}|$ rounds. Cluster join message and TDMA schedules are exchanged using CSMA between $T3$ and $T4$. During $T4$ to $T5$ nodes send the routine data in their time slots. At T5, all cluster heads wake up, and the inter-cluster communication is triggered. Cluster heads transmit control messages and data packets using CSMA. Similar to CDFD algorithm, the local diagnostics and the routine data are aggregated through the spanning tree up to the sink. Thus, at T6, the sink has the global fault state view of the WSN. Synchronization is important for the operation of MAUCR and MCDFD algorithms. This work assumes that all sensor nodes are synchronized and start MCDFD algorithm phases at the same time. This could be achieved by having the sink periodically broadcast synchronization pulses. In this work at T6, the sink node broadcasts the synchronization message along with the global view such that all nodes in the WSN will receive this message.

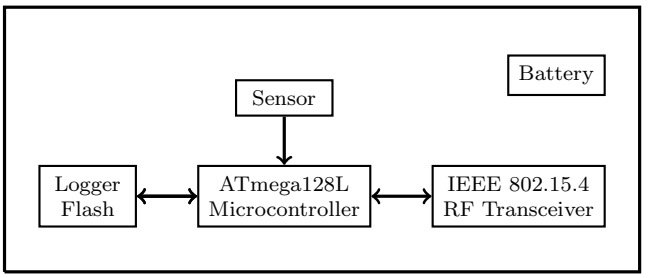


Figure 6.5: MPR2400 Block Diagram.

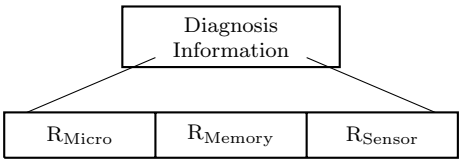


Figure 6.6: Information contents of diagnosis field of *Hello message*.

6.3.2 Analysis of the Algorithm

The Test Pattern

We describe the selection of the test pattern considering MICAZ mote with a temperature sensor. However, the same description can be extended for other motes and sensors. The functional block diagram of MPR2400 [112] is shown in Figure 6.5. The test pattern will be executed on different functional blocks of a sensor node to obtain the test result set. A node is hard faulty when the blocks namely IEEE 802.15.4 RF transceiver and battery are not functioning and thus do not require any explicit test. To test the sensing device, two successive readings are taken in a small interval Δt, i.e., at t and $t + \Delta t$. The test result for a sensing device is defined as

$$R_{Sensor} = \begin{cases} 0 & \text{if } |Temp_t - Temp_{t+\Delta t}| < \sigma_2 \\ 1 & \text{otherwise} \end{cases} \tag{6.9}$$

where σ_2 is function of node velocity v, i.e., $\sigma_2 = f(v)$.

To test the processing and memory block, micro-controller fetches the predefined data stored in the memory and then manipulates the data. The predefined data and the kind of manipulating operations performed on the data are the same for all the nodes, which ensure a uniform test pattern. The fetched data R_{Memory}, the manipulated data R_{Micro} and R_{Sensor} are embedded in the diagnosis

information field of the*Hello message* (see Figure 6.6).

Expected Neighbor Time

The performance of the clustering algorithm is mostly affected by the expected neighbor time $E[T_N]$, which is the expected time period within which any neighbor node v_j stays within the transmission range of a reference node v_i. The expected neighbor time can be calculated by using the latest mobility information (speed, direction, position) of the nodes. The expected neighbor time can be calculated with aid of Figure 6.7(a) and 6.7(b). As shown in Figure 6.7(a), node v_i moves with a random velocity U_1 and node v_j moves with a random velocity U_2. Figure 6.7(b) shows the relative velocity U_R for direction difference ϕ.

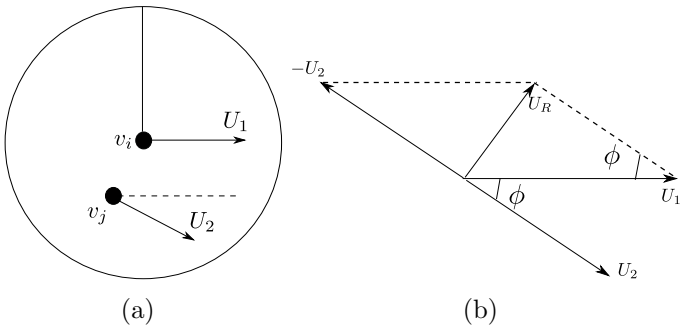


Figure 6.7: (a) Analytical model for neighborhood interval. (b) Relative velocity V_R of nodes S and U.

The magnitude of U_R is given by

$$U_R = \sqrt{u_1^2 + u_2^2 - 2v_1u_2cos\phi} \tag{6.10}$$

where u_1 and u_2 are the magnitude of U_1 and U_2 respectively. The mean value of U_R is computed as

$$E[U_R] = \frac{1}{\pi(u_{max} - u_{min})^2} \int_{u_{min}}^{u_{max}} \int_{u_{min}}^{u_{max}} \int_0^{\pi} U_R d\phi du_1 du_2 \tag{6.11}$$

There is no closed-form solution for the integral in (6.11). We use the numerical approximation where the integration range of each variable is divided into h fragments and summing the integration results over all the fragments. Solution

for (6.11) can be computed as

$$\begin{aligned} E[U_R] &\approx \frac{1}{\pi(u_{max}-u_{min})^2} \sum_{\phi,u_1,u_2} U_R \left(\frac{\pi}{h}\right) \left(\frac{u_{max}-u_{min}}{h}\right)^2 \\ &= \frac{1}{h^3} \sum_{\phi,u_1,u_2} U_R \end{aligned} \tag{6.12}$$

The value of h should be determined such that the error introduced in the approximation is very small. In order to choose a proper value of h, we calculate the approximation for h with step size of h where $h = 1, 2, 3, \cdots$. We suggest using h for which the difference between two consecutive approximations with step size h tends to zero, i.e., the difference is negligible small. In this approach, we have used $h = 2000$. Assuming uniform node distribution and the direction of movement of each node distributed uniformly over $[0, 2\pi]$ from [113] the mean value of T_N is computed as

$$E[T_N] = \frac{\pi A}{E[U_R]L} \tag{6.13}$$

where A is the area of the transmission range and L is the perimeter of this area.

Determination of θ_5

In the worst case, the clustering algorithm ends with n number of cluster heads in a n-node WSN. In the worst case, the height of the cluster head backbone or the spanning tree constructed is $n-1$. Thus, the worst-case time taken to reach a message from the leaf cluster head to the sink node is $(n-1)(T_p+\gamma)$. Where T_p is an upper bound to the time needed to propagate a message and γ is the processing time at each cluster head. Thus, the value for θ_5 is $(n-1)(T_p+\gamma)$.

Effect of Mobility

This section elaborates the effect of mobility on the performance of the proposed work. The performance in detecting both hard and soft faults is not affected by the node mobility. This can be well-explained using the example network shown in Figure 6.8. Let a fault-free node-23 is at location in cluster-B at time t_0, the time a node initiates its diagnosis round by sending a *Hello message*. The node migrates to a new location in the cluster-A at time t_1. Node-23 receives *Hello messages* from the neighbors at new location, and a decision about soft faults is taken

by comparing its own test result set with the test result sets of new neighbors. Similarly, let node-16 be at a position as shown in Figure 6.8 during diagnosis round i and has moved to a new location in the cluster-D in diagnosis round $i+1$. Let the node become hard faulty between these two successive diagnosis rounds.

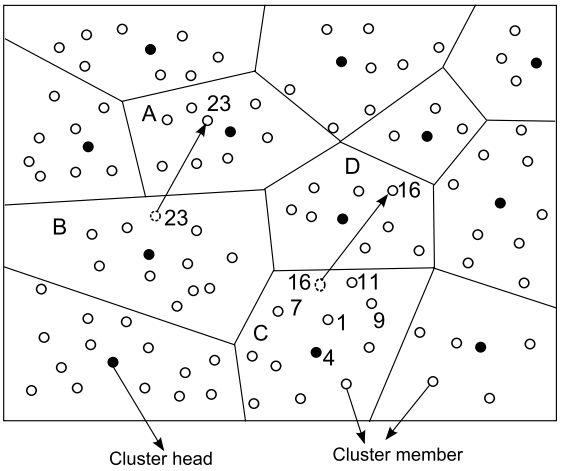


Figure 6.8: Example network to demonstrate effect of node mobility.

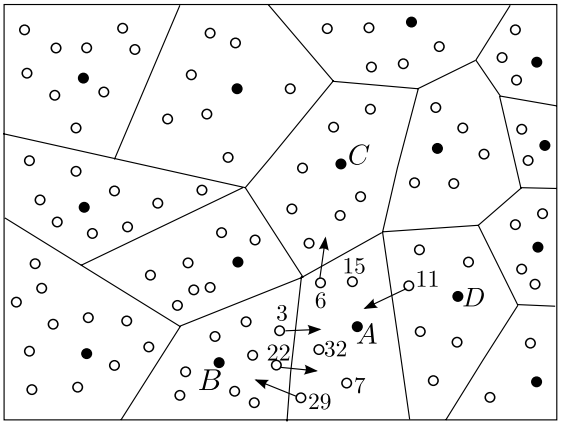
In the $(i+1)^{th}$ diagnosis round the nodes 1, 4, 9, 7 and 11 will not receive the *Hello message* from node-16 as it has migrated away from their transmission range. Thus, node-16 is marked as hard faulty by these nodes. In addition, the nodes neighbor to node-16 in the new location in the cluster-D are unaware of it as it became hard faulty during its transition and cannot communicate. Therefore, the presence of hard faults is correctly detected. In summary, the proposed scheme is not affected by the node mobility in building local diagnostic views.

Obtaining a correct set of faults, i.e., the global view is negligibly affected by node mobility as dissemination of local views is carried out by a spanning tree of relatively stable cluster heads. In summary, we can claim that the proposed work is robust from mobility perspective.

Adaptive TDMA Schedule Creation

Upon receiving join request from nodes the cluster head creates a number of TDMA time slots based upon the number of nodes. The cluster head first calculates the expected neighbor time and allocates the time slot in an ascending order. A member node with lowest expected neighbor time with the cluster head

is assigned the first-time slot, and the member node with highest neighbor time is assigned the last time slot within the TDMA frame. This enables a node with lowest neighbor time with the cluster head to send data packets successfully before detached from the cluster head. In this approach, at $T3$ a cluster head first checks for member nodes those have left it and sensor nodes from which it has received the join request messages. It first allocates vacant time slots, if present, to the nodes, those have sent join request messages. If such slots are not available, it appends their time slots based on neighbor time and communicate them. It then sorts the list in ascending order and informs the affiliated nodes regarding their new time slots during data request. The affiliated nodes follow this new TDMA schedule in the consecutive data rounds.



(a) Migration of mobile nodes.

Round- i	29	7	15	32	6	
Round - i+1	11	7	15	32	3	22
Round - i+2	7	11	15	3	22	32

(b) TDMA Scheduling.

Figure 6.9: Example of adaptive TDMA schedule creation.

The adaptive schedule creation can be well explained through an example shown in Figure 6.9(a). As shown in the Figure 6.9(a), in round-i, nodes with IDs 29, 7, 15, 32 and 6 are affiliated with the cluster head "A" and the time slots are allotted based on expected neighbor time of these nodes with cluster head. At

round-$i+1$ nodes with IDs 29 and 6 have left the cluster and nodes with IDs 3, 11 and 22 have sent their joining request to cluster head "A". Let's say that the expected neighbor times of these newly joined nodes is in the ascending order 11, 3 and 22. As shown in Figure 6.9(b), the cluster head assigns the empty time slots to nodes 11 and 3 and appends node 22. These allotted time slots are informed to these newly joined nodes. Next, the cluster head sort the new list in ascending order and inform the affiliated nodes about the new assigned slots when it sends the data request message following old schedule. The affiliated nodes follow the new schedule in round-$i+2$. This new schedule will be followed in the subsequent rounds by the cluster "A" until there is no change in its structure.

6.4 Simulation Experiments

The performance of the proposed scheme through simulations is presented in this section. This work uses Castalia-2.3b [52], a state-of-art WSN simulator based on the OMNET++ [53] platform. For the simulation purpose, a communication scenario has been generated with simulation parameters as summarized in Table 6.2. To represent different mobility scenarios, we specify five mobility patterns by tuning the node moving speed and pause time (Table 6.2). The Hello interval is set to 100 *sec*.

Table 6.2: Mobility Pattern.

Pattern	τ (min.)	$[v_{min}, v_{max}]$ (m/sec.)
MP1	10	[1,2]
MP2	8	[1,4]
MP3	6	[1,6]
MP4	4	[1,8]
MP5	2	[1,10]

In this simulation we consider the following performance parameters—

- *Mean cluster lifetime.* It is the time duration before which all the cluster members leave the cluster head or the residual energy falls below the threshold.

- *Mean inter-cluster lifetime.* It is the time duration before which link between

cluster heads break.

- *Packet delivery ratio.* It is the ratio between total received packets to the total generated packets.

In addition to these performance parameters, this chapter also considers the parameters discussed in earlier chapters like DA, FAR, network lifetime, message complexity and detection latency.

6.4.1 Experiment 1: Parameter Tuning

There are several parameters in MAUCR protocol, namely R_0, c, TD_MAX, and k. In this experiment, we tune these parameters with regard to the network lifetime, mean cluster lifetime, mean inter-cluster link lifetime, and packet delivery ratio. In this experiment, we have deployed 100 faulty nodes ($p = 0.1$) randomly. All the simulations are conducted considering the mobility pattern MP5. For better analysis, we consider only soft faults. As suggested in [90], we set k to 2.

As discussed in section 6.3.1 and shown in Figure 6.10, for a fixed value of c, the number of clusters decreases for an increase in R_0 and for a fixed value of R_0, the number of cluster increases with c. The reason is that the competition range (R_i) decreases either by increasing c while keeping R_0 constant or by decreasing R_0 while keeping c constant.

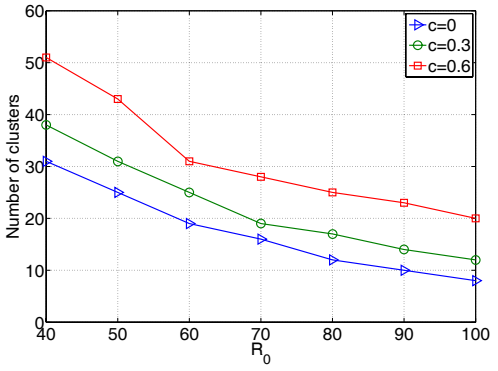


Figure 6.10: The number of clusters.

Figure 6.11(a), depicts the network lifetime for different settings of R_0 and c. It is observed that there is a tradeoff between R_0, c, and the network lifetime.

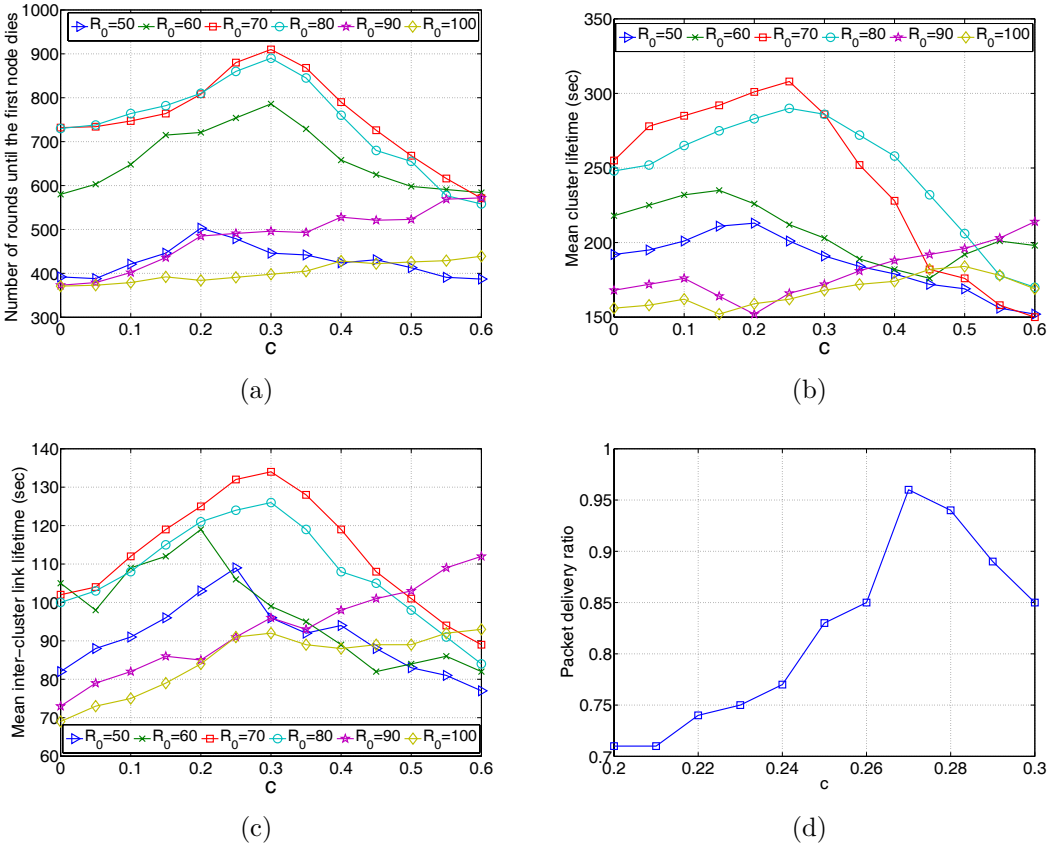


Figure 6.11: Parameter tuning: (a) The network lifetime. (b) Mean cluster lifetime. (c) Mean inter-cluster link lifetime. (d) Packet delivery ratio.

The mean cluster lifetime for different settings of R_0 and c is shown in Figure 6.11(b) where a tradeoff between R_0, c, and the mean cluster lifetime is observed. Similarly, Figure 6.11(c) explains the tradeoff between R_0, c, and the mean inter-cluster link lifetime. It is observed that, R_0 is the dominant factor that impacts this three lifetime metrics. The reason is that the number of clusters in a given network size and mobility pattern is mainly determined by R_0. This is because the competition range increases with R_0 for constant c. This in turn increases the number of affiliated cluster members under a cluster head and thus decreases the probability that a cluster head will be detached from all its members. However, the energy overhead of the cluster head increases with affiliated members and the cluster head depletes energy faster, and will need faster re-clustering. In addition for large value of R_0, the distance between two cluster heads is more. Thus, the probability of communication link failure is more since a small movement

of cluster heads will move them away from their transmission ranges. It is observed that for $R_0 = 70m$, all these lifetimes are prolonged furthest.

When $c = 0$, MAUCR protocol performs as an equal clustering approach. The energy consumption gradually balanced among cluster heads with an increase of c, therefore, the network lifetime increases. However, the lifetime decreases when c is too large. This is because the number of clusters produced closer to the base station will be more, and each of them will transmit their data to the base station directly, which causes a waste of energy. It is observed that the cluster lifetime and inter-cluster link lifetime increases with c. However, it decreases for large value of c. The reason is that for a fixed value of R_0, the competition range decreases for large value of c (see (6.1)) which in turn increases number of clusters (see Figure 6.10). The number of affiliated members under each cluster head will be less if the number of clusters in a given network scale is more. This in turn increase the probability that all its cluster members will move away in high-mobility environment, and the cluster head initiates re-clustering and thus impacting the cluster lifetime and inter-cluster link lifetime. Both the network lifetime and the mean inter-cluster link lifetime are comparably high for $c = 0.3$ and comparably high value for the mean cluster lifetime is obtained for $c = 0.25$. Therefore, there exists an optimal value of c for $R_0 = 70m$ that could best extend these lifetimes. To further tune c we conducted a set of simulation experiments to find the impact of c on the packet delivery ratio. Figure 6.11(d) shows the packet delivery ratio for different values of c and $R_0 = 70m$. It is observed that the WSN achieves the highest packet delivery ratio for $c = 0.27$ which is approximately the optimal value of c for $R_0 = 70m$.

Next, we investigate the impact of TD_MAX on the network lifetime. TD_MAX decides the area where cluster heads should directly send their data to the base station. In order to save and balance the energy consumption the size of this area should be properly tuned. If TD_MAX becomes larger, a larger group of cluster heads communicates directly with the base station, resulting in a waste of energy. In contrary, for smaller TD_MAX, the energy hole problem may not be addressed since the average load of cluster heads in this direct communication area is too high. Therefore, there exists an optimal value of TD_MAX that could best

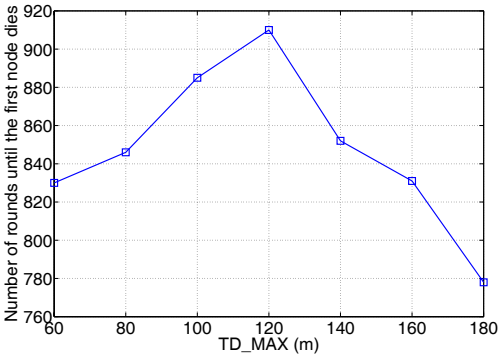
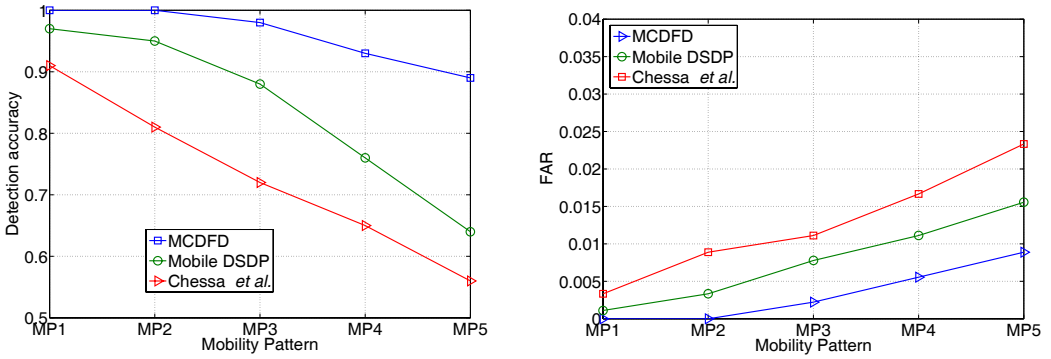


Figure 6.12: Parameter tuning: Network lifetime.

extend the network lifetime. Figure 6.12 depicts the network lifetime for different settings TD_MAX. As shown in Figure 6.12, the optimal value of TD_MAX for $R_0 = 70m$ is $120m$.

6.4.2 Experiment 2: Robustness with regard to node mobility

In this experiment, the performance parameters namely DA and FAR are evaluated with regard to node mobility. These two performance parameters are compared with the state-of-art schemes namely Mobile-DSDP [57] and Chessa *et al.* scheme [37]. We consider the aforementioned example network where all nodes are moving by following the mobility patterns as shown in Table 6.2. 100 faulty nodes are randomly deployed. To show the effectiveness, we consider an equal number of hard and soft faults.



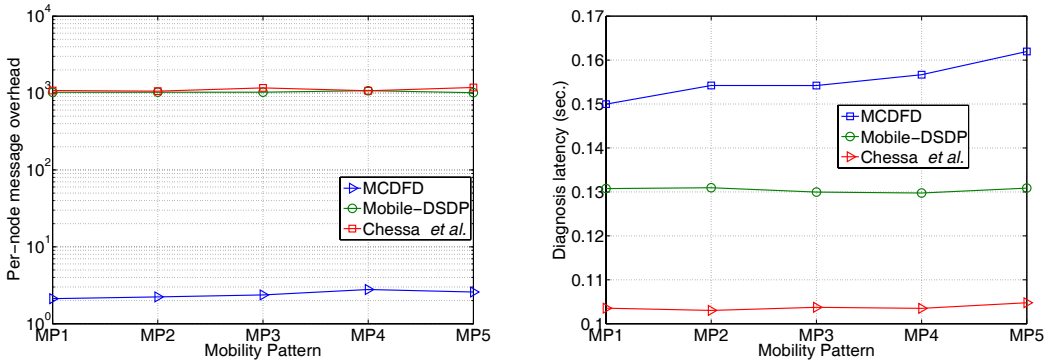
(a) DA (b) FAR

Figure 6.13: DA and FAR at varying mobility patterns.

As shown in Figure 6.13(a), the detection accuracy of the proposed diagnosis algorithm remains very close to 1 in the low-mobility patterns (i.e., MP1, MP2 and MP3). However, manageable performance degradation in regard to DA is reported in the high-mobility patterns (i.e., MP4 and MP5). The reason is that the local diagnostic view is not affected by node mobility. In addition, a soft faulty node wrongly detected as fault-free only when it has more than $0.5(N-1)$ faulty neighbors, and all of them produce same result set $\{TR\}$. The probability of mentioned neighbors producing $\{TR\}$ is very small. Dissemination of local diagnostics to obtain the global diagnostic view is negligibly affected by high node mobility patterns because dissemination is carried out through relatively stable cluster heads. Mobile-DSDP shows a comparable result to our model at low mobility patterns. However, it suffers in high-mobility patterns because a node may not get a chance to initiate its diagnosis session and thus remains undiagnosed. The reason is that in Mobile-DSDP, a mobile node that receives a test request or a test response for the first time will discover that a diagnosis session has been initiated. Mobile-DSDP adopts a flooding-based dissemination strategy. Though flooding-based technique ensures dissemination of diagnostic information, it fails in high node mobility. The model proposed by Chessa *et al.* is worse affected for a high-mobility pattern as hard-faulty nodes cannot be distinguished from fault-free nodes that migrated out of the testing node's transmitting range. In addition, the assumption made in detecting soft faulty nodes is hard to quantify unless some restrictions on the mobility of the nodes are imposed.

The FAR for different mobility patterns is reported in Figure 6.13(b). In WSNs, the FAR is important as high FAR isolates fault-free nodes from the WSN, thus reducing available resources and impacting reliability. To clarify the reason for not achieving a better FAR in Mobile-DSDP and Chessa *et al* scheme, we considered the following scenario where a fault-free node before responding to the test request has migrated out of all the testing nodes transmitting range. These two schemes will mark this fault-free node as faulty. This becomes even worse for sparse WSNs as well for high-dynamic WSNs. In contrary, in the proposed detection algorithm, a fault-free node fails to pass the threshold test is later diagnosed as fault-free by the fault-free neighbor(s). In addition, as discussed in Chapter 3, the use of

optimal threshold makes the algorithm to perform better in sparse WSNs.



(a) Diagnosis message overhead

(b) Diagnostic latency

Figure 6.14: Per-node message overhead and diagnosis latency: $P_{cerr} = 1 \times 10^{-3}$.

6.4.3 Experiment 3: Time and message efficiency

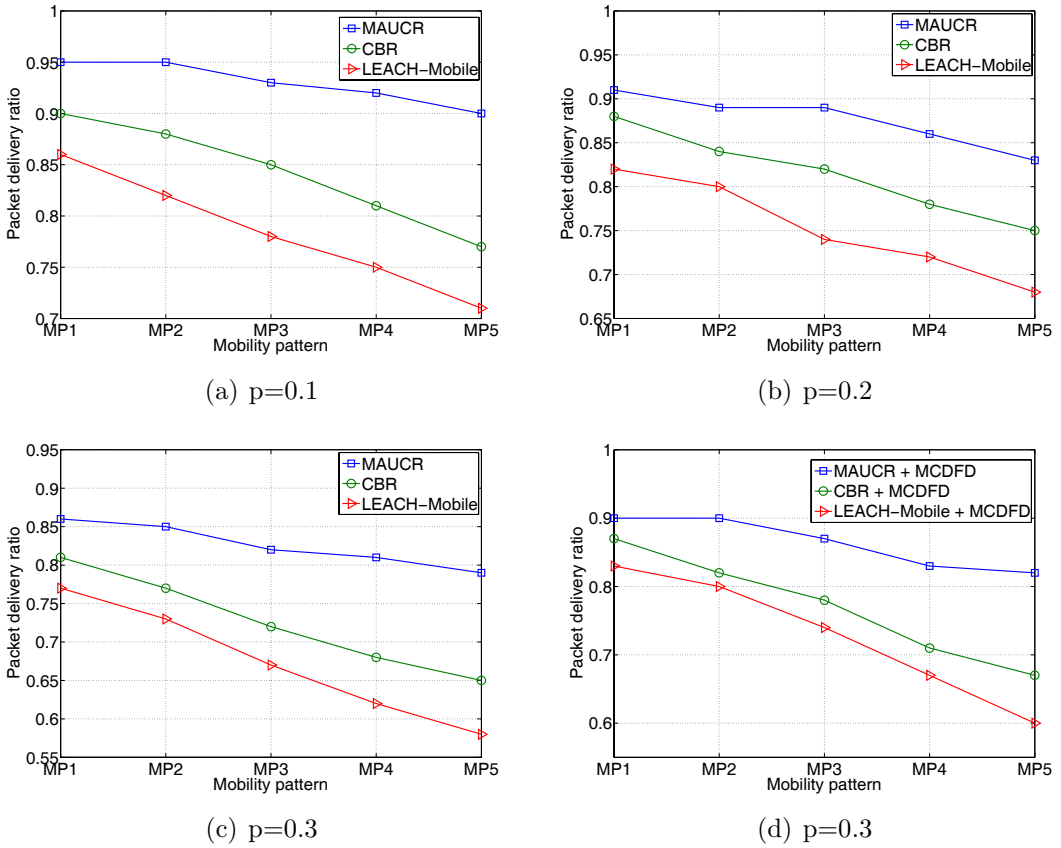
Figures 6.14(a) and 6.14(b) compare the performance parameters namely average per-node message overhead and diagnostic latency with mobile-DSDP and Chessa et al. scheme. The average values are obtained by repeating this simulation over 100 random topologies. Figure 6.14(a) shows the per node message overhead and the advantage that proposed algorithm presents over mobile-DSDP and Chessa et al. scheme. Figure 6.14(b) shows the diagnostic latency with regard to various mobility patterns. When a node moves in a dynamic WSN, it may be attached to different clusters at different times, which results in a frequent path rediscoveries each time it changes the point of attachment. Thus, as expected, the diagnostic latency of the proposed scheme is manageably high as compare to mobile-DSDP and Chessa et al. scheme.

If these results are put into context, it is observed that the proposed scheme outperforms both Mobile-DSDP and Chessa *et al.* model from message complexity perspective. Since the proposed schemes will be used in WSNs, where some or all the nodes may rely on batteries or other exhaustible means for their energy, it would be preferable for a proposed scheme to be energy efficient. Thus, a diagnosis scheme should be communication-efficient contrary to time-efficient.

6.4.4 Experiment 4: Efficiency with regard to packet delivery ratio

In this experiment, we evaluate the performance of MCDFD algorithm in regard to the packet delivery ratio. We compare MACUCR with sate-of-art clustering protocols namely CBR [114] and LEACH-Mobile protocol [115]. The performance of these clustering protocols with and without integrating the proposed diagnosis algorithm is evaluated. We consider the afore mentioned example network where, sensor nodes are assumed to be faulty with probabilities of 0.10, 0.20, 0.30 respectively. All nodes are moving by following the aforementioned mobility patterns. To show the effectiveness, we consider an equal number of hard and soft faults. The average values are obtained by repeating this simulation over 100 random topologies. First, we implement the protocols without considering the diagnosis algorithm. Figures 6.15(a), 6.15(b) and 6.15(c) depict the packet delivery ratio for different mobility patterns and fault rates without considering the fault detection. It is observed that the MAUCR protocol outperforms both LEACH-Mobile and CBR protocol. This is because MAUCR protocol considers the neighbor time as a primary parameter while constructing the clusters and considers both residual energy and neighbor time while constructing the cluster head backbone. The cluster head backbone is used for data delivery to the sink. In contrary, nodes in LEACH-Mobile and CBR choose the cluster head according to received signal strength and do not consider the node mobility. It is observed that the packet delivery ratio in MAUCR, CBR and LEACH-Mobile protocols decreases for an increase in node velocity. The reason is that nodes will keep changing their affiliating clusters more frequently and faster with node velocity. This results in many disconnection periods which in turn causes high packet loss. The percentage of successfully received packets suffers more when the disconnection periods are more frequent and extended for long time. However, it is observed that MAUCR protocol is less affected by an increase in node velocity. This is because of the stable link between the sensor nodes and their affiliating cluster heads, and the stable links between relay cluster heads created by MAUCR protocol.

An improvement in the packet delivery ratio is observed when the proposed



(a) p=0.1 (b) p=0.2

(c) p=0.3 (d) p=0.3

Figure 6.15: Packet delivery ratio: (a)(b)(c) Without fault diagnosis. (d) With fault diagnosis.

diagnosis algorithm is embedded in the clustering protocols. This is shown in Figure 6.15(d). The reason of this improvement can be explained as follows—(i) A faulty node elected as cluster head may drop some or all the packets routed through it. (ii) If the detection algorithm is executed before clustering, the nodes detected as faulty will not be allowed to participate in cluster head selection process.

6.4.5 Experiment 5: Network lifetime

In this experiment, we evaluate the energy efficiency of MAUCR protocol. First, we compare the network lifetime with CBR [114] and LEACH-Mobile [115] without considering the fault diagnosis. We consider the example network of Experiment 4 where, sensor nodes are assumed to be faulty with probabilities of 0.1. A node is considered dead if it has lost 99 percent of its initial energy. As expected and shown in Figure 6.16(a) MAUCR protocol outperforms both CBR and LEACH-Mobile.

The reason is that MAUCR protocol addresses the hot spot problem by creating unequal clusters. In addition relatively stable nodes are elected as cluster heads. The proposed greedy geographic routing protocol ensures a stable cluster head backbone for inter-cluster communication. Frequent re-clustering is avoided in MAUCR protocol since a sable link is created between non-cluster-head nodes and cluster head nodes. This in turn saves energy and prolongs the network lifetime.

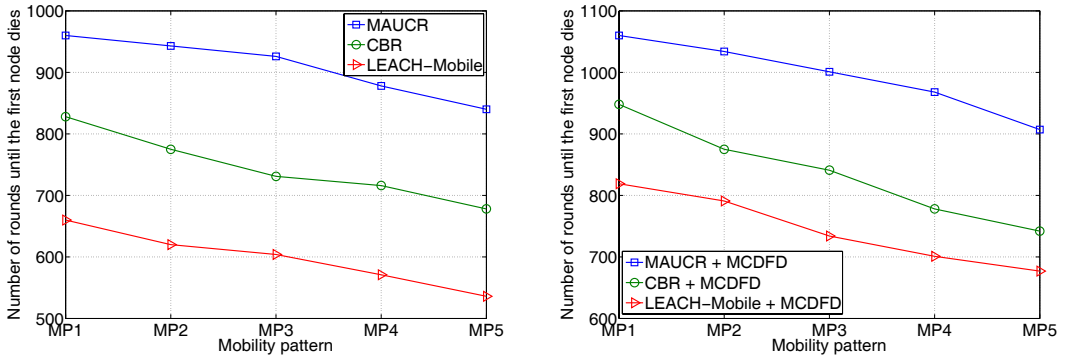


Figure 6.16: Network lifetime: (a) Without fault diagnosis: $p = 0.2$. (b) With fault diagnosis: $p = 0.2$.

An improvement in a network lifetime is observed when the proposed diagnosis algorithm works in conjunction with the clustering protocols. This is shown in Figure 6.16(b). If faulty nodes are allowed to send their data, then relay nodes dissipate energy in forwarding this erroneous data to the sink node. This becomes even worse for multimedia sensor networks where the amount of data generated by each node is large. In this approach since the erroneous data generated by the faulty nodes are discarded, wastage of energy in relaying these erroneous data is avoided. This in turn increases the network lifetime. In cluster-based routing if the non-cluster-head nodes are unaware of the failure at the head sensor node, they send meaningless data and therefore, waste energy. As discussed earlier, the non-cluster-head nodes are informed about the fault state of their cluster heads at each data-gathering round.

6.5 Summary

We have extended the unequal cluster based routing protocol [90] for WSNs by integrating mobility and node fault detection. Neighbor time is considered to elect relatively stable nodes as cluster heads. Greedy geographic routing protocol is proposed, which constructs a stable cluster head backbone for inter-cluster data communication and dissemination of local diagnostics. The adaptive TDMA scheduling and the integrated fault diagnosis algorithm ensures a better packet delivery ratio (> 0.8) in highly dynamic WSN with fault rate as high as 0.3. The efficiency of proposed MCDFD is substantiated by the network lifetime which is greater than 900 data-gathering rounds in highly dynamic WSN with fault rate as high as 0.2. In general, it is observed that the proposed algorithm outperforms the existing state-of-art algorithms.

Chapter 7

Conclusions and Future Work

Large-scale deployment of low-cost sensor nodes in uncontrolled, harsh or hostile environments is the inherent property of WSNs. It is common for the sensor nodes in WSNs to become faulty and unreliable. The normal operation of a WSN suffers from faulty data since it decreases the judgment accuracy of the base station, it increases the traffic in the WSNs, and it wastes a considerable amount of energy from a sensor node's limited energy. In addition, the sensors are often used to compute control actions, where sensor faults can cause catastrophic events. For the last one decade, researchers across the globe have been working to efficiently diagnose faults in WSNs and quite significant volumes of literature are available in this area. Owing to the ill-posed nature of WSNs, the problem is still open and needs substantial research.

In this thesis, algorithms have been proposed to diagnose faults in WSNs and evaluations are made analytically as well as through simulations using Castalia-2.3b, a state-of-art WSN simulator based on the OMNET++ platform. The proposed algorithm CDFD exploits the spatial correlation between sensor measurements and works in conjunction with the UCR protocol. CDFD is lightweight since it imposes a negligible extra cost in the WSNs. The diagnostic messages are sent as the output of the routine tasks of the WSNs. To obtain the diagnostic global view, CDFD uses the cluster head backbone constructed by the UCR protocol and thereby reducing the overheads. An optimal value for the threshold is derived, which ensures the algorithm to perform better in sparse WSNs or WSNs with sparse areas. The high detection accuracy and low

false alarm rate make the algorithm efficient from network lifetime perspective. The message complexity of CDFD is $O(n)$ and the number of bits exchanged to diagnose the WSN are $O(n \log_2 n)$. Comparative analysis demonstrates the efficacy of the CDFD algorithm.

The diagnosis of intermittent faults in WSNs is modeled as a multiobjective optimization problem. The two objectives such as detection latency and energy overhead are taken into consideration while considering detection error as a constraint. The two-lbests-based multiobjective particle swarm optimization (2LB-MOPSO) algorithm is used as a tool to find trade-offs accounting for the relative importance of detection accuracy, diagnosis latency and energy overhead. A fuzzy-based mechanism is used to find out the best compromised solution on the optimal Pareto front. The tuned detection parameters were used by the detection algorithm. The performance difference between 2LB-MOPSO and NSGA-II based parameter tuning was observed and 2LB-MOPSO based approach was found more suitable for the proposed application. A high level (> 0.95) of DA is achieved while keeping the FAR low (< 0.01) for sparse WSNs. The proposed CDIFD algorithm effectively tolerates faults in communication channels. CDIFD algorithm is energy efficient since the normalized total energy consumption is very less.

A count and threshold-based mechanism is used to discriminate transient from intermittent or permanent faults. The detection parameters namely inter-test interval, reward counter, and penalty counter were tuned to effectively discriminate the persistence of faults. The diagnosis algorithm details the recovery phase of a sensor node as an integral part of the diagnostic process.

A mobility aware fault diagnosis algorithm to diagnose permanent faults has been presented. A mobility and energy aware clustering technique is proposed, where the neighbor time, residual energy, and fault state are considered to elect relatively stable nodes as cluster heads. A greedy geographic routing protocol is proposed which constructs a stable cluster head backbone for inter-cluster data communication. The proposed topology adaptive TDMA scheduling and the integrated fault diagnosis algorithm ensure a better data delivery (> 0.8) in highly dynamic WSNs with a fault rate as high as 0.3. The proposed MCDFD algorithm is energy efficient. The network lifetime is reported as greater than 900

data-gathering rounds in highly dynamic WSNs with a fault rate as high as 0.2. All the functional blocks of a sensor node are checked for error by the suggested test pattern. The robustness of the proposed algorithm to mobility is compared with the state-of-art fault diagnosis schemes.

The research findings made out of this thesis have opened several auxiliary research directions, which can be further investigated. The proposed MCDFD algorithm can be extended to discriminate persistence of faults in WSNs with mobile nodes. Since an event also causes abnormal data to be sensed by the nearby sensor nodes, the proposed schemes can naturally be extended to cope with the fault-event disambiguation problem. In this thesis, we consider only static faults, i.e., the state of a node is not allowed to change during diagnosis round. Another promising research direction to pursue is to bring robustness of the fault diagnosis algorithm to attacks of malicious nodes. A malicious node alone can hardly affect the decision-making process of the proposed diagnosis schemes simply by sending incorrect sensing data or incorrect decision. However, malicious sensor nodes cooperating may isolate fault-free sensor nodes in such a way that they diagnose incorrectly themselves to be faulty.

Bibliography

[1] M. Barborak, A. Dahbura, and M. Malek, "The consensus problem in fault-tolerant computing," *ACM Computing Survey*, vol. 25, pp. 171–220, 1993.

[2] N. Ramanathan, T. Schoellhammer, D. Estrin, M. Hansen, T. Harmon, E. Kohler, and M. Srivastava, "The final frontier: Embedding networked sensors in the soil," UCLA, Center for Embedded Networked Computing, Tech. Rep. CENS-TR-68, 1995.

[3] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, pp. 2292–2330, 2008.

[4] M. Perillo, Z. Cheng, and W. Heinzelman, "An analysis of strategies for mitigating the sensor network hot spot problem," in *Proceedings of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, 2005, pp. 474–478.

[5] J. F. M. Faye, "An ultra hierarchical clustering-based secure aggregation protocol for wireless sensor networks," *Advances in Information Sciences and Service Sciences*, vol. 3, no. 9, pp. 309–319, 2001.

[6] H. Luo, K. Wu, Z. Guo, L. Gu, and L. Ni, "Ship detection with wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 7, pp. 1336–1343, 2012.

[7] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[8] G. Simon, M. Maróti, A. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton, "Sensor network-based countersniper system," in *Proceedings of the 2nd international conference on Embedded networked sensor systems.* ACM, 2004, pp. 1–12.

[9] K. Casey, A. Lim, and G. Dozier, "A sensor network architecture for tsunami detection and response," *International Journal of Distributed Sensor Networks*, vol. 4, pp. 28–43, 2008.

[10] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke, "Data collection, storage, and retrieval with an underwater sensor network," in *Proceedings of the*

3rd international conference on Embedded networked sensor systems. ACM, 2005, pp. 154–165.

[11] D.-J. Kim and B. Prabhakaran, "Motion fault detection and isolation in body sensor networks," *Pervasive and Mobile Computing*, vol. 7, no. 6, pp. 727–745, 2011.

[12] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Communication ACM*, vol. 47, pp. 34–40, 2004.

[13] K. Tan, S. Huang, Y. Zhang, and T. Lee, "Distributed fault detection in industrial system based on sensor wireless network," *Computer Standards & Interfaces*, vol. 31, no. 3, pp. 573–578, 2009.

[14] D. Sudharsan, J. Adinarayana, A. K. Tripathy, M. H. S. Ninomiya, T. Kiura, U. B. Desai, S. N. Merchant, D. R. Reddy, and G. Sreenivas, "Geosense: A multimode information and communication system," *ISRN Sensor Networks*, vol. 2012, pp. 40–52, 2012.

[15] M. Yu, H. Mokhtar, and M. Merabti, "Fault management in wireless sensor networks," *IEEE Wireless Communications*, vol. 14, no. 6, pp. 13–19, 2007.

[16] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentell, "Fault tolerance techniques for wireless ad hoc sensor networks," in *Proceedings of IEEE Sensors*, vol. 2, 2002, pp. 1491–1496.

[17] K. Ni, N. Ramanathan, M. N. H. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava, "Sensor network data fault types," *ACM Transaction on Sensor Networks*, vol. 5, pp. 1–29, 2009.

[18] P. Jalote, *Fault tolerance in distributed systems.* Pentice Hall, April 1994.

[19] S. Chessa, "Self-diagnosis of grid-interconnected systems, with application to self-test of vlsi wafers," Tech. Rep., 1999.

[20] M. Elhadef, A. Boukerche, and H. Elkadiki, "A distributed fault identification protocol for wireless and mobile ad hoc networks," *Journal of Parallel and Distributed Computing*, vol. 68, no. 3, pp. 321–335, 2008.

[21] D. P. Siewiorek and R. S. Swmlz, *Reliable Computer System Design and Evaluation.* Digital Press, 1992.

[22] A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and F. Grandoni, "Threshold-based mechanisms to discriminate transient from intermittent faults," *IEEE Transaction on Computers*, vol. 49, no. 3, pp. 230–245, 2000.

[23] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transaction on Dependable Secure Computing*, vol. 1, pp. 11–33, 2004.

[24] C. P. Fuhrman, "Comparison-based diagnosis in fault-tolerant, multiprocessor systems," Ph.D. dissertation, Swiss Federal Institute of Technology in Lausanne, 1996.

[25] H. Wang, N. Agoulmine, M. Ma, and Y. Jin, "Network lifetime optimization in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 7, pp. 1127 –1137, 2010.

[26] F. P. Preparata, G. Metze, and R. T. Chien, "On the connection assignment problem of diagnosable systems," *IEEE Transactions on Electronic Computers*, vol. EC-16, no. 6, pp. 848–854, 1967.

[27] E. P. Duarte, Jr., R. P. Ziwich, and L. C. Albini, "A survey of comparison-based system-level diagnosis," *ACM Computing Survey*, vol. 43, pp. 1–56, 2011.

[28] M. Malek, "A comparison connection assignment for diagnosis of multiprocessor systems," in *Proceedings of the 7th annual symposium on Computer Architecture*. ACM, 1980, pp. 31–36.

[29] M. C. Vuran, zgr B. Akan, and I. F. Akyildiz, "Spatio-temporal correlation: theory and applications for wireless sensor networks," *Computer Networks*, vol. 45, no. 3, pp. 245–259, 2004.

[30] S. Hakimi and A. Amin, "Characterization of connection assignment of diagnosable systems," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 86–88, 1974.

[31] J. Chen, S. Kher, and A. Somani, "Distributed fault detection of wireless sensor networks," in *Workshop on Dependability issues in wireless ad hoc networks and sensor networks*. ACM, 2006, pp. 65–72.

[32] G. Jian-Liang, X. Yong-Jun, and L. Xiao-Wei1, "Weighted-median based distributed fault detection for wireless sensor networks," *Journal of Software*, vol. 18, no. 5, pp. 1208–1217, 2007.

[33] M.-H. Lee and Y.-H. Choi, "Fault detection of wireless sensor networks," *Computer Communications*, vol. 31, no. 14, pp. 3469–3475, 2008.

[34] P. Jiang, "A new method for node fault detection in wireless sensor networks," *Sensors*, vol. 9, no. 2, pp. 1282–1294, 2009.

[35] J.-Y. Choi, S.-J. Yim, Y. J. Huh, and Y.-H. Choi, "A distributed adaptive scheme for detecting faults in wireless sensor networks," *WSEAS Transactions on Communications*, vol. 8, pp. 269–278, 2009.

[36] X. Miao, K. Liu, Y. He, Y. Liu, and D. Papadias, "Agnostic diagnosis: Discovering silent failures in wireless sensor networks," in *INFOCOM*, 2011, pp. 1548–1556.

[37] S. Chessa and P. Santi, "Comparison-based system-level fault diagnosis in ad hoc networks," in *20th IEEE Symposium on Reliable Distributed Systems*, 2001, pp. 257–266.

[38] ——, "Crash faults identification in wireless sensor networks," *Computer Communications*, vol. 25, no. 14, pp. 1273–1282, 2002.

[39] C. Hajiyev, F. Caliskan, and C. Hajiyev, *Fault Diagnosis And Reconfiguration In Flight Control Systems*. Springer, October 2003.

[40] S. Gobriel, S. Khattab, D. Mosse, J. Brustoloni, and R. Melhem, "Ridesharing: Fault tolerant aggregation in sensor networks using corrective actions," in *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, 2006, pp. 595–604.

[41] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 1–13.

[42] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons from a sensor network expedition," in *Wireless Sensor Networks*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004, pp. 307–322.

[43] H. Li, M. Price, J. Stott, and I. Marshall, "The development of a wireless sensor network sensing node utilising adaptive self-diagnostics," in *Self-Organizing Systems*, 2007, pp. 30–43.

[44] E. Elnahrawy and B. Nath, "Cleaning and querying noisy sensors," in *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*. ACM, 2003, pp. 78–87.

[45] N. Ramanathan, L. Balzano, M. Burt, D. Estrin, T. Harmon, C. Harvey, J. Jay, E. Kohler, S. Rothenberg, and M. Srivastava, "Rapid deployment with confidence: Calibration and fault detection in environmental sensor networks," Center for Embedded Networked Sensing, UCLA and Department of Civil and Environmental Engineering, MIT, Tech. Rep., 2006.

[46] D. M. Blough and H. W. Brown, "The broadcast comparison model for on-line fault diagnosis in multicomputer systems: Theory and implementation," *IEEE Transactions on Computers*, vol. 48, no. 5, pp. 470–493, 1999.

[47] X. Yang, G. M. Megson, and D. J. Evans, "A comparison-based diagnosis algorithm tailored for crossed cube multiprocessor systems," *Microprocessors and Microsystems*, vol. 29, no. 4, pp. 169–175, 2005.

[48] X. Yang and Y. Y. Tang, "Efficient fault identification of diagnosable systems under the comparison model," *IEEE Transactions on computers*, vol. 56, no. 12, pp. 1612–1618, 2007.

[49] S.-Y. Hsieh and Y.-S. Chen, "Strongly diagnosable product networks under the comparison diagnosis model," *IEEE Transactions on Computers*, vol. 57, no. 6, pp. 721–732, 2008.

[50] G.-Y. Chang, "(t, k)-diagnosability for regular networks," *IEEE Transactions on Computers*, vol. 59, no. 9, pp. 1153–1157, 2010.

[51] B. Krishnamachari and S. Iyengar, "Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 241–250, 2004.

[52] A. Boulis, *Castalia: A simulator for wireless sensor networks and body area networks*, National ICT Australia Ltd, Australia, 2009.

[53] A. Varga and R. Hornig, "An overview of the OMNET++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 2008, pp. 1–10.

[54] M. Yu, H. Mokhtar, and M. Merabti, "Fault management in wireless sensor networks," *IEEE Wireless Communications*, vol. 14, no. 6, pp. 13–19, 2007.

[55] A. Weber, A. R. Kutzke, and S. Chessa, "Diagnosability evaluation for a system-level diagnosis algorithm for wireless sensor networks," in *IEEE Symposium on Computers and Communications*, 2010, pp. 241–244.

[56] A. Weber, A. Kutzke, and S. Chessa, "Energy-aware test connection assignment for the self-diagnosis of a wireless sensor network," *Journal of the Brazilian Computer Society*, pp. 1–9, 2012.

[57] M. Elhadef, A. Boukerche, and H. Elkadiki, "Diagnosing mobile ad-hoc networks: two distributed comparison-based self-diagnosis protocols," in *Proceedings of the 4th ACM international workshop on Mobility management and wireless access*. ACM, 2006, pp. 18–27.

[58] C. Hsin and M. Liu, "Self-monitoring of wireless sensor networks," *Computer Communications*, vol. 29, no. 4, pp. 462–476, 2006.

[59] X. Luo, M. Dong, and Y. Huang, "On distributed fault-tolerant detection in wireless sensor networks," *IEEE Transactions on Computers*, vol. 55, no. 1, pp. 58–70, 2006.

[60] S.-J. Yim and Y.-H. Choi, "An adaptive fault-tolerant event detection scheme for wireless sensor networks," *Sensors*, vol. 10, no. 3, pp. 2332–2347, 2010.

[61] X.-Y. Xiao, W.-C. Peng, C.-C. Hung, and W.-C. Lee, "Using sensorranks for in-network detection of faulty readings in wireless sensor networks," in *Proceedings of the 6th ACM international workshop on Data engineering for wireless and mobile access*. ACM, 2007, pp. 1–8.

[62] S. Guo, Z. Zhong, and T. He, "Find: faulty node detection for wireless sensor networks," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2009, pp. 253–266.

[63] S. Rost and H. Balakrishnan, "Memento: A health monitoring system for wireless sensor networks," in *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, vol. 2, 2006, pp. 575–584.

[64] L. Gheorghe, R. Rughini, R. Deaconescu, and N. pu, *Adaptive Trust Management Protocol Based on Fault Detection for Wireless Sensor Networks*, 2010, pp. 216–221.

[65] A. Abdul-Rahman and S. Hailes, "Supporting trust in virtual communities," in *33rd Annual Hawaii International Conference on System Sciences*, 2000, pp. 1–9.

[66] S. Harte, A. Rahman, and K. Razeeb, "Fault tolerance in sensor networks using self-diagnosing sensor nodes," in *The IEE International Workshop on Intelligent Environments*, 2005, pp. 7–12.

[67] D. Rakhmatov and S. Vrudhula, "Time-to-failure estimation for batteries in portable electronic systems," in *International Symposium on Low Power Electronics and Design*, 2001, pp. 88–91.

[68] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "A discrete-time battery model for high-level power estimation," in *Proceedings of the conference on Design, automation and test in Europe*. ACM, 2000, pp. 35–41.

[69] G. Gupta and M. Younis, "Fault-tolerant clustering of wireless sensor networks," in *IEEE Wireless Communications and Networking*, vol. 3, 2003, pp. 1579–1584.

[70] C. Jaikaeo, C. Srisathapornphat, and C.-C. Shen, "Diagnosis of sensor networks," in *IEEE International Conference on Communications*, vol. 5, 2001, pp. 1627–1632.

[71] A. Tai, K. Tso, and W. Sanders, "Cluster-based failure detection service for large-scale ad hoc wireless network applications," in *International Conference on Dependable Systems and Networks*, 2004, pp. 805–814.

[72] O. Younis, S. Fahmy, and P. Santi, "An architecture for robust sensor network communications," *International Journal of Distributed Sensor Networks*, vol. 1, no. 3–4, 2005.

[73] P. Wang, J. Zheng, and C. Li, "An agreement-based fault detection mechanism for under water sensor networks," in *IEEE Global Telecommunications Conference*, 2007, pp. 1195–1200.

[74] G. Venkataraman, S. Emmanuel, and S. Thambipillai, "Energy-efficient cluster-based scheme for failure management in sensor networks," *IET Communications*, vol. 2, no. 4, pp. 528–537, 2008.

[75] M. Asim, H. Mokhtar, and M. Merabti, "A fault management architecture for wireless sensor network," in *Wireless Communications and Mobile Computing*. IEEE, 2008, pp. 779–785.

[76] W. Wang, B. Wang, Z. Liu, and L. Guo, "A cluster-based real-time fault diagnosis aggregation algorithm for wireless sensor networks," *Information Technology Journal*, vol. 10, no. 1, pp. 80–88, 2011.

[77] K. Sakib, "Asynchronous failed sensor node detection method for sensor networks," *International Journal of Network Management*, vol. 22, no. 1, pp. 27–49, 2011.

[78] Z. Ji, W. Bing-shu, M. Yong-guang, Z. Rong-hua, and D. Jian, "Fault diagnosis of sensor network using information fusion defined on different reference sets," in *International Conference on Radar*, 2006, pp. 1–5.

[79] A. Jabbari, R. Jedermann, and W. Lang, "Application of computational intelligence for sensor fault detection and isolation," in *World Academy of Science, Engineering and Technology*, 2007, pp. 265–270.

[80] A. Moustapha and R. Selmic, "Wireless sensor network modeling using modified recurrent neural networks: Application to fault detection," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 5, pp. 981–988, 2008.

[81] J. Barron, A. Moustapha, and R. Selmic, "Real-time implementation of fault detection in wireless sensor networks using neural networks," in *Fifth International Conference on Information Technology: New Generations*, 2008, pp. 378–383.

[82] 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*. ACM, 2000, pp. 255–265.

[83] A. Patcha and A. Mishra, "Collaborative security architecture for black hole attack prevention in mobile ad hoc networks," in *Radio and Wireless Conference*, 2003, pp. 75–78.

[84] T. Instruments, *MSP430x13x, MSP430x14x Mixed Signal Microcontroller, Datasheet,*, 2001.

[85] C. AS, *CC1000 Single Chip Very Low Power RF Transceiver Data Sheet*, 2004.

[86] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell System Technical Journal*, pp. 1253–1265, 1960.

[87] E. O. Elliott, "Estimates of error rates for codes on burst error channels," *Bell System Technical Journal*, vol. 42, pp. 1977–1997, 1963.

[88] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[89] T. S. Rappaport, *Introduction to wireless communication systems.* Prentice Hall, 2002.

[90] G. Chen, C. Li, M. Ye, and J. Wu, "An unequal cluster-based routing protocol in wireless sensor networks," *Wireless Networks*, vol. 15, pp. 193–207, 2009.

[91] L. Hu, "Distributed code assignments for cdma packet radio networks," *IEEE/ACM Transactions on Networking*, vol. 1, no. 6, pp. 668–677, 1993.

[92] R. Horst, D. Jewett, and D. Lenoski, "The risk of data corruption in microprocessor-based systems," in *The Twenty-Third International Symposium on Fault-Tolerant Computing*, 1993, pp. 576–585.

[93] D. P. Siewiorek and R. S. Swmlz, *The Theory and Practice of Reliable System Design.* Digital Equipment Corporation,, 1982.

[94] S. Z. Zhao and P. N. Suganthan, "Two-lbests based multi-objective particle swarm optimizer," *Engineering Optimization*, vol. 43, no. 1, pp. 1–17, 2011.

[95] J. S. Dhillon, S. C. Parti, and D. P. Kothari, "Stochastic economic emission load dispatch," *Electric Power Systems Research*, vol. 26, no. 3, pp. 179–186, 1993.

[96] M. Breuer, "Testing for intermittent faults in digital circuits," *IEEE Transactions on Computers*, vol. C-22, no. 3, pp. 241–246, 1973.

[97] R. E. Barlow and F. Prochan, *Mathematical Theory of Reliability.* John Wiley & Sons, 1965.

[98] K. Deb, *Multi-objective optimization using evolutionary algorithms.* Wiley, 2001.

[99] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

[100] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[101] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications,," Ph.D. dissertation, Swiss Federal Institute of Technology, 1999.

[102] J. Schott, "Fault tolerant design using single and multi-criteria genetic algorithms," Master's thesis, Massachusetts Institute of Technology, 1995.

[103] H. Wu and A. A. Abouzeid, "Error resilient image transport in wireless sensor networks," *Computer Networks*, vol. 50, no. 15, pp. 2873–2887, 2006.

[104] M. Serafini, A. Bondavalli, and N. Suri, "On-line diagnosis and recovery: On the choice and impact of tuning parameters," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 4, pp. 295–312, 2007.

[105] K. Xu and M. Gerla, "A heterogeneous routing protocol based on a new stable clustering scheme," in *MILCOM*, vol. 2, 2002, pp. 838–843.

[106] R. Ghosh and S. Basagni, "Limiting the impact of mobility on ad hoc clustering," in *Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*. ACM, 2005, pp. 197–204.

[107] D.-S. Kim and Y.-J. Chung, "Self-organization routing protocol supporting mobile nodes for wireless sensor network," in *First International Multi-Symposiums on Computer and Computational Sciences*, 2006, pp. 622–626.

[108] S. A. B. Awwad, C. K. Ng, N. K. Noordin, and M. F. A. Rasid, "Cluster based routing protocol for mobile nodes in wireless sensor network," in *International Symposium on Collaborative Technologies and Systems*, 2009, pp. 233–241.

[109] P. S. Kumar, S. Ramachandram, and C. R. Rao, "Impact of node mobility and network size on the performance of zone routing protocol in mobile ad hoc networks," in *OBCOM*, 2006, pp. 170–175.

[110] Y. Xu and W. Wang, "Topology stability analysis and its application in hierarchical mobile ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 3, pp. 1546–1560, 2009.

[111] K. Sha and W. Shi, "Modeling the lifetime of wireless sensor networks," *Sensor Letters*, vol. 3, pp. 1–10, 2005.

[112] *MicaZ Mote data sheet*, www.openautomation.net/uploadsproductos/micaz_datasheet.pdf.

[113] H. Xu and J. Garcia-Luna-Aceves, "Neighborhood tracking for mobile ad hoc networks," *Computer Networks*, vol. 53, no. 10, pp. 1683–1696, 2009.

[114] S. Awwad, C. Ng, N. Noordin, and M. Rasid, "Cluster based routing protocol for mobile nodes in wireless sensor network," *Wireless Personal Communications*, vol. 61, pp. 251–281, 2011.

[115] G. Santhosh Kumar, M. Vinu Paul, and K. Poulose Jacob, "Mobility metric based leach-mobile protocol," in *16th International Conference on Advanced Computing and Communications*, 2008, pp. 248–253.

Dissemination

Journals

1. Arunanshu Mahapatro and Pabitra Mohan Khilar. Fault Diagnosis in Wireless Sensor Networks: A Survey. Accepted for Publication in *IEEE Communications Surveys and Tutorials, IEEE*.

2. Arunanshu Mahapatro and Pabitra Mohan Khilar. Online Distributed Fault Diagnosis in Wireless Sensor Networks. *Wireless Personal Communication, Springer*, Vol. 71, no. 3, pp 1931-1960, 2013.

3. Arunanshu Mahapatro and Pabitra Mohan Khilar. An Energy-Efficient Distributed Approach for Clustering-Based Fault Detection and Diagnosis in Image Sensor Networks. *IET Wireless Sensor Systems, IET*, Vol. 3, no. 1, 2013.

4. Arunanshu Mahapatro and Pabitra Mohan Khilar. An Adaptive Approach to Discriminate the Persistence of Faults in Wireless Sensor Networks. *ISRN Sensor Networks, Hindawi Publishing Corporation*, 2012. doi:10.5402/2012/342461.

5. Arunanshu Mahapatro and Pabitra Mohan Khilar. Detection and Diagnosis of Node failure in Wireless Sensor Networks: A Multiobjective Optimization Approach. Accepted for Publication in *Swarm and Evolutionary Computation, Elsevier*.

6. Arunanshu Mahapatro and Pabitra Mohan Khilar. Mobility Aware Distributed Diagnosis of Mobile Ad Hoc Sensor Networks. *Networking Science, Springer*, Vol. 2, no. 1-2, pp 52-65 2013.

Conferences

1. Arunanshu Mahapatro and Pabitra Mohan Khilar. Mobility and Energy Aware Distributed Clustering Protocol for Ad Hoc Sensor Networks. In *International Conference on Engineering Sustainable Solutions (INDICON), IEEE*, India, 2011.

2. Arunanshu Mahapatro and Pabitra Mohan Khilar. SDDP: Scalable Distributed Diagnosis Protocol for Wireless Sensor Networks. In *International Conference on Contemporary Computing (IC3), Springer*, India, 2011.

3. Arunanshu Mahapatro and Pabitra Mohan Khilar. newblock On distributed self fault diagnosis for wireless multimedia sensor networks In *International Conference on Communication, Computing and Security (ICCCS), ACM*, India, 2011.

Arunanshu Mahapatro

Research Scholar
Computer Science and Engineering Department,
National Institute of Technology Rourkela,
Rourkela – 769008, India.

Ph: `+91-680-2281757 (R), +91-9437438294 (M)`

e-mail: `arun227@gmail.com`

Qualification

- Doctor of Philosophy *(Continuing)*
  National Institute of Technology Rourkela, India
- Master of Technology
  Electronics and Communication Engineering
  KIIT University,Bhubaneswar, India, [8.46 CGPA]
- Bachelor of Engineering
  Electronics Engineering
  CEPN, Pune University, India, [First division]
- Diploma in Engineering
  Electronics and Communication Engineering
  SMIT, Berhampur, Odisha, India, [First division]
- Matriculation
  Medical Campus High School, Berhampur, Odisha, India, [First division]

Publications

- 06 Journal Articles
- 07 Conference Papers
- 02 Book Chapters

Permanent Address

Plot# A1-50, Arabinda Nagar 1^{st} Lane,
Berhampur – 760001, Odisha, India.

Date of Birth

June 26, 1975