Modeling and Analysis of Fault Detection and Fault Tolerance in Wireless Sensor Networks

ARSLAN MUNIR, University of Nevada, Reno
JOSEPH ANTOON, National Instruments
ANN GORDON-ROSS, University of Florida, Gainesville

3

Technological advancements in communications and embedded systems have led to the proliferation of Wireless Sensor Networks (WSNs) in a wide variety of application domains. These application domains include but are not limited to mission-critical (e.g., security, defense, space, satellite) or safety-related (e.g., health care, active volcano monitoring) systems. One commonality across all WSN application domains is the need to meet application requirements (e.g., lifetime, reliability). Many application domains require that sensor nodes be deployed in harsh environments, such as on the ocean floor or in an active volcano, making these nodes more prone to failures. Sensor node failures can be catastrophic for critical or safety-related systems. This article models and analyzes fault detection and fault tolerance in WSNs. To determine the effectiveness and accuracy of fault detection algorithms, we simulate these algorithms using ns-2. We investigate the synergy between fault detection and fault tolerance and use the fault detection algorithms' accuracies in our modeling of Fault-Tolerant (FT) WSNs. We develop Markov models for characterizing WSN reliability and Mean Time to Failure (MTTF) to facilitate WSN application-specific design. Results obtained from our FT modeling reveal that an FT WSN composed of duplex sensor nodes can result in as high as a 100% MTTF increase and approximately a 350% improvement in reliability over a Non-Fault-Tolerant (NFT) WSN. The article also highlights future research directions for the design and deployment of reliable and trustworthy WSNs.

Categories and Subject Descriptors: C.3 [**Computer Systems Organization**]: Special-Purpose and Application-Based Systems—*Real-time and embedded systems*; C.4 [**Computer Systems Organization**]: Performance of Systems—*Fault tolerance, modeling techniques, reliability, availability, and serviceability*

General Terms: Reliability, Design, Performance

Additional Key Words and Phrases: Fault-tolerance, fault detection, reliability, Markov modeling, wireless sensor networks

ACM Reference Format:
Arslan Munir, Joseph Antoon, and Ann Gordon-Ross. 2015. Modeling and analysis of fault detection and fault tolerance in wireless sensor networks. ACM Trans. Embedd. Comput. Syst. 14, 1, Article 3 (January 2015), 43 pages.
DOI: http://dx.doi.org/10.1145/2680538

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSERC.
Authors' addresses: A. Munir, 1664 N. Virginia St., University of Nevada, Reno, MS-171, Reno, Nevada, 89557 USA; email: arslan@unr.edu; J. Antoon and A. Gordon-Ross, University of Florida, P. O. Box 116200, 319 Benton Hall, Gainesville, FL 32611 USA; emails: joe.antoon@ufl.edu, ann@ece.ufl.edu.

DOI: http://dx.doi.org/10.1145/2680538

1. INTRODUCTION AND MOTIVATION

Wireless Sensor Networks (WSNs) consist of spatially distributed autonomous sensor nodes that collaborate with each other to perform an application task. A *sensor node* comprises a sensing unit (a *sensing unit* contains sensors such as temperature and humidity sensors), a processing unit, a storage unit, a communication unit, a power unit, an actuator unit, and an optional location finding unit [Munir and Gordon-Ross 2010].

Figure 1 shows the WSN architecture that we consider in this article [Akyildiz et al. 2002; Winkler et al. 2008; Jiang et al. 2009]. The sensor nodes distributed in the *sensor field* gather information (sensed data or statistics) about an observed phenomenon (e.g., environment, target) using attached sensors. A group of sensor nodes located geographically close to each other is called a *WSN cluster*. Each WSN cluster has one *cluster head* (WSN cluster formation and cluster head determination/maintenance is beyond the scope of this article). Sensed data within a WSN cluster are collected by the cluster head and relayed to a *sink node* (or base station) via the sensor nodes' ad hoc network. The sink node transmits the received information back to the *WSN designer* and/or *WSN manager* via a gateway node connected to a computer network. The WSN designer is responsible for designing the WSN for a particular application to meet application requirements such as lifetime, reliability, and throughput. After WSN design and deployment, the WSN manager manages WSN operations, such as data analysis, monitoring alive and dead sensor nodes, and alarm conditions (e.g., forest fire, volcano eruption).

WSN research and design has gained significance in recent years because of the increasing proliferation of WSNs in a wide variety of application domains. These application domains include but are not limited to mission-critical (e.g., security, defense, space, satellite) or safety-related (e.g., health care, active volcano monitoring) systems. The utility of WSNs for safety-related systems can be illustrated by a volcano monitoring example. Studying active volcanoes typically requires sensor arrays to collect seismic and infrasonic (low-frequency acoustic) signals. Distributed sensor nodes in a WSN provide a correlated spatial distribution of seismic and infrasonic events that greatly facilitates scientific studies of wave propagation phenomena and volcanic source mechanisms [Werner-Allen et al. 2006a]. The analytics of seismic and infrasonic data obtained by a WSN can help scientists better predict volcanic events, such as volcanic eruption and earthquakes. Although WSNs can benefit numerous applications, the realization of WSNs for an application requires addressing various design challenges.

A crucial challenge in WSN design is to meet varying application requirements (e.g., lifetime, throughput, reliability) given the limited energy, computation, and storage available on sensor nodes. Sensor nodes are often deployed in hostile environments, such as forests, the ocean floor [Yifan and Peng 2008], animal habitats [Mainwaring et al. 2002], and active volcanoes [Werner-Allen et al. 2006b], that further exacerbates the design challenge of meeting application requirements. Unattended and/or hostile deployment environments makes sensor nodes more susceptible to failures than other systems [Werner-Allen et al. 2006b], and manual inspection of sensor nodes after deployment can be impractical. Although faults can occur in any of the sensor node components, sensors and actuators have significantly higher fault rates than other semiconductor-based systems. For example, NASA aborted the launch of space shuttle Discovery [NASA 2011] because of a sensor failure in the shuttle's external tank [Moustapha and Selmic 2007]. Sensor failure can occur due to deployment impact, accidents (animal, vehicular), fire, extreme weather, or aging [Bredin et al. 2010]. Failed sensor nodes may result in sensor network partitioning (i.e., sensor nodes become isolated and are disconnected from the sensor network), reduced WSN availability, and WSN failure. To meet application requirements in the presence of sensor failures, incorporation of fault detection and Fault-Tolerance (FT) mechanisms in WSNs is imperative.

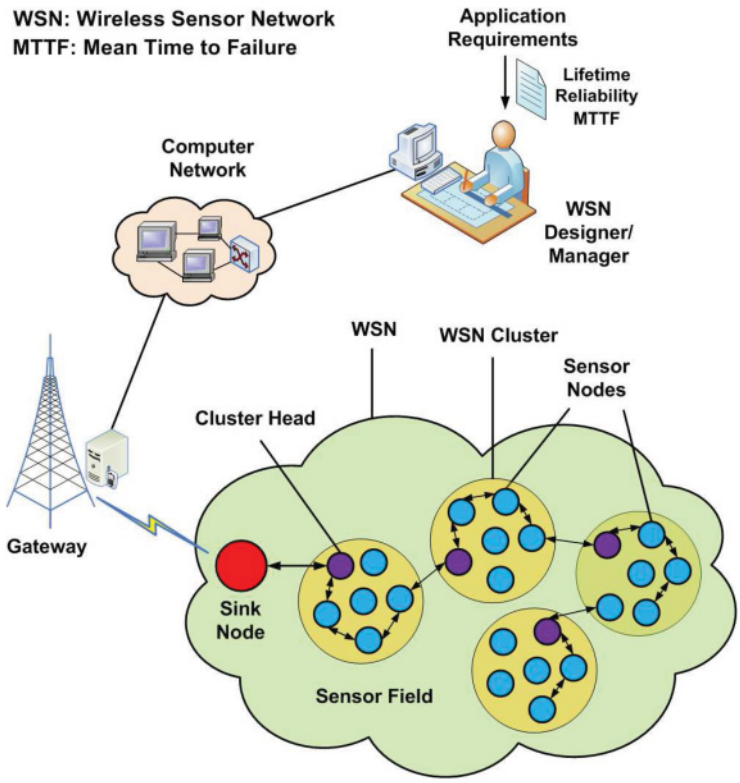


Fig. 1. Wireless sensor network architecture.

Although traditional reliability models can be readily applied to FT systems, faults do not always result in failure. Transient, malicious errors occur due to noise, voltage levels, and broken components. Therefore, it is imperative to model the ability for fault detection to provide coverage against such faults to protect mission-critical data. Fault detection includes Distributed Fault Detection (DFD) algorithms that identify faulty sensor readings to diagnose a faulty sensor. DFD algorithms do not incur additional transmission cost because they use existing network traffic to identify sensor failures. The *accuracy* of a fault detection algorithm signifies the algorithm's ability to accurately identify faults. We analyze and study two well-cited fault detection algorithms and compare their performance under realistic conditions. We implemented and simulated the DFD algorithms incorporating the protocol stack and a realistic WSN topology, as opposed to only algorithm simulation that was done in most previous works [Chen et al. 2006; Ding et al. 2005].

Although fault detection helps in isolating faulty sensors, FT incorporation in WSNs is imperative to reliably accomplish application tasks. A prominent FT technique is to add redundant hardware and/or software [Koren and Krishna 2007]. Stringent design constraints (e.g., power, cost) differentiate WSNs from other systems, and, consequently, the added redundancy for FT must justify the additional cost. Since sensors (e.g., temperature, light, motion) attached to sensor nodes have comparatively higher fault rates than do other components (e.g., processor, transceiver) [Moustapha and Selmic 2007; Sharma et al. 2007; Koushanfar et al. 2002], sensor redundancy would be most effective to enhance the FT capability of sensor nodes. Fortunately, sensors are cheap, and adding redundant spare sensors would add little to the individual sensor node's cost.

Even though FT is a well-studied research field [Hopkins et al. 1978; Wensley et al. 1978; Avizienis 1985; Somani and Vaidya 1997; Sklaroff 1976; Avizienis and Laprie 1986], fault diagnosis and FT for WSNs are relatively unstudied. Varying FT requirements across different applications increase the complexity of fault detection and FT for WSNs. For instance, mission-critical applications have relatively high reliability requirements as compared to non–mission-critical applications such as ambient conditions monitoring. To the best of our knowledge, there exists no sensor node model to provide better reliability for mission-critical applications. Since applications are typically designed to operate reliably for a certain period of time (i.e., lifetime requirement), *FT metrics* such as reliability and Mean Time to Failure (MTTF) need to be considered in WSN design. WSN designers require a model to estimate FT metrics and determine necessary redundancy during design time. Unfortunately, the literature provides no rigorous mathematical model that provides insights into WSN reliability and MTTF. Previous works study fault detection and FT in isolation, and their synergistic relationship has not been investigated in the context of WSNs.

Our main contributions in this article are:

—Fault diagnosis in WSNs through various fault detection algorithms.
—Simulation of fault detection algorithms using ns-2 to determine the algorithms' accuracy and false alarm rates. ns-2 is a discrete event simulator that provides support for simulation of networks and routing protocols over wired and wireless networks [ns2 2014]. We further analyze the effectiveness of fault detection algorithms under conditions modeled from real-world data.
—Proposal of an FT sensor node model consisting of duplex sensors (i.e., one active sensor and one inactive spare sensor) that exploits the synergy of fault detection and FT. Although our model can be extended for sensors with N-Modular Redundancy (NMR) [Koren and Krishna 2007], we suggest duplex sensor nodes to minimize the additional cost.
—Characterization of FT parameters, such as coverage factor, by exploiting the synergy between fault detection and FT.
—Proposal of hierarchical Markov models to characterize WSN reliability and MTTF. For the first time, we delineate reliability and MTTF hierarchically at the sensor node, WSN cluster, and WSN levels (Figure 1).
—Determination of iso-MTTF (isoreliability) for WSN clusters and the WSN. We define iso-MTTF (isoreliability) of a WSN as how many redundant WSN clusters an NFT WSN requires over an FT WSN to achieve an equal MTTF (reliability). Similarly, we define iso-MTTF (isoreliability) of a WSN cluster as how many redundant sensor nodes an NFT WSN cluster requires over an FT WSN cluster to achieve an equal MTTF (reliability).
—Research challenges and future research directions for the design and deployment of reliable and trustworthy WSNs.

In our duplex sensor model, we assume that the redundant sensor is in a cold standby mode, which ideally consumes no power. Although we focus on sensor failures within the sensor node in this study due to higher sensor failure rates as compared to other sensor node components [Moustapha and Selmic 2007; Sharma et al. 2007], our model can be extended to include failures for other components within the sensor node, such as the processor and transceiver. Our FT WSN modeling serves as a first step toward FT WSN modeling and can potentially facilitate further research in FT WSN modeling and evaluation.

Our Markov models are comprehensive and characterize sensor nodes, WSN clusters, and overall WSN reliability and MTTF. The proposed Markov modeling enables WSN designers to investigate the effects of different types of FT sensor nodes (e.g., duplex,

NMR), number of WSN clusters, and the number of sensor nodes in the cluster on the FT of the overall WSN. Our hierarchical WSN Markov models enable designers to select an efficient WSN topology to better meet application requirements.

The remainder of this article is organized as follows. Section 2 gives a review of related work. Section 3 elaborates on fault diagnosis in WSNs. Section 4 discusses two fault detection algorithms to elucidate the fault diagnosis mechanism in WSNs. Section 5 describes our Markov models for characterizing WSN reliability and MTTF. Implementation and simulation of fault detection algorithms using ns-2 are presented in Section 6. Numerical results from our Markov modeling are presented in Section 7. Section 8 highlights research challenges and future research directions for the design and deployment of reliable and trustworthy WSNs. Finally, Section 9 concludes our study.

2. RELATED WORK

Despite fault detection and FT being well-studied research fields [Hopkins et al. 1978; Somani and Vaidya 1997; Avizienis and Laprie 1986], little work exists in WSN fault detection and FT. This section summarizes some of the previous works in literature related to fault detection and FT.

2.1. Fault Detection

Jiang [2009] proposed a DFD scheme that detected faulty sensor nodes by exchanging data and mutually testing neighboring nodes. Jian-Liang et al. [2007] proposed a weighted median fault detection scheme that used spatial correlations among the sensor measurements (e.g., temperature, humidity). Lee and Choi [2008] presented a DFD algorithm that identified faulty sensor nodes based on comparisons between neighboring sensor nodes' data. The DFD algorithm used time redundancy to tolerate transient faults in sensing and communication. Khilar and Mahapatra [2007] proposed a probabilistic approach to diagnose intermittent WSN faults. The simulation results indicated that the DFD algorithm's accuracy increased as the number of diagnostic rounds increased when neighboring sensor nodes exchanged measurements in each round.

Ding et al. [2005] proposed algorithms for faulty sensor identification and FT event boundary detection. The algorithms considered that both the faulty and normal sensors in an event region could generate abnormal readings (readings that deviate from a typical application-specific range). Krishnamachari and Iyengar [2004] proposed a distributed Bayesian algorithm for sensor fault detection and correction. The algorithm considered that measurement errors due to faulty equipment are likely to be uncorrelated. Wu et al. [2007] presented a fault detection scheme in which the fusion center (the node that aggregated data from different nodes) attempted to identify faulty sensor nodes through temporal sequences of received local decisions using a majority voting technique. Lo et al. [2013] presented a distributed, reference-free fault detection algorithm based on local pairwise verification between sensor nodes monitoring the same physical phenomenon. The authors observed that a linear relationship existed between the outputs of a pair of sensor nodes that could be exploited to detect faulty sensor nodes. Since the fault detection was done pairwise, the algorithm was able to conserve energy. Results revealed that the proposed algorithm could achieve a detection accuracy of 84% and a false alarm rate of 0.04%.

Miao et al. [2013] proposed agnostic diagnosis for detecting silent failures (i.e., failures with unknown types and symptoms). The proposed detection technique exploited the fact that a sensor node's metrics (e.g., radio on-time, number of packets transmitted in a time interval) exhibited certain correlation patterns, the violation of which indicated potential silent failures. The detection accuracy of the proposed scheme was close to 100% for small WSNs, whereas the detection accuracy decreased sharply and

the false alarm rate increased as the WSN size increased. The proposed technique required a sink node to collect data from all the sensor nodes in the WSN, which resulted in rapid energy depletion of the sink node and sensor nodes near the sink node. Furthermore, the fault detection latency of the proposed scheme was high due to its centralized nature.

There exists some work related to anomaly detection in WSNs. Bhargava and Raghuvanshi [2013] proposed a method for anomaly detection in WSNs based on S-transform (an extension of continuous wavelet transform). The S-transform extracted the features from sensor nodes' datasets, which were used to train a Support Vector Machine (SVM). The SVM was then used for the classification of normal and anomalous data. Results revealed that the proposed scheme's accuracy for data classification ranged from 78% to 94%. Salem et al. [2013] proposed an anomaly detection algorithm for medical WSNs. The proposed algorithm first classified instances of sensed patient attributes as normal and abnormal. The algorithm then used regression prediction to discern between a faulty sensor reading and a patient entering into a critical state. The results demonstrated the algorithm's ability to achieve a relatively low false alarm rate ($\sim$1%) and a good detection accuracy (the attained accuracy was not specified) for the medical WSN application.

2.2. Fault Tolerance

In work related to FT for WSNs, Koushanfar et al. [2002] proposed an FT scheme that provided backup for one type of sensor using another type of sensor, but they did not propose any FT model. Clouqueur et al. [2004] presented algorithms for collaborative target detection in the presence of faulty sensors. Chiang et al. [2004] designed system-level test interfaces for remote testing, repair, and software upgrade for sensor nodes. The authors evaluated the proposed test interfaces on Texas Instrument's MSP430 microcontroller-based sensor nodes. Experimental results revealed that the test interfaces with double, triple, and quadruple redundancy increased the WSN's availability.

Krasniewski et al. [2005] proposed a protocol to diagnose and mask arbitrary sensor node failures in an event-driven WSN. The protocol considered a clustered WSN with rotating cluster heads. The protocol assigned each sensor node a *trust index* to indicate the sensor node's track record in reporting past events correctly. The cluster head analyzed the event reports using the trust index and made event decisions. The authors simulated the protocol using ns-2 and observed that the protocol was able to detect events correctly even if more than 50% of the sensor nodes were compromised. Sun et al. [2012] introduced a trust mechanism to evaluate the trustworthiness of a sensor node and the sensor node's data. The authors defined *memory depth* to characterize the temporal correlation and *aggregation bandwidth* to characterize spatial correlation. A sensor node used memory depth to calculate the self-trustworthiness of the sensor node's data based on stored historical/past data and the current data. When a sensor node reported its sensed data to the aggregator, the aggregator's trustworthiness to the reported result depended both on the trustworthiness of the sensor node reporting data as well as on the trustworthiness of the sensor node to its sensed data. Subsequently, each aggregator reported the aggregated result and the aggregator's self-trust opinion to the upper layer aggregator progressively, and so on, until the aggregated result reached the sink node. Results demonstrated the effectiveness of the proposed mechanism for continuous media streaming and discrete data in wireless multimedia sensor networks.

There exists some work on providing FT in WSNs by deploying relay nodes and considering *connectivity* as an FT metric. Relay nodes communicate with sensor nodes, other relay nodes, and sink nodes to prolong WSN lifetime. Zhang et al. [2007]

developed approximation algorithms for determining a minimum number of relay nodes, along with the relay nodes' placement, to achieve certain connectivity requirements. Han et al. [2010] considered the problem of deploying relay nodes to provide FT in heterogeneous WSNs where sensor nodes had different transmission radii. They developed approximation algorithms for *full* and *partial* FT relay node placement. Whereas full FT relay node placement deployed a minimum number of relay nodes to establish disjoint paths between every sensor and/or relay node pair, partial FT relay node placement only considered sensor node pairs. Baldi et al. [2009] evaluated gossip algorithms, which are distributed algorithms that distribute the computational burden across all nodes, and these authors also considered connectivity as an FT metric.

Sen et al. [2006] introduced *region-based connectivity*: an FT metric defined as the minimum number of nodes within a region whose failure would disconnect the network. Alwan and Agarwal [2009] provided a survey of FT routing techniques in WSNs. Souza [2007] presented a framework for failure management in WSNs focusing on fault diagnosis and recovery techniques. The presented FT framework mitigated the failure propagation in a business (enterprise) environment by implementing different FT techniques.

2.3. WSN Reliability Modeling

There exists some work on FT WSN modeling. Cai et al. [2006] presented a reliability model to prolong the network lifetime and availability based on connectivity and coverage constraints. Zhu and Papavassiliou [2003] presented a model that characterized sensor connectivity and investigated the tradeoffs among sensor node connectivity, power consumption, and data rate. They also discussed the impact of sensor connectivity on system reliability. Vasar et al. [2009] presented Markov models for WSN reliability analysis. They presented a reliability comparison for various numbers of defective components' replacements with hot-standby redundant components. Xing and Michel [2006] presented WSN reliability and security modeling in an integrated fashion. Their modeling technique differentiated two types of WSN failures: security failures due to malicious intrusions and traditional failures due to malfunctioning components.

Moustapha and Selmic [2007] used recurrent neural networks to model sensor node dynamics for sensor fault detection. Their network model corresponded to the WSN topology such that the recurrent neural network input was taken from the modeled sensor node and neighboring sensor nodes. Kannan and Iyengar [2004] developed a game-theoretic model of reliable length- and energy-constrained routing in WSNs. They showed that optimal length-constrained paths could be computed in polynomial time in a distributed manner using geographic routing. Mukhopadhyay et al. [2009] presented a method that used sensor data properties to enable reliable data collection. The method consisted of predictive models based on temporal correlation in sensor data. They demonstrated that their method could handle multiple sources of errors simultaneously and could correct transient errors arising in sensor node hardware and wireless communication channels.

Even though DFD algorithms were proposed in the literature for detecting sensor faults, the fault detection was not leveraged to provide FT. There exists work in literature regarding FT in WSNs [Koushanfar et al. 2002; Zhang et al. 2007; Han et al. 2010; Sen et al. 2006; Cai et al. 2006], however, FT metrics such as reliability and MTTF were not investigated rigorously in the context of WSNs. Specifically, reliability and MTTF were not considered hierarchically at the sensor node, WSN cluster, and WSN levels. This hierarchical characterization of FT metrics facilitates a WSN designer to determine an appropriate WSN topology (i.e., number of clusters and sensor nodes with required FT capability in each cluster).

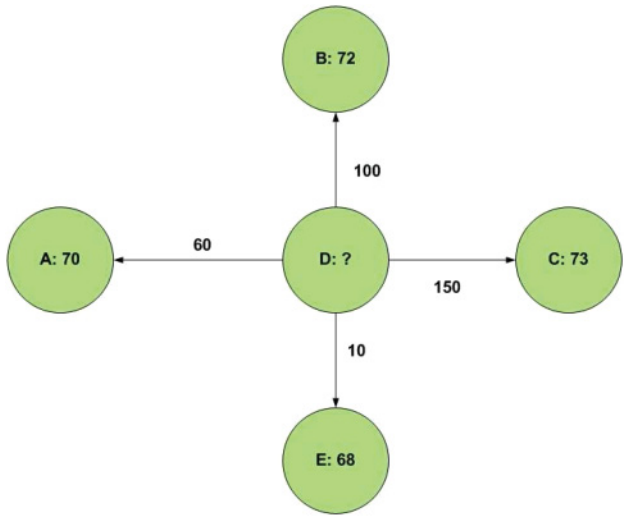


Fig. 2. Byzantine faulty behavior in WSNs.

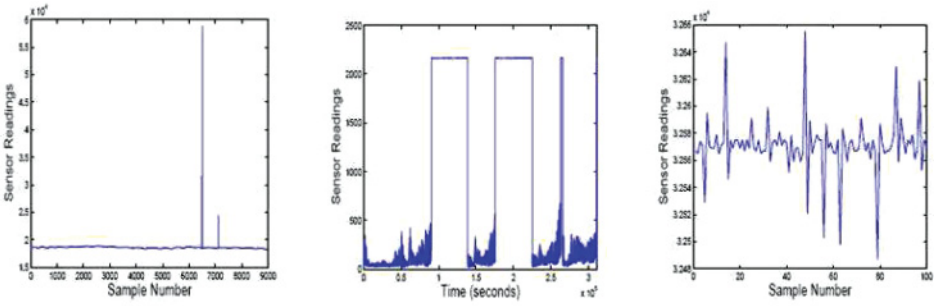
3. FAULT DIAGNOSIS IN WSNs

The sensor nodes in WSNs are often deployed in unattended and possibly hostile environments that make sensor nodes susceptible to failures. Faults in sensor nodes can occur at various levels, such as the processing unit, transceiver, storage unit, power unit, sensors, and actuators. Since faults are inevitable, it is imperative to determine faulty and fault-free sensor nodes in the WSN. Traditional fault diagnosis techniques developed for multiprocessor systems are not directly applicable to WSNs due to the sensor nodes' stringent resource constraints. This section describes sensor faults and then elaborates on fault diagnosis in WSNs.

3.1. Sensor Faults

The faults in sensor nodes include permanent faults, in which a node becomes dead, and Byzantine faults, in which the node behaves arbitrarily or maliciously. Figure 2 shows an example of Byzantine faults in which sensor nodes A, B, C, D, and E are equipped with temperature sensors and process the measured reading to give temperature in Fahrenheit. The sensor nodes send the sensed temperature readings to other neighboring nodes. The temperature sensor in node D behaves maliciously and sends inconsistent and arbitrary temperature values to other nodes (60, 100, 150, and 10 to A, B, C, and E, respectively). The rest of the sensor nodes send consistent values to other neighboring nodes. As a consequence, nonfaulty sensor nodes obtain different global information about the temperature in the region and may conclude on a false value for the overall temperature of the region. The fault detection algorithm needs to be robust to such inconsistent behavior that can jeopardize collaboration in the WSN [Clouqueur et al. 2004].

Several efforts have been made to classify malicious sensor behavior [Sharma et al. 2007; Ni et al. 2009]. Although different sensor fault taxonomies have been proposed, three particular behaviors are targeted by DFD methods. *Outlier faults* occur when a sensor signal spikes in value (Figure 3(a)) and can be detected by comparing sensor readings to previous readings or neighboring sensor nodes' readings. The *stuck-at faults* occur when the output remains at a constant level (Figure 3(b)). The *noisy faults* occur when the signal-to-noise ratio of the sensor's output is low, resulting in random data



(a) Outlier faults (b) Stuck-at faults (c) Noisy faults

Fig. 3. Various types of sensor faults [Sharma et al. 2007].

(Figure 3(c)). An effective fault detection algorithm must be able to identify the broadest possible range of malicious output while minimizing fault positives. Because the broad range of sensor failures results in a mixed detection rate for different algorithms, accurate simulation is important for studying sensor failures [Sharma et al. 2007].

3.2. Taxonomy for Fault Diagnosis Techniques

A *fault diagnosis system* is a monitoring system that detects faulty sensor nodes and their location in the WSN. Fault diagnosis techniques are classified based on the nature of tests, correlation between sensor readings, and characteristics of sensor nodes [Mahapatro and Khilar 2013]. The key component of a fault diagnosis system is a *fault detection algorithm*, and key terminology includes correctness, latency, detection accuracy, and false alarm rate. A fault diagnosis is *correct* if no fault-free sensor nodes are mistakenly diagnosed as faulty. *Latency* is defined as the time elapsed since the appearance of the fault to the isolation of the faulty sensor node. *Detection accuracy* is defined as the ratio of the number of faulty sensor nodes detected to the actual number of faulty sensor nodes in the network. The accuracy of the fault detection algorithm is a crucial factor in maintaining reliable WSN operation. The ratio of the number of fault-free sensor nodes diagnosed as faulty to the actual number of fault-free sensor nodes is the *false alarm rate*. Fault detection techniques for WSN can be broadly categorized into *centralized* and *distributed* approaches [Mahapatro and Khilar 2013].

3.2.1. Centralized Approaches. In a centralized approach, a geographically or logically centralized arbiter (i.e., sink node or base station) with higher computational power, larger memory size, and a greater energy supply than ordinary sensor nodes is responsible for fault detection and diagnosis of the overall WSN. The sink node periodically sends diagnostic queries into the network to obtain the state of the individual sensor nodes. The sink node then analyzes the received diagnostic response messages to determine faulty and fault-free sensor nodes. Centralized approaches are accurate; however, these approaches cannot be scaled for large-scale WSNs. The scalability of centralized approaches is limited because it is expensive for the sink node to accumulate and analyze diagnostic information from all the sensor nodes in the WSN. Furthermore, centralized approaches lead to rapid energy depletion in certain regions of the network, particularly in nodes closer to the sink node. The energy depletion in sensor nodes closer to the sink node causes network partitions, which results in unmonitored areas in the WSN. Moreover, the detection latency of centralized approaches is large due to multihop communication. Due to these limitations of centralized approaches, distributed approaches are highly preferable in WSNs.

3.2.2. Distributed Approaches. In distributed approaches, each sensor node executes the fault detection algorithm and generates a localized fault view. The localized fault view is a sensor node's view regarding the fault states of the sensor node's one-hop neighbors. This localized fault 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 network. Distributed approaches conserve the sensor nodes' energy and consequently prolong the WSN lifetime. Distributed approaches can be classified into various types [Mahapatro and Khilar 2013]:

Test-Based Approaches: In test-based approaches, different tests (tasks) are assigned to sensor nodes, and faulty sensor nodes are identified based on test results. Test-based approaches are further classified into *invalidation-based* and *comparison-based* approaches.

In *invalidation-based* approaches, every sensor node tests a set of sensor nodes. Each sensor node then passes the test results to other sensor nodes based on which a consensus is made on the faulty sensor nodes. Each sensor node must be tested by at least t one-hop neighbors to achieve a t-diagnosability. The greater the t, the greater the message overhead. In most invalidation-based approaches, the number of faulty sensor nodes that can be detected is upper bounded by t. The detection accuracy of these approaches is close to 100%, and the false alarm rate is close to zero; however, the detection latency exhibited by most of these approaches is high.

In *comparison-based* approaches, different tests (tasks) are assigned to a pair of sensor nodes, and the results of these tasks are compared. The agreement and disagreement between these results form the basis for diagnosing faulty sensor nodes. Comparison-based approaches have comparatively less message and time complexity overhead than do invalidation-based approaches.

Neighbor Coordination Approaches: In neighbor coordination approaches, one-hop neighbors coordinate with each other to determine whether to disregard their own sensor readings based on the neighboring sensors readings or on the weights based on physical distances from the event and trustworthiness of the sensor node's measurements. Neighbor coordination approaches are further categorized into *majority voting* and *weighted majority voting* approaches.

Majority voting approaches exploit the fact that the faulty measurements are uncorrelated whereas nonfaulty measurements are spatially correlated. For example, assume a sensor node s_i is a neighbor of s_j, and r_i and r_j are the readings for s_i and s_j, respectively. Sensor readings r_i and r_j are similar when $|r_i - rj| < \delta$ where δ is application dependent and typically a small number. Majority voting techniques can provide good detection accuracy and low false alarm rates. However, majority voting techniques are *topology*-dependent because the effectiveness of these approaches depends on the node degree in the WSN.

Weighted majority voting approaches weigh the sensing measurements based on properties such as physical distance from the event and trustworthiness of the measurements. Unlike simple majority voting, these weighted measurements are used to decide the state of a sensor node (i.e., faulty or nonfaulty) in a WSN. Weighted majority voting approaches are computationally more expensive than simple majority voting approaches. Weighted majority voting approaches are also topology-dependent, like simple majority voting approaches (i.e., majority voting accuracy degrades drastically in sparse networks).

Hierarchical Detection: In hierarchical detection, a spanning tree with the sink node as the root node is constructed that spans all fault-free sensor nodes in the WSN. This spanning tree is then used to determine faults at each level of the tree. The fault results

are disseminated across the spanning tree such that each node in the network correctly diagnoses the fault states of all the sensor nodes in the WSN. The detection latency in a hierarchical detection approach is high because the diagnosis process is started either by the sink node or the leaf nodes and requires multihop communication of diagnostic messages. Furthermore, similar to the centralized approach, the hierarchical detection approach leads to rapid energy depletion in certain regions of the WSN, particularly in sensor nodes close to the sink.

Node Self-Detection: In sensor node self-detection, the sensor node detects its own status (faulty or nonfaulty) with the help of additional hardware incorporated in the sensor node. Since the sensor node self-detection approach requires additional hardware, the approach increases hardware complexity, weight, and energy consumption.

Clustering-Based Approaches: Clustering-based approaches divide the overall WSN into clusters, which are groups of sensor nodes located geographically close to each other. Each cluster head executes a fault detection algorithm in the cluster head's group using a centralized or distributed approach. Clustering-based diagnostic approaches are communication-efficient; however, the cluster head requires more energy for leading the diagnostic process.

Watchdog Approaches: In watchdog approaches, a sensor node monitors whether the sensor node's packets are forwarded by the sensor node's one-hop neighbor by overhearing the communication channel. If a sensor node's neighboring node does not forward a packet within a certain time, the neighboring node is viewed as misbehaving. When the misbehaving rate exceeds a threshold, the misbehaving node is diagnosed as faulty. The source node then sends packets along other routes that avoid the diagnosed faulty node.

Probabilistic Approaches: Probabilistic approaches exploit the fact that sensor failure probability for different sensor nodes is not identical in a time interval that is not infinitesimally small. Probabilistic approaches normally leverage Bayesian or other statistical methods for fault diagnosis.

Event-Driven Diagnosis: In event-based WSNs (as opposed to data- or query-driven WSNs), sensor nodes only report events of interest to the sink node in a timely manner. Fault diagnosis in event-driven WSNs requires special consideration for fault-event disambiguation because an event also causes abnormal data that could be interpreted as a faulty measurement to be sensed by the the sensor nodes. Event detection approaches work effectively in dense networks with relatively low fault probability of sensor nodes. Most of the event detection approaches fail in distinguishing between events and faults if faults are located at the event boundary.

4. DISTRIBUTED FAULT DETECTION ALGORITHMS

DFD algorithms are those in which each sensor node executes a fault detection algorithm to diagnose the status of the sensor node's neighboring sensor nodes and to obtain a localized fault view (Section 3.2.2). The localized fault view is then shared with neighboring sensor nodes to reach a consensus on faulty sensor nodes. We analyze different fault detection algorithms [Chen et al. 2006; Ding et al. 2005; Jiang 2009; Jian-Liang et al. 2007; Krishnamachari and Iyengar 2004]; however, we present two fault detection algorithms that explain the fault detection algorithms' functionalities:

4.1. Fault Detection Algorithm 1: The Chen Algorithm

The first algorithm, which we refer to as the Chen algorithm [2006], relies on simple majority counts to determine faults. The approach is practical because sensor nodes

Table I. Summary of Notations Used in the DFD Algorithms

Notation	Description
S	set of all the sensors involved in the DFD algorithm
$N(S_i)$	set of neighbors of sensor node S_i
x_i	measurement of S_i
d_{ij}^t	measurement difference between S_i and S_j at time t (i.e., $d_{ij}^t = x_i^t - x_j^t$)
$\triangle t_l$	measurement time difference, $t_{l+1} - t_l$
$\triangle d_{ij}^{\triangle t_l}$	measurement difference between S_i and S_j from time t_l to t_{l+1} (i.e., $\triangle d_{ij}^{\triangle t_l} = d_{ij}^{t_{l+1}} - d_{ij}^{t_l}$)
c_{ij}	test between S_i and S_j, $c_{ij} \in \{0, 1\}$, $c_{ij} = c_{ji}$
θ_1 & θ_2	two predefined thresholds
T_i	tendency value of a sensor: $T_i \in$ {LG, LF, GD, FT}

ALGORITHM 1: The Chen algorithm [2006] for WSN fault detection.

Input: Sensors S_i with measurements
Output: Tendency of good and faulty sensors

Step 1: For each sensor node S_i, compute the bit vector $C_i\{S\}$, where a 1 means S_i and S_j are consistent;
Let $S_{ij}(t) = |S_i(t) - S_j(t)|$;
$C_i\{S_j\} = 1$ if $S_{ij}(t) > \theta_1$ and $|S_{ij}(t) - S_{ij}(t+1)| > \theta_2$;

Step 2: Calculate the tendency of each sensor:
$T_i =$ Likely Good (LG) if $\sum C_i > \frac{N_S}{2}$, else Likely Faulty (LF) where N_s is the number of neighboring sensor nodes;

Step 3: Determine if the sensor node is good:
$T_i =$ Good (GD) if $N_C - \overline{N_C} > \frac{N_S}{2}$;
where N_C and $\overline{N_C}$ denote the number of consistent and inconsistent neighbors respectively;

Step 4: If a node is not sure, find Good or Faulty neighbors (set N). For each sensor:
if $S_i \in$ {LG, LF}, then for each $N_j \in N$:
$T_i =$ GD if ($T_j =$ GD and $C_i\{N_j\} = 1$) or ($T_j =$ FT and $C_i\{N_j\} = 0$);
$T_i =$ FT if ($T_j =$ FT and $C_i\{N_j\} = 1$) or ($T_j =$ GD and $C_i\{N_j\} = 0$);
Repeat until all sensor nodes have a state of Good (GD) or Faulty (FT);

have limited computing ability, and hence lightweight calculations are preferred over intensive calculations. Each sensor node $S_i \in S$ can exist in four states: Good (GD), Faulty (FT), Likely Good (LG), or Likely Faulty (LF). A sensor node in likely states is unsure if the sensor node is faulty and uses the sensor nodes neighbors' states to finalize the sensor node's decision. The algorithm's implementation complexity is low, and the probability of correct diagnosis is high.

Table I lists the notations used in the Chen algorithm [2006]. The neighboring sensor nodes in a WSN are those that are within the transmission range of each other. Each sensor node sends the sensor node's sensed values to the sensor node's neighboring sensor nodes. A sensor S_i generates test result c_{ij} based on the sensor node's neighbor S_j's measurements using variables d_{ij}, $\triangle d_{ij}^{\triangle t_l}$ and threshold values θ_1 and θ_2. The sensors S_i and S_j are both good or both faulty if $c_{ij} = 0$; however, the sensor nodes have a different good or faulty status if $c_{ij} = 1$. Each sensor node sends the sensor node's tendency values (i.e., estimated state) to all the sensor node's neighboring sensor nodes. The test values from neighboring sensors determine the tendency value of sensors to be LG or LF, and the number of LG sensors with the same results determines whether the

sensors are GD or FT. Precisely, $\forall S_j \in N(S_i)$ and $T_j = LG$, $\sum(1-c_{ij}) - \sum c_{ij} = \sum(1-2c_{ij})$ must be greater than or equal to $\lceil |N(S_i)/2| \rceil$ for S_i to be identified as good. The algorithm diagnoses a good sensor S_i as GD in the first round if the sensor node has less than $k/4$ bad neighbors. The Chen algorithm's important steps are depicted in Algorithm 1.

4.2. Fault Detection Algorithm 2: The Ding Algorithm

The second algorithm, which we refer to as the Ding algorithm [2005], is a WSN fault detection algorithm with a low computational overhead. The results indicate that the algorithm can identify faulty sensors with relatively high accuracy even when the sensor failure probability is as high as 20%. The Ding algorithm's important steps are shown in Algorithm 2.

ALGORITHM 2: The Ding algorithm [2005] for WSN fault detection.

Input: Sensors S_i with measurements
Output: Set of faulty sensors $\mathcal{F}$

Step 1: Identify the neighboring sensor nodes of sensor node S_i. For each sensor node S_i perform Steps 2–4;

Step 2: Compute d_i for each sensor node S_i using Equation (1);

Step 3: Compute y_i for each sensor node S_i using Equation (4);

Step 4: If $|y_i| \geq \theta$, assign S_i to $\mathcal{F}$, otherwise treat S_i as a normal sensor;

The algorithm compares the sensor node S_i's reading with the sensor node's k neighbors $S_{i1}, S_{i2}, \ldots, S_{ik}$ with measurements $x_1^{(i)}, x_2^{(i)}, \ldots, x_k^{(i)}$. The comparison is done by comparing x_i with the median med_i of $\{x_1^{(i)}, x_2^{(i)}, \ldots, x_k^{(i)}\}$; that is,

$$d_i = x_i - \text{med}_i. \tag{1}$$

If there are n sensors in total, the algorithm computes $d_i \, \forall n$ (i.e., $D = \{d_1, d_2, \ldots, d_i, \ldots, d_n\}$). The mean $\hat{\mu}$ of D is given by Ding et al. [2005]:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} d_i. \tag{2}$$

The standard deviation $\hat{\sigma}$ of D is given as [Ding et al. 2005]:

$$\hat{\sigma} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (d_i - \hat{\mu})^2}. \tag{3}$$

Standardizing the dataset D yields $Y = \{y_1, y_2, \ldots, y_i, \ldots, y_n\}$; that is:

$$\begin{aligned} y_1 &= \frac{d_1 - \hat{\mu}}{\hat{\sigma}} \\ y_2 &= \frac{d_2 - \hat{\mu}}{\hat{\sigma}} \\ &\vdots \\ y_i &= \frac{d_i - \hat{\mu}}{\hat{\sigma}} \\ &\vdots \\ y_n &= \frac{d_n - \hat{\mu}}{\hat{\sigma}}. \end{aligned} \tag{4}$$

Table II. Summary of Notations Used in Our Markov Models

Notation	Description
n	total number of sensors in a WSN
c	coverage factor
p	cumulative sensor failure probability
λ_s	sensor failure rate
t_s	time over which sensor failure rate is specified
$P_i(t)$	probability of being in state i at time t
$R_{s_d}(t)$	reliability of FT (duplex) sensor node
MTTF_{s_d}	mean time to failure of an FT sensor node
k	average number of neighbor sensor nodes
$\lambda_{s_d}(k)$	FT sensor node failure rate with k neighbors
$R_c(t)$	WSN cluster reliability
$\lambda_c(n)$	WSN cluster failure rate with n sensor nodes
N	number of clusters in the WSN
$R_{wsn}(t)$	WSN reliability

If $|y_i| \geq \theta$, then the Ding algorithm detects S_i as faulty where $\theta > 1$ is a predefined threshold value. Mathematically, $S_i \in \mathcal{F}$ if $|y_i| \geq \theta$, where $\mathcal{F}$ denotes the set of sensors claimed as faulty.

Section 6 discusses the implementation and effectiveness, particularly in detection accuracy, that is used implicitly in our Markov model as the coverage factor c, of these fault detection algorithms.

5. FAULT-TOLERANT MARKOV MODELS

In this section, we present our Markov models for FT WSNs. Our Markov models are comprehensive and hierarchically encompass the sensor node, WSN cluster, and WSN levels (Figure 1). We adopt a bottom-up paradigm in our modeling by first developing a sensor node model and determining the model's reliability and MTTF [Munir and Gordon-Ross 2011]. The sensor node MTTF gives the average sensor node failure rate, which is utilized in the WSN cluster model. The WSN cluster model gives the WSN cluster reliability and MTTF, which determines the WSN cluster average failure rate. The WSN cluster average failure rate is then utilized in the WSN model to determine WSN reliability and MTTF. Our bottom-up approach enables WSN reliability and MTTF characterization by leveraging sensor node and WSN cluster models. We note that some WSN architectures do not employ a cluster-based hierarchy due to the additional energy cost associated with cluster formation and cluster head election. However, our modeling approach is equally applicable for WSN architectures without a cluster-based hierarchy by assuming that the WSN is composed of just one cluster, where the cluster head is the sink node. For clarity, Table II summarizes important notations used in our Markov models.

5.1. Fault-Tolerance Parameters

Fault-tolerance in WSNs is highly dependent on fault diagnosis in WSNs (Section 3). In this section, we characterize the FT parameters by exploiting the synergy between fault detection and FT in WSNs. The FT parameters leveraged in our Markov model are the *coverage factor* and *sensor failure rate*.

5.1.1. Coverage Factor. The coverage factor c is defined as the probability that the faulty active sensor is correctly diagnosed, disconnected, and replaced by a good inactive spare sensor. The c estimation is critical in an FT WSN model and can be determined by:

$$c = c_k - c_c, \tag{5}$$

where c_k denotes the accuracy of the fault detection algorithm in diagnosing faulty sensors and c_c denotes the probability of an unsuccessful replacement of the identified faulty sensor with the good spare sensor. Whereas c_c depends on the sensor switching circuitry and is usually a constant, c_k's estimation is challenging because different fault detection algorithms have different accuracies. We analyze different fault detection algorithms [Ding et al. 2005; Jiang 2009; Jian-Liang et al. 2007; Krishnamachari and Iyengar 2004] and observe that the accuracy of a fault detection algorithm depends on the average number of sensor node neighbors k and the cumulative probability of sensor failure p. We model $c_k : c_k \leq 1$ with the empirical relation:

$$c_k = \frac{k \times (1-p)}{k^{(k/M(p))^{1/M(p)}} + (1 - k/M(p))^k}. \tag{6}$$

where $M(p)$ is a function of p and denotes an adjustment parameter that may correspond loosely to the desired average number of neighboring sensor nodes required to achieve a good fault detection accuracy for a given p. We derived Equation (6) by experimenting with the relationship among the fault detection algorithms' parameters (i.e., the average number of sensor node neighbors), sensor failure probability, and the fault detection algorithms' accuracy. We point out that whereas Equation (6) provides a good estimate of c_k in general for any fault detection algorithm, exact c_k for a particular fault detection algorithm can be derived from the algorithm's mathematical model. Equation (6) approximates the relationship among a fault detection algorithm's parameters to obtain the fault detection algorithm's accuracy (in case the accuracy is unknown). In practice, a fault detection algorithm's accuracy should be determined accurately by the algorithm's designer; this will alleviate the need for an approximation (as in Equation (6)). To clarify further, we point out that our Markov models are independent of c_k's determination methodology and are equally applicable to any c_k value.

5.1.2. Sensor Failure Rate. The sensor failure rate can be represented by exponential distribution with a failure rate of λ_s over the period t_s [Johnson et al. 1994]. The exponential distribution, which has a property of constant failure rate, is a good model for the long, flat, intrinsic failure portion of the *bathtub curve*. The exponential distribution is applicable to a variety of practical situations since many embedded components and systems spend most of their lifetimes in the flat portion of the bathtub curve. Furthermore, any failure rate curve can be approximated by piecewise exponential distribution segments patched together. Each exponential distribution segment in the overall approximation specifies a constant failure rate over a small time unit (e.g., daily, weekly, or monthly) that is the average of the failure rates during the respective time duration. The constant failure rate assignment for each exponential distribution segment is justifiable because many natural phenomena have a constant failure rate (or occurrence rate) property (e.g., the arrival rate of cosmic ray alpha particles). The failure rate curve approximation by piecewise exponential distributions is analogous to a curve approximation by piecewise straight-line segments.

The exponential model works well for those interarrival times where the total number of events in a given time period is given by the Poisson distribution. When these events trigger failures, the exponential lifetime distribution model naturally applies [NIST 2011]. The Cumulative Distribution Function (CDF) for the sensors with an exponentially distributed failure rate is:

$$F_s(t_s; \lambda_s) = p = 1 - exp(-\lambda_s t_s), \tag{7}$$

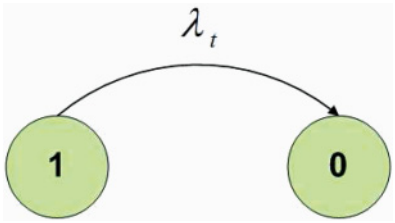


Fig. 4. A Non-FT (NFT) sensor node Markov model.

where p denotes the cumulative probability of sensor failure (for simplicity) and t_s signifies the time over which p is specified. Solving Equation (7) for λ_s gives:

$$\lambda_s = -\frac{1}{t_s} ln(1-p). \tag{8}$$

5.2. Fault-Tolerant Sensor Node Model

As a base case, we describe a Non-FT (NFT) sensor node Markov model (Figure 4) containing one sensor (a temperature sensor in this case, but the sensor type is arbitrary). The NFT sensor node model consists of two states: state 1 (good state) and state 0 (failed state). The NFT sensor node fails when the node transitions from state 1 to state 0 due to a sensor failure. The differential equations describing the NFT sensor node Markov model are:

$$\begin{aligned} P_1'(t) &= -\lambda_t P_1(t) \\ P_0'(t) &= \lambda_t P_1(t), \end{aligned} \tag{9}$$

where $P_i(t)$ denotes the probability that the sensor node will be in state i at time t, and $P_i^{'}(t)$ represents the first order derivative of $P_i(t)$. λ_t represents the failure rate of an active temperature sensor.

Solving Equation (9) with the initial conditions $P_1(0) = 1$ and $P_0(0) = 0$ yields:

$$\begin{aligned} P_1(t) &= e^{-\lambda_t t} \\ P_0(t) &= 1 - e^{-\lambda_t t}. \end{aligned} \tag{10}$$

The reliability of the NFT sensor node is:

$$R_s(t) = 1 - P_0(t) = P_1(t) = e^{-\lambda_t t}. \tag{11}$$

The MTTF of the NFT sensor node is:

$$\text{MTTF}_s = \int_0^\infty R_s(t)\,dt = \frac{1}{\lambda_t}. \tag{12}$$

The average failure rate of the NFT sensor node is:

$$\lambda_s = \frac{1}{\text{MTTF}_s} = \lambda_t. \tag{13}$$

Because sensors have comparatively higher fault rates than other components [Moustapha and Selmic 2007; Sharma et al. 2007; Koushanfar et al. 2002], we propose an FT duplex sensor node model consisting of one active sensor and one inactive spare sensor. Using TMR [Koren and Krishna 2007] for FT is a possible scenario, but we consider a duplex sensor node model to minimize the additional cost because the additive cost of spare sensors can be prohibitive for large WSNs. In addition, a duplex model limits the increase in sensor node size. We point out that our modeling methodology can be extended to sensor nodes with NMR.

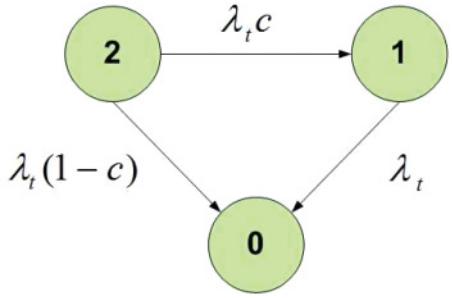


Fig. 5. FT sensor node Markov model [Munir and Gordon-Ross 2011].

In our duplex sensor node, the inactive sensor becomes active only when the active sensor is declared faulty by the fault detection algorithm. We refer to our duplex sensor node as an FT sensor node, whose Markov model is depicted in Figure 5. The states in the Markov model represent the number of good sensors. The differential equations describing the FT sensor node Markov model are:

$$\begin{aligned} P_2'(t) &= -\lambda_t P_2(t) \\ P_1'(t) &= \lambda_t c P_2(t) - \lambda_t P_1(t) \\ P_0'(t) &= \lambda_t(1-c)P_2(t) + \lambda_t P_1(t), \end{aligned} \tag{14}$$

where $P_i(t)$ denotes the probability that the sensor node will be in state i at time t, and $P_i'(t)$ represents the first-order derivative of $P_i(t)$. λ_t represents the failure rate of an active temperature sensor, and $c\lambda_t$ is the rate at which recoverable failure occurs. The probability that the sensor failure cannot be recovered is $(1-c)$, and the rate at which unrecoverable failure occurs is $(1-c)\lambda_t$.

Solving Equation (14) with the initial conditions $P_2(0) = 1$, $P_1(0) = 0$, and $P_0(0) = 0$ yields:

$$\begin{aligned} P_2(t) &= e^{-\lambda_t t} \\ P_1(t) &= c\lambda_t t e^{-\lambda_t t} \\ P_0(t) &= 1 - P_1(t) - P_2(t). \end{aligned} \tag{15}$$

The reliability of the FT sensor node is:

$$R_{s_d}(t) = 1 - P_0(t) = P_2(t) + P_1(t) = e^{-\lambda_t t} + c\lambda_t t e^{-\lambda_t t}, \tag{16}$$

where subscript d in $R_{s_d}(t)$ stands for duplex. The MTTF of the FT sensor node is:

$$\text{MTTF}_{s_d} = \int_0^\infty R_{s_d}(t)\,dt = \frac{1}{\lambda_t} + \frac{c}{\lambda_t}. \tag{17}$$

The average failure rate of the FT sensor node depends on k (since the fault detection algorithm's accuracy depends on k (Section 5.1)):

$$\lambda_{s_d(k)} = \frac{1}{\text{MTTF}_{s_d(k)}}, \tag{18}$$

where $\lambda_{s_d(k)}$ and $\text{MTTF}_{s_d(k)}$ denote, respectively, the failure rate and MTTF of an FT sensor node with k neighbors.

5.3. Fault-Tolerant WSN Cluster Model

A typical WSN consists of many clusters (Figure 1), and we assume for our model that all nodes in a cluster are neighbors to each other (either one- or two-hop neighbors

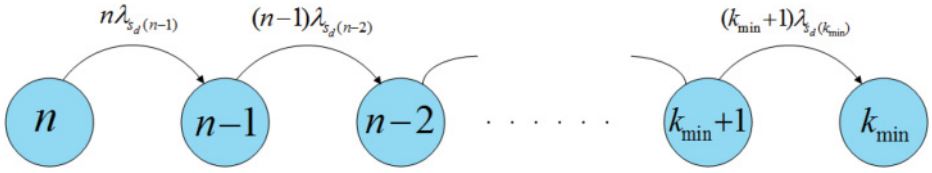


Fig. 6. WSN cluster Markov model [Munir and Gordon-Ross 2011].

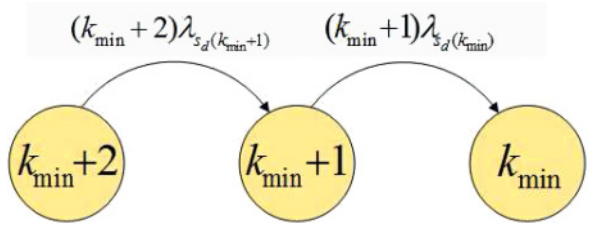


Fig. 7. WSN cluster Markov model with three states [Munir and Gordon-Ross 2011].

depending on the topology). We note that our model does not impose any specific topology on a sensor network cluster. If the average number of nodes in a cluster is n, then the average number of neighboring nodes per sensor node is $k = n - 1$. Figure 6 depicts our Markov model for a WSN cluster. We assume that a cluster fails (i.e., fails to perform its assigned application task) if the number of alive (nonfaulty) sensor nodes in the cluster reduces to k_{min}. The differential equations describing the WSN cluster Markov model are:

$$\begin{aligned} P'_n(t) &= -n\lambda_{s_d(n-1)}P_n(t) \\ P'_{n-1}(t) &= n\lambda_{s_d(n-1)}P_n(t) - (n-1)\lambda_{s_d(n-2)}P_{n-1}(t) \\ &\vdots \\ P'_{k_{min}}(t) &= (k_{min}+1)\lambda_{s_d(k_{min})}P_{k_{min}+1}(t), \end{aligned} \tag{19}$$

where $\lambda_{s_d(n-1)}$, $\lambda_{s_d(n-2)}$, and $\lambda_{s_d(k_{min})}$ represent the FT (duplex) sensor node failure rate (Equation (18)) when the average number of neighboring sensor nodes are $n-1$, $n-2$, and k_{min}, respectively. For mathematical tractability and a closed-form solution, we analyze a special (simple) case when $n = k_{min} + 2$, which reduces the Markov model to three states (Figure 7). The differential equations describing the WSN cluster Markov model when $n = k_{min} + 2$ are:

$$\begin{aligned} P'_{k_{min}+2}(t) &= -(k_{min}+2)\lambda_{s_d(k_{min}+1)}P_{k_{min}+2}(t) \\ P'_{k_{min}+1}(t) &= (k_{min}+2)\lambda_{s_d(k_{min}+1)}P_{k_{min}+2}(t) \\ &\quad -(k_{min}+1)\lambda_{s_d(k_{min})}P_{k_{min}+1}(t) \\ P'_{k_{min}}(t) &= (k_{min}+1)\lambda_{s_d(k_{min})}P_{k_{min}+1}(t). \end{aligned} \tag{20}$$

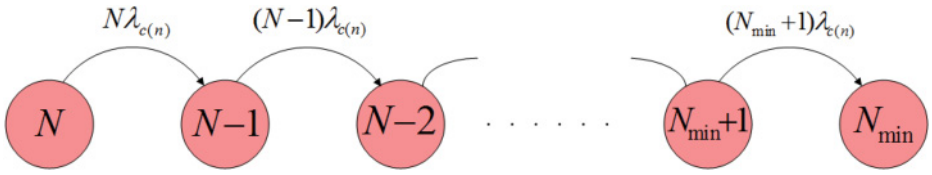


Fig. 8. WSN Markov model [Munir and Gordon-Ross 2011].

Solving Equation (20) with the initial conditions $P_{k_{min}+2}(0) = 1$, $P_{k_{min}+1}(0) = 0$, and $P_{k_{min}}(0) = 0$ yields:

$$\begin{aligned} P_{k_{min}+2}(t) &= e^{-(k_m+2)\lambda_{s_d(k_m+1)}t} \\ P_{k_{min}+1}(t) &= \frac{(k_m+2)\lambda_{s_d(k_m+1)}e^{-(k_m+2)\lambda_{s_d(k_m+1)}t}}{(k_m+1)\lambda_{s_d(k_m)} - (k_m+2)\lambda_{s_d(k_m+1)}} \\ &\quad + \frac{(k_m+2)\lambda_{s_d(k_m+1)}e^{-(k_m+1)\lambda_{s_d(k_m)}t}}{(k_m+2)\lambda_{s_d(k_m+1)} - (k_m+1)\lambda_{s_d(k_m)}} \\ P_{k_{min}}(t) &= 1 - P_{k_{min}+1}(t) - P_{k_{min}+2}(t), \end{aligned} \tag{21}$$

where k_{min} denotes k_m in Equation (21) for conciseness. The reliability of the WSN cluster is:

$$\begin{aligned} R_c(t) &= 1 - P_{k_{min}}(t) \\ &= e^{-(k_{min}+2)\lambda_{s_d(k_{min}+1)}t} \\ &\quad + \frac{(k_{min}+2)\lambda_{s_d(k_{min}+1)}e^{-(k_{min}+2)\lambda_{s_d(k_{min}+1)}t}}{(k_{min}+1)\lambda_{s_d(k_{min})} - (k_{min}+2)\lambda_{s_d(k_{min}+1)}} \\ &\quad + \frac{(k_{min}+2)\lambda_{s_d(k_{min}+1)}e^{-(k_{min}+1)\lambda_{s_d(k_{min})}t}}{(k_{min}+2)\lambda_{s_d(k_{min}+1)} - (k_{min}+1)\lambda_{s_d(k_{min})}}. \end{aligned} \tag{22}$$

The MTTF of the WSN cluster is:

$$\begin{aligned} \text{MTTF}_c &= \int_0^\infty R_c(t)\,dt = \frac{1}{(k_m+2)\lambda_{s_d(k_m+1)}} \\ &\quad + \frac{1}{(k_m+1)\lambda_{s_d(k_m)} - (k_m+2)\lambda_{s_d(k_m+1)}} \\ &\quad + \frac{(k_m+2)\lambda_{s_d(k_m+1)}}{(k_m+2)(k_m+2)\lambda_{s_d(k_m)}\lambda_{s_d(k_m+1)} - (k_m+1)^2\lambda^2_{s_d(k_m)}}, \end{aligned} \tag{23}$$

where k_{min} denotes k_m in Equation (23) for conciseness. The average failure rate of the cluster $\lambda_{c(n)}$ depends on the average number of nodes in the cluster n at deployment time:

$$\lambda_{c(n)} = \frac{1}{\text{MTTF}_{c(n)}}, \tag{24}$$

where $\text{MTTF}_{c(n)}$ denotes the MTTF of a WSN cluster of n sensor nodes.

5.4. Fault-Tolerant WSN Model

A typical WSN consists of $N = n_s/n$ clusters, where n_s denotes the total number of sensor nodes in the WSN, and n denotes the average number of nodes in a cluster. Figure 8 depicts our WSN Markov model. We assume that the WSN fails to perform

the WSN's assigned task when the number of alive clusters reduces to N_{min}. The differential equations describing the WSN Markov model are:

$$\begin{aligned} P'_N(t) &= -N\lambda_{c(n)} \\ P'_{N-1}(t) &= N\lambda_{c(n)}P_N(t) - (N-1)\lambda_{c(n)}P_{N-1}(t) \\ &\vdots \\ P'_{N_{min}}(t) &= (N_{min}+1)\lambda_{c(n)}P_{N_{min}+1}(t), \end{aligned} \tag{25}$$

where $\lambda_{c(n)}$ represents the average cluster failure rate (Equation (24)) when the cluster contains n sensor nodes at deployment time. For mathematical tractability, we analyze a special (simple) case when $N = N_{min} + 2$, which reduces the Markov model to three states. The differential equations describing the WSN Markov model when $N = N_{min}+2$ are:

$$\begin{aligned} P'_{N_{min}+2}(t) &= -(N_{min}+2)\lambda_{c(n)}P_{N_{min}+2}(t) \\ P'_{N_{min}+1}(t) &= (N_{min}+2)\lambda_{c(n)}P_{N_{min}+2}(t) \\ &\quad -(N_{min}+1)\lambda_{c(n)}P_{N_{min}+1}(t) \\ P'_{N_{min}}(t) &= (N_{min}+1)\lambda_{c(n)}P_{N_{min}+1}(t). \end{aligned} \tag{26}$$

Solving Equation (26) with the initial conditions $P_{N_{min}+2}(0) = 1$, $P_{N_{min}+1}(0) = 0$, and $P_{N_{min}}(0) = 0$ yields:

$$\begin{aligned} P_{N_{min}+2}(t) &= e^{-(N_{min}+2)\lambda_{c(n)}t} \\ P_{N_{min}+1}(t) &= (N_{min}+2)\lambda_{c(n)} \\ &\quad \times [e^{-(N_{min}+1)\lambda_{c(n)}t} - e^{-(N_{min}+2)\lambda_{c(n)}t}] \\ P_{N_{min}}(t) &= 1 - P_{N_{min}+1}(t) - P_{N_{min}+2}(t). \end{aligned} \tag{27}$$

The WSN reliability is:

$$\begin{aligned} R_{wsn}(t) &= 1 - P_{N_{min}}(t) \\ &= e^{-(N_{min}+2)\lambda_{c(n)}t} + (N_{min}+2)\lambda_{c(n)} \\ &\quad \times [e^{-(N_{min}+1)\lambda_{c(n)}t} - e^{-(N_{min}+2)\lambda_{c(n)}t}]. \end{aligned} \tag{28}$$

The WSN MTTF when $N = N_{min} + 2$ is:

$$\begin{aligned} \text{MTTF}_{wsn} &= \int_0^\infty R_{wsn}(t)\,dt \\ &= \frac{1}{(N_{min}+2)\lambda_{c(n)}} + \frac{N_{min}+2}{N_{min}+1} - 1. \end{aligned} \tag{29}$$

Discussion: Our hierarchical Markov modeling exploits the relationship between failure rates at the sensor component, sensor node, WSN cluster, and WSN levels. Markov modeling of sensor nodes (Section 5.2) takes the failure rate of the sensor node's components λ_t (e.g., temperature sensor) as input and then derives the failure rate of the sensor node λ_s (for simplicity, the failure rate of one component is considered for illustration; however, the failure rates of different components can be considered or combined into a single representative failure rate).

For an FT sensor node model (e.g., duplex sensor node or triple modular redundant sensor node), the average failure rate of the FT sensor node implicitly depends on k since the fault detection algorithm's accuracy depends on k (Sections 4 and 5.1.1). Hence, the failure rate of an FT sensor node is calculated for different k values in our

numerical results (Section 7). The WSN cluster Markov model relies on the failure rates calculated from the FT sensor node Markov model for different values of k. For example, for a cluster with n sensor nodes (Figure 6), and assuming each sensor node has all other sensor nodes in the cluster as neighbors, the number of neighboring sensor nodes for a given sensor node is $(n-1)$. Thus, the failure rate of an FT sensor node is calculated from the sensor node Markov model with $k = n-1$, which is then used in the WSN cluster Markov model (Figure 6). When a sensor node fails, the remaining operational (healthy) neighboring nodes of a given sensor node in a cluster become $(n-2)$; thus, the failure rate of an FT sensor node is calculated from the sensor node's Markov model with $k = n-2$ and so on. The cluster becomes nonoperational when the number of operational sensor nodes in the cluster reaches a minimum k_{min}, and, consequently, the overall failure rate of a WSN cluster with n sensor nodes is calculated for that k_{min}.

The WSN cluster failure rate with n sensor nodes obtained from the WSN cluster Markov model is then used in the WSN Markov model (Section 5.4). The WSN is assumed to consist of N clusters, and the WSN becomes nonoperational when the number of alive (operational) clusters reduces to N_{min}. The WSN Markov model is then used to calculate the MTTF and reliability of a given WSN using the cluster failure rate calculated from the WSN cluster Markov model.

6. SIMULATION OF DISTRIBUTED FAULT DETECTION ALGORITHMS

In this section, we discuss our implementation of two DFD algorithms: the Chen and Ding algorithms (Section 4). Our analysis is challenging because it is difficult to compare fault detection algorithms without a standard benchmark (Section 8.2). Since different DFD algorithms can use different fault models, a comparison between DFD algorithms can be made by observing the performance of the algorithms under similar fault models and similar topologies.

6.1. Using ns-2 to Simulate Faulty Sensors

In prior work, the Chen and Ding algorithms were simulated strictly algorithmically using simplified fault models that approximated a Poisson process. To compare these two models as realistically as possible, we use ns-2 [ns2 2014], which is a widely used discrete event simulator for both wired and wireless networks and a custom fault model. ns-2 allows us to see how these algorithms perform atop a detailed network stack over a wireless channel. We have leveraged ns-2 instead of ns-3 because ns-2 provides better support for WSN and Mobile Ad Hoc Network (MANET) protocols [Wikipedia 2014].

Wireless entities in ns-2 are called *nodes* and have modules called *agents* representing network layers stacked atop the nodes. Agents are typically used to generate application traffic, but we use agents as fault detection units in a sensor node (Figure 9). The agent first receives scheduled, scripted commands to read values from a `SensorNodeData` module. The `SensorNodeData` module generates random sensor readings based on our implemented fault models.

After recording a data sample, the sensor node broadcasts the data as a packet and builds a record of each broadcast packet that the sensor node receives from other sensor nodes. After a listening period, a separately scheduled event commands the detection agent to apply the two fault detection algorithms (i.e., the Ding and Chen algorithms). A fault tracker unit records statistics of detected and undetected errors. In the case of the Chen algorithm, each round of tendency broadcasting occurs in 2-second intervals. The sensor readings are also broadcast similarly in 2-second intervals.

A duplex configuration was implemented for permanent errors. When a sensor node passes a threshold of detected errors, the sensor "fails" and is replaced by a spare sensor. For the simplified fault model, this replacement helps little because Poisson

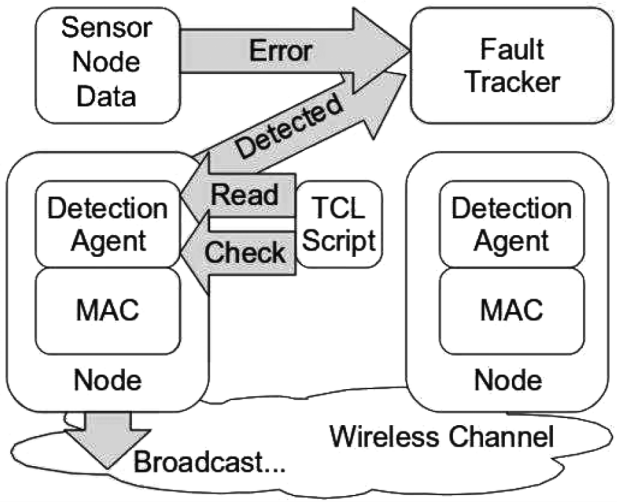


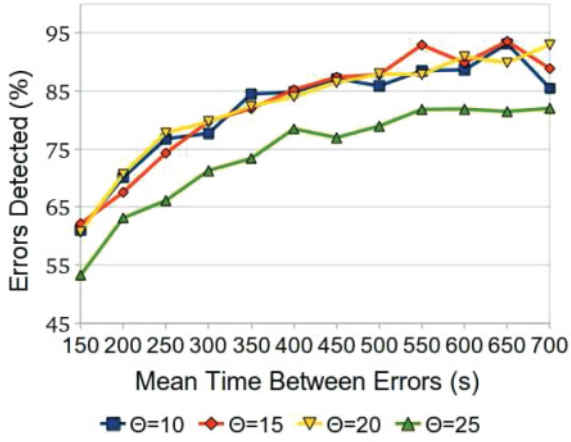
Fig. 9. The ns-2–based simulation architecture.

events are memoryless, and a sensor node with a new sensor and a sensor node with an old sensor will have the same error rate. However, for realistic data, constant errors are permanent and are repaired when the sensor is replaced. The realistic data usage also results in a penalty of false positives and causes more sensor nodes to be identified as faulty. The following subsections describe our experiments with both simulated and real-world data.

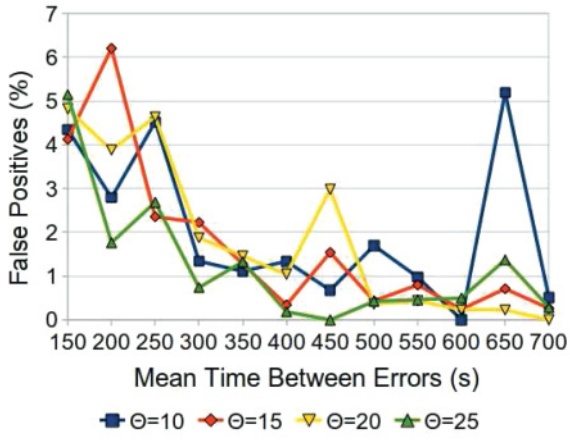
6.2. Experimental Setup for Simulated Data

To compare the performance of each algorithm, it is useful to observe each algorithm using a simplified fault model; thus, we consider the scenario used by Chen et al. [2006] where the average number of neighbors for each sensor node is approximately 7. Each sensor records a uniformly random temperature between 70 and 75 degrees. Errors occur as a Poisson process, with an arbitrary, scalable mean and corrupt a single sensor on a single sensor node during the event. A corrupted sample is randomly distributed between 100 and 105 degrees. We simulate each sensor node for 4 hours and take samples every 30 seconds.

Figures 10(a) and 10(b) depict the detection performance of the Chen algorithm across different parameters and error rates. We observe that, for the Chen algorithm, the error rate affects detection performance significantly; however, the θ value has a relatively minor impact. Our experimental results also match closely with the results from Chen et al. [2006], i.e., the detection accuracy for 7 neighbors and 10 neighbors decreases when the fault probability becomes larger. However, the fault detection accuracy is still about 97% when there are about 25% of the sensors being faulty. Figures 11(a) and 11(b) depict the detection performance of the Ding algorithm across different parameters and error rates. For the Ding algorithm, as θ is scaled, the number of false positives scales proportionately. For both algorithms, low error rates improve detection accuracy because low error rates make errors more distinguishable. In the simulated data scenario, the Chen algorithm performs better than the Ding algorithm in terms of percentage of false positives, and the detection accuracy is relatively less dependent on θ. However, this simulated scenario is very specific and consistent, which may not always be the case.



(a) Error detection accuracy for the Chen algorithm



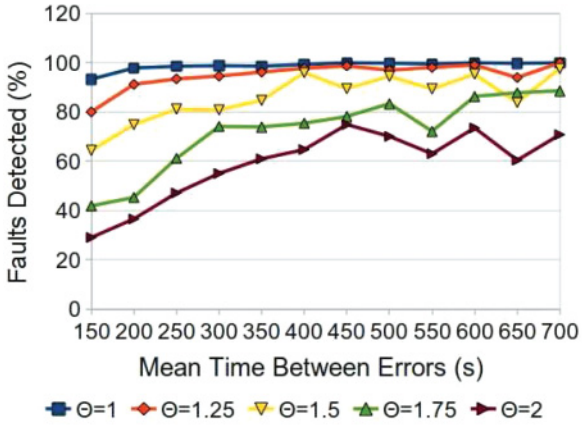
(b) False positive rate of Chen algorithm

Fig. 10. Effectiveness and false-positive rate of the Chen algorithm.

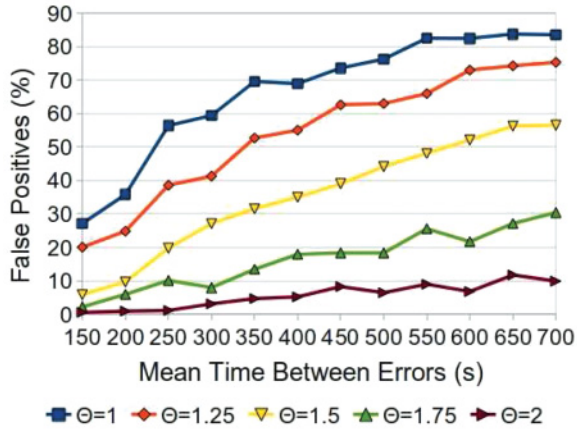
6.3. Experiments Using Real-World Data

Because transient faults are manifested in different ways, a more realistic error model that closely follows a real-world scenario is desirable for analysis. For our real-world data case study, we leverage publicly available sensor data from the Intel Berkeley research laboratory [Intel 2009]. The data come from an indoor sensor network deployment that measures temperature, humidity, and other metrics. Using temperature data from 53 sensor nodes, we produced a variety of empirical observations on which we based our simulation.

We note a few empirical observations from the Intel Berkeley data. One observation is that the real-world data include noise. Noise in sensor readings is correlated with past values (autocorrelated) and can have high power levels for certain time periods. To measure the effect of noise on sensor data, we apply a de-noising filter (available in the MATLAB Wavelet toolkit) to the Intel Berkeley data to approximate a reference "noise-free" signal. We measure the power level of noise in the data (Figure 12) and record the values in a text file. We play back the recorded noise power level values



(a) Error detection accuracy for the Ding algorithm



(b) False positive rate for the Ding algorithm

Fig. 11. Effectiveness and false-positive rate for the Ding algorithm.

during simulation and multiply the noise values with simulated white Gaussian noise to imitate the bursty behavior of noisy faults.

Another observation from the Intel Berkeley data is that, for nearly all sensor nodes, sensor samples level off as they approach their end-of-life, resulting in a "constant" fault. A histogram of the times at which this occurs (Figure 13) reveals that the distribution of sensor node lifetimes fits well into a Student's t-distribution, which is supported by MATLAB's distribution fitting tools. For our experiments, we recorded a set of several thousand random numbers fitting the Student's t-distribution. The sensor nodes then randomly select a value from the recorded t-distribution values to select a time for the fault occurrence.

Finally, we observe that voltage spikes are not frequent in the Intel Berkeley dataset, occurring in only 6 out of 53 nodes. Because of the rare occurrence of voltage spikes, our simulations inject a spike once during the lifetime of each sensor node. The time of the spike occurrence is modeled using a uniform process. The base signal is a sine wave with a period of 2 hours between 20 and 30 degrees Celsius. To allow minor variations in temperature readings sensed by sensor nodes, the temperature values are treated

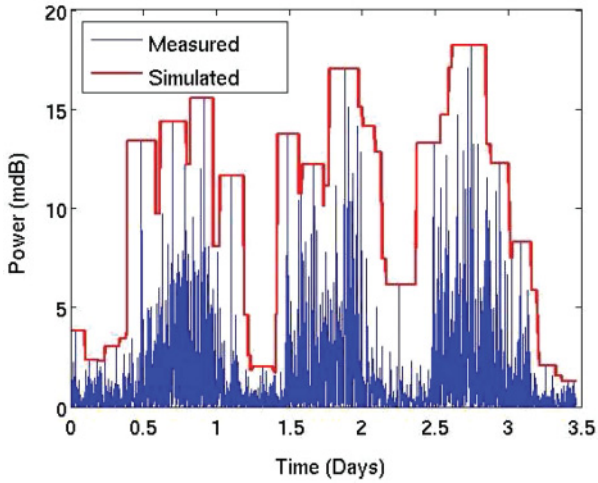


Fig. 12. Noise power levels in the Intel-Berkeley sample.

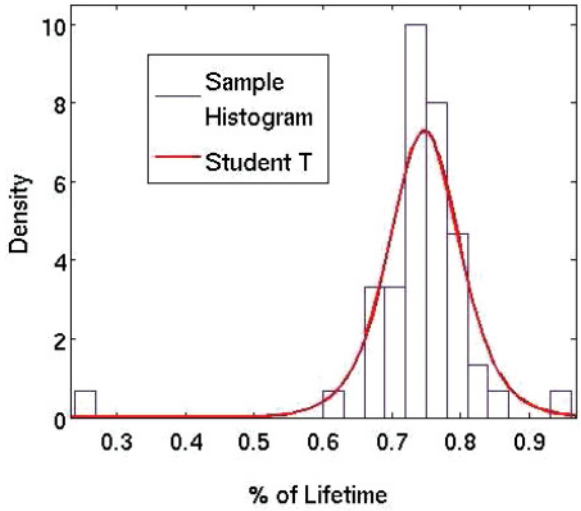


Fig. 13. Distribution of constant error occurrences.

as being generated from a radiating heat source. The nodes farthest from the center have a lower temperature, proportional to the inverse of the distance squared, as in the free-space path loss model. The noise and spike errors are added to the signal, and the constant fault causes the signal to hang at the signal's last valid value during an error. A signal is considered erroneous if the variation is more than half a degree from the signal's true value. In our experiments, the faulty signals are never repaired unless faults accumulate; ultimately, the sensor is replaced if diagnosed as faulty.

Figure 14 shows the performance of the Chen algorithm for the real-world data. Despite detecting more than 90% of the faults in the simplified model, the algorithm fails to perform well with real-world data, detecting fewer than 50% of erroneous readings. The poor performance of the Chen algorithm on real-world data is due to the nature of the algorithm. For an error to be detected, the error must not only be disruptive, causing a node to have different values from the node's neighbors, but the

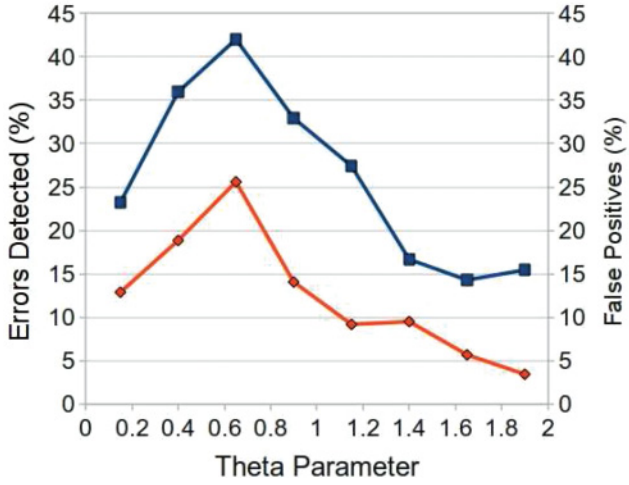


Fig. 14. Error detection and false positive rate for the Chen algorithm using real-world data.

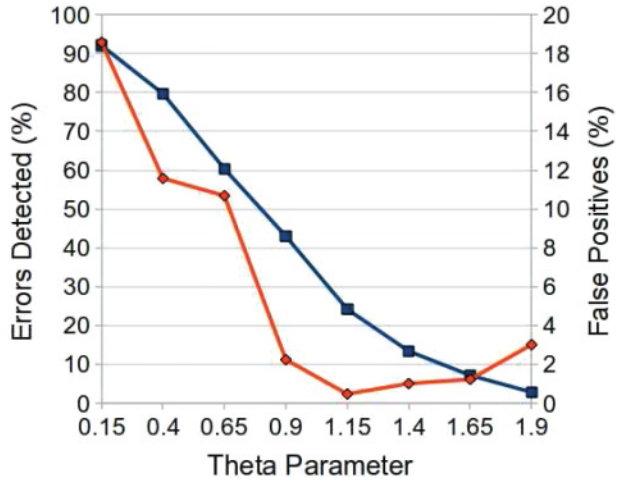


Fig. 15. Error detection and false-positive rate for the Ding algorithm using real-world data.

error must also be transient and have a sharp transition boundary. Since noise events last for multiple samples, the Chen algorithm is not equipped to handle these errors.

The Ding algorithm's performance with real-world data is shown in Figure 15. Like the Chen algorithm, the Ding algorithm's performance degrades as compared to the previous simplified model environment. However, the Ding algorithm still detects a majority of occurring faults. As with the previous simplified model experiment, the Ding algorithm is highly dependent on the algorithm's parameters. The higher the detection rate, the higher the rate of false positives for the Ding algorithm; however, the false-positive rate remains low, peaking at 20%.

7. NUMERICAL RESULTS

In this section, we present reliability and MTTF results for our Markov models (Section 5) implemented in the Symbolic Hierarchical Automated Reliability and Performance Evaluator (SHARPE) Software Package [Sahner et al. 1996; SHARPE 2014].

Table III. Estimated Values for a Fault Detection Algorithm's Accuracy

p	$M(p)$	k				
		5	6	7	10	15
0.05	25	0.979	1	1	1	1
0.1	50	0.858	0.895	0.921	0.957	0.96
0.2	56	0.755	0.717	0.81	0.845	0.851
0.3	65	0.652	0.679	0.699	0.732	0.742
0.4	66	0.558	0.5813	0.599	0.627	0.636
0.5	67	0.464	0.484	0.498	0.522	0.53
0.6	68	0.371	0.386	0.398	0.417	0.424
0.7	69	0.278	0.29	0.298	0.313	0.318
0.8	70	0.185	0.193	0.198	0.208	0.212
0.9	71	0.092	0.096	0.099	0.104	0.106
0.99	72	0.0092	0.00962	0.0099	0.0104	0.0106

SHARPE provides MTTF results directly based on our models' implementations; although SHARPE does not provide reliability results directly, reliability results can be calculated from state probabilities. We present example reliability calculations as well as detailed reliability and MTTF results for an NFT as well as for an FT sensor node, WSN cluster, and WSN using our bottom-up Markov modeling paradigm.

7.1. Experimental Setup

In this subsection, we describe our FT sensor node, WSN cluster, and WSN model implementation in the SHARPE Software Package [Sahner et al. 1996; SHARPE 2014]. We also present a typical fault detection algorithm's accuracy for different cumulative sensor failure probability p values. Due to SHARPE limitations that take only exponential polynomials, we assume our sensor failure rate to be exponentially distributed (Section 5.1.2).

SHARPE is a software tool for performance, reliability, and performability model specification and analysis. The SHARPE toolkit provides a specification language and solution methods for commonly used performance, reliability, and performability model types. SHARPE-supported models include combinatorial models, state-space (e.g., Markov and semi-Markov reward) models, and stochastic Petri nets. SHARPE allows computation of steady-state, transient, and interval measures. SHARPE allows output measures of a model to be used as parameters of other models to facilitate the hierarchical combination of different model types.

Our Markov model exploits the synergy of fault detection and FT for WSNs. Table III depicts c_k values estimated for a DFD algorithm using Equation (6) for different p and k values. These estimated c_k values approximate the accuracy of the DFD algorithms proposed in [Ding et al. 2005; Jiang 2009; Jian-Liang et al. 2007; Krishnamachari and Iyengar 2004]. We note that any inaccuracy in the c_k estimation does not affect our results calculations since these calculations leverage c_k values but are independent of whether these values accurately reflect a particular DFD algorithm's accuracy. We assume $c_c = 0$ in Equation (5) (i.e., once a faulty sensor is identified, the faulty sensor is perfectly replaced by a good spare sensor), which gives $c = c_k$ in Equation (5).

7.2. Reliability and MTTF for an NFT and an FT Sensor Node

In this subsection, we present the reliability and MTTF results for an NFT and an FT sensor node for the two cases: $c \neq 1$ and $c = 1$, when $k = 5$ and $p = 0.05$. The case $c \neq 1$ corresponds to a real-world fault detection algorithm (typical values are shown in Table III), whereas the case $c = 1$ corresponds to an ideal fault detection algorithm,

Table IV. Reliability for an NFT and an FT Sensor Node

Prob. p	NFT k, c : N/A	FT ($k = 5$, $c \neq 1$)	FT ($k = 10$, $c \neq 1$)	FT ($k = 15$, $c \neq 1$)	FT ($c = 1$)
0.05	0.94999	0.99770	0.99872	0.99872	0.99872
0.1	0.89996	0.98135	0.99074	0.99102	0.99482
0.2	0.80011	0.93481	0.95088	0.95195	0.97853
0.3	0.69977	0.86265	0.88263	0.88513	0.94959
0.4	0.59989	0.77094	0.79209	0.79485	0.90643
0.5	0.5007	0.66086	0.68097	0.68374	0.84662
0.6	0.40012	0.53610	0.55295	0.55552	0.76663
0.7	0.30119	0.40167	0.41432	0.41612	0.66262
0.8	0.19989	0.25943	0.26683	0.26812	0.52171
0.9	0.10026	0.12148	0.12424	0.12470	0.33086
0.99	0.01005	0.01048	0.01053	0.01056	0.05628

which detects faulty sensors perfectly for all p values. We calculate reliability at $t = 100$ days since we assume $t_s = 100$ days [Munir and Gordon-Ross 2009] in Equation (8); however, we point out that reliability can be calculated for any other time value using our Markov models.

For an NFT sensor node reliability calculation, we require the sensor failure rate λ_t, which can be calculated using Equation (8) (i.e., $\lambda_t = (-1/100)\ln(1-0.05) = 5.13 \times 10^{-4}$ failures/day). SHARPE gives $P_1(t) = e^{-5.13\times10^{-4}t}$ and sensor node reliability $R_s(t) = P_1(t)$. Evaluating $R_s(t)$ at $t = 100$ gives $R_s(t)|_{t=100} = 0.94999$.

For an FT sensor node when $c \neq 1$, different reliability results are obtained for different k because the fault detection algorithm's accuracy and coverage factor c depend on k. For $k = 5$, $c = 0.979$ (Table III), SHARPE gives $P_2(t) = e^{-5.13\times10^{-4}t}$ and $P_1(t) = 5.0223 \times 10^{-4}te^{-5.13\times10^{-4}t}$. The reliability $R_s(t) = P_2(t) + P_1(t)$, which when evaluated at $t = 100$ gives $R_s(t)|_{t=100} = 0.99770$. For an FT sensor node when $c = 1$ for all k, p, SHARPE gives $P_2(t) = e^{-5.13\times10^{-4}t}$ and $P_1(t) = 5.13 \times 10^{-4}te^{-5.13\times10^{-4}t}$. Using Equation (16), the reliability $R_s(t)|_{t=100} = 0.99872$.

Table IV shows the reliability for an NFT and an FT sensor node evaluated at $t = 100$ days for k values of 5, 10, and 15. As expected, the results show that reliability decreases for both NFT and FT sensor nodes as p increases, and the reliability attained by an FT sensor node (both for $c \neq 1$ and $c = 1$) is always better than an NFT sensor node. For example, the percentage reliability improvement achieved by an FT sensor node with $c \neq 1$, $k = 15$ over an NFT sensor node is 18.98% when $p = 0.2$. However, an FT sensor node with $c = 1$ outperforms both an FT sensor node with $c \neq 1$ and an NFT sensor node for all p values. For example, the percentage improvement in reliability for an FT sensor node with $c = 1$ over an NFT sensor node is 69.09%, and the percentage improvements in reliability for an FT sensor node with $c = 1$ over an FT sensor node ($c \neq 1$) with k equal to 5, 10, and 15, are 28.11%, 24.32%, and 23.82%, respectively, for $p = 0.5$. The percentage improvement in reliability attained by an FT sensor node with $c = 1$ over an NFT sensor node is 230%, and the percentage improvements in reliability for an FT sensor node with $c = 1$ over an FT sensor node ($c \neq 1$) with k equal to 5, 10, and 15, are 172.36%, 166.31%, and 165.32%, respectively, for $p = 0.9$. These results reveal that the percentage improvement in reliability attained by an FT sensor node with $c = 1$ increases as $p \to 1$ because, for an FT sensor node with $c \neq 1$, c decreases as $p \to 1$ (Table III). The sensor node reliability analysis reveals that a robust fault detection algorithm with $c \approx 1$ for all p and k values is necessary to attain good reliability for an FT sensor node.

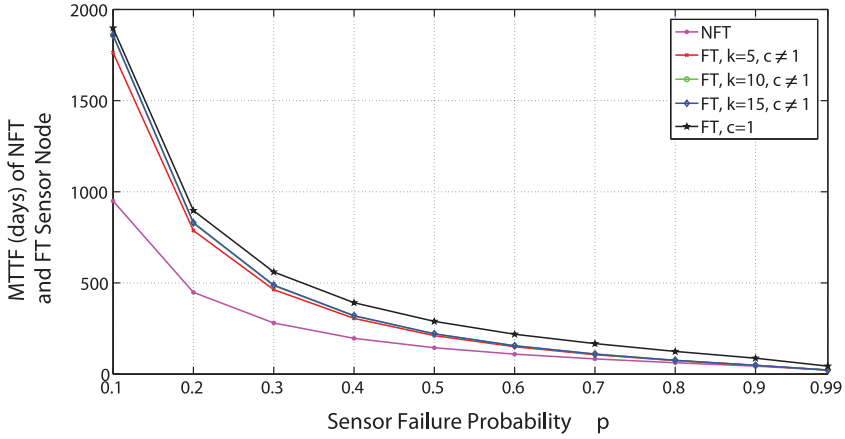


Fig. 16. MTTF in days for an NFT and an FT sensor node [Munir and Gordon-Ross 2011].

Table V. Percentage MTTF Improvement for an FT Sensor Node as Compared to an NFT Sensor Node

Prob. p	FT ($k = 5, c \neq 1$)	FT ($k = 10, c \neq 1$)	FT ($c = 1$)
0.1	85.75	95.65	100
0.2	75.61	84.63	100
0.3	65.14	72.99	100
0.5	46.25	52.49	100
0.7	27.62	31.23	100
0.9	9.37	10.52	100
0.99	0.875	1.34	100

Figure 16 depicts the MTTF for an NFT and an FT sensor node for k values of 5, 10, and 15 versus the sensor failure probability p when t_s in Equation (7) is 100 days [Ding et al. 2005; Krishnamachari and Iyengar 2004]. The results show that the MTTF for an FT sensor node improves with increasing k. However, the MTTF shows negligible improvement when $k = 15$ over $k = 10$ because the fault detection algorithm's accuracy improvement gradient (slope) decreases for large k values.

Figure 16 also compares the MTTF for an FT sensor node when $c = 1 \forall k, p$. Whereas $c \neq 1$ for existing fault detection algorithms, comparison with $c = 1$ provides insight into how the fault detection algorithm's accuracy affects the sensor node's MTTF. Figure 16 shows that the MTTF for an FT sensor node with $c = 1$ is always greater than an FT sensor node with $c \neq 1$.

Figure 16 shows that the MTTF for an NFT and an FT sensor node decreases as p increases; however, the FT sensor node maintains better MTTF than the NFT sensor node for all p values. This observation reveals that a sensor with a lower failure probability achieves better MTTF for both the NFT and FT sensor nodes. Figure 16 also shows that the MTTF for both the NFT and the FT sensor nodes approaches zero as p approaches 1 (i.e., MTTF $\rightarrow 0 \iff p \rightarrow 1$). This observation is intuitive because a faulty sensor (with $p = 1$) is unreliable and leads to a failed sensor node with zero MTTF and suggests that, depending on the application's reliability and MTTF requirements, the faulty sensor should be replaced before p approaches 1.

Table V depicts the percentage MTTF improvement gained by an FT sensor node over an NFT sensor node for different values of p. We calculate the percentage improvement

as % MTTF Improvement $= (\text{MTTF}_{FT} - \text{MTTF}_{NFT})/\text{MTTF}_{NFT} \times 100$. The table shows that the MTTF percentage improvement for an FT sensor node decreases as p increases when $c \neq 1$. The percentage MTTF improvement for an FT sensor node with $k = 5$ and $k = 10$ are 85.75% and 95.65%, respectively, for $p = 0.1$, and drops to 0.875% and 1.34%, respectively, for $p = 0.99$. This observation reveals that having more neighboring sensor nodes (a higher k value) improves MTTF because, in general, a fault detection algorithm's accuracy and c improve with increasing k. Table V also shows that the MTTF percentage improvement for an FT sensor node over an NFT sensor node is 100% on average when $c = 1$, thus highlighting the importance of a robust fault detection algorithm.

Model Validation: We compare the MTTF results for an NFT sensor node obtained using our Markov model with other existing models in literature, as well as with practical sensor node implementations, to provide insights into the accuracy of our models. We note that a detailed comparison and verification of our model with existing models is not possible because there are no similar/related models that leverage FT constructs. [Jung et al. 2007] modeled over the lifetime of trigger-driven sensor nodes with different event arrival rates (trigger-driven sensor nodes perform processing based on events as opposed to duty-cycle-based sensor nodes that perform processing based on a duty cycle), which can be interpreted as sensing events that depend on the sensor node's sensing frequency. Results were obtained for three trigger-driven sensor node platforms: XSM, MicaZ, and Telos. XSM and MicaZ consisted of an ATmega128L 8-bit processor with a maximum processor frequency of 16MHz and a maximum performance of 16 Million Instructions Per Second (MIPS). Telos consisted of a TI MSP430 16-bit processor with a maximum processor frequency of 8MHz and a maximum performance of 16 MIPS. Results revealed that Telos, XSM, and MicaZ delivered lifetimes of 1,500, 1,400, and 600 days, respectively, with an event arrival rate of 0.1 events/hour, which dropped to 350, 100, and 0 days, respectively, with an event arrival rate of 100 events/hour. The lifetime values for an NFT sensor node obtained from our Markov model ranges from 949 days to 22 days for different sensor failure probabilities. Depending on the sensor node energy consumption, the lifetime model proposed in Sha and Shi [2005] estimated that the sensor node lifetime varied from 25 days to 215 days. These comparisons indicate that our Markov model yields sensor node lifetime values in the range obtained from other existing models.

We also compare the NFT sensor node lifetime estimation from our model with an experimental study on sensor node lifetimes [Nguyen et al. 2011]. This comparison verifies conformity of our model with real measurements. For example, with a sensor node battery capacity of 2,500mA-h, experiments indicate a sensor node lifetime ranging from 72 to 95 hours for a 100% duty cycle for different battery brands (e.g., Ansmann, Panasonic Industrial, Varta High Energy, Panasonic Extreme Power) [Nguyen et al. 2011]. The sensor node lifetime for a duty cycle of 18% can be estimated as $95/0.18 = 528$ hours ≈ 22 days and for a duty cycle of 0.42% as $95/0.0042 = 22{,}619$ hours ≈ 942 days. This comparison reveals that our NFT sensor node lifetime model estimates lifetime values in the range obtained by experimental sensor node implementations.

7.3. Reliability and MTTF for an NFT and an FT WSN Cluster

In this subsection, we present the reliability and MTTF results for two WSN clusters that contain on average $n = k_{min} + 2$ and $n = k_{min} + 5$ sensor nodes at deployment time ($k_{min} = 4$). The selection of two different n values provides insight on how the size of the cluster affects reliability and MTTF (other n and k_{min} values depicted similar trends). The NFT WSN cluster consists of NFT sensor nodes and the FT WSN cluster consists of FT sensor nodes. For brevity, we present example WSN cluster reliability calculations for $n = k_{min} + 2$ ($k_{min} = 4$) and $p = 0.1$.

Table VI. Reliability for an NFT and an FT WSN Cluster when $n = k_{min} + 2$ ($k_{min} = 4$)

p	NFT	FT ($c \neq 1$)	FT ($c = 1$)
0.05	0.96720	0.99058	0.99098
0.1	0.88562	0.95969	0.96552
0.2	0.65567	0.84304	0.87474
0.3	0.41968	0.66438	0.74422
0.4	0.23309	0.45925	0.59388
0.5	0.10942	0.26595	0.43653
0.6	0.04098	0.11837	0.28732
0.7	0.01114	0.03548	0.16279
0.8	1.5957×10^{-3}	4.3470×10^{-3}	0.06723
0.9	5.5702×10^{-5}	1.4838×10^{-4}	0.01772
0.99	6.1056×10^{-10}	1.0057×10^{-9}	5.5702×10^{-5}

Leveraging our bottom-up approach model, the NFT WSN cluster uses the failure rate λ_s of an NFT sensor node. By using Equation (8), $\lambda_t = 1.054 \times 10^{-3}$ failures/day. Since the coverage factor c does not appear in the reliability (and MTTF) calculation for an NFT sensor node, the failure rate $\lambda_s = \lambda_t = 1/\text{MTTF}_s = 1.054 \times 10^{-3}$ failures/day. In other words, for an NFT sensor node $\lambda_{s(k_{min})} = \lambda_{s(k_{min}+1)}$, $\lambda_{s(k_{min})} = \lambda_{s(4)}$, and $\lambda_{s(k_{min})+1} = \lambda_{s(5)}$, which denote the sensor node failure rate when the average number of neighboring sensor nodes are 4 and 5, respectively. The NFT WSN cluster reliability calculation utilizes $\lambda_{s(4)}$ and $\lambda_{s(5)}$, which gives $P_{k_{min}+2}(t) = e^{-6.324\times10^{-3}t}$ and $P_{k_{min}+1}(t) = 6\times e^{-5.27\times10^{-3}t} - 6 \times e^{-6.324\times10^{-3}t}$. The reliability $R_c(t) = P_{k_{min}+2}(t) + P_{k_{min}+1}(t)$, which when evaluated at $t = 100$ days gives $R_c(t)|_{t=100} = 0.88562$.

For an FT WSN cluster when $c \neq 1$, the reliability calculation requires $\lambda_{s_d(k_{min})}$ and $\lambda_{s_d(k_{min}+1)}$, which depends on k as the fault detection algorithm's accuracy, and c depends on k, yielding different reliability results for different k_{min} values. Using Table III and Equation (18), $\lambda_{s_d(k_{min})} = \lambda_{s_d(4)} = 1/\text{MTTF}_{s_d(4)} = 1/1.715 \times 10^3 = 5.83 \times 10^{-4}$ failures/day and $\lambda_{s_d(k_{min}+1)} = \lambda_{s_d(5)} = 1/\text{MTTF}_{s_d(5)} = 1/1.763 \times 10^3 = 5.67 \times 10^{-4}$ failures/day. SHARPE gives $P_{k_{min}+2}(t) = e^{-3.402\times10^{-3}t}$ and $P_{k_{min}+1}(t) = 6.9856\times e^{-2.915\times10^{-3}t} - 6.9856\times e^{-3.402\times10^{-3}t}$. The reliability of the FT WSN cluster is calculated as $R_c(t)|_{t=100} = 0.95969$.

For an FT WSN cluster when $c = 1$, the reliability calculation does not depend on k, which gives $\lambda_{s_d(k_{min})} = \lambda_{s_d(4)} = 1/\text{MTTF}_{s_d(4)} = 1/1898 = 5.269 \times 10^{-4}$ failures/day and $\lambda_{s_d(k_{min}+1)} = \lambda_{s_d(5)} = 1/\text{MTTF}_{s_d(5)} = 1/1898 = 5.269 \times 10^{-4}$ failures/day. SHARPE gives $P_{k_{min}+2}(t) = e^{-3.1614\times10^{-3}t}$ and $P_{k_{min}+1}(t) = 6 \times e^{-2.6345\times10^{-3}t} - 6 \times e^{-3.1614\times10^{-3}t}$, from which we calculate $R_c(t)|_{t=100} = 0.96552$.

Table IV shows the reliability for an NFT and an FT WSN cluster (for $c \neq 1$ and $c = 1$) evaluated at $t = 100$ days when $n = k_{min} + 2$ ($k_{min} = 4$). We observe similar trends as with sensor node reliability (Table IV), where the reliability of both NFT and FT WSN clusters decreases as p increases (i.e., reliability $R_c \to 0 \iff p \to 1$) because decreased individual sensor node reliability decreases the overall WSN cluster reliability. Table VI shows that an FT WSN cluster with $c = 1$ outperforms an FT WSN cluster with $c \neq 1$ and an NFT WSN cluster for all p values. For example, the percentage improvement in reliability for an FT WSN cluster with $c = 1$ over an NFT WSN cluster and an FT WSN cluster with $c \neq 1$ is 77.33% and 12.02% for $p = 0.3$ and 601.12% and 142.73% for $p = 0.6$, respectively. These results show that the percentage improvement in reliability attained by an FT WSN cluster increases as p increases because a fault detection algorithm's accuracy and c decrease as p increases (Table III). This trend is similar to the percentage improvement in reliability for an FT sensor node (Section 7.2). The results show that an FT WSN cluster always performs better than an NFT WSN

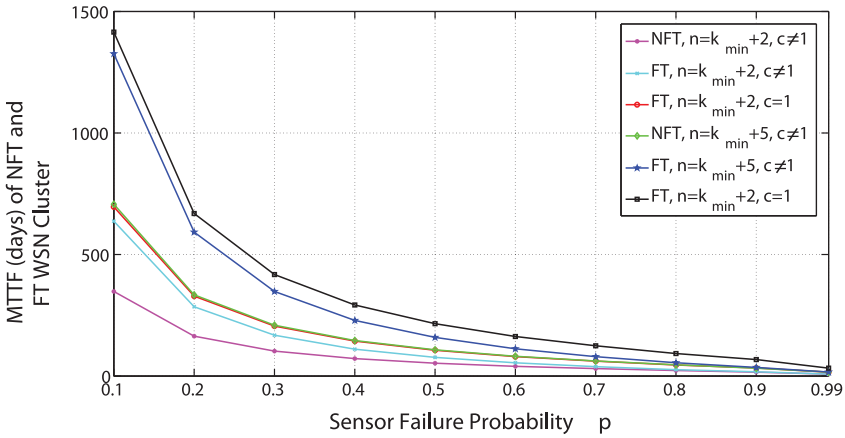


Fig. 17. MTTF in days for an NFT WSN cluster and an FT WSN cluster with $k_{min} = 4$ [Munir and Gordon-Ross 2011].

Table VII. Percentage MTTF Improvement for an FT WSN Cluster as Compared to an NFT WSN Cluster ($k_{min} = 4$)

Prob. p	FT ($n = k_{min} + 2$, $c \neq 1$)	FT ($n = k_{min} + 5$, $c \neq 1$)	FT ($n = k_{min} + 2$, $c = 1$)
0.1	83.04	87.55	100
0.2	73.84	77.24	100
0.3	63.58	66.51	100
0.5	44.97	47.22	100
0.7	26.4	28.71	100
0.9	9.81	9.57	100
0.99	2.26	2.47	100

cluster. For example, the percentage improvement in reliability for an FT WSN cluster with $c \neq 1$ over an NFT WSN cluster is 58.31% for $p = 0.3$.

Figure 17 depicts the MTTF for an NFT WSN cluster and an FT WSN cluster versus p when $k_{min} = 4$ for average cluster sizes of $n = k_{min} + 2$ and $n = k_{min} + 5$ sensor nodes, respectively, at deployment time. The figure reveals that the FT WSN cluster's MTTF is considerably greater than the NFT WSN cluster's MTTF for both cluster sizes. Figure 17 also compares the MTTF for an FT WSN cluster when $c = 1$ with $c \neq 1$ and shows that the MTTF for an FT WSN cluster with $c = 1$ is always better than an FT WSN cluster with $c \neq 1$. This observation again verifies the significance of a robust (ideal) fault detection algorithm, which can ideally provide c values close to 1 (i.e., $c \approx 1$ for all p values). We note that both an NFT and an FT WSN cluster with $n > k_{min}$ have redundant sensor nodes and can inherently tolerate $n - k_{min}$ sensor node failures. The WSN cluster with $n = k_{min} + 5$ has more redundant sensor nodes than the WSN cluster with $n = k_{min} + 2$, resulting in a comparatively greater MTTF. Figure 17 shows that the MTTF for both an NFT and an FT WSN cluster approaches 0 as p approaches 1 and further solidifies the necessity of low failure probability sensors to achieve better MTTF in WSN clusters. The MTTF variation for a WSN cluster with varying p is similar to the MTTF variation for a sensor node (Figure 16) because a WSN cluster is composed of sensor nodes and reflects the MTTF variation of a WSN cluster's constituent sensor nodes.

Table VII shows the percentage MTTF improvement for an FT WSN cluster as compared to an NFT WSN cluster for cluster sizes of $n = k_{min} + 2$ and $n = k_{min} + 5$ sensor

Table VIII. Iso-MTTF for WSN Clusters ($k_{min} = 4$)

Prob. p	MTTF FT (days) ($n = k_{min} + 2, c = 1$)	MTTF NFT (days) ($n = k_{min} + 5$)	NFT $\Re^s_{NFT}$
0.1	695.89	707.43	3
0.3	205.3	208.86	3
0.5	105.97	107.6	3
0.7	61.213	62.136	3
0.99	15.942	16.209	3

$\Re^s_{NFT}$ denotes the redundant sensor nodes required by an NFT WSN cluster to achieve a comparable MTTF as that of an FT WSN cluster.

nodes. The percentage improvements are calculated separately for the two cluster sizes, and we compare the MTTFs of clusters of the same size. We observe that the MTTF improvement for an FT cluster is slightly greater when $n = k_{min} + 5$ as compared to when $n = k_{min} + 2$; however, the MTTF improvement decreases with increasing p when $c \neq 1$. This observation reveals that a WSN cluster consisting of more sensor nodes can achieve better MTTF improvements when compared to a WSN cluster containing fewer sensor nodes. The MTTF percentage improvement for an FT WSN cluster with $n = k_{min} + 2$, $c \neq 1$, is 83.04% for $p = 0.1$ and drops to 2.26% for $p = 0.99$. Similarly, the percentage MTTF improvement for an FT WSN cluster with $n = k_{min} + 5$, $c \neq 1$, is 87.55% for $p = 0.1$ and drops to 2.47% for $p = 0.99$. The percentage MTTF improvement for both cluster sizes is 100% on average when $c = 1$ (for brevity, in Table VII, results are not shown for $n = k_{min} + 5$ when $c = 1$). We observed that the MTTF percentage improvement for an FT WSN cluster with $n = k_{min} + 5$ over $n = k_{min} + 2$ is 103.35% on average. Intuitively, these results confirm that clusters with more redundant sensor nodes are more reliable and have a better MTTF than do clusters with fewer redundant sensor nodes.

Because SHARPE provides information on MTTF directly from Markov models, whereas reliability calculation requires manual substitution in state transition probabilities, we present iso-MTTF results for WSN clusters. Table VIII depicts the MTTFs for an FT WSN cluster and an NFT WSN cluster for p varying from 0.1 to 0.99 and when $k_{min} = 4$. Results indicate that an NFT WSN cluster requires three redundant NFT sensor nodes to achieve a comparable MTTF as that of an FT WSN cluster.

7.4. Reliability and MTTF for an NFT and an FT WSN

This subsection presents the reliability and MTTF results for an NFT WSN and an FT WSN containing $N = N_{min}+2$ and $N = N_{min}+5$ clusters at deployment time ($N_{min} = 0$). We consider WSNs containing different number of clusters to provide an insight on how the number of clusters affects WSN reliability and MTTF (other N and N_{min} values depicted similar trends). The FT WSN contains FT sensor nodes and the NFT WSN contains NFT sensor nodes. We present example WSN reliability calculations for $N = N_{min} + 2$ ($N_{min} = 0$; i.e., each WSN fails when there are no more active clusters) and $p = 0.2$. We assume that both WSNs contain clusters with $n = k_{min} + 5$ ($k_{min} = 4$) = 9 sensor nodes on average, with a cluster failure rate $\lambda_{c(9)}$.

The reliability calculation for an NFT WSN requires the NFT WSN cluster failure rate $\lambda_{c(9)}$, which can be calculated using Equation (24) with $n = 9$ (i.e., $\lambda_{c(9)} = 1/MTTF_{c(9)} = 1/3.34 \times 10^2 = 2.99 \times 10^{-3}$ failures/day). Using $\lambda_{c(9)}$, SHARPE gives $P_{N_{min}+2}(t) = e^{-5.98\times10^{-3}t}$ and $P_{N_{min}+1}(t) = 2 \times e^{-2.99\times10^{-3}t} - 2 \times e^{-5.98\times10^{-3}t}$. The WSN reliability $R_{wsn}(t) = P_{N_{min}+2}(t) + P_{N_{min}+1}(t)$ when evaluated at $t = 100$ days gives $R_{wsn}(t)|_{t=100} = 0.93321$.

Table IX. Reliability for an NFT WSN and an FT WSN when $N = N_{min} + 2$ ($N_{min} = 0$)

p	NFT	FT ($c \neq 1$)	FT ($c = 1$)
0.05	0.99557	0.99883	0.99885
0.1	0.98261	0.99474	0.99534
0.2	0.93321	0.97583	0.98084
0.3	0.85557	0.93775	0.95482
0.4	0.75408	0.87466	0.91611
0.5	0.63536	0.78202	0.86218
0.6	0.51166	0.65121	0.78948
0.7	0.36303	0.49093	0.69527
0.8	0.20933	0.30328	0.55494
0.9	0.08807	0.11792	0.39647
0.99	4.054×10^{-3}	4.952×10^{-3}	0.08807

For an FT WSN when $c \neq 1$, the reliability calculation requires the FT WSN cluster failure rate $\lambda_{c(9)}$ (for $c \neq 1$) (i.e., $\lambda_{c(9)} = 1/MTTF_{c(9)} = 1/5.92 \times 10^2 = 1.69 \times 10^{-3}$ failures/day). Using $\lambda_{c(9)}$, SHARPE gives $P_{N_{min}+2}(t) = e^{-3.38\times10^{-3}t}$, $P_{N_{min}+1}(t) = 2 \times e^{-1.69\times10^{-3}t} - 2 \times e^{-3.38\times10^{-3}t}$. The WSN reliability $R_{wsn}(t)|_{t=100} = 0.97583$.

For an FT WSN cluster when $c = 1$, the reliability calculation requires the FT WSN cluster failure rate $\lambda_{c(9)}$ (for $c = 1$) (i.e., $\lambda_{c(9)} = 1/MTTF_{c(9)} = 1/668.73 = 1.49 \times 10^{-3}$ failures/day). Using $\lambda_{c(9)}$, SHARPE gives $P_{N_{min}+2}(t) = P_2(t) = e^{-2.98\times10^{-3}t}$, $P_{N_{min}+1}(t) = P_1(t) = 2 \times e^{-1.49\times10^{-3}t} - 2 \times e^{-2.98\times10^{-3}t}$, which gives $R_{wsn}(t)|_{t=100} = 0.98084$.

Table IX shows the reliability for an NFT WSN and an FT WSN evaluated at $t = 100$ days when $N = N_{min} + 2$ ($N_{min} = 0$) for clusters with nine sensor nodes on average. We observe similar trends as with sensor node reliability (Table IV) and WSN cluster reliability (Table VI), where reliability for both an NFT WSN and an FT WSN decreases as p increases (i.e., reliability $R_{wsn} \to 0 \iff p \to 1$) because a WSN contains clusters of sensor nodes and decreased individual sensor node reliability with increasing p decreases both WSN cluster and WSN reliability. Table IX shows that an FT WSN with $c = 1$ outperforms an FT WSN with $c \neq 1$ and an NFT WSN for all p values. For example, the percentage improvement in reliability for an FT WSN with $c = 1$ over an NFT WSN and an FT WSN with $c \neq 1$ is 5.1% and 0.51% for $p = 0.2$ and 350.18% and 236.22% for $p = 0.9$, respectively. These results show that the percentage improvement in reliability attained by an FT WSN increases as p increases because the fault detection algorithm's accuracy and c decrease as p increases (Table III). This trend is similar to the percentage improvement in reliability for an FT sensor node (Section 7.2) and an FT WSN cluster (Section 7.3). The results show that an FT WSN always performs better than an NFT WSN. For example, the percentage improvement in reliability for an FT WSN with $c \neq 1$ over an NFT WSN is 4.57% for $p = 0.2$.

Figure 18 depicts the MTTF for an NFT WSN and an FT WSN containing on average $N = N_{min} + 2$ and $N = N_{min} + 5$ clusters ($N_{min} = 0$) at deployment time. The figure reveals that an FT WSN improves the MTTF considerably over an NFT WSN for both number of clusters. Figure 18 also shows that the MTTF for an FT WSN when $c = 1$ is always greater than the MTTF for an FT WSN when $c \neq 1$. We observe that since the MTTF for an FT WSN drops nearly to that of an NFT WSN as $p \to 1$, building a more reliable FT WSN requires low failure probability sensors. This WSN MTTF variation with p follows trends similiar to those observed in WSN clusters (Figure 17) and sensor nodes (Figure 16). Similar MTTF variations for a WSN, WSN cluster, and a sensor node result from our bottom-up modeling approach (Section 5) where each hierarchical level captures the MTTF variation trends of lower levels. We observe that the MTTF for a WSN with $N = N_{min}+5$ is always greater than the MTTF for a WSN with $N = N_{min}+2$.

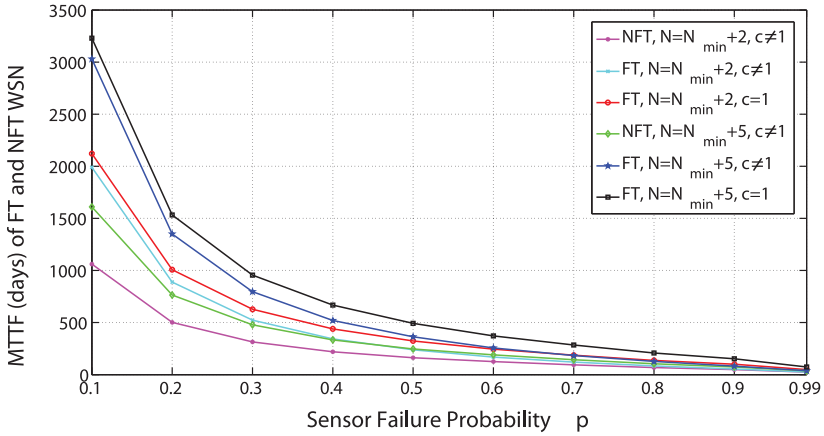


Fig. 18. MTTF in days for an NFT WSN and an FT WSN with $N_{min} = 0$ [Munir and Gordon-Ross 2011].

Table X. Iso-MTTF for WSNs ($N_{min} = 0$)

Prob. p	MTTF FT (days) ($N = N_{min} + 2$, $c = 1$)	MTTF NFT (days) ($N = N_{min} + 11$)	NFT $\Re^c_{NFT}$
0.1	2121.6	2135.7	9
0.3	627.62	631.77	9
0.5	323.28	326.12	9
0.7	186.8	188.74	9
0.99	48.387	48.71	9

$\Re^c_{NFT}$ denotes the redundant WSN clusters required by an NFT WSN to achieve a comparable MTTF as that of an FT WSN.

This observation is intuitive because WSNs with $N = N_{min} + 5$ have more redundant WSN clusters (and sensor nodes) and can survive more cluster failures before reaching the failed state ($N = 0$) as compared to a WSN with $N = N_{min} + 2$.

We observe the percentage MTTF improvement for an FT WSN over an NFT WSN containing on average $N = N_{min} + 2$ and $N = N_{min} + 5$ clusters. We observe that the MTTF improvement for both numbers of clusters decreases with increasing p when $c \neq 1$ (a trend similar to that for a WSN cluster and for a sensor node). The MTTF percentage improvement for an FT WSN with $N = N_{min}+2, c \neq 1$, is 87.56% for $p = 0.1$ and drops to 3.3% for $p = 0.99$. Similarly, the MTTF percentage improvement for an FT WSN with $N = N_{min}+5, c \neq 1$, is 88.2% for $p = 0.1$ and drops to 3.26% for $p = 0.99$. We observe that the MTTF improvement for an FT WSN with $c = 1$ is 100% on average for all p values and is greater than an FT WSN with $c \neq 1$. The MTTF percentage improvement for an FT WSN with $N = N_{min} + 5$ over an FT WSN with $N = N_{min} + 2$ is 52.22% on average.

We also investigate iso-MTTF for WSNs. Table X depicts the MTTFs for an FT WSN and an NFT WSN for p varying from 0.1 to 0.99 and when $N_{min} = 0$. Results indicate that an NFT WSN requires nine redundant NFT WSN clusters to achieve an MTTF comparable to that of an FT WSN.

8. RESEARCH CHALLENGES AND FUTURE RESEARCH DIRECTIONS

WSNs are susceptible to a plethora of different fault types and external attacks after the WSN's deployment. Fault detection and tolerance in WSNs is paramount, especially for safety-critical WSNs such as military target applications, biomedical applications

(body sensor networks), and volcanic eruption detection systems. Although there has been research done in fault diagnosis and FT in WSNs, various issues remain to be addressed. This section highlights research challenges and future research directions for development and deployment of reliable and trustworthy WSNs.

8.1. Accurate Fault Detection

Our Markov modeling results in Section 7 reveal that the fault detection algorithm's accuracy plays a crucial role in realizing FT WSNs. An accuracy close or equal to 100% and a false alarm rate close or equal to 0% is desirable for fault detection algorithms. There is a need to develop novel fault detection algorithms and/or improve the existing fault detection algorithms with the objective of attaining ∼100% fault detection accuracy.

8.2. Benchmarks for Comparing Fault Detection Algorithms

There is a need to develop standard benchmarks for comparing different fault detection algorithms for various fault models and topologies. The development of these benchmarks would facilitate designers in selecting an appropriate fault detection algorithm for WSNs with given FT requirements.

8.3. Energy-Efficient Fault Detection and Tolerance

Because sensor nodes have stringent resource constraints, fault detection and FT approaches for WSNs need to be energy-efficient. Although initiatives have been taken in developing DFD approaches to minimize energy overhead, the area requires further research. Moreover, FT techniques developed for WSNs need to be energy-efficient. The impact of added redundancy in sensor nodes to provide FT has not only economic cost overhead but can also have power consumption overhead, which needs to be considered and minimized during the design. Energy-efficiency in fault detection and tolerance can also be achieved by exploiting Near-Threshold Computing (NTC) in sensor node design. NTC refers to using a supply voltage (V_{DD}) that is close to a single transistor's threshold voltage V_t (generally V_{DD} is slightly above V_t in near-threshold operation, whereas V_{DD} is below V_t for subthreshold operation). Lowering the supply voltage reduces power consumption and increases energy efficiency by lowering the energy consumed per operation. The widespread adoption of NTC in sensor node designs for reduced power consumption requires addressing NTC challenges such as increased process, voltage, and temperature variations; subthreshold leakage power; and soft error rates.

8.4. Machine Learning-Inspired Fault Detection

Machine learning-based approaches can be leveraged to classify normal and anomalous data. Since sensor nodes are resource constrained, innovative approaches are required to extract features from the sensed data. The machine learning techniques for sensor nodes need to reduce the training data size for the classifier (e.g., SVM) based on extracted features. The goal of machine learning-inspired fault detection is to classify normal and anomalous data in or near real-time. There exist some initiatives in machine learning-inspired fault detection [Bhargava and Raghuvanshi 2013]; however, further work is required in this domain for accurate and energy-efficient fault detection.

8.5. FT in Multimedia Sensor Networks

Fault detection and tolerance in Wireless Multimedia Sensor Networks (WMSNs) is a relatively new research field [Sun et al. 2012]. FT data aggregation in WMSNs is imperative to reduce the throughput of data transmission, energy conservation, and

accuracy of event detection, and to alleviate the interference from compromised nodes. Another research challenge in WMSNs is to ensure the trustworthiness of aggregated results. Since multimedia data sensed by sensor nodes require compute-intensive processing, multicore-based sensor node designs could meet the performance, energy, and FT requirements of WMSNs. Accurate fault detection, FT, reliability, and security in WMSNs require further research endeavors.

8.6. Security

Security is critical for many WSNs, such as military target tracking and biomedical applications. Security aspects are crucial to consider in developing a reliable, robust, and trustworthy WSNs. Since sensor nodes use wireless communication, WSNs are susceptible to eavesdropping, message injection, replay, and other attacks. In many situations, an adversary is capable of deploying malicious nodes in the WSN or compromising some legitimate sensor nodes. Since the sink node is a gateway between a WSN and the outside world, compromising the sink node could render the entire WSN useless. Hence, security integration in sink nodes is imperative. The security integration objective in WSNs is to provide confidentiality, integrity, authenticity, and availability of all messages in the presence of adversaries. In a secure and reliable WSN, every eligible receiver should receive all messages intended for that receiver. Furthermore, the receiver should be able to verify the integrity of each message, as well as the identity of the sender.

Integration of security and privacy in sensor nodes is challenging because of the sensor nodes' resource constraints in terms of computation, communication, memory/storage, and energy supply. The security mechanisms for prevalent end-to-end systems (e.g., Secure Socket Layer (SSL)) are not directly applicable to WSNs. In traditional end-to-end systems, it is neither necessary nor desirable for the contents of the message (except the message headers) to be available to intermediate routers. On the contrary, in WSNs, the dominant traffic pattern is many-to-one (i.e., from sensor nodes to the sink node) [Du and Chen 2008]. Each intermediate sensor node in a WSN also needs access to message contents to perform in-network processing, such as data aggregation and data compression, to conserve energy. Security incorporation in WSNs has several aspects, some of which are discussed here.

Time Synchronization: Due to the collaborative nature of sensor nodes, secure time synchronization is important for various WSN operations, such as coordinated sensing tasks, sensor scheduling, target tracking, data aggregation, and Time Division Multiple Access (TDMA) medium access control. The Network Time Protocol (NTP), which is used for synchronization in the Internet, cannot be directly used by WSNs due to a sensor node's constrained resources. Some time synchronization protocols for WSNs have been proposed in the literature, such as Reference-Broadcast Synchronization (RBS) [Sichitiu and Veerarittiphan 2003] and Timing-Sync Protocol for Sensor Networks (TPSN) [Ganeriwal et al. 2003]; however, security is not considered in the design of these synchronization protocols.

Sensor Location Discovery: Sensor node location is important for many WSN applications, such as target tracking and environmental monitoring. The geographical routing protocols developed for WSNs also require sensor node location information to make routing decisions. Sensor location discovery protocols make use of special nodes, known as *beacon nodes*, which are assumed to know their own location via Global Positioning System (GPS) or manual configuration. Nonbeacon nodes receive radio signals (reference signals) from beacon nodes. These reference signals contain the location of beacon nodes. The nonbeacon nodes then determine their own location based on features of the

reference messages, such as using Received Signal Strength Indicator (RSSI). Without any protection, the adversary has the potential to mislead the location estimation at sensor nodes. For example, an attacker may provide incorrect location references by replaying the beacon signals intercepted in different locations. A more serious attack launched by an adversary could be to compromise a beacon node and distribute malicious location references by manipulating the beacon signals (e.g., changing the signal strength if RSSI is used to estimate the distance). In the presence of such attacks, nonbeacon nodes will determine their location incorrectly. There are a few methods to counter these attacks and to increase the reliability and trustworthiness of WSN. For instance, voting-based location reference uses voting of different received location references from different beacon nodes; however, this technique would increase the cost of the WSN because the approach requires more beacon nodes equipped with GPS.

Secure Routing: An adversary can also impact routing within a WSN, which consequently degrades the reliability of the WSN. Secure routing protocols for mobile ad hoc networks or wired networks are not suitable for WSNs because these protocols are computationally expensive and are meant for routing between any pair of nodes and not for the many-to-one traffic pattern prevalent in WSNs.

Intrusion Detection: Most of the existing WSNs are designed without considering security and intrusion detection. WSNs are more prone to malicious intruders as compared to traditional wired networks due to an open medium, a low degree of physical security, the dynamic topology, and a limited power supply. Security vulnerabilities in WSNs can be alleviated using an Intrusion Detection System (IDS). Intrusion detection is a security technology that attempts to identify those who are trying to break into and misuse a system without authorization, as well as those who have legitimate access to the system but are abusing their privileges [Sun et al. 2007]. By modeling the behavior of normal sensor node activities and operations, an IDS can identify potential intruders and maintain the trustworthy operation of a WSN. IDSs are of two types: misuse-based detection and anomaly-based detection. A *misuse-based* IDS encodes known attack signatures and vulnerabilities and stores this information in a database. An alarm is generated by the IDS if a discrepancy exists between the current activities and stored signatures. A misuse-based IDS cannot detect novel attacks because of the lack of corresponding signatures. An *anomaly-based* IDS creates normal profiles of sensor node behavior and compares them with current activities. If a significant deviation in a sensor node's behavior is observed, the anomaly-based IDS raises an alarm. An anomaly-based IDS can detect unknown attacks; however, developing profiles of a sensor node's normal behavior is challenging.

8.7. WSN Design and Tuning for Reliability

WSNs are more susceptible to failure than are wired networks due to a WSN's deployment in unattended and often hostile environments. The vulnerability of sensor nodes to failures and attacks from malicious intruders requires reliability and security incorporation in the design of sensor nodes and WSNs. For instance, in the case of failures of some sensor nodes or some sensor nodes being compromised by an adversary, a WSN should be able to detect these failures and then adapt accordingly to remain operational. For example, if a few nodes are diagnosed as misbehaving or failed, the routing protocol should be able to adapt to the situation and find alternative routes. This adaptation may require some sensor nodes to increase their transmission energy to get their sensed information to healthy nodes. Furthermore, the frequency of packet transmission can be reduced in such situations to conserve the energy of sensor nodes to compensate for the increased transmission energy. This tuning of sensor nodes and

the WSN would maintain the reliable operation of WSN, albeit at a reduced efficiency. Moreover, the WSN manager should be notified of the situation so that the manager can take necessary actions to increase the efficiency of the WSN, including removal of identified malicious nodes and placement of new healthy sensor nodes in the WSN.

8.8. Novel WSN Architectures

To maintain reliable and secure operation of WSNs in the presence of failures and malicious intruders, innovative WSN architectures are desired. For instance, a heterogeneous hierarchical Multicore Embedded Wireless Sensor Network (MCEWSN) can better meet the performance and reliability requirements of a WSN [Munir et al. 2014]. The heterogeneity in the architecture subsumes the integration of numerous single-core embedded sensor nodes and several multicore embedded sensor nodes. A hierarchical architecture comprising various clusters and a sink node is appropriate for large WSNs since, in small WSNs, sensor nodes can send the sensed data directly to the sink node. Each cluster comprises several leaf sensor nodes and a cluster head. Leaf sensor nodes embed a single-core processor and are responsible for sensing, preprocessing, and transmitting the sensed data to the cluster head nodes. Cluster head nodes embed a multicore processor and are responsible for coalescing/fusing the data received from leaf sensor nodes for transmission to the sink node. Multicore embedded sensor nodes enable energy savings over conventional single-core embedded sensor nodes in two ways. First, multicore embedded sensor nodes reduce the energy expended in communication by performing in situ computation of sensed data and transmitting only processed information. Second, a multicore embedded sensor node permits the computations to be split across multiple cores while running each core at a lower processor voltage and frequency, as compared to a single-core system, which results in energy savings. Multicore cluster heads are also amenable for supervising a fault diagnosis algorithm within the cluster. Additionally, security primitives could be integrated into these multicore cluster heads to thwart attacks from malicious intruders or malicious sensor nodes.

9. CONCLUSION

This article provided comprehensive research on modeling and analysis of fault detection and tolerance in WSNs. To elucidate fault detection in WSNs, we provided a taxonomy for fault diagnosis in WSNs. We simulated prominent fault detection algorithms using ns-2 to determine the algorithms' accuracies and false alarm rates. We further analyzed the effectiveness of fault detection algorithms under conditions modeled from real-world data. We proposed an FT duplex sensor node model based on the novel concept of determining the coverage factor using a sensor node's fault detection algorithm's accuracy. We developed comprehensive Markov models that hierarchically encompass sensor nodes, WSN clusters, and the overall WSN. Our Markov models characterized WSN reliability and MTTF for different sensor failure probabilities. Our models could assist WSN designers to better meet application requirements by determining the reliability and MTTF in the predeployment phase.

Our ns-2 simulation results for fault detection algorithms suggested that both the simulated and real-world data needed to be considered for rigorous evaluation of fault detection algorithms. Although some algorithms performed better than others on simulated data, these algorithms performed inferiorly to other algorithms on real-world data. We observed that the fault detection algorithm's accuracy played a crucial role in FT WSNs by comparing our results against a perfect fault detection algorithm ($c = 1$). The results indicated that the percentage improvement in reliability for an FT sensor node with $c = 1$ over an NFT sensor node is 230%, and the percentage improvements

in reliability for an FT sensor node with $c = 1$ over an FT sensor node ($c \neq 1$) with an average number of neighbors k equal to 5, 10, and 15, was 172.36%, 166.31%, and 165.32%, respectively, for $p = 0.9$. The percentage improvement in reliability for an FT WSN cluster with $c = 1$ over an NFT WSN cluster and an FT WSN cluster with $c \neq 1$ was 601.12% and 142.73%, respectively, for $p = 0.6$. The percentage improvement in reliability for an FT WSN with $c = 1$ over an NFT WSN and an FT WSN with $c \neq 1$ was 350.18% and 236.22%, respectively, for $p = 0.9$.

Results indicated that our FT model could provide, on average, a 100% MTTF improvement with a perfect fault detection algorithm, whereas the MTTF improvement varied from 95.95% to 1.34% due to a fault detection algorithm's typical poor performance at high sensor failure rates. We also observed that redundancy in WSNs plays an important role in improving WSN reliability and MTTF. Results revealed that just three redundant sensor nodes in a WSN cluster resulted in an MTTF improvement of 103.35% on average. Similarly, redundancy in WSN clusters contributes to reliability and MTTF improvement, and the results indicated that three redundant WSN clusters could improve the MTTF by 52.22% on average. The iso-MTTF results indicated that an NFT WSN cluster required three redundant sensor nodes to attain an MTTF comparable to that of an FT WSN cluster. Iso-MTTF results further indicated that an NFT WSN required nine redundant WSN clusters to achieve an MTTF comparable to that of an FT WSN.

REFERENCES

I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. 2002. Wireless sensor networks: A survey. *Elsevier Computer Networks* 38, 4 (March 2002), 393–422.

Hind Alwan and Anjali Agarwal. 2009. A survey on fault tolerant routing techniques in wireless sensor networks. In *Proceedings of IEEE SENSORCOMM*.

A. Avizienis. 1985. The n-version approach to fault-tolerant software. *IEEE Transactions on Software Engineering* 11, 12 (December 1985), 1491–1501.

Algirdas Avizienis and J. Laprie. 1986. Dependable computing: From concepts to design diversity. *Proceedings of the IEEE* 74, 5 (May 1986), 629–638.

Marco Baldi, Franco Chiaraluce, and Elma Zanaj. 2009. Fault tolerance in sensor networks: Performance comparison of some gossip algorithms. In *IEEE WISES*.

Arpita Bhargava and A. S. Raghuvanshi. 2013. Anomaly detection in wireless sensor networks using s-transform in combination with SVM. In *Proceedings of 5th International Conference on Computational Intelligence and Communication Networks (CICN'13)*. 111–116.

Jonathan Bredin, Erik Demaine, Mohammad Hajiaghayi, and Daniela Rus. 2010. Deploying sensor networks with guaranteed fault tolerance. *IEEE/ACM Transactions on Networking* 18, 1 (February 2010), 216–228.

Wenyu Cai, Xinyu Jin, Yu Zhang, Kangsheng Chen, and Jun Tang. 2006. Research on reliability model of large-scale wireless sensor networks. In *Proceedings of IEEE WiCOM*.

Jinran Chen, Shubha Kher, and Arun Somani. 2006. Distributed fault detection of wireless sensor networks. In *ACM DIWANS*.

M. Chiang, Z. Zilic, J. Chenard, and K. Radecka. 2004. Architectures of increased availability wireless sensor network nodes. In *Proceedings of IEEE ITC*.

Thomas Clouqueur, Kewal Saluja, and Parameswaran Ramanathan. 2004. Fault tolerance in collaborative sensor networks for target detection. *IEEE Transactions on Computers* 53, 3 (March 2004), 320–333.

Min Ding, Dechang Chen, Kai Xing, and Xiuzhen Cheng. 2005. Localized fault-tolerant event boundary detection in sensor networks. In *Proceedings of IEEE INFOCOM*.

Xiaojiang Du and Hsiao-Hwa Chen. 2008. Security in wireless sensor networks. *IEEE Wireless Communications* 15, 4 (August 2008), 60–66.

Saurabh Ganeriwal, Ram Kumar, and M. B. Srivastava. 2003. Timing-sync protocol for sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys'03)*. 138–149.

Xiaofeng Han, Xiang Cao, Errol Lloyd, and Chien-Chung Shen. 2010. Fault-tolerant relay node placement in heterogeneous wireless sensor networks. *IEEE Transactions on Mobile Computing* 9, 5 (May 2010), 643–656.

A. Hopkins, T. Basil Smith, and J. Lala. 1978. FTMP—A highly reliable fault-tolerant multiprocessor for aircraft. *Proceedings of the IEEE* 66, 10 (October 1978), 1221–1239.

Intel. 2009. Intel-Berkeley Research Lab. Available: http://db.csail.mit.edu/labdata/labdata.html.

Gao Jian-Liang, Xu Yong-Jun, and Li Xiao-Wei. 2007. Weighted-median based distributed fault detection for wireless sensor networks. *Journal of Software* 18, 5 (May 2007), 1208–1217.

Mingxing Jiang, Zhongwen Guo, Feng Hong, Yutao Ma, and Hanjiang Luo. 2009. OceanSense: A practical wireless sensor network on the surface of the sea. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom'09)*. Galveston, Texas.

Peng Jiang. 2009. A new method for node fault detection in wireless sensor networks. *Sensors* 9, 2 (May 2009), 1282–1294.

Norman Johnson, Samuel Kotz, and N. Balakrishnan. 1994. *Continuous Univariate Distributions*. John Wiley and Sons, Inc.

Deokwoo Jung, Thiago Teixeira, Andrew Barton-Sweeney, and Andreas Savvides. 2007. Model-based design exploration of wireless sensor node lifetimes. In *Proceedings of the ACM 4th European Conference on Wireless Sensor Networks (EWSN'07)*.

Rajgopal Kannan and Sitharama Iyengar. 2004. Game-theoretic models for reliable path-length and energy-constrained routing with data aggregation in wireless sensor networks. *IEEE Journal on Selected Areas in Communications (JSAC)* 22, 6 (August 2004), 1141–1150.

P. Khilar and S. Mahapatra. 2007. Intermittent fault diagnosis in wireless sensor networks. In *Proceedings of IEEE ICIT*.

Israel Koren and Mani Krishna. 2007. *Fault-Tolerant Systems*. Morgan Kaufmann.

Farinaz Koushanfar, Miodrag Potkonjak, and Alberto Sangiovanni-Vincentelli. 2002. Fault tolerance techniques for wireless ad hoc sensor networks. In *Proceedings of IEEE Sensors*.

Mark Krasniewski, Padma Varadharajan, Bryan Rabeler, Saurabh Bagchi, and Y. C. Hu. 2005. TIBFIT: Trust index based fault tolerance for arbitrary data faults in sensor networks. In *Proceedings of IEEE International Conference on Dependable Systems and Networks (DSN'05)*. 672–681.

Bhaskar Krishnamachari and Sitharama Iyengar. 2004. Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *IEEE Transactions on Computers* 53, 3 (March 2004), 241–250.

M. Lee and Y. Choi. 2008. Fault detection of wireless sensor networks. *Elsevier Computer Communications* 31, 14 (September 2008), 3469–3475.

Chun Lo, J. P. Lynch, and Mingyan Liu. 2013. Distributed reference-free fault detection method for autonomous wireless sensor networks. *IEEE Sensors Journal* 13, 5 (May 2013), 2009–2019.

Arunanshu Mahapatro and M. Pabitra Khilar. 2013. Fault diagnosis in wireless sensor networks: A survey. *IEEE Communications Surveys & Tutorials* 15, 4 (2013), 2000–2026.

A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and Anderson. 2002. Wireless sensor networks for habitat monitoring. In *Proceedings of ACM WSNA*. Atlanta, Georgia.

Xin Miao, Kebin Liu, Yuan He, Dimitris Papadias, Qiang Ma, and Yunhao Liu. 2013. Agnostic diagnosis: Discovering silent failures in wireless sensor networks. *IEEE Transactions on Wireless Communications* 12, 12 (December 2013), 6067–6075.

Azzam Moustapha and Rastko Selmic. 2007. Wireless sensor network modeling using modified recurrent neural networks: Application to fault detection. In *Proceedings of IEEE ICNSC*.

Shoubhik Mukhopadhyay, Curt Schurgers, Debashis Panigrahi, and Sujit Dey. 2009. Model-based techniques for data reliability in wireless sensor networks. *IEEE Transactions on Mobile Computing* 8, 4 (April 2009), 528–543.

Arslan Munir and Ann Gordon-Ross. 2009. An MDP-based application oriented optimal policy for wireless sensor networks. In *Proceedings of IEEE/ACM CODES+ISSS*.

Arslan Munir and Ann Gordon-Ross. 2010. Optimization approaches in wireless sensor networks. In *Sustainable Wireless Sensor Networks*, Winston Seah and Yen Kheng Tan (Eds.). INTECH.

Arslan Munir and Ann Gordon-Ross. 2011. Markov modeling of fault-tolerant wireless sensor networks. In *Proceedings of IEEE International Conference on Computer Communication Networks (ICCCN'11)*.

Arslan Munir, Ann Gordon-Ross, and Sanjay Ranka. 2014. Multi-core embedded wireless sensor networks: Architecture and applications. *IEEE Transactions on Parallel and Distributed Systems* 25, 6 (June 2014), 1553–1562.

NASA. 2011. NASA Kennedy Space Center: NASA Orbiter Fleet. Available: http://www.nasa.gov/centers/kennedy/shuttleoperations/orbiters/orbitersdis.html.

Hoang Nguyen, Anna Forster, Daniele Puccinelli, and Silvia Giordano. 2011. Sensor node lifetime: An experimental study. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom'11)*. Seattle, Washington.

Kevin Ni, Nithya Ramanathan, Mohamed Chehade, Laura Balzano, Sheela Nair, Sadaf Zahedi, Greg Pottie, Mark Hansen, Mani Srivastava, and Eddie Kohler. 2009. Sensor network data fault types. *ACM Transactions on Sensor Networks* 5, 3 (May 2009).

NIST. 2011. Engineering Statistics Handbook: Exponential Distribution. Available: http://www.itl.nist.gov/div898/handbook/apr/section1/apr161.htm.

ns2. 2014. The Network Simulator—ns-2. Available: http://www.isi.edu/nsnam/ns/.

Robin Sahner, Kishor Trivedi, and Antonio Puliafito. 1996. *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package*. Kluwer Academic.

Osman Salem, Alexey Guerassimov, Ahmed Mehaoua, Anthony Marcus, and Borko Furht. 2013. Sensor fault and patient anomaly detection and classification in medical wireless sensor networks. In *Proceedings of IEEE International Conference on Communications (ICC'13)*. 4373–4378.

Arunabha Sen, Bao Shen, Ling Zhou, and Bin Hao. 2006. Fault-tolerance in sensor networks: A new evaluation metric. In *Proceedings of IEEE INFOCOM*.

Kewei Sha and Weisong Shi. 2005. Modeling the lifetime of wireless sensor networks. *Sensor Letters* 3 (2005), 126–135.

Abhishek Sharma, Leana Golubchik, and Ramesh Govindan. 2007. On the prevalence of sensor faults in real-world deployments. In *Proceedings of IEEE SECON*.

SHARPE. 2014. The SHARPE Tool & The Interface (GUI). Available: http://people.ee.duke.edu/~chirel/IRISA/sharpeGui.html.

M. L. Sichitiu and C. Veerarittiphan. 2003. Simple, accurate time synchronization for wireless sensor networks. In *Proceedings of IEEE Wireless Communications and Networking (WCNC'03)*.

J. Sklaroff. 1976. Redundancy management technique for space shuttle computers. *IBM Journal of Research and Development* 20, 1 (January 1976), 20–28.

A. Somani and N. Vaidya. 1997. Understanding fault tolerance and reliability. *IEEE Computer* 30, 4 (April 1997), 45–50.

Luciana Souza. 2007. FT-CoWiseNets: A fault tolerance framework for wireless sensor networks. In *Proceedings of IEEE SENSORCOMM*.

Bo Sun, Lawrence Osborne, Yang Xiao, and Sghaier Guizani. 2007. Intrusion detection techniques in mobile ad hoc and wireless sensor networks. *IEEE Wireless Communications* 14, 5 (October 2007), 56–63.

Yan Sun, Hong Luo, and S. K. Das. 2012. A trust-based framework for fault-tolerant data aggregation in wireless multimedia sensor networks. *IEEE Transactions on Dependable and Secure Computing* 9, 6 (December 2012), 785–797.

Cristian Vasar, Octavian Prostean, Ioan Filip, Raul Robu, and Dan Popescu. 2009. Markov models for wireless sensor network reliability. In *Proceedings of IEEE ICCP*.

John Wensley, L. Lamport, J. Goldberg, M. Green, N. Levitt, P. Melliar-Smith, R. Shostak, and C. Weinstock. 1978. SIFT: Design and analysis of a fault-tolerant computer for aircraft control. *Proceedings of the IEEE* 66, 10 (October 1978), 1240–1255.

Geoffrey Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees. 2006a. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing* 10, 2 (March-April 2006), 18–25.

G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees. 2006b. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing* 10, 2 (March 2006), 18–25.

Wikipedia. 2014. ns (simulator). Available at: http://en.wikipedia.org/wiki/Ns_(simulator).

Michael Winkler, Klaus-Dieter Tuchs, Kester Hughes, and Graeme Barclay. 2008. Theoretical and practical aspects of military wireless sensor networks. *Journal of Telecommunications and Information Technology* (2008), 37–45.

J. Wu, D. Duh, T. Wang, and L. Chang. 2007. On-line sensor fault detection based on majority voting in wireless sensor networks. In *Proceedings of the 24th Workshop on Combinatorial Mathematics and Computation Theory (ALGO'07)*.

Liudong Xing and Howard Michel. 2006. Integrated modeling for wireless sensor networks reliability and security. In *Proceedings of IEEE/ACM RAMS*.

K. Yifan and J. Peng. 2008. Development of data video base station in water environment monitoring oriented wireless sensor networks. In *Proceedings of IEEE ICESS*.

Weiyi Zhang, Guoliang Xue, and Satyajayant Misra. 2007. Fault-tolerant relay node placement in wireless sensor networks: Problems and algorithms. In *Proceedings of IEEE INFOCOM*.

Jin Zhu and Symeon Papavassiliou. 2003. On the connectivity modeling and the tradeoffs between reliability and energy efficiency in large scale wireless sensor networks. In *Proceedings of IEEE WCNC*.

Received December 2013; revised July 2014; accepted September 2014