Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2014, Article ID 323754, 16 pages
http://dx.doi.org/10.1155/2014/323754

Research Article

Energy Efficient Distributed Fault Identification Algorithm in Wireless Sensor Networks

Meenakshi Panda and P. M. Khilar

Department of Computer Science and Engineering, National Institute of Technology, Rourkela 769008, India

Correspondence should be addressed to Meenakshi Panda; meenakshi.nitrkl@gmail.com

Received 29 May 2014; Revised 2 November 2014; Accepted 5 November 2014; Published 1 December 2014

Academic Editor: Liansheng Tan

A distributed fault identification algorithm is proposed here to find both hard and soft faulty sensor nodes present in wireless sensor networks. The algorithm is distributed, self-detectable, and can detect the most common byzantine faults such as stuck at zero, stuck at one, and random data. In the proposed approach, each sensor node gathered the observed data from the neighbors and computed the mean to check whether faulty sensor node is present or not. If a node found the presence of faulty sensor node, then compares observed data with the data of the neighbors and predict probable fault status. The final fault status is determined by diffusing the fault information from the neighbors. The accuracy and completeness of the algorithm are verified with the help of statistical model of the sensors data. The performance is evaluated in terms of detection accuracy, false alarm rate, detection latency and message complexity.

1. Introduction

Recent advances in microsensor technology, low-power very large scale integration (VLSI), and wireless communication have led to the development of distributed wireless sensor network (WSN) [1, 2]. Due to their several popular applications, efficient design and implementation of WSNs become a hot research area in recent years. The major problems such as the limited battery power, less computing capabilities, and inefficient use of communication resources make the WSN deployment challenging particularly when WSN is deployed to function for a long duration. Among these obstacles, the most difficult one is the mysterious data which is sent by a faulty sensor node either to the fusion center or to the neighboring nodes on demand.

During the life span of a WSN, a number of unexpected situations such as the misbehavior of sensor nodes, providing unexpected results and not receiving information from specific sensor nodes, are observed [3–6]. This affects the performance of WSN. This may be caused due to a number of valid reasons such as improper functioning of hardware and software units, malicious interference, battery exhaustion, and natural calamities. In fact, these behaviors of wireless sensor nodes are characterized as Byzantine faulty behavior. Therefore, it is necessary to identify the Byzantine faulty sensor nodes in WSN.

In the literature, the faults in WSN are broadly classified into two types such as hard fault (permanent fault or static fault) [7–12] and soft fault (or dynamic fault) [13–16]. When a sensor node fails to communicate with the rest of the nodes in the network, that node is considered as hard faulty sensor node. When few sensor nodes are able to sense the environment and communicate with the other sensor nodes but transmit some erroneous messages at a particular time, such sensor nodes are considered as soft faulty sensor nodes. The sensor nodes that are capable of transmitting data, receiving data, and computing the desired task correctly are said to be fault free sensor nodes.

Depending on the faulty behavior of the sensor nodes at different time instants, the soft faulty nodes are further classified into three different categories, namely, transient fault [17], intermittent fault [8], and Byzantine fault [18]. When the sensor nodes are subjected to transient fault, they do not perform their desired task for a short duration of time. During this faulty duration, they send some arbitrary data to other sensor nodes instead of the correct value. When a sensor node suffers from intermittent fault, they behave correctly for some period of time and behave incorrectly for some another period of time. It becomes difficult to predict the fault status (faulty or fault free) of a sensor nodes. Many algorithms have been proposed for intermittent fault detection in WSN based on the probabilistic approach [8] in the literature. When the sensor nodes are subjected to Byzantine fault, the faulty node behaves arbitrarily which is also difficult to predict. The Byzantine faults are further classified as stuck at zero, stuck at one, and random fault. The detailed description is explained in fault model under Section 3.

The proposed fault identification algorithm considers the Byzantine behavior of sensor nodes during the diagnosis time. Though many detection algorithms exist in the literature to detect permanent or hard faults in WSNs, but few algorithms consider transient and intermittent faulty sensor nodes [8]. For the best of our knowledge, none of the Byzantine soft fault detection algorithm for WSN has considered the following types of faults such as stuck-at-zero, stuck-at-one, random, and hard faults.

The performance of the sensor network degrades if a faulty sensor node sends erroneous data. In both centralized [19] and distributed approach [20], the information from all the sensor nodes in the network is required for either global decision or estimation (in case of global optimization). If the network is unaware of the faulty sensor nodes, then the fusion center or central processor leads to an erroneous solution due to wrong information from all the faulty sensor nodes.

In the literature, both centralized [19] and distributed soft fault [21, 22] identification methods are proposed. In centralized approach, the sensor nodes send their data over a long distance to the fusion center. Then the fusion center identifies the faulty sensor node using a fault detection algorithm. The major disadvantage of the centralized method is the quick drainage of sensor node's energy due to more communication overhead, specially for the nodes nearer to the fusion center. And this method also contains an ultrareliable node that maintains the status about the entire WSN. In fact, the failure of this node results in catastrophic situation due to single point of failure.

Due to these shortcomings in centralized method, distributed fault detection algorithms are proposed by various researchers. Every node runs the fault detection algorithm by accumulating information from the neighbors and then maintains the fault status of the entire network. The distributed methods available in the literature [21, 22] identify the soft faulty sensor nodes by collecting the information from the neighbors for multiple times. Since each sensor node communicates multiple times with the neighboring sensor nodes, the distributed algorithm requires more energy which makes the algorithm energy inefficient. To minimize the communication overhead, miss prediction of the faulty sensor nodes, and increasing the overall performance of the network, a novel distributed fault identification algorithm is proposed.

In the proposed self-detectable distributed algorithm for identifying the Byzantine soft faulty sensor nodes, every sensor node in the network shares their sensed data to the neighbors and predicts the probable fault status of every other sensor node. After sharing the probable fault status, the majority voting scheme is used for identifying the final fault status of sensor nodes. The proposed approach reduces the communication overhead to make the fault identification algorithm energy efficient. The main contribution of this paper is on (i) the design and evaluation of an efficient distributed self-fault detection algorithm for identifying soft faulty sensor nodes in WSN, (ii) calculating the mean to know the presence of faulty sensor node in the neighborhood which reduces the computational time, (iii) implementation of the algorithms using NS3 [23], and (iv) comparing the performance of the algorithm with the existing algorithm [21, 22].

The remaining part of the paper is organized as follows. In Section 2, the related work presents an exhaustive review about the previous work on soft fault identification. The network model used for the development of the algorithm is discussed in Section 3. The proposed distributed fault identification algorithm is given in Section 4. The analytical model has been provided in Section 5 which proves the correctness of the proposed algorithm. We described the simulation results and compared the performance with the fault identification algorithm in Section 6. Finally, Section 7 concludes the paper with discussions.

2. Related Work

In this section, the work proposed by authors on soft fault identification in WSNs is briefly discussed. The emphasis is given to the distributed soft fault detection methods. A probabilistic soft fault detection scheme in multiprocessor system is proposed in [8]. In this technique, all processors are assigned a specific task. Each processor evaluates the task and sends the result to the processor which is assigned this task. The set of results after evaluating the task are obtained by the processors is known as a syndrome. It performs the above operation repeatedly, collects multiple syndromes, and stores them locally in its memory. Finally, each processor analyzes the syndrome to identify the faulty processors present in the system. This approach requires $O(M^c)$ time where M represents the maximum number of processors tested by a processor and c is the maximum times required to collect the results.

In [10], the authors have proposed a Byzantine soft fault detection algorithm in WSNs. The technique is centralized sequence based soft fault detection approach where the entire rectangular terrain is partitioned into number of subregions. Each subregion is assigned an identifier based on its distance to the rest of the sensor nodes present in the network. When an event occurs, each sensor node sends their sensed data to the fusion center. Then the fusion center reestimates a sequence based on the received signal strength from the received data. This method has many demerits. First, the central node keeps all identifiers of the subregions for which large amount of the memory is required. Since each node required a shortest path for forwarding sensed data to the fusion center so that extra time and message required finding the shortest path for forwarding sensed data to the fusion center. Secondly, during the transmission the signal strength may change due to different environmental causes. Since the signal strength is the major parameter, it may result in error while detecting faults in a dynamic environment.

Since centralized approach requires more communication resources, the algorithm is energy inefficient. Therefore, a distributed soft fault detection scheme for WSNs has been proposed by authors in [17]. Each sensor node performs local comparisons between own sensed data and its neighbor at a particular time instant and is stored locally in a table. This process is repeated for constant (c) times (say) and at each time the comparison results are stored in the table. In the final step, every sensor node calculates own fault status by analyzing the data stored in the table. The disadvantage of this distributed approach is that every sensor node collects data from their neighbors for multiple times which causes more communication overhead.

In [24], a localized fault identification algorithm in WSNs is analyzed. It is a distributed fault identification algorithm, where each sensor node compares its own sensed data with the median of the neighbor's data in order to determine its own status. The performance of localized diagnosis is limited due to the nonuniform nature of the sensor node in WSNs. Chen et al. in [21] have proposed a distributed fault identification algorithm such as that given in [22, 25, 26]. Each sensor node compares its own sensed data with its neighbors and sends back the results to the neighboring nodes. Each sensor node is tagged with a name called likely fault free or likely faulty. Each likely fault free sensor node has been identified as fault free sensor sensor nodes by using some rigid criteria. Finally, the remaining likely fault free or faulty sensor nodes are determined to be fault free or faulty with the help of the known fault free sensors or its own tendency value, respectively. In this algorithm, more communication overhead needed as every sensor node send their data multiple times to its neighbors in order to take a decision.

In [18], a Byzantine fault identification method is proposed where each sensor node sends a set of messages to a group of sensor nodes and also receives messages from the same group. If the number of sending and receiving messages is equal, then the sensor node is identified as fault free; otherwise it is considered as a faulty sensor node. This approach needs multihop communication and requires coordination among the nodes to identify the faulty node. Ssu et. al. presented a fault detection method in WSNs [14] where each sensor node establishes a disjoint shortest path between two sensor nodes [27] and sends their message using the established path. If the node receives the same message which is sent by the node then that node is identified as fault free; otherwise it is labeled as faulty. This approach is used for multihop communication and requires more time to establish a path. However, the proposed distributed method does not need a multihop communication as it uses data from the one hop neighbors only for diagnosis purpose.

In [28], a Byzantine fault diagnosis algorithm in Wireless Sensors Network is proposed which diagnoses the faulty sensor nodes based on the replication of services. This method needs additional information for multiple times from the neighboring sensor nodes to diagnose the Byzantine faulty sensor node. Due to this reason, the energy of the sensor node depletes quickly and as a result life span of the network reduces quickly.

In [13], the authors Geeta et al. have proposed a battery power and interference model based fault tolerance mechanism to identify all the faulty nodes present in WSN. Hand-off mechanism is used to identify those faulty nodes which are going to be hard faulty due to low battery power. When a node suffers from low battery power, it transfers all the services to its neighboring node having highest battery power and remains idle so that quality of the network will not be degraded. Fault tolerance against interference is provided by dynamic power level adjustment mechanism by allocating the time slot to all the neighboring nodes. If a particular node wishes to transmit the sensed data, it enters active status and transmits the packet with maximum power; otherwise it enters into sleep status having minimum power that is sufficient to receive hello messages and to maintain the connectivity. The performance of the algorithm is evaluated in terms of packet delivery ratio, control overhead, memory overhead, and fault recovery delay. In [3], the authors Banerjee et al. have proposed an efficient fault detection algorithm based on the cellular automaton which diagnoses both hard and soft faulty sensor nodes.

In [29], Panda and Khilar have proposed a modified three-sigma edit test based self-fault diagnosis algorithm to diagnose the soft faulty sensor nodes and time out mechanism to identify the hard faulty sensor nodes. In modified three-sigma edit test, the normalized median absolute deviation of neighborhood data for a sensor node is computed to find the fault status of sensor nodes. This approach is not suitable for sparsely deployed sensor node.

The notations and their description which are used for developing and analyzing the proposed DFI algorithm are shown in Notations section.

3. Network Model for Fault Identification

In this section, the network model for the purposed distributed fault identification algorithm is described. The network model refers to system, fault, and radio model for energy calculation. The detailed description is given as follows.

3.1. System Model. Consider a sensor network with N distributed sensor nodes randomly deployed in a terrain of size $R \times R$. Each sensor node s_i, $1 \leq i \leq N$ is located in the two-dimensional Euclidean plane $\mathscr{R}^2$. Each sensor node s_i knows about its position $P_i(x_i, y_i)$, where $0 \leq x_i,\ y_i \leq R$. Sensor node s_i interacts with each other and employs a one-to-many broadcast primitive in their basic transmission mode. Consider all the sensor nodes are homogeneous and have a same transmission range T_r. The sensor network follows disk model [30] in order to generate network topology. The T_r of s_i is the radius of the circle where the node is presented at the center. Figure 1 depicts the arbitrary network topology based on the disk model. $s_1, s_2, \ldots, s_{12}$ are a set of sensor nodes, and

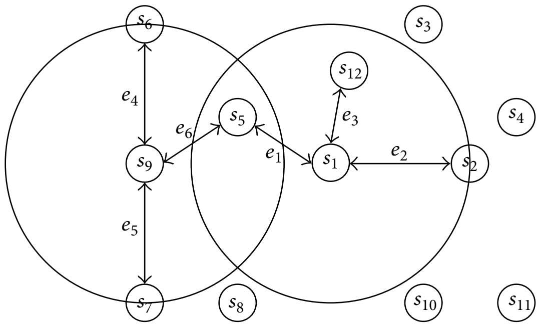


Figure 1: Arbitrary network topology based on unit disk model.

$e_1, e_2, \ldots, e_6$ are the communication links between the sensor nodes. A sensor node s_1 can communicate with its immediate neighbors (s_2, s_5, s_{12}) if the radius of the sensor node s_1 is within T_r. The sensor nodes communicate with each other through an overlapping transmission range so that most of the rectangular terrain can be covered by the deployed sensor nodes. IEEE 802.15.4 is used as the MAC layer protocol to communicate with neighboring nodes. The degree of sensor node s_i is (N_i) which is defined as the number of one hop immediate neighbors associated with it.

3.2. Fault Model. The deployment of sensor nodes in hostile and human inaccessible environment makes the sensor nodes suffer from many environment particle changes for which sensor nodes are subjected to various kinds of faults. In this paper, the sensor nodes are assumed to be suffering from Byzantine faults such as soft and hard fault. The soft fault includes stuck at zero, stuck at one, and random data fault, respectively [31]. A faulty sensor node is subjected to stuck at zero faults, if the value provided by the sensor node remains zero during identification period. When the sensor node provides maximal value (that can be the full scale value) then that type of fault is known as stuck at one. Similarly, in case of random fault, the data provided by a sensor node is random, varies from one time period to another, and ranges from minimum sensing value to maximum sensing value. The soft faulty sensor node suffers with either receiver circuit or sensor circuit failure (environment changes due to alpha and beta particles). However, the hard faulty sensor node suffers from transceiver circuit failure, microcontroller failure, or energy drainage (battery constraint and not rechargeable). The hard faulty sensor node remains silent throughout the life span of the network.

Let the set S_F represents the randomly chosen sensor nodes, which are subjected to either hard or soft fault. More specifically, let S_1, S_2, S_3, and S_4 be the set of randomly chosen sensor nodes suffering from stuck at zero, stuck at one, random, and hard fault, respectively. The fault free sensor nodes in the network are $S_G = S - S_F$, where $S_F = S_1 \cup S_2 \cup S_3 \cup S_4$ and $|S - S_F| \gg |S_1 \cup S_2 \cup S_3 \cup S_4|$ and $N = |S - S_F| + |S_1 \cup S_2 \cup S_3 \cup S_4|$.

The sensor nodes can disseminate its own sensed data to its neighbors Neg_i and also collect the observations $x_j, s_j \in \text{Neg}_i$ from the neighbors at time instant t. In the sensor network, some sensor nodes are subjected to a fault, whereas links are assumed to be fault free. The link faults can be detected by using error detecting and correcting codes which are usually implemented in the underlying networks. The fault free sensor node always provides accurate measured data within acceptable range, whereas faulty sensor node gives arbitrary value in different time.

3.3. Radio Model for Energy Calculation. It is well known that the sensor nodes use low power batteries (at most 1 joule) for data processing and communication. The battery power needs to be utilized in efficient manner. For data communication, each sensor is equipped with a wireless transceiver. Transmitter requires transmitting electronics and amplifier whereas receiver needs only receiving electronics for data transmission. Let, α_1, α_2, and α_3 be the amount of energy required for the transmitting electronics, amplifier, and receiving electronics, respectively. The α_1 and α_3 depend on factors such as the digital coding and modulation, whereas the α_2 depends on the transmission distance and the acceptable bit-error rate. For data transmission and reception, the free space (fs) fading channel models are used because every sensor node needs communication to only their neighboring nodes in a single hop. Depending on the distance between the transmitter and receiver, the free space coefficient is chosen. Let $E_T(m, d)$ and $E_R(m, d)$ be the amount of energy to transmit and receive m bytes of data over an Euclidean distance d. The total amount of energy is the sum of $E_T(m, d)$ and $E_R(m, d)$ which is given in [32]

$$\begin{aligned} E_T(m,d) &= m * (\alpha_1 + \alpha_2 * d^{\alpha}), \\ E_R(m,d) &= m * \alpha_3, \end{aligned} \tag{1}$$

where the free space coefficient α is defined in [33]

$$\alpha = \begin{cases} 2, & d_o \leq d \\ 4, & d_o > d, \end{cases} \tag{2}$$

where d_o is the minimum Euclidean distance between any two sensor nodes.

4. Distributed Fault Identification (DFI) Algorithm

The proposed distributed fault identification (DFI) algorithm based on neighbor coordination approach has two phases such as partial self-fault identification and self-diagnosis phase. In partial self-fault identification phase, every sensor node in the network exchanges their sensed data with the neighboring sensor nodes. The probable fault status of own as well as its neighbors are estimated in this phase. The estimated status are exchanged among themselves in self-diagnosis phase. Each sensor node receives its probable fault status from the neighbors and diffuses the received status. Each sensor node compares its computed status with diffused status to predict its own status. All the notations used for describing the steps of the DFI algorithm are summarized in Notations section. Detailed description of different phases is given below.

4.1. Partial Self-Fault Identification Phase. Every sensor node $s_i \in S$ exchanges their measured data x_i with neighboring nodes Neg_i. Each sensor node s_i keeps the received data from the neighboring nodes Neg_i in Nx_i. After receiving the data, the partial self- and neighboring nodes fault status are computed based on the following observations.

Case 1. The remaining battery power Re_i of a sensor node s_i is computed with a constant battery power ζ to identify the hard faulty sensor node and the value for ζ is constant for all sensor nodes.

Let MinSense and MaxSense be the minimum and maximum sensing value of the sensor node s_i. The value of MinSense and MaxSense is constant and common to all the sensor nodes present in WSN. Cases 2 and 3 are based on MinSense and MaxSense value and used for identifying stuck at zero and stuck at one fault as given below.

Case 2. If the sensed data x_i of the sensor node s_i is MinSense, then the sensor node s_i is suffering from stuck at zero fault.

Case 3. If the sensed data x_i of the sensor node s_i is MaxSense, then the sensor node s_i is suffering from stuck at one fault.

Cases 2 and 3 are based on the fact that when the observed data x_i of a sensor node s_i is the value of either MinSense or MaxSense, the sensor node s_i does not depend on the neighbors to identify its own fault status. However, Case 4 is based on the fact that if the observed data of the sensor node s_i is the value of neither MinSense nor MaxSense, the sensor node s_i needs to find its own status and its neighbors status as the observed data is random between MinSense and MaxSense.

Case 4. If the sensed data x_i of the sensor node s_i is between MinSense and MaxSense, then it performs the operation defined in (3) over the collected data from the neighboring nodes Neg_i and own sensed data x_i to identify self- and neighbors probable fault status:

$$\hat{\mu} = \left(x_i - \frac{1}{N_i} \left(\sum_{s_j \in \mathrm{Neg}_i} x_j \right) \right) \le \lambda_1, \qquad (3)$$

where λ_1 is the threshold value and the optimum value of λ_1 is discussed in Section 5.

When condition (3) is satisfied by s_i then the node s_i and all its neighbors $s_j \in \mathrm{Neg}_i$ are fault free and become the members of set S_G. Otherwise the sensor node s_i and its neighboring nodes are suspected as faulty sensor node. To identify the exact status of sensor node s_i and neighboring nodes Neg_i, the sensor node s_i recomputes over the received data $x_j, x_j \in Nx_i$ to identify the probable faulty sensor nodes. If the data $x_j, x_j \in Nx_i$ matched with MinSense or MaxSense then assign the sensor node s_j to S_2 or S_3, respectively. Otherwise, the following operations over the collected data Nx_i are performed in order to identify the probable fault status of neighboring nodes Neg_i. Case 4 is further partitioned into four subcases which are given below.

Case 4.1 ($|x_i - x_j| > \lambda_1$ and $x_j \le \lambda_2$). In this case, the sensor node s_j is added to the set PFFN_i and the sensor node s_i is detected as faulty sensor node.

Case 4.2 ($|x_i - x_j| > \lambda_1$ and $x_j > \lambda_2$). In this case, both the sensor nodes s_i and s_j are faulty and the sensor node s_j is added to PFN_i.

Case 4.3 ($|x_i - x_j| \le \lambda_1$ and $x_j \le \lambda_2$). In this case, both the sensor nodes s_i and s_j have fault free status and the sensor node s_j is added to PFFN_i.

Case 4.4 ($|x_i - x_j| \le \lambda_1$ and $x_j > \lambda_2$). In this case, the sensor node s_i is fault free, the sensor node s_j is faulty and added to PFN_i.

The test outcome is 0, if a sensor node s_i is found to be fault free after performing partial self-fault identification phase; otherwise it is 1. After performing the self-fault identification phase, self-diagnosis phase is carried out as follows.

4.2. Self-Diagnosis Phase. The self-diagnosis phase is based on the majority voting scheme to diagnose whether a sensor node is faulty or fault free [11]. In this phase, each sensor node s_i exchanges its neighbor status (i.e., 0 or 1) and also receives status from its neighboring nodes Neg_i. Therefore the sensor node s_i predicts its own status by analyzing the status received from its neighboring nodes Neg_i; that is, each sensor node s_i counts number of 0's and 1's it has received. If numbers of 0's at s_i are more than numbers of 1's at s_i, then s_i is diagnosed as fault free and belongs to set S_G; otherwise it is faulty and include to set S_4, respectively.

The detailed description about the algorithm DFI is given in Algorithm 1. The notations used for developing the Algorithm 1 are summarized in Notations section.

4.3. Complexity of the Algorithm DFI. The message complexity, energy complexity, and detection latency are the three important parameters to compare the performance of the proposed DFI algorithm with the existing algorithms Algo1 [21] and Algo2 [22]. The description of these parameters is as follows.

4.3.1. Message Complexity. The message complexity of the algorithm is determined by considering the total number of messages exchanged over the network in both the partial self-fault identification and self-diagnosis phases. In the partial self-fault identification phase, each sensor node s_i exchanges its own sensing data among the neighboring sensor nodes which requires N number of message exchange over the network. In self-diagnosis phase, each sensor node s_i estimates the probable fault status of its neighboring nodes and sends the status information to the neighboring nodes. This requires N number of message is exchanged over the network. To complete the DFI algorithm, total $2N$ number of messages is exchanged over the network. Therefore, the total number of message exchanges is $2N \approx O(N)$.

```
**Data**: $\mathcal{N}_{\mathcal{J}}$ Nodes, $Nx_i$, $Re_i$
**Result**: Calculate $S_1$, $S_2$, $S_3$, $S_4$, and $S_G$
Initialize $S_1 = \phi$, $S_2 = \phi$, $S_3 = \phi$, $S_4 = \phi$, and $S_G = \phi$
**Partial self fault identification Phase**
**If** $Re_i <= \zeta$ **then**
    $S_4 = s_i$;
**else**
    **if** $x_i = MinSense$ **then**
        $S_1 = s_i$;
    **end**
    **if** $x_i = MaxSense$ **then**
        $S_2 = s_i$;
    **end**
    Move to Algorithm 2
**end**
**Self Diagnosis Phase**
$s_i \in S$ send PFN to neighbors $s_j \in \mathrm{Neg_i}$ and receives PFN from $s_j$ which is computed by the neighbors $s_j$.
From received data the sensor node $s_i$ prepares $RS_i$.
**if** $N_z(RS_i) > N_o(RS_i)$ **then**
    Node $s_i$ is detected as fault free sensor node. $S_G = s_i$
**else**
    Node $s_i$ is detected as random faulty sensor node. $S_3 = s_i$
**end**
```

Algorithm 1: Distributed fault identification in WSN.

```
**Data**: $Nx_i$
**Result**: Calculate $S_G$, PFN and PFFN
$S_G = \phi$, PFN $= \phi$ and PFFN $= \phi$
$\mathrm{CRDN}_i = 0$;
**for** $j = 1, \ldots, |\mathrm{Neg}_i|$ and $s_j \in \mathrm{Neg}_i$ **do**
    $\mathrm{CRDN}_i = \mathrm{CRDN}_i + x_j$;
**end**
$\mathrm{CRDN}_i = \mathrm{CRDN}_i / N_i$
**if** $|x_i - \mathrm{CRDN}_i| \leq \theta_1$ **then**
    The node $s_i$ and $s_j \in \mathrm{Neg}_i$ are identified as likely fault free nodes;
    Assign the node $s_i$ to $S_G$
**else**
    **for** $j = 1, \ldots, |\mathrm{Neg}_i|$ **do**
        **if** $x_j = MinSenseor\ x_j = MaxSense$ **then**
            PFFN $= s_j$
        **else**
            **if** $|x_i - x_j| \succ \theta_1$ and $x_j \succ \theta_2$ **then**
                PFFN $= s_j$
            **end**
            **if** $|x_i - x_j| \leq \theta_1$ and $x_j \leq \theta_2$ **then**
                PFN $= s_j$
            **end**
            **if** $|x_i - x_j| \leq \theta_1$ and $x_j \succ \theta_2$ **then**
                PFN $= s_j$
            **end**
            **if** $|x_i - x_j| \succ \theta_1$ and $x_j \leq \theta_2$ **then**
                PFFN $= s_j$
            **end**
        **end**
    **end**
**end**
```

Algorithm 2: Random fault identification algorithm.

Table 1: Comparison study on existing method with proposed scheme.

Comparison parameters	DFI algorithm	Algo1 (paper in [21])	Algo2 (paper in [22])
Message complexity	$O(2N)$	$O(5N)$	$O(3N)$
Detection latency	$2T_{\text{out}}$	$5T_{\text{out}}$	$3T_{\text{out}}$
Energy	$(m+p)N(\alpha_1+\alpha_2 d^{\alpha}+d_i\alpha_3)$	$(2m+3)N(\alpha_1+\alpha_2 d^{\alpha}+d_i\alpha_3)$	$(2m+1)N(\alpha_1+\alpha_2 d^{\alpha}+d_i\alpha_3)$

4.3.2. Energy Complexity. We have calculated the energy requirement of the network to detect the faulty sensor nodes by using the DFI algorithm. As the energy consumed in communication dominates the energy consumption in processing (because of the development of low power VLSI and computing architecture), we have considered only energy complexity due to data transmission and reception during the fault diagnosis of sensor nodes [34]. The DFI algorithm needs message exchange twice by each sensor node. The energy calculation for each message transmission is provided separately.

(A) The Energy Required for Exchanging the Sensed Data x_i. Let $E_1, E_2, \ldots, E_N$ be the energy which dissipates by the sensor nodes $s_1, s_2, \ldots, s_N$, respectively. Let m be the message size of sense data and let T_r (transmission range) be the maximum distance a sensor node can transmit the message. Thus, the amount of energy required by a sensor for the transmission of m bits of message data is

$$E_{Ti}(m, T_r) = m\left[\alpha_1 + \alpha_2 T_r^{\alpha}\right]. \tag{4}$$

This transmission energy is common for all the senor nodes in the network. However, the energy required to receive the data from the neighbors is different, because the degree of the sensor nodes is different. Therefore, the energy required by a sensor node s_i to receive data from all the neighbors is given as

$$E_{Ri}(m, T_r) = N_i m \alpha_3, \tag{5}$$

where N_i is the degree of sensor node s_i.

The total amount of energy required by s_i for data transmission and reception is

$$E1_i(m, T_r) = E_{Ti}(m, T_r) + E_{Ri}(m, T_r). \tag{6}$$

(B) The Energy Required for Exchanging the Probable Fault Status. The sensor node exchanges p bits of information to its neighbors. According to the procedure given in Section 4.3.2(A), the total energy required by s_i here is given as

$$E2_i(p, T_r) = E_{Ti}(p, T_r) + E_{Ri}(p, T_r), \tag{7}$$

where $E_{Ti}(p, T_r) = p[\alpha_1 + \alpha_2 T_r^{\alpha}]$ and $E_{Ri}(p, T_r) = N_i p \alpha_3$.

Therefore, the total energy required for each sensor node to detect soft faulty sensor nodes in the network is given as

$$\begin{aligned} E_i(m+p, T_r) &= E1_i(m, T_r) + E2_i(p, T_r) \\ &= \alpha_1(m+p) + \alpha_2 T_r^{\alpha}(m+p) \\ &\quad + N_i\alpha_3(m+p) \\ &= (m+p)(\alpha_1 + \alpha_2 T_r^{\alpha} + N_i\alpha_3). \end{aligned} \tag{8}$$

The total energy dissipated by the network for identifying the faulty sensor nodes is

$$E_{\text{total}}(m, d) = \sum_{i=1}^{N} E_i(m+p, T_r). \tag{9}$$

4.3.3. Detection Latency. The detection latency is defined as the maximum time required by a network to identify the fault status of every sensor node present in the network. The processing time of the sensor nodes is negligible due to the faster embedded processor and communication time is more than the processing time. Thus, the detection latency of the DFI algorithm is calculated by considering only transmission and reception time that is communication time. Let T_{out} be the maximum time set by the timer by each sensor node while exchanging the data with neighbors. In partial self-fault identification and self-diagnosis phase, each sensor node s_i exchanges one message in each phase. Therefore, the total time required by s_i is $2T_{\text{out}}$. A comparison study between DFI and existing Algo1 and Algo2 algorithms is tabulated in Table 1.

5. Analysis of the Proposed DFI Algorithm

In this section, the proposed DFI algorithm has been mathematically analyzed to ensure the correctness of the proposed approach. In a WSN, every sensor node senses environmental data, converts it into a suitable packet format, and then transmits it to the neighboring nodes or fusion center on demand. While performing this, the noise is likely to be added with sensed data. So, we can mathematically model the sensors measured data as the sum of true value and additive noise. The additive noise is considered as normally distributed Gaussian noise.

It is assumed that all the sensor nodes measured same physical data and few sensor nodes can be faulty. The mean of the measurement for all sensor nodes is constant w, but the noise is different for sensor nodes. The data generated for each sensor node by using normal distribution have w mean and σ_i^2 variance, that is, $\mathcal{N}(w, \sigma_i^2)$. It is a common assumption in WSNs literature that all the sensor nodes measure same physical data with constant mean.

The data model of the sensor node s_i is given as

$$x_i = w + n_i, \quad \text{where } i = 1, 2, 3, \ldots, N, \tag{10}$$

where w is mean of the measured data which is common at all sensor nodes and n_i is the additive noise. Here n_i is assumed to be independent over time and space, respectively. The *probability density function* (pdf) of x_i is given by [35]

$$f_X(x_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-(x_i-w)^2/2\sigma_i^2}, \tag{11}$$

where σ_i^2 is the variance of noise present at the sensor node s_i.

The probability of $x_i \in X$ that lies in the range of $(-\infty, x_i]$ can be expressed in terms of its *cumulative distribution function* (cdf). As sensor data follows a normal distribution its cdf is defined as

$$F(x) = \int_{-\infty}^{x} f_x(y)\,dy = \Phi\left(\frac{x_i - w}{\sigma_i}\right). \tag{12}$$

Now the cdf can be expressed in terms of *error function* (erf). The erf is defined as

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-y^2} dy. \tag{13}$$

The cdf is rewritten in terms of erf as

$$F(x) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x_i - w}{\sigma_i\sqrt{2}}\right)\right], \quad x_i, w \in R. \tag{14}$$

The probability of a random variable x_i lies in between $(w_i - a)$ and $(w + a)$ (where w is the mean of measured data) and is calculated by using its cdf as

$$\begin{aligned} &\operatorname{pr}(w - a \le x_i \le w + a) \\ &\quad = F\left(w + a, w, \sigma_i^2\right) - F\left(w - a, w, \sigma_i^2\right) \\ &\quad = \operatorname{erf}\left(\frac{a}{\sigma_i\sqrt{2}}\right). \end{aligned} \tag{15}$$

As we know that the variance of a random variable indicates the spread of its pdf around the mean, hence it is better if we choose the constant a in terms of variance. For example, if the constant $a = 3\sigma_i$ then the probability of the random variable x_i lies in between $(w - 3\sigma_i)$ to $(w + 3\sigma_i)$ and is

$$\begin{aligned} &\operatorname{pr}(w - 3\sigma_i \le x_i \le w + 3\sigma_i) \\ &\quad = \operatorname{erf}\left(\frac{3\sigma_i}{\sigma_i\sqrt{2}}\right) \quad \text{(From (15))} \\ &\quad = 0.9973. \end{aligned} \tag{16}$$

This reflects that if the variance of the noise at s_i is low, there is a maximum probability to get an error free measurement. If the sensor node is working properly and the transmission in wireless channel is noise free then the variance is very low (around 0.001). In general, maximum noise is added by the channel while transmission if the node is faulty. Let us take the variance of measurement and transmission noise associated with a fault free node is 1. There is 99.73% probability that the value is deviated around ±3. When the node is faulty due to sensor processing error or transmission failure the measured data is corrupted using noise having high variance. In our model, we assumed that the noise variance of faulty node is 100 times as compared to that of fault free sensor node.

We compare any two sensor node data x_i and x_j at the observed time t. The difference $x_{i,j}$ is given as

$$x_{i,j} = x_j - x_i. \tag{17}$$

The sensed data by a sensor node s_i is temporarly and spatially independent from the amount noise associated to the data. Therefore x_i and x_j are independent in nature. From the definition, $x_{i,j}$ is a random with mean 0 and variance $\sigma_{i,j}^2$, respectively, which are calculated as

$$\sigma_{i,j}^2 = \sigma_i^2 + \sigma_j^2, \tag{18}$$

where σ_i, σ_j are the noise variances of sensor nodes s_i and s_j, respectively.

When the sensor nodes are deployed in a particular environment, the sensed data for neighboring sensor nodes are nearly the same. The difference is caused due to additive noise associated with the sensor data. It is considered that every sensor nodes measure same data, which is the mean of the distribution. In general practice, for most applications of WSNs, we need the average of measured data from all the nodes. The theory of statistical estimation provides the mean estimator as the best *minimum variance unbiased* (MVU) estimator. We compared the sensor's own measured data with the mean of neighbor's data for fault identification. Let N_a be the average degree of the sensor nodes in the sensor network. The mean and variance of neighbor data excluding itself is written as

$$\theta_i = \frac{1}{N_a}\sum_{j=1}^{N_a} w_j = w, \qquad \rho_i^2 = \frac{1}{N_a^2}\sum_{j=1}^{C} \sigma_j^2. \tag{19}$$

In fact, there are two cases such that either all neighbor nodes are fault free or some nodes are faulty. In first case, when all the neighboring nodes are fault free having same variance of measurement σ^2 then the mean variance is

$$\rho_i^2 = \frac{\sigma^2}{N_a}. \tag{20}$$

The difference between own measured data of node s_i with the mean of its neighbors data is

$$x_i^m = x_i - \theta_i, \tag{21}$$

which have zero mean and $((1 + N_a)/N_a)\sigma^2$ variance, respectively. Let λ_1 be a constant which is used for comparing the difference such that

$$\left|x_i - \theta_i\right| \le \lambda_1. \tag{22}$$

If we choose the constant $\lambda_1 = 3((1 + N_a)/N_a)\sigma^2$ assuming all neighboring nodes are fault free with respect to node s_i, then there is 99.73% (from (16)) of probability such that the absolute difference is less than λ_1.

In the second case, if any one of the neighboring node is faulty then the mean θ_i remains unchanged as we are assuming all sensor nodes have same measured data. The variance of faulty node is very high compared to fault free node so that $\rho_k^2 >> \sigma^2$. We cannot choose the constant λ_1 which is very high to satisfy the condition in (22) because when the single neighboring node is faulty for high variation of degree the $\rho_i^2 \approx \sigma^2$. It may happen that the faulty node is detected as fault free. Therefore, in that case we may lose the comparison when this comparison equation is not satisfied, and then the ith sensor node compares its data with the neighboring sensor nodes data using another constant λ_2. In this case, if ith node is comparing its own temperature value with a faulty node having variance σ_f^2 which is different from normal variance σ^2, therefore, difference in the variance is given as

$$\sigma_{ij}^2 = \sigma_i^2 + \sigma_j^2 = \sigma^2 + \sigma_f^2. \tag{23}$$

In general, the variance of faulty node is nearly 100 times the variance of fault free nodes. The magnitude of the difference must be compared with a higher threshold. Therefore, in the proposed algorithm, we choose the threshold value of λ_2 as 33σ.

During the comparison process, there are four different situations that may arise, such as (i) both nodes (compared and comparing node) are fault free (ii) both nodes (compared and comparing node) are faulty, (iii) faulty node comparing with fault free node and (iv) fault free node comparing with faulty node. When both the nodes are fault free, then the difference of their variance is very low, so it may always satisfy with the condition for the threshold λ_2 with high probability. If both sensor nodes are faulty with high variance, then the difference is much higher than the threshold λ_2 which indicates that one faulty node can detect the status of another faulty node as faulty. It is trivial that when a fault free node compares with the sensed value of a faulty node it finds a faulty node as faulty. When a faulty node compares with fault free node data, then the faulty node make fault free node as faulty. Due to randomness of data the results are always not 100% accurate, to overcome this particular situation, we employed majority voting on the data collected from different neighboring sensor nodes before taking final decision about the fault status of a node.

6. Simulation Model

In this section, the proposed distributed fault identification (DFI) algorithm is implemented in network simulator NS3 [23] and the performances are compared with the existing algorithms [21, 22]. The network parameters used for developing the network model are given in Table 2. After developing the network model, different types of faults are injected into the network. It is assumed that the occurrences of various faults are independent of each other. The detection accuracy (DA), false alarm rate (FAR), detection latency (DL), and energy consumption (EC) [21, 22] are used for measuring the performance of the algorithms. These parameters are defined as follows.

TABLE 2: The Simulation parameters.

Simulation parameters	
Topologies	Arbitrary network with $N = 512$
Propagation loss model	Range propagation loss model
MAC	IEEE 802.15.4
Simulation time	300 s
T_r	(40, 54, 60, 67) m
Network grid	From (0, 0) to (100, 100) m
α_1	50 nJ/bit
α_2	10 pJ/bit/m^2
α_3	50 nJ/bit

(1) *Detection accuracy (DA)* is defined as the ratio between the number of faulty sensor nodes detected as faulty and the total number of faulty sensor nodes present in the network.

(2) *False alarm rate (FAR)* is defined as the ratio of the number of fault free sensor nodes detected as faulty to the total number of fault free sensor nodes present in the network.

(3) *Detection latency* is defined as the maximum time required by the node to identify its own fault status.

(4) *Energy consumption (EC)* is defined as the total energy consumed by the network to identify the faulty node present in the network.

6.1. Simulation Results. The performance of the proposed algorithm is analyzed and compared with different existing algorithms for different values of the number of sensor nodes N, the average degree of sensor nodes in the network, the probability that a sensor node is faulty p, and the predefined threshold values (λ_1 and λ_2) used for fault detection in the proposed DFI algorithm. After random deployment of the sensor nodes, the topology of the sensor network has been generated using the transmission range of the sensor nodes given in Table 2. The performances are measured by varying the fault probability from 0.05 to 0.4 with step size of 0.05. The threshold values λ_1 and λ_2 used in DFI algorithm are taken as 3 and 40, respectively. All the algorithms (DFI, Algo1, and Algo2) are implemented in NS3 over the fault model discussed in Section 3.

In the simulation model, the data of a fault free sensor node are generated by using *normal distribution* function with mean $w = 30$ and variance $\sigma^2 = 1$. The faulty sensor nodes are assumed to have the same mean as fault free node, but the variance is taken 100.

6.2. Performance of the Algorithm with respect to DA and FAR. The DA and FAR versus the fault probability for different average degree of the algorithms are plotted in Figures 2(a) to 2(d) and Figures 3(a) to 3(d), respectively. As can be seen from

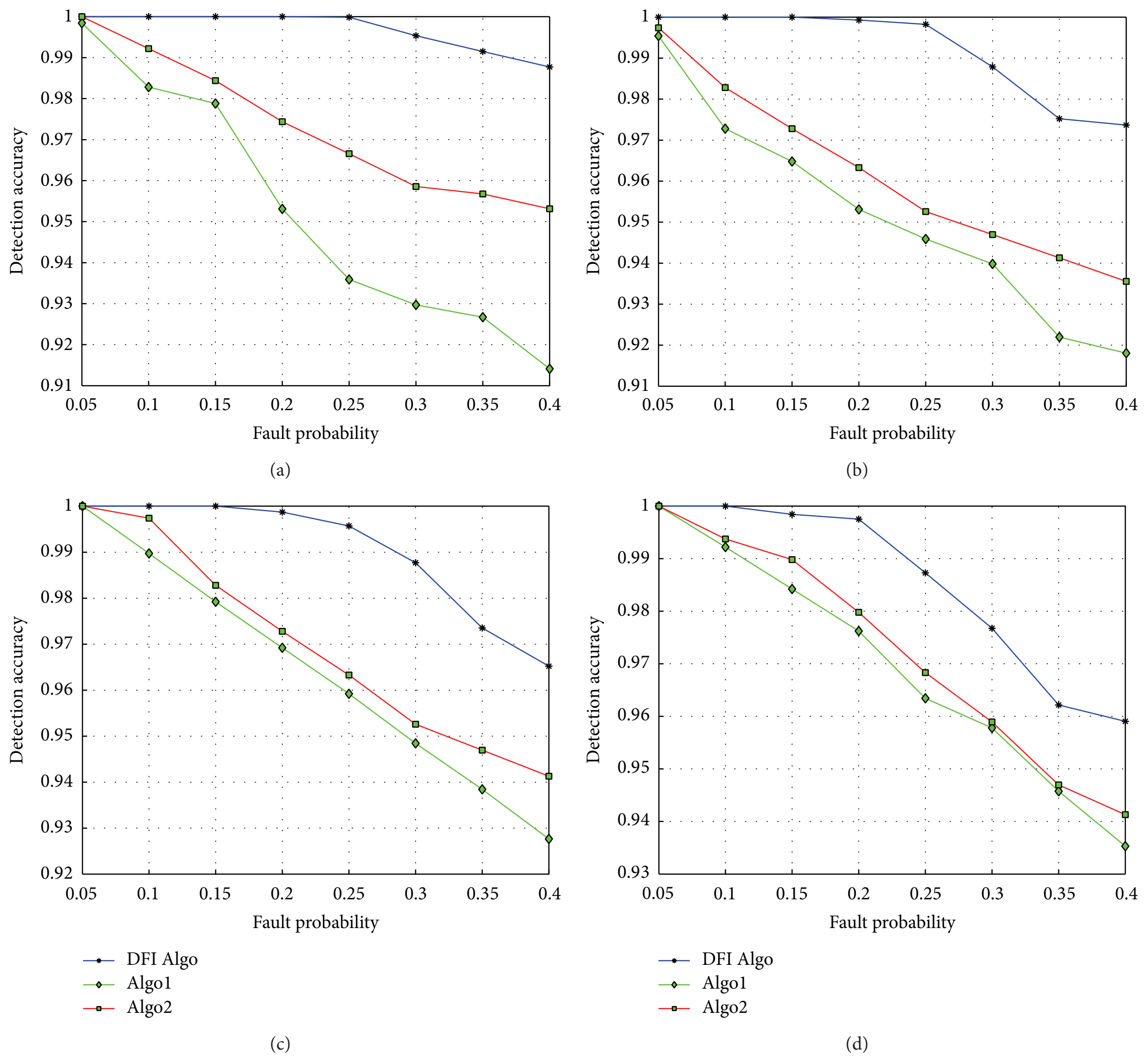


Figure 2: DA versus fault probability plots obtained by the proposed DFI, Algo1, and Algo2 algorithms: (a) for average degree $N_a = 10$; (b) for average degree $N_a = 15$; (c) for average degree $N_a = 20$; (d) for average degree $N_a = 25$.

Figures 2 and 3, the proposed DFI algorithm yields significant superior performance over existing Algo1 [21] and Algo2 [22] algorithm as a whole, by demonstrating higher DA and lower FAR.

The superior performance of the proposed algorithm over the existing algorithm is due to the statistical property of the mean, which is used for comparison of own observation. Further, each node does not take its own fault decision by simply comparing own data with one of the neighbors. Instead, the fault status is diagnosed by each of the neighboring sensor nodes. A voting scheme among the probable fault status measured by the neighboring nodes is used to take the final correct decision about the fault status.

Ideally, the DFI algorithm aims to achieve the 100% DA and 0% FAR, respectively. In Figures 2 and 3, the proposed algorithm attains these ideal performances for lower fault probability but degrades for higher fault probabilities as compared to the existing algorithms which never attain these ideal performance. In the worst case scenario (40% fault probability) the DA and FAR of the proposed algorithm are 0.95 and 0.07, respectively. In the meantime the Algo1 [21] and Algo2 [22] algorithms the value of DA and FAR performances is around 0.92 and 0.35, respectively. The detail comparison of the performance among the algorithms is given in Tables 3 and 4.

6.3. Message Complexity. The message complexity of different algorithms is compared with the DFI algorithm. The total number of messages exchanged in all the algorithms depends upon the number of nodes present in the network. Usually

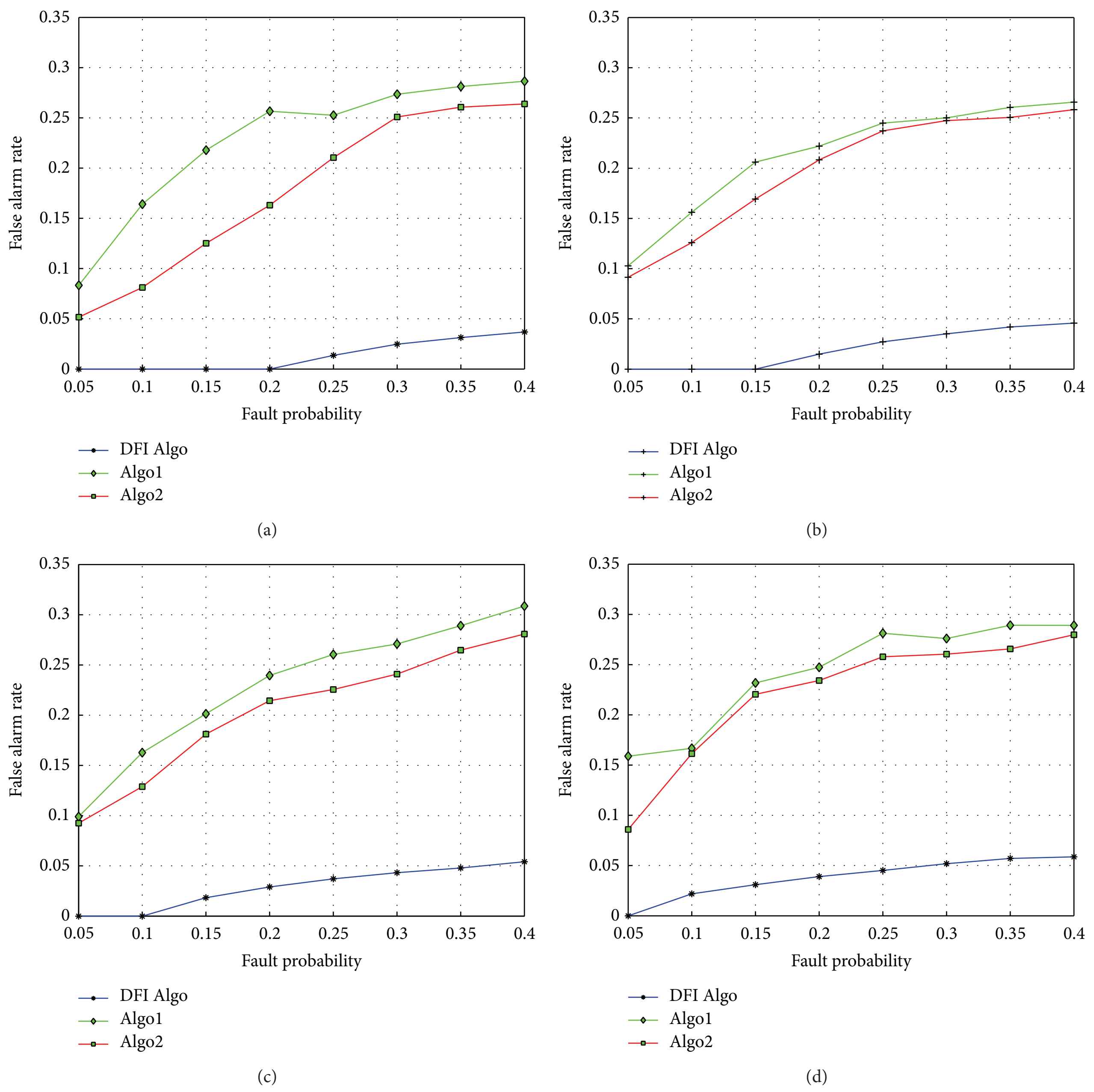


FIGURE 3: FAR versus fault probability plots for the proposed DFI, Algo1, and Algo2 algorithms: (a) for average degree $N_a = 10$; (b) for average degree $N_a = 15$; (c) for average degree $N_a = 20$; (d) for average degree $N_a = 25$.

the message complexity is independent of fault probability, because in soft fault detection method, it is assumed that all the nodes communicate to their neighbors by using one hop communication. The Algo1 [21] and Algo2 [22] algorithms require more message overhead as compared to the DFI algorithm. This is due to the requirement of multiple data from the neighboring nodes for fault detection, unlike the DFI algorithm needs only one data from the neighboring nodes to do the same. In the worst case, Algo1 and Algo2 algorithms need 5 and 3 messages, respectively, to identify the fault status of the faulty sensor node present in the network whereas the DFI needs only 2 messages.

In Table 5, the total number of messages exchanged required by different algorithms for different average degree of the network is provided. We know the communication consumes much power compared to other operation. Further, from Table 5 it is found that the DFI algorithm needs less number of message exchange. Therefore, the proposed DFI algorithm is energy efficient.

6.4. Detection Latency. The detection latency (DL) is one of the important parameters for fault identification algorithm because it provides the time required to detect all the faulty nodes in the network. The DL versus fault probability of

Table 3: Comparison of DA for different algorithms.

Fault prob.	Detection accuracy								
	Avg. degree = 10			Avg. degree = 15			Avg. degree = 25		
	DFI algo.	Algo1 algo.	Algo2 algo.	DFI algo.	Algo1 algo.	Algo2 algo.	DFI algo.	Algo1 algo.	Algo2 algo.
0.05	1	0.9984	1	1	0.9954	0.9974	1	1	1
0.1	1	0.9828	0.9921	1	0.9728	0.9828	1	0.99374	0.9922
0.15	1	0.9788	0.9843	1	0.9648	0.9728	0.9984	0.9898	0.984
0.2	1	0.9531	0.9743	0.9992	0.9531	0.9633	0.9975	0.9798	0.9762
0.25	0.9998	0.9359	0.9665	0.9982	0.9459	0.9525	0.9872	0.9683	0.9634
0.3	0.9953	0.9297	0.9585	0.9878	0.9398	0.9469	0.9767	0.9589	0.9578
0.35	0.9914	0.9267	0.9567	0.9752	0.9219	0.9412	0.9621	0.9469	0.9457
0.4	0.9877	0.9141	0.9531	0.9736	0.9180	0.9355	0.959	0.9412	0.9353

Table 4: Comparison of FAR for different algorithms.

Fault prob.	False alarm rate								
	Avg. degree = 10			Avg. degree = 15			Avg. degree = 25		
	DFI algo.	Algo1 algo.	Algo2 algo.	DFI algo.	Algo1 algo.	Algo2 algo.	DFI algo.	Algo1 algo.	Algo2 algo.
0.05	0	0.0516	0.0833	0	0.0913	0.102	0	0.0859	0.158
0.1	0	0.0812	0.164	0	0.1258	0.156	0.021	0.161	0.166
0.15	0	0.125	0.217	0	0.169	0.206	0.03	0.223	0.231
0.2	0	0.163	0.256	0.014	0.208	0.221	0.072	0.234	0.247
0.25	0.013	0.2104	0.252	0.027	0.2369	0.244	0.039	0.257	0.281
0.3	0.024	0.2508	0.273	0.035	0.247	0.25	0.045	0.26	0.276
0.35	0.031	0.2606	0.281	0.041	0.2504	0.26	0.051	0.265	0.289
0.4	0.036	0.2638	0.286	0.045	0.2581	0.265	0.057	0.279	0.289

Table 5: Total number of message exchanged for different algorithms.

Algorithm	Proposed DFI algorithm	Algo1 algorithm	Algo2 algorithm
Average degree = 10	1024	2560	1536
Average degree = 15	1024	2560	1536
Average degree = 20	1024	2560	1536
Average degree = 25	1024	2560	1536

the DFI, Algo1 [21], and Algo2 [22] algorithms for different average node degree is plotted in Figure 4. From the figures it has been shown that the DFI algorithm has less DL as compared to Algo1 and Algo2 algorithm. It is because the DFI needed less message exchange and the DL fully depends on the number of message exchange. It remains almost same for varying fault probability because each node including the soft faulty node is involved in message exchange and fault diagnosis.

6.5. Energy Consumption (EC). Figure 5 depicts the total energy consumed in the network for fault identification by the DFI, Algo1 [21], and Algo2 [22] algorithms for different fault probabilities. The result shows that as average degree of the network increases the energy consumption increases. The energy consumption in DFI is 28% and 56% less consumed as compared to that of Algo2 and Algo1 algorithm, respectively. The number of message receptions is varied due to packet loss in the network for a fixed number of message transmission. As more energy is required for message transmission than reception, the DFI requires less number of messages for transmission and thereby consuming less energy compared to existing Algo1 and Algo2 algorithms. It is noted that the DFI algorithm does not use any special message for identifying the faulty nodes rather the message containing observed data for the sensor nodes is utilized for fault identification purposes.

7. Conclusion

A novel distributed fault identification algorithm for wireless sensor networks is proposed. The distributed algorithm is based on a Byzantine fault model such as stuck at zero, stuck at one, random, and hard fault. In the proposed algorithm, each sensor node gathered the information from the one hop neighbors and then detect the probable fault status of each of the neighboring sensor node. The final fault status

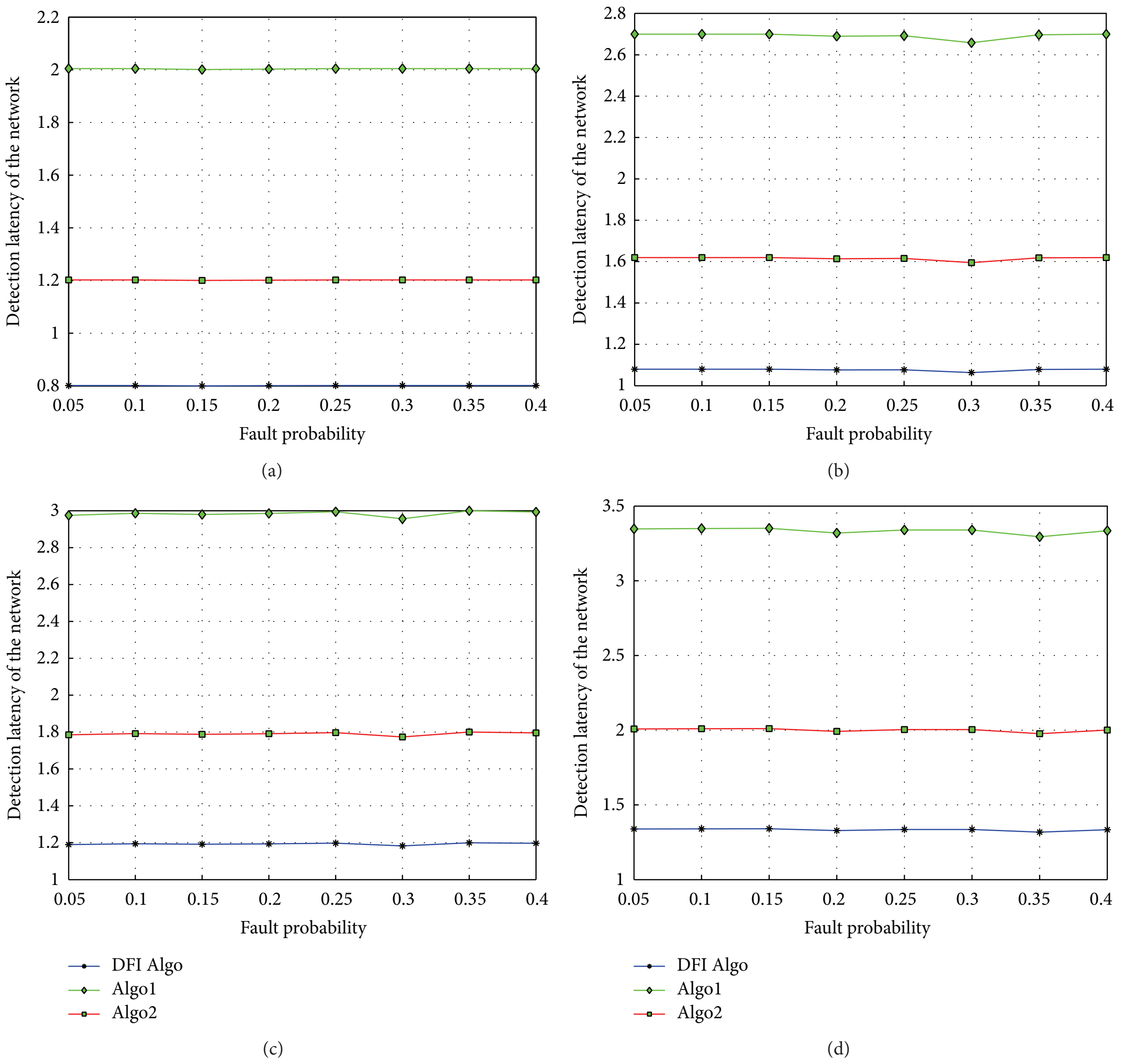


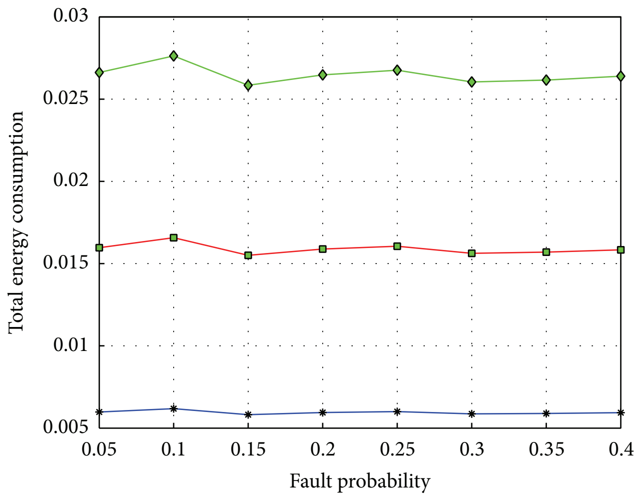
FIGURE 4: Detection latency versus fault probability for the DFI, Algo1, and Algo2 algorithms: (a) for average degree $N_a = 10$; (b) for average degree $N_a = 15$; (c) for average degree $N_a = 20$; (d) for average degree $N_a = 25$.

is computed by diffusing the probable fault information received from the neighboring nodes. The accuracy and completeness of the proposed algorithm have been analyzed and proved to be correct.

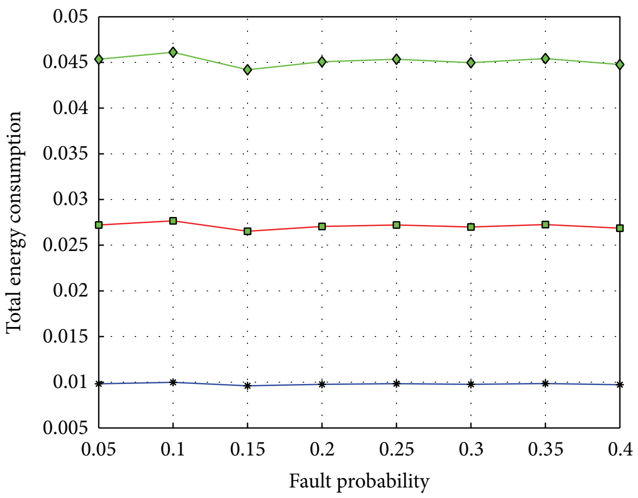
From the simulation, it is observed that the performance of the DFI algorithm is much better than the existing algorithm. The detection accuracy and false alarm rate of the proposed algorithm for lower degree and less fault probability are nearly 1 and 0, respectively, but degrade for higher fault probability. The message and time complexity are very less compared to other existing algorithms. In future, instead of comparing the observation with the mean of their neighbor, we may use robust statistical measure to detect the faulty sensor nodes.

Notations

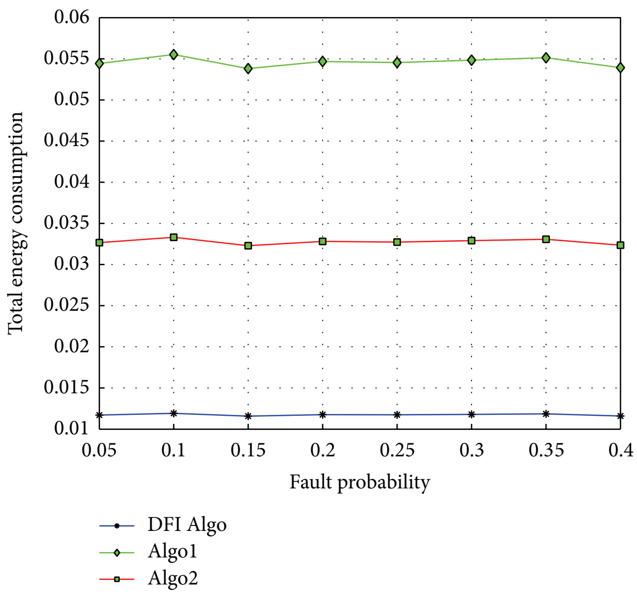
S:	Set of sensor nodes in the sensor network
s_i:	A sensor node deployed at $P_i(xco_i, yco_i)$, $s_i \in S$
N:	Total number of sensor nodes deployed
Neg_i:	A set contains all the neighboring sensor nodes of s_i
CRDN_i:	Cumulative sum of received data of all neighbors of sensor node s_i
θ_1, θ_2:	Threshold value used by each sensor node for estimating the status of the neighboring sensor nodes and itself
PFFN:	A set contains probable fault free sensor nodes estimated by s_i



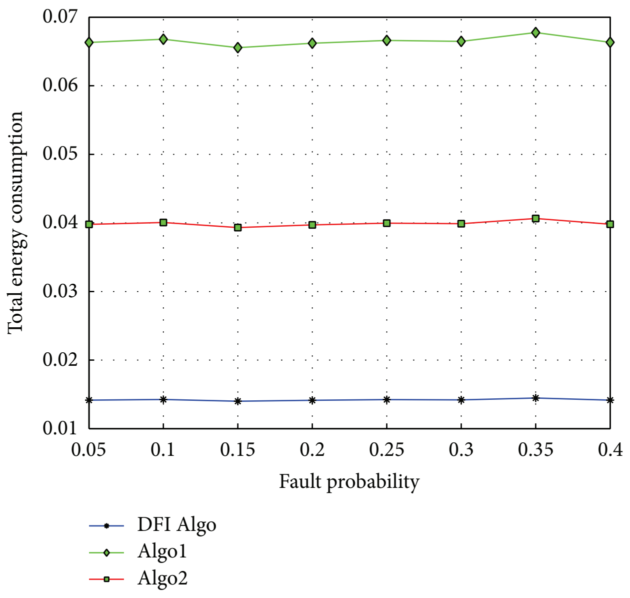
(a)



(b)



(c)



(d)

FIGURE 5: Total energy consumption versus fault probability for the DFI, JSA, and Jiang algorithms: (a) for average degree $N_a = 10$; (b) for average degree $N_a = 15$; (c) for average degree $N_a = 20$; (d) for average degree $N_a = 25$.

PFN:	A set contains probable faulty sensor nodes estimated by s_i
RS_i:	A set contains the status of s_i calculated by $s_j \in \text{Neg}_i$
$Nz(RS_i)$:	Number of zeros present in the set RS_i
$No(RS_i)$:	Number of ones present in the set RS_i
x_i:	Sensed data of sensor node s_i
MaxSense:	Maximum sensing value of the sensor node
MinSense:	Minimum sensing value of the sensor node
$G(S, C)$:	An undirected graph describing the interconnection among the sensor nodes to form an arbitrary network topology
C:	Set contains all the communication edges between the sensor nodes present in S
T_r:	Transmission range of each sensor s_i which is constant for all sensor nodes present in the sensor network
S_1:	Set of sensor nodes suffering from stuck at zero fault
S_2:	Set of sensor nodes suffering from stuck at one fault
S_3:	Set of sensor nodes suffering from random fault
S_4:	Set of sensor nodes suffering from hard fault
S_F:	Set of all faulty sensor nodes, $S_F = S_1 \cup S_2 \cup S_3 \cup S_4$
S_G:	Set of fault free sensor nodes
N_i:	Degree of the sensor node s_i
N_a:	Average degree of sensor nodes in the network
Nx_i:	A set contains received data from the neighbors Neg_i of s_i

ζ: The threshold for energy at which a sensor node s_i works normally
Re_i: Remaining battery power of the sensor node s_i.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–105, 2002.

[2] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[3] I. Banerjee, P. Chanak, H. Rahaman, and T. Samanta, "Effective fault detection and routing scheme for wireless sensor networks," *Computers and Electrical Engineering*, vol. 40, no. 2, pp. 291–306, 2014.

[4] P. Barooah, H. Chenji, R. Stoleru, and T. Kalmár-Nagy, "Cut detection in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 3, pp. 483–490, 2012.

[5] E. P. Duarte Jr., A. Weber, and K. V. O. Fonseca, "Distributed diagnosis of dynamic events in partitionable arbitrary topology networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 8, pp. 1415–1426, 2012.

[6] C.-H. Tsai, "A quick pessimistic diagnosis algorithm for hypercube-like multiprocessor systems under the PMC model," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 259–267, 2013.

[7] M. Barborak, A. Dahbura, and M. Malek, "The consensus problem in fault-tolerant computing," *ACM Computing Surveys*, vol. 25, no. 2, pp. 171–220, 1993.

[8] D. M. Blough, G. F. Sullivan, and G. M. Masson, "Intermittent fault diagnosis in multiprocessor systems," *IEEE Transactions on Computers*, vol. 41, no. 11, pp. 1430–1441, 1992.

[9] S. Chessa and P. Santi, "Crash faults identification in wireless sensor networks," *Computer Communications*, vol. 25, no. 14, pp. 1273–1282, 2002.

[10] S. Guo and Z. Zhong, "FIND: faulty node detection for wireless sensor networks," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*, pp. 1–14, Berkeley, Calif, USA, November 2009.

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

[12] S. Mallela and G. M. Masson, "Diagnosable systems for intermittent faults," *IEEE Transactions on Computers*, vol. 27, no. 6, pp. 560–566, 1978.

[13] D. D. Geeta, N. Nalini, and R. C. Biradar, "Fault tolerance in wireless sensor network using hand-off and dynamic power adjustment approach," *Journal of Network and Computer Applications*, vol. 36, no. 4, pp. 1174–1185, 2013.

[14] K.-F. Ssu, C.-H. Chou, H. C. Jiau, and W.-T. Hu, "Detection and diagnosis of data inconsistency failures in wireless sensor networks," *Computer Networks*, vol. 50, no. 9, pp. 1247–1260, 2006.

[15] Z. You, X. Zhao, H. Wan, W. N. N. Hung, Y. Wang, and M. Gu, "A novel fault diagnosis mechanism for wireless sensor networks," *Mathematical and Computer Modelling*, vol. 54, no. 1-2, pp. 330–343, 2011.

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

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

[18] A. Haeberlen, P. Kouznetsov, and P. Druschel, "The case for byzantine fault detection," in *Proceedings of the 2nd conference on Hot Topics in System Dependability (HOTDEP '06)*, vol. 2, p. 5, 2006.

[19] B. C. P. Lau, E. W. M. Ma, and T. W. S. Chow, "Probabilistic fault detector for wireless sensor network," *Expert Systems with Applications*, vol. 41, no. 8, pp. 3703–3711, 2014.

[20] T.-Y. Wang, L.-Y. Chang, D.-R. Duh, and J.-Y. Wu, "Distributed fault-tolerant detection via sensor fault detection in sensor networks," in *Proceedings of the 10th International Conference on Information Fusion*, pp. 1–6, July 2007.

[21] J. Chen, S. Kher, and A. Somani, "Distributed fault detection of wireless sensor networks," in *Proceedings of the Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks (DIWANS '06)*, pp. 65–71, ACM, Los Angeles, Calif, USA, September 2006.

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

[23] Network Simulator, http://www.nsnam.org.

[24] M. Ding, D. Chen, K. Xing, and X. Cheng, "Localized fault-tolerant event boundary detection in sensor networks," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '05)*, vol. 2, pp. 902–913, March 2005.

[25] M. Panda and P. M. Khilar, "Distributed soft fault detection algorithm in wireless sensor networks using statistical test," in *Proceedings of the 2nd IEEE International Conference on Parallel, Distributed and Grid Computing (PDGC '12)*, pp. 195–198, December 2012.

[26] M. Panda and P. M. Khilar, "Energy efficient soft fault detection algorithm in wireless sensor networks," in *Proceedings of the 2nd IEEE International Conference on Parallel, Distributed and Grid Computing (PDGC '12)*, pp. 801–805, Solan, India, December 2012.

[27] T. J. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, McGraw-Hill, Palo Alto, Calif, USA, 2nd edition, 1990.

[28] R. Klempous, Ł. Radosz, J. Nikodem, and N. Raus, "Byzantine algorithms in wireless sensors network," in *Proceedings of the International Conference on Information and Automation (ICIA '06)*, pp. 319–324, Shandong, China, December 2006.

[29] M. Panda and P. Khilar, "Distributed self fault diagnosis algorithm for large scale wireless sensor networks using modified three sigma edit test," *Ad Hoc Networks*, 2014.

[30] X.-Y. Li, P.-J. Wan, and O. Frieder, "Coverage in wireless ad hoc sensor networks," *IEEE Transactions on Computers*, vol. 52, no. 6, pp. 753–763, 2003.

[31] D. Armstrong, "On finding a nearly minimal set of fault detection tests for combinational logic nets," *IEEE Transactions on Electronic Computers*, vol. 15, no. 1, pp. 66–73, 1966.

[32] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS ’00)*, vol. 8, pp. 8020–8030, 2000.

[33] Y. J. Zhao, R. Govindan, and D. Estrin, “Residual energy scan for monitoring sensor networks,” in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC ’02)*, vol. 2, pp. 256–262, 2002.

[34] L. Alazzawi and A. Elkateeb, “Performance evaluation of the WSN routing protocols scalability,” *Journal of Computer Systems, Networks, and Communications*, vol. 2008, Article ID 481046, 9 pages, 2008.

[35] S. M. Kay, *Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory*, PHI Publication, 1st edition, 1993.