Journal of Network and Computer Applications 78 (2017) 267–287

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

An analysis of fault detection strategies in wireless sensor networks

Thaha Muhammed*, Riaz Ahmed Shaikh

Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

ARTICLE INFO

MSC:
23-557

Keywords:
Wireless sensor networks
Fault-detection
Taxonomy
Faults

ABSTRACT

Wireless sensor networks have emerged as a key technology which is used in many safety critical applications. The sensors in wireless sensor network have to be deployed in hostile, harsh and unattended environments for long periods of time. This creates a great challenge in providing a good quality of service. This results in introductions of faults, sensor failures, communication failures and changes in topology. Hence, efficient fault detection techniques are required for good quality of service. In this article, we survey various fault detection techniques and provide a new taxonomy to integrate new fault detection techniques. We perform a qualitative comparison of the latest fault detection algorithms. From a qualitative analysis, we select a list of techniques that are analyzed quantitatively. We also discuss the shortcomings, advantages and future research directions for fault detection in wireless sensor networks.

1. Introduction

Wireless sensor networks consist of a large number of independent sensor nodes connected together to form a network. Each of these individual sensor nodes in a wireless sensor network has sensing and processing capability. Wireless communication is used as a medium for communicating between the nodes. It forms an ad hoc network with peer–peer communication. Wireless sensors are low power devices with limited computational power, memory, battery and storage. They are deployed in hostile and harsh conditions to report time-critical events such as landslide monitoring (Ramesh, 2014), agricultural monitoring (Pantazis et al., 2013), military operations (Akyildiz et al., 2002), infrastructure monitoring, scientific data collection, intruder detection system (Casey et al., 2008), navigation (Varshney, 2004), and environmental monitoring (Casey et al., 2008). Sensor networks are also used to monitor the health of patients in hospitals (Kim and Prabhakaran, 2011).

Since, sensor networks are deployed in hostile and harsh conditions, e.g., rain, snow, wind, thunder, etc., they are susceptible to frequent and unexpected errors. The faults may be due to hardware or software failure. These faults result in erroneous results in normal operation. The occurrence of faults during normal operation results in harsh consequences involving loss of human life, economic and environmental loss as sensor networks are used in safety catastrophic disasters. For example, if the wireless sensor detecting the activity of the volcano malfunctions and gives incorrect readings, it might result in unneeded panic or loss of lives due to the absence of warning.

The presence of faults in wireless sensor data, may increase the network traffic, decrease the fault detection efficiency of the base station and wastes the battery and power. They need to conserve battery as they are supposed to operate for periods of time ranging from hours to years. Moreover, replacing the battery is not feasible since sensors are normally deployed at inaccessible locations. The transmission of collected data from nodes to sink is an expensive process and results in congestion (Ni and Pottie, 2012, Yu et al., 2007; Liu et al., 2006; Rosberg et al., 2010; Anisi et al., 2013). The presence of faulty sensor nodes increases the congestion by transmitting unusable and misleading data. Therefore, fault detection techniques are required for proper management of the sensor network. Fault detection algorithms enhance bandwidth utilization and data reliability. However, the energy consumption by the nodes increases due to complex fault detection techniques. Hence, there is a tradeoff between conservation of energy versus maintaining a high Quality of Service (Yu et al., 2007). Fault detection techniques detect the faulty node in a wireless sensor network and create a record of all faulty nodes, which can be used for fault recovery of the node or replacement of the faulty node or isolate faulty sensor nodes from the network (Mahapatro and Khilar, 2013).

Yu et al. (2007) briefly discuss fault management in WSN. The fault detection strategy classification provided in Yu et al. (2007) is outdated. Moreover, it does not incorporate a discussion of faults in WSN. A discussion on the faults in WSN can be found in Jurdak et al. (2011) where they provide a classification of faults based on the source of faults. Jurdak et al. (2011) provide insight about the suitability of

* Corresponding author.
E-mail addresses: m.thaha.h@ieee.org (T. Muhammed), riaz289@gmail.com (R.A. Shaikh).

http://dx.doi.org/10.1016/j.jnca.2016.10.019
Received 27 October 2015; Received in revised form 20 April 2016; Accepted 24 October 2016
Available online 01 November 2016
1084-8045/

fault detection tools for various faults along with the usability of fault detection tools. The discussion of fault detection tools and techniques in existing surveys is too brief and concise to be constructive. Moreover, the classification of fault detection techniques provided is limited. Although existing classifications provide the basic architecture, they do not provide a further classification based on latest fault detection techniques in WSN. Mahapatro and Khilar (2013) also discuss various fault detection techniques in WSN but none of these surveys provide a quantitative analysis.

In this survey, we provide a discussion on latest fault detection techniques based on our proposed taxonomy. We also classify and mathematically model the faults that occur in wireless sensor networks. By analyzing the existing literature work, we outline prospective future research challenges and issues. We qualitatively and quantitatively analyze and compare various latest fault detection schemes for wireless sensor networks. We use the Intel-Berkeley data set (http://db.csail.mit.edu/labdata/labdata.html) for quantitative analysis. To the best of our knowledge, this survey is the first of its kind to provide *quantitative comparisons* in this area of research.

The rest of the paper is organized as follows. In Section 2, we discuss various faults that occur in WSN based on proposed taxonomy. In Section 3, we propose a new technique based taxonomy for wireless fault detection. Thereafter, in Section 4 we discuss the state of art fault detection techniques. We present a quantitative and qualitative analysis of various fault detection techniques in Section 5. In Section 6 we outline future research issues and directions for fault detection in WSN and finally Section 7 summarizes our observations on current state and trends in fault detection techniques of WSN.

2. Fault classification and modeling

Faults in WSN can be classified by two aspects: the time span of the fault and the locality of the fault. The timespan of the fault indicates the duration of the faults. Some faults are temporary faults and occur for a certain duration. Hence, based on the time span of the fault, faults can be further classified into two categories: (1) persistent faults and (2) transient faults. Transient faults are temporary faults that occur due to certain conditions such as network congestion, changing weather conditions, etc. These faults are not permanent and disappear after a short time. Fault detection techniques for detecting transient faults have been discussed in Sharma and Sharma (2016), Mahapatro and Panda (2014), and Sahoo and Khilar (2014a,b). Persistent faults are the faults that are permanent; these faults exist until a fault recovery is performed. In a majority of cases, the faults are local to specific component of WSN. Faults are extremely local in the network and only affect a few components of the network (Kutten and Peleg, 1999). Generally, the entire WSN is not faulty. Faults only affect a small number of components of the network instead of the whole global network. Due to the increasing reliability of the networks, the growth of the faults is slower than that of network size (Kutten and Peleg, 1995). Hence, the detection has to be based on the locality than the entire global system. Therefore, by fault locality, we classify the faults broadly into two categories: (1) data-centric and (2) system-centric. The fault classification is not disjoint and can overlap with each other.

Let us consider the data originating from a sensor node be modeled as a time series, $d(n, t, f(t))$, where n is the node id, t is the instance of time in which the value was sensed, and $f(t)$ represents the value sensed by node n during the time instance t. The $f(t)$ can be modeled as $\alpha + \beta x + \eta$, where α is an additive constant called offset, β is a multiplicative constant called gain, x is the non-faulty sensor value at time t, and η is the external noise in the data. In an ideal case, $f(t)$ will be x but in real world cases a fault-free node will have $f(t) = x + \eta$.

2.1. Data-centric perspective

Data-centric perspective takes into consideration the characteristics of the sensed data in determining faults. They are also called as soft faults. It can be categorized into various categories as shown in Fig. 1.

2.1.1. Offset fault

Offset fault refers to a deviation in sensed data by an additive constant from the expected data. This might occur due to improper calibration of the sensor. An offset fault can be modeled as $x' = \alpha + x + \eta$, where $x' \in f(t)$ and α is the constant value that gets added to the normal reading. Offset fault is depicted in Fig. 4a. Techniques for detecting offset faults have been discussed in Warriach and Tei (2013), Feng et al. (2014), Panda and Khilar (2012, 2014, 2015), Mo et al. (2015), Panda et al. (2014), Saihi et al. (2013), Obst (2014), Banerjee et al. (2011), Nitesh and Jana (2015), Sharma and Sharma (2016), Titouna et al. (2015a,b), Abid et al. (2015), Ghorbel et al. (2015), and Nguyen et al. (2013).

2.1.2. Gain fault

A fault is said to be a gain fault if the rate of change of sensed data does not match with expectation over an extended period of time. In gain fault, a constant value gets multiplied to the non-faulty sensor data. This also might be caused by improper calibration of the sensors. A gain fault can be modeled as $x' = \beta x + \eta$, where $x' \in f(t)$ and β is the constant value that gets multiplied to the normal reading. Gain fault is depicted in Fig. 4b. Warriach and Tei (2013), Feng et al. (2014), Panda and Khilar (2012, 2014, 2015), Mo et al. (2015), Panda et al. (2014), Saihi et al. (2013), Obst (2014), Banerjee et al. (2011), Nitesh and Jana (2015), Nguyen et al. (2013), and Titouna et al. (2015a,b) have discussed techniques to detect gain faults in WSN.

2.1.3. Stuck-at fault

A fault is said to be a stuck-at fault when the difference or the variance of data from the data series of a node is zero which implies that the sensed data is constant. A stuck-at fault can be either transient or persistent. A stuck-at fault can be modeled as $x' = \alpha$, where $x' \in f(t)$ and α is the constant value that is sensed. Stuck-at fault found at node15 from the Intel-Berkeley data set (http://db.csail.mit.edu/

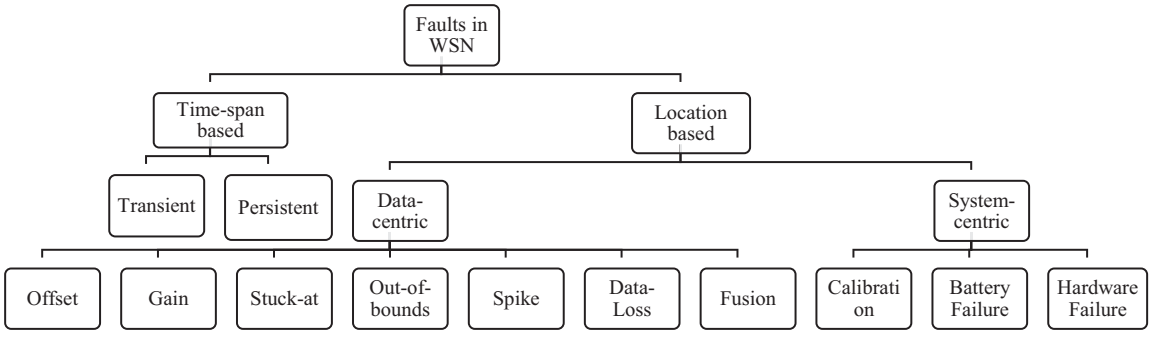


Fig. 1. Taxonomy of faults in WSN.

labdata/labdata.html) is depicted in Fig. 4c. Stuck at fault can be further classified as stuck-at-one and stuck-at-zero. When α is the maximum value that can be sensed by the sensor, it is called as stuck-at-one and when it is minimum, it is called stuck-at-zero. Some of the recent techniques that detect this work has been discussed in Warriach and Tei (2013), Feng et al. (2014), Mo et al. (2015), Obst (2014), Sharma and Sharma (2016), Guo et al. (2014), and Nguyen et al. (2013).

2.1.4. Out of bounds

A fault is said to be an out of bound fault if the sensed data lies beyond the thresholds defined by the problem requirement. A node is said to have out of bounds error if $x' > \theta$ or $x' < \theta_1$, where $x' \in f(t)$, θ and θ_1 are the application thresholds. Out of bound fault is depicted in Fig. 2. Schemes to detect out of bound faults have been discussed in Warriach and Tei (2013), Panda and Khilar (2014, 2015), Mo et al. (2015), Obst (2014), Banerjee et al. (2011), Sharma and Sharma (2016), Titouna et al. (2015a,b), Abid et al. (2015), Guo et al. (2014), and Nguyen et al. (2013).

2.1.5. Spike faults

If the rate of change of measured time series with the predicted time series is more than the acceptable changing trend, then the faults are said to be spike faults. If $|f(t) - f^p(t)|/t > \lambda$, where λ is the normal changing trend, and $f^p(t)$ is the predicted time series data at time t, then it is said to be spike fault. Detection of spike faults has been discussed in Warriach and Tei (2013), Panda and Khilar (2014), Mo et al. (2015), and Obst (2014).

2.1.6. Data loss fault

A data loss fault occurs when sensed data is missing from the time series for a given node. If $f(t) = \Phi \ \& \ t > \tau$, where Φ is the null set and τ is the maximum time required for sensing, then it is called data loss error. Data loss fault is depicted in Fig. 4d. Techniques to identify data loss faults have been considered in Warriach and Tei (2013), Jin et al. (2015), Mo et al. (2015), Nguyen et al. (2013), Ni and Pottie (2012), and Yuvaraja and Sabrigiriraj (2015).

2.1.7. Aggregation/fusion error

An aggregation error is said to occur when

$$(\sum |f(t) - f^p(t)|/t > \lambda^*) \& (|f(t) - f^p(t)|/t < \lambda) \quad (1)$$

where λ^* is the acceptable total error bound and λ is the normal changing trend. Detection of such errors can utilize techniques discussed in Xu et al. (2014), Sahoo and Khilar (2014a,b), Huang (2015), and Sharma and Sharma (2016).

2.2. System-centric perspective

The system-centric classification considers the characteristics and properties of the system that is used in the WSN. They are also known as hard faults. It can be classified into three categories as shown in Fig. 1.

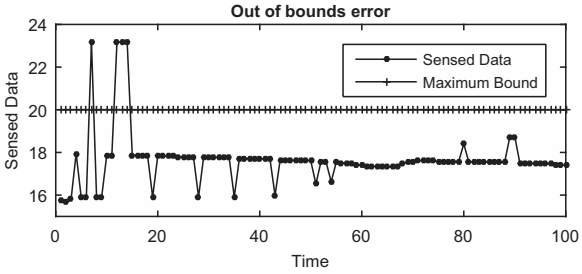


Fig. 2. Out of bounds error in WSN.

2.2.1. Calibration fault

Calibration is a major cause for faults in WSN. Many papers talk about the difficulty in calibrating the sensors. Calibration errors give rise to gain faults, drift faults and offset faults. Drift faults occur when the performance drifts away from original calibration formulas. Gain faults and offset faults have been discussed along with data-centric errors. Drift faults have been discussed in the literatures (Balzano and Nowak, 2007; Buonadonna et al., 2005; Bychkovskiy et al., 2003; Ramanathan et al., 2006; Nguyen et al., 2013).

2.2.2. Battery failure

Battery failure is a major cause for faulty data (Szewczyk et al., 2004). Depletion of the batteries leads to transmission of faulty data by the sensors. In Tolle et al. (2005), the authors conclude that most of the faults in the sensor data are caused by battery failures. Fig. 3 depicts the graph of sensed data in relation with the battery voltage of node 15 in the Intel-Berkeley WSN (http://db.csail.mit.edu/labdata/labdata.html). We can observe that when the battery starts depleting the data started exhibiting stuck-at fault. Battery failure detection has been discussed in Warriach and Tei (2013), Lau et al. (2014), Mo et al. (2015), Panda and Khilar (2014), Banerjee et al. (2011), Jin et al. (2015), Yang et al. (2014), Nguyen et al. (2013), Kaur and Sharma (2010), and Yuvaraja and Sabrigiriraj (2015).

2.2.3. Hardware failure

Communication and hardware failures occur due to the malfunction of hardware components of WSN. These are mostly permanent faults that require the replacement of faulty hardware. As WSNs are deployed in harsh conditions hardware errors are quite frequent. One such instance of hardware fault is when the hardware short circuits due to the presence of water content (Szewczyk et al., 2004). Other techniques that discuss detection of hardware faults can be found in Warriach and Tei (2013), Lau et al. (2014), Mo et al. (2015), Jin et al. (2015), Banerjee et al. (2011), Panda and Khilar (2014), Yuvaraja and Sabrigiriraj (2015), Kaur and Sharma (2010), Kamal et al. (2014), Yang et al. (2014), and Nguyen et al. (2013).

3. Fault detection taxonomy

In recent couple of years, the popularity and use of wireless sensor networks have grown in leaps and bounds which ensued in the advancement of fault detection techniques. This provides us a motivation to develop a new taxonomy for wireless sensor network based on detection techniques. In this section, we discuss a new classification of fault detection techniques.

Fault detection techniques can be broadly classified into *centralized*, *distributed* and *hybrid* as shown in Fig. 5. The centralized approach consists of single central node or base station which monitors and analyzes the condition of remaining nodes. Centralized approach increases the traffic in the network and ensues in network congestion, which resulted in Distributed approaches. In distributed approach, the task of monitoring and analysis is distributed among each node in WSN. A fault detection algorithm is run at each node to produce a local

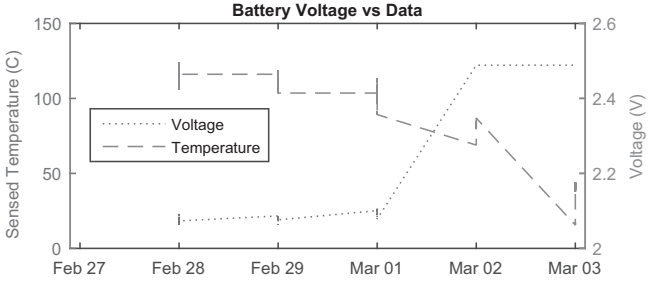


Fig. 3. Data variation with battery voltage at node 15 from Intel-Berkeley data set.

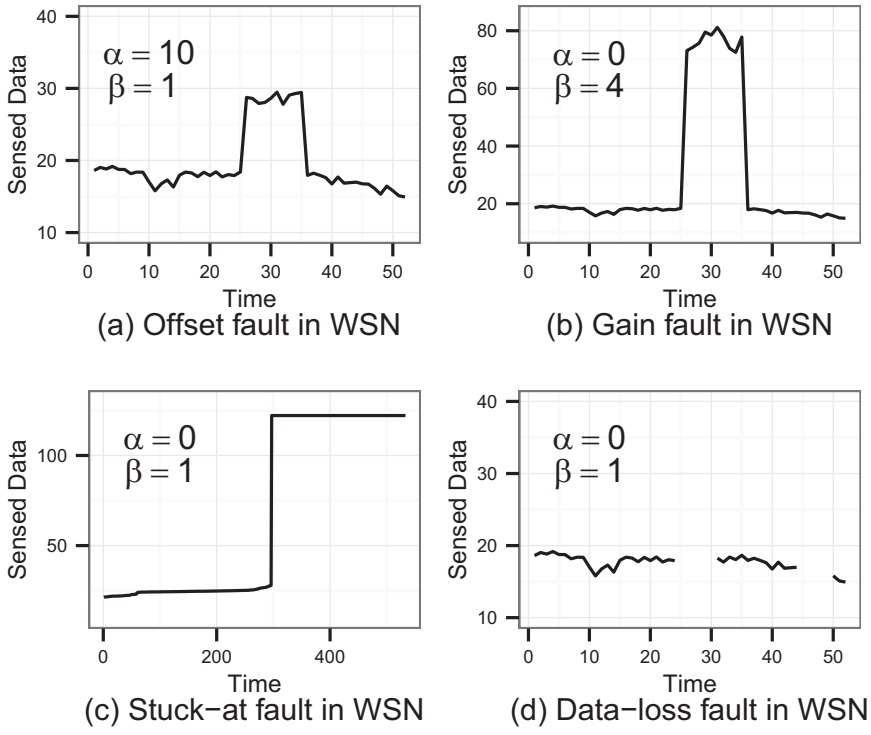


Fig. 4. Faults in WSN data.

status of the node which is propagated throughout the network. Hybrid techniques combine both distributed and centralized approaches. These approaches create a multi-tiered WSN architecture which combines both centralized and distributed aspects.

Centralized approach can be classified into three categories: (1) statistics based and (2) soft computing based. Statistic based techniques use statistic techniques such as mean and sigma test. The soft computing based algorithms (Sanchez et al., 1997; Michie et al., 1994) are algorithms that are mainly based on machine learning techniques.

Distributed approach can be further classified into six categories: (1) neighborhood-based, (2) statistics-based, (3) probability-based, (4) soft computing-based, (5) self-detection and (6) cloud-based. Wireless sensor data are spatiotemporally correlated. The data obtained at an instance is related with previous and succeeding data. Similarly, there is a relation between sensor data from sensors in a geographically close proximity. Neighborhood techniques use the spatial–temporal correlation between the nodes to detect faults in sensor reading. Neighborhood techniques (Chen et al., 2006) can be classified into Majority voting and Weighted Majority voting. Majority voting considers the majority of fault status of neighboring nodes to determine the fault status of nodes. Majority weighted algorithm uses a weight for each node in WSN and collects weighted votes from all the neighboring nodes, and gives the prediction that has a higher vote.

Statistical algorithms are the algorithms that use statistical techniques to detect the outliers in the data. They can be categorized further into three categories namely Time-series analysis, Descriptive statistics, and Bayesian statistics. Time-series technique analyzes time series data to detect similarities in the data and measure the amount of deviation. They use tests such as Kolmorgov–Smirnov test (Massey, 1951) and Kuiper test (Press et al., 1990; Louter and Koerts, 1970) to detect outliers in wireless sensor data. Time series models such as Auto-regressive model are used to model the WSN which is analyzed by these statistical tests. Descriptive statistics (Peatman, 1947) deal with the statistical methods that use one of the central tendency measures such as mean or median of neighborhood nodes to determine the faults. It may also include other statistical methods that are used to measure and describe data. Bayesian techniques (Lee, 2012) are used to find the likelihood a sensor is faulty based on Bayes theorem (Lindley, 1972).

Probability based fault detection techniques utilize the probability of node failure to determine the fault status of nodes. The fault probability of a node and its neighbors are used to calculate posterior fault probability to determine the faulty nodes.

Soft computing based algorithms (Sanchez et al., 1997; Michie et al., 1994) are algorithms that are mainly based on machine learning techniques. This can be further classified into supervised detection techniques and non-supervised detection techniques. Supervised detection techniques use training data sets to train the difference between the actual data and faulty data and predicts the faults in sensor data. Neural network based techniques use the degree of nodes and node dynamics to predict errors in the data. Unsupervised learning techniques are like density estimation problem in statistics. They are not trained with any datasets. Clustering techniques and certain neural networks such as self-organizing maps (SOM) (Kohonen, 1998) come under this category. Clustering based techniques cluster the nodes into various clusters and associates it with a cluster head which analyzes the node.

In self-detecting algorithms, the final fault status of a sensor node is determined by the sensor nodes in the network itself. These techniques use data from neighboring nodes. However, the final decision is taken by individual nodes.

Cloud-based techniques use cloud-based infrastructure to reduce the computational complexity. It is a parallel and distributed technique that uses Map-Reduce algorithm and Hadoop (Dean and Ghemawat, 2010; Lam, 2010). The main idea in this scheme to transfer the data values from the sensor nodes to cloud storage and use map reduces to parallelize the process of fault detection that would reduce the fault detection time. In our knowledge, Yang et al. (2015) are the only work that uses cloud for WSN data fault detection. We discuss this technique in detail in Section 4 when we discuss distributed fault detection techniques.

Hybrid algorithms are used in multi-tiered WSN in which nodes are organized into clusters with cluster head. Nodes in each cluster send its data to corresponding cluster heads. After that cluster heads forward it

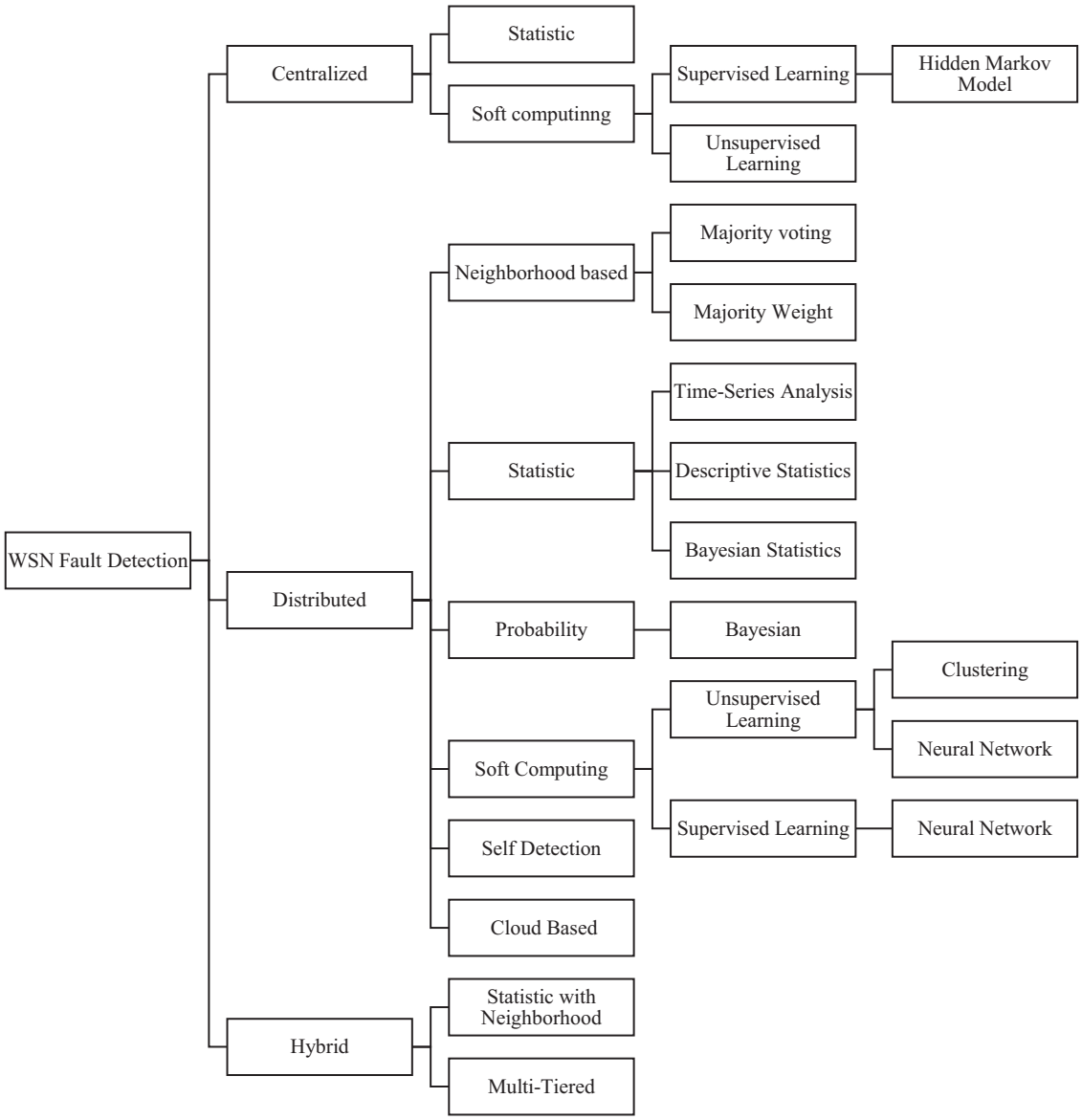


Fig. 5. Taxonomy of fault detection techniques of WSNs.

to a central base station for data analysis. Trust matrix is used to determine the trustworthiness of data in trust matrix technique. Hybrid algorithms also consist of a combination of various detection techniques that are discussed above. The combination of neighborhood algorithm with descriptive statistical techniques such as mean and median belongs to this category.

4. Fault detection techniques

4.1. Centralized approaches

Panda et al. (2014) proposed a centralized algorithm for fault detection in sensor networks based on the statistical method called *z*-value (Maronna et al., 2006). In a customary centralized algorithm, the sensed data is represented as $x(k) = A + r(k)$, where A is the actual data, $x(k)$ is the sensed data and $r(k)$ is the noise. If the data is assumed to be normal, then the absolute difference $d(k) = |x(k) - A|$ will lie in between $A - 3\sigma$ and $A + 3\sigma$, where σ is the standard deviation. The absolute difference $d(k)$ is the absolute value of the noise $r(k)$. If $d(k) < 3\sigma$ the node is considered as fault free else it is considered as faulty. But it is difficult to know the value of A in real environments, hence in this algorithm, the authors first estimate the mean, and then calculate variance using this mean. Hence, the recalculated absolute difference is $d(k) = |x(k) - mean|$. The *z*-value (Maronna et al., 2006) is defined as $z = (x - \mu)/(\sigma/N^{1/2})$ where x is the sensed data, μ is the estimated mean, σ is the standard deviation and N is the number of nodes. If $d(x) < 3z$, the node is not faulty else the node is faulty. The authors only consider data faults, mainly offset and gain faults in this technique.

Warriach and Tei (2013) introduced a supervised machine learning approach based on Hidden Markov Model (Rabiner and Juang, 1986) to detect and identify faults in WSN. They compare and analyze Hidden Markov Models (HMM) of fault free environments and faulty environment to detect the errors, and classify system and data faults. The HMM is characterized by five parameters namely the number of states, the set of possible data values, a probability distribution, the state transition probability and initial state distribution. In order to estimate these parameters, the authors have used supervised learning. In supervised learning, the data set is partitioned into two, training set and test set. Faults are injected into the fault free training set and each data is labeled as faulty or fault-free along with the fault type. This is the data that is used to assign the parameters of Hidden Markov Model.

In this technique, the sensor data from various nodes are sent to the base station. At the base station, initially all the data from sensors are combined based on a time window t_w. The authors first define a set of visible states v_g, a set of hidden states a_g, and set of faulty states f_g for the sensed phenomenon. Next they construct two Hidden Markov models, one with hidden states and visible states of the environment (HMM_{AV}) and the other with hidden states and faulty states of the environment (HMM_{AF}). Later, the authors examine the two HMMs based on known fault model and identify the error type. Based on this a Markov Model M_A that shows fault free states is created. The authors consider data faults especially offset faults, gain faults, and stuck-at faults.

Jin et al. (2015) propose a passive diagnosis approach based on Auto-regressive model (Andel, 1976), Kuiper test (Press et al., 1990) and Kolmogorov–Smirnov test (Kar and Mohanty, 2004; Massey, 1951). In this technique, the normal signals are filtered out to highlight the faulty signals. The Auto-regressive model is a time series model that gives us a linear relationship between the input and output variables (Cong et al., 2016). It is defined as $x[n] = \sum_{i=1}^{p} a[i]x[n-i] + e[n]$, where $a[i]$ is the coefficient of the model, p is the order of the model and $e[n]$ is the white noise with mean zero and variance sigma squared. The $a[i]$ is determined with the help of Levinson–Durbin recursion (Franke, 1985) method and the order is determined by Akaike Information Criterion (Book reviews, 1988). The Auto-regressive series is used to pre-whiten the data before testing. Kolmogorov–Smirnov test and Kuiper tests are tests in statistics to check the similarities between two given distributions. Faults in nodes cause variations in cumulative distributive function (CDF) of data time series. We can compare the two CDFs, $F_{N1}(x)$ and $F_{N2}(x)$, using Kuiper test as $V = \max[F_{N1}(x) - F_{N2}(x)] + \max[F_{N2}(x) - F_{N1}(x)]$, where V represents Kuiper statistics. The mathematical function for V for computing probability is defined as $Q_{KP}(\lambda) = 2\sum_{i=1}^{\infty}(4i^2\lambda^2 - 1)e^{-2i^2\lambda^2}$, where $\lambda = (N_e^{1/2} + 0.155 + (0.24/N_e^{1/2}))V$ and $N_e = (N_1N_2)/(N_1 + N_2)$, where N_1 and N_2 are the number of sensor nodes in first and second distributions respectively. $Q_{KP}(\lambda)$ is a monotonic function with limits $Q_{KP}(0) = 1$ and $Q_{KP}(\infty) = 0$. The Kolmogorov–Smirnov test is defined as $D = \max|F_{N1}(x) - F_{N2}(x)|$. But in the K–S test, differentiating small change is a big challenge and hence the authors replace D by the *Kuiper statistics*. The D value and the V values are used to determine the health of a given node in wireless sensor network. The authors target system faults, mainly the hardware failure of the nodes.

Lau et al. (2014) presented a new centralized algorithm based on Naïve Bayes framework named CNBD, in which the end to end transmission time is collected at the sink and is analyzed using the Naïve Bayes framework. Only hardware faults at nodes are considered in this algorithm. The authors have divided CNBD into two phases: the training phase and the testing phase. In the training phase, we use the data (time taken for the packet to reach sink) from the normal scenario, and the minimum time value for each node is selected as the threshold for anomaly detection. Using this data both normal and conditional probability density functions are estimated by Maximum likelihood attribute (MLE), defined mathematically as $l(\theta) = \sum_{-\infty}^{\infty} \ln(f(s_i|\theta))$ where $f(s_i|\theta)$ represents the joint density of training attribute values $S = \{s_1, s_2, \ldots, s_n\}$. Next the marginal probability is estimated both for the faulty and normal nodes. In the next phase, the received packet at the sink is analyzed. The packets are grouped at the sink depending upon the source it originated. To differentiate between heavy traffic network and a network with a faulty node, we compare all the packets from a particular source with its threshold, if all of them produces false condition, we assume that the node is faulty else we assume that there is traffic in the network. Further, the authors have calculated the statistical mode value, and for each source the mode value is compared with the normal conditional probability to detect congestion in the network. Last the authors compare the last five transmission packets for a given node using the Naïve Bayes classification. The result from the classifier overrides all other previous assumptions and the nodes are classified into faulty or not based on the results of this phase. The authors target system nodes, mainly hardware or node failures.

Kamal et al. (2014) propose a fault detection scheme called Sequence Based Fault Detection (SBFD). Unlike previously proposed techniques, this technique does not depend upon periodic transmission of observed data nor does it depend upon the passive information collection. Rather the proposed technique utilizes tagging of normal data packets that goes to the sink using Fletcher checksum and a path analysis at the server side to sense the route of the packets from the nodes to the sink. The sink analyzes the path to detect any changes in the path which indicates a network failure. The proposed fault detection scheme consists of 4 components: (1) In-network packet tagging (IPT), (2) Network Database (NDB), (3) Network path analysis, and (4) Fault detection and identification. In the IPT phase each node tags the data packets that traverse to the sink with the path checksum. All the nodes in the routing path update the path checksum information using their ID and the path checksum. Fletcher checksum algorithm is used to calculate the checksums. The Network database stores the original routing paths and associated checksums of the route that packets from each node is supposed to traverse to reach the sink. The normal tagged data packets on their arrival at the sink is stripped down of the extra information, and the path information is obtained using the checksums from the NDB. The route is noted, and path statistics are updated. The FDI analyzes the statistics to determine whether the network is faulty. If the FDI doubts that a node is faulty, it will send a control message to the aforementioned node, and the fault status decision depends upon the node's reply. They target system faults, mainly hardware failure.

Yang et al. (2014) develop a tool to detect the node failure/silence in remote WSNs based on power diagnosis of the sensor nodes called telediagnostic powertracer. The proposed tool uses external power measurements of the node to determine the welfare of the nodes. For external power measurements, the authors develop their power meter with a low bandwidth radio. The power measurements are then transmitted to the sink which further diagnose and decide the fault status of the node. A power meter is attached to each node with its radio to sample its energy consumption. Bits of the sampled power consumption is transmitted to the sink wirelessly. The sink then starts up a diagnosis scheme that analyzes the sample power traces and decides the fault status. The authors propose two diagnostic schemes, a passive scheme, and an active sampling scheme. In the passive scheme, before the diagnostic module kicks in, the power traces from various nodes in different failure scenarios are used to train a classifier. During the run time passively collected power samples are classified using the trained classifier. The major disadvantage of this diagnostic scheme is that the possible number of fault scenarios increases exponentially with the number of applications. In the second scheme instead of using a passive technique, a module is inserted into all the sensor nodes that inject distinct power patterns to the power traces that represents the status of the nodes. This scheme eliminates the training process of classifiers. The authors target system faults such as battery failure, hardware issues such as faulty antennas, radio damage, network disconnection, system crashes, short circuit due to water damage, and the node router failure.

Guo et al. (2014) propose a fault detection algorithm to detect data faults in WSNs. The proposed technique is called FIND and does not assume any particular sensing model nor does it require event injections. After an event is detected the technique ranks the nodes based on the sensed data as well based on the proximity to the event. If the difference between these is significant, then the technique assumes that the node is faulty. Initially, the location is partitioned, and each of the partitions is called a face. Each face has a unique sequence called the distance sequence which is the sorted sequence containing the IDs of the sensor nodes from an arbitrary point. When an event occurs, different nodes sensed data differs depending on the proximity to the event. The sensing results of the nodes are ordered to obtain a detected sequence. The detected sequence is mapped with the distance sequence

of the corresponding face to get an estimated sequence. The estimated sequence is compared with the detected sequence, to produce a list of faulty nodes. They target sensor applications in which the sensor readings vary with respect to distance such as thermal radiation and acoustic volume. The authors consider data outliers occurring in WSNs in which the data attenuates with distance. They also target Byzantine faults. Byzantine faults are those faults that seems different to different observers. They are also known as random faults.

Abid et al. (2015) propose a centralized data outlier detection technique for WSN using KNN (K-Nearest Neighbor) and Euclidean Distance algorithm. In this technique, all the sensor nodes send their data to the sink node. For each node, the time series of data values are sorted in descending order, and the values are aggregated into clusters of K-Neighbors. For each value, the Euclidean distances between its successor and predecessor are calculated, and a link is added to the smallest distance. The comparison is made between the different distance values and the values that produce bigger distance is eliminated as it is detected as faulty. The authors target data outliers faults, mainly gain faults and offset faults.

4.2. Distributed approaches

Feng et al. (2014) proposed a distributed fault detection algorithm based on weighted distance. In this algorithm, the weighted sensed value of a node is compared to original sensed value to judge faulty nodes. For a given node s_i, having neighbors s_j, they compute the weighted distance from node s_i to node s_j using the formula, $\rho_i = \sum_{j=1}^{M}(1/M-1)(1-(E(s_i)_j/\sum_{j=1}^{M}E(s_i)_j))$ where M is the number of neighboring nodes of node s_i, $E(s_i)_j$ is the distance between the nodes s_i and s_j and ρ_i is the weighted distance from node s_i and all of its neighbors s_j. Then the weighted values for the nodes s_i is defined as $x'_i = \rho_i x_j$ where x_j is the sensed value of node s_j and x'_i is the weighted value. After that, they compare the weighted value of the node s_i and the original data at s_i with a fixed parameter. If the absolute difference between x_i and x'_i is less than the parameter, then the node s_i is assumed to be correct else it is declared faulty. The authors discuss the detection of data faults such as gain faults, stuck-at faults, and offset faults.

Panda and Khilar (2015) propose a distributed, self-diagnosing algorithm based on modified three sigma edit test (Maronna et al., 2006) for detecting soft and hard faults (DSFD). It consists of two phases: initialization phase and self-diagnosis phase. In the initialization phase, the sensor s_i transmits a message containing the sensed data x_i to all its neighbors s_j. And waits for an estimation time Ett_i, during which it accepts messages containing the sensed data, x_j from its neighbors, s_j. Once the Ett_i expires, the node s_i creates a neighboring table from the sensed data received from the neighbors and create a neighboring table Nt_i. The neighboring table consists of neighboring node ID and corresponding sensed data. In this phase, all the nodes are considered as non-faulty. In self-diagnosing phase, the sensed data of each neighboring node in the neighboring table is analyzed. If no sensed data is present, it indicates a hard fault at the corresponding node. If sensed data is present then we measure the outlyingness tr^i. The tr^i is calculated as $(x_i - median_i)/MADn(x_i)$, where $median_i$ is the median of neighboring nodes sensed data x_j and x_i is the sensed data at node x_i. MADN is the normalized median absolute deviation (Leys et al., 2013) similar to standard deviation and is given by $MADn(x_i) = Med\{|x_i - median_i|\}/0.675$. The sensed node is said to be a faulty node if $tr^i > 3$. The authors discuss the detection of system faults such as hardware failure and data faults such as out-of bounds, offset faults, gain faults, and stuck-at faults.

Saihi et al. (2013) proposed a distributed algorithm that is based on error functions which are proportional to the deviation between the measurements. This algorithm depends upon majority voting. They define the two error functions as $erf(d_{ij}(t)) = erf^1 = (2/\pi^{1/2})\int_0^{d_{ij}(t)} e^{u^2}du$ and $erf(\Delta d_{ij}(t)) = erf^2 = (2/\pi^{1/2})\int_0^{\Delta d_{ij}(t)} e^{u^2}du$ using Gaussian Distribution and reduced centered normal distribution, where $d_{ij}(t)$ is the measurement difference between the nodes s_i and its neighboring node s_j at a given time t. Moreover, $\Delta d_{ij}(t)$ is the measurement difference between the nodes s_i and its neighboring node s_j between a given time t and $t+1$. This can be mathematically expressed as $\Delta d_{ij}(t) = d_{ij}(t+1) - d_{ij}(t) = (x_i(t+1) - x_j(t+1)) - (x_i(t) - x_j(t))$. In this method, the authors calculate error function 1 ($erf1$) and error function 2 ($erf2$), and find the product C_{ij} of both error functions. The nodes are first initialized as possibly faulty (LT) or possibly normal (LG). LT, LG, FT, and GD are called the tendency values (T_i) of node s_i and they respectively indicate whether the node is probably faulty, probably good, a sure fault in the node, and that it is surely working. If $\sum_{s_j \in N(s_i)} C_{ij} < N(s_i) - 1$ where s_j is the neighboring node and s_i is the source node, then it sets $T_i = LG$ (possibly correct) else it is set it as LT (possibly faulty). After setting all nodes to LG or LT we check if $\sum_{s_i \in N(s_i), T_i = LG} C_{ij} < Num(N(s_i)_{(T_j = LG)}) - 1$, where $Num(N(s_i)_{(T_j=LG)})$ is the degree of s_i having LG as node fault tendency. If the condition is satisfied then the nodes are set as normal else it is set as faulty. The fault detection technique targets data faults such as gain faults, offset faults, and out-of bound faults.

Xu et al. (2014) proposed an energy efficient distributed algorithm that takes into consideration both the temporal and spatial characteristics of the sensor node. The data at consecutive time instance is compared and a conclusion is reached whether it has transient faults or not. If transient faults are present then the data is corrected using the data from the instance in which it was marked normal. In the second phase, they compare the data with the neighbors to reach a conclusion whether it is faulty or not. They consider the data at node s_i at time t, along with data from q previous time instances. Using this q number of data for the particular node, a matrix is created as follows:

$$M_{m\times n} = \begin{cases} 0, \text{ if } |x_i^m - x_i^n| \le \xi_1 \\ 1, \text{ otherwise} \end{cases} \quad m, n = \{0, 1, \ldots, q-1\} \tag{2}$$

For each row in matrix M, c_i^r is calculated with

$$c_i^r = \begin{cases} 0, & \text{if } \sum_{j=t-q+1}^{t} M_{ij} < \frac{q}{2} \\ 1, & \text{otherwise} \end{cases} \quad t-q+1 \le r \le t \tag{3}$$

The value of c_i^r at time t is corrected as follows:

$$c_i^t = \begin{cases} 0, & \text{if } \sum_{j=t-q+1}^{t} c_i^r < \frac{q}{2} \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

The value of time t is any measure when c_i^r is zero. Initial states of sensor nodes can be determined as

$$T_i = \begin{cases} 0, & \text{if } \sum_{j=t-q+1}^{t} c_i^r < \frac{q}{2} \\ 1, & \text{otherwise} \end{cases} \tag{5}$$

The value of T_i indicates the probable fault status. If $T_i = 0$, it is fault-free else it is probably faulty. For a given node s_i which is probably fault free, its neighbors reading is obtained, whose initial fault state is zero. Then c_{ij}^t is determined, which is used for final determination of faulty node s_i:

$$c_{ij}^t = \begin{cases} 0, & \text{if } |x_i^t - x_j^t| \le \xi_2 \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

Using c_{ij}^t, the final fault status of the node s_i is calculated, which is denoted by GT_i:

$$GT_i = \begin{cases} 0 & \text{if } \sum_{s_j \in Neig(s_i) \text{ and } T_j = 0} C_{ij}^t < \frac{Num(N(s_i) \text{ and } T=0)}{2} \\ 1, & \text{otherwise} \end{cases} \tag{7}$$

$Num(N(s_i) \text{ and } T=0)$ indicates the number of neighbor nodes that

have a fault status, probably fault free. If the value of GT_i is zero, then it is fault free else the node is faulty. The authors target data faults like random faults, gain faults, offset faults, stuck-at faults, and out of bounds fault.

Panda and Khilar (2014) proposed a distributed fault detection algorithm. In this algorithm, the mean of neighboring nodes is computed to check whether faulty sensor node is present or not. If a faulty node is found then observed data is compared with neighbor's data to predict probable fault status. The final fault status is determined by combining the fault information from the neighbors. Faults are detected in two phases, partial self-identification phase and self-detection phase. In partial self-identification phase, a node s_i receives data from its neighbors and creates a neighboring table. Initially remaining battery power for each node is calculated for detecting the hard faults, then data is compared with minimum value and maximum value to detect stuck at zero and stuck at one fault. If the detected value is between the minimum value and maximum value, then the mean of neighboring data is computed and the difference between the data and mean is compared to a threshold as, $\mu = x_i - mean_j \leq \lambda_1$. If the condition is satisfied then the nodes s_i and s_j are said to be probable fault-free nodes. Otherwise, there are four cases:

Case 1: $|x_i - x_j| > \lambda_1$ and $x_i \leq \lambda_2$. In this case, x_i is added to PFF (Probably faulty free) but x_j is added to PF (Probably faulty).

Case 2: $|x_i - x_j| > \lambda_1$ and $x_j > \lambda_2$. In this case, both the nodes are added to PF list.

Case 3: $|x_i - x_j| \leq \lambda_1$ and $x_j \leq \lambda_2$. In this case, both the nodes are added to PFF list.

Case 4: $|x_i - x_j| \leq \lambda_1$ and $x_i > \lambda_2$. In this case, x_j is added to PFF list but x_i is added to PF list.

The authors mainly target data fault called random faults, which gives random readings.

Yuan et al. (2015) proposed a new distributed Bayesian algorithm to detect node errors in wireless sensor networks. The fault probability of the sensor node is calculated using Bayesian networks and is improved by exploiting the border nodes to increase the fault detection accuracy. Initially, the authors examine whether a node and its neighbor have the same fault status. Each sensor node s_i transmits its sensor reading and fault probability to its neighboring nodes periodically. If $x_i - x_j$ less than a threshold r^t then both the nodes have the same status else they differ in status. If the variation is less than parameter r^t we set $f_{ij} = 1$ else $f_{ij} = 0$. Posterior fault probability is calculated as

$$P(s_i/N(s_i)) = \frac{P_i \prod_{j=1}^{n} |1 - f_{ij} - P_i|}{(1 - P_i) \prod_{j=1}^{n} |f_{ij} - p_j| + p_j \prod_{j=1}^{n} |1 - f_{ij} - P_i|} \quad (8)$$

where p_i and p_j are the fault probabilities of node s_i and its neighbor s_j. P_i and P_j are the prior fault probabilities of the node s_i and its neighbor s_j. If the number of faulty nodes among the neighboring nodes are higher, this results in falsely identifying the node as faulty and vice versa. Hence, a common border node is found to which the neighboring nodes calculate and sends the confidence. The confidence, c_{n1}, is calculated as $c_{n1} = n_{eq} - n_{neq} - (n(s_{n1})/2)$. c_{n1} is the confidence of sensor node s_{n1}, n_{eq} is the number of nodes supporting s_{n1} and n_{neq} is the number of nodes that does not support s_{n1}. If the confidence of all neighboring nodes is less than zero, then their probabilities will be set to 0.5. If the confidence is greater than zero, we select the neighbor, s_k, with the maximum confidence. If f_{ij} is one, then the probability of fault is the same for both, else s_i will have fault probability of $1 - p_{sk}$. The authors discuss detection of gain faults, stuck-at faults and out-of bounds fault.

Obst (2014) proposes an anomaly detection using spatially organized distributed echo state networks (SODESN) (Obst, 2009). The computation is distributed throughout the entire WSN. The SODESN distributes a recurrent neural network over a WSN and imposes a topology on its connectivity matrix. Each sensor node estimates its own true values based on information from its neighbors in the training period. Using the estimated true values and a threshold, each sensor node can decide if it can be assumed to work correctly. A recurrent neural network has an input layer, an output layer, and a hidden reservoir. Initially, the Echo State Network (ESN) is trained. The connections between the hidden layer and the output layer are the only connections that are trained. At each time step t, training data is used to drive the network and activations of all input and hidden units are saved as a new column to a state matrix S. The output synaptic connection weight is determined and squared. The SODESN is trained and is used to predict the next sensor value. A sensor node is considered to be abnormal if the difference between the predicted value and the actual sensor value exceeds a threshold θ. The authors discuss stuck-at faults and random faults.

Yang et al. (2015) proposed a time efficient approach to detect the faults in wireless sensor data on the cloud. This method exploits the scale-free topology of the wireless sensor network and clustering technique to boost the effective time to detect errors in data. This algorithm effectively detects data errors and the source of errors rapidly. The wireless sensor networks are clustered and the data is collected from each cluster and sent to the base station. At the base station, the data is send along with the cluster information to the cloud. It is assumed that the data in the same cluster lie at close quarters. A Hadoop-based MapReduce algorithm (Dean and Ghemawat, 2008) is used to process the clusters in parallel. The proposed fault detection is a two-phase detection technique. The first phase is the error detection phase. Three inputs, the graph of the WSN network, the fault patterns, and the collected data, are passed to the error detection algorithm. The error detection algorithm runs on the cloud. The fault detection algorithm initially verifies whether the given graph is a scale-free graph. This technique works efficiently on scale-free graphs. The WSNs that are modelled as a clustered WSN are similar to scale-free graphs. Next based on the time stamp of the data the algorithm determines stuck at fault or flat line fault and it checks if any data is missing (Data loss error). Based on the error patterns the algorithm checks for boundary errors. The spike errors are detected next using the time series time stamp that was passed to the algorithm. This fault detection algorithm produces a fault set D', that consists of data that has an error. In the second phase, an error localization algorithm is used to find the location of the error, and it produces a graph G' that consists of the nodes that produced the error. The advantage of this technique is that it is fast, as the graph is partitioned and the detection algorithm is run in parallel on all the sub-graphs. The authors target data faults such as fusion faults, spike faults, stuck-at fault, and data loss error.

Banerjee et al. (2014) propose a strategy that does not exclusively detect fault but replaces the detected faulty node with a healthy node. This strategy exploits the sensor nodes to their fullest before declaring it as faulty. In this strategy, they monitor three components of the sensor network, namely the peripheral unit, the sensor circuit and the receiver circuit. The peripheral unit comprises processor, memory unit, battery, and transmitter circuit. For each of the three components, we store faulty status and based on these statuses we make final node fault status. If the peripheral unit bit is zero, the node is detected as faulty as the sensor cannot function without power. In addition to the peripheral unit bit, either of the sensor or receiver bits should be set to one for the sensor to detect fault free. This is due to the fact that the sensors with only sensing capability can still keep sensing and the node with only transceiver can still serve as an intermediate node for message transmission. The status of the peripheral unit is changed when one of the components stops functioning or when the energy is depleted. Moreover, the difference between the sensed data and the mean of neighboring nodes is compared with a threshold to determine the fault status of transceiver bit. In addition, the fault status of receiver bit is determined by analyzing the acknowledgment signal time. Thereafter,

the detected nodes are replaced by fault-tolerance strategy that makes use of regression plan. The authors detect both hardware and data faults. Hardware faults such as hardware failure including transmitter circuit fault, microcontroller fault, sensor circuit fault, battery fault and receiver failure are detected by the proposed detection technique.

Mo et al. (2015) proposed a fault detection strategy based on Time domain features of sensed data (TDSD). It is a passive technique wherein, initially one-dimensional Gabor transform is used to extract and analyze features of the sensed data and after that Self-organized maps are used to diagnose and classify the data from preceding phase. Finally, the fault status of the data is determined. The Gabor transform is normally used for local feature extraction for signals and is given by $G_f = \int_{-\infty}^{+\infty} f(t)g(t-\tau)e^{-iwt}dt$, where $g(t)$ is the time window and $f(t)$ is the signal. The Self-organizing maps (SOM) are unsupervised neural networks that are used to train and classify values. The data from the Gabor transform with known faults are used to train the SOM, which creates a library of known fault values. The test data is analyzed and compared with the library to finally classify the nodes as faulty. The authors target network failures such as network failures (link failures), node failures, and data failures such as gain faults, off-set faults, and bound faults.

Yuvaraja and Sabrigiriraj (2015) propose a fault detection scheme that would improve the lifetime of the WSN. In the proposed scheme, the base station (sink) generates an Agent packet and the agent develops a query path towards the faulty node. The sink periodically generates an agent packet which is sent to all of its neighbors. An agent packet consists of information such as message Id, sender Id, receiver Id, the number of active nodes in the current path, the time at which the packet was sent and the maximum time the packet will live (TTL). When the neighboring nodes receive the agent packets, they store the sender Id, Message Id, and the number of active nodes to their query list and they send an Ack packet back to the sender. Now the receiving node makes a random decision whether to transmit the agent packet further to other neighboring nodes. This to avoid congestion in the network and reduce the number of queries in the network. The TTL of the package is decreased by one as it traverses from node to node. If the sender did not receive an acknowledgment from its neighbor within a time t, the node is assumed to be dead and the sender broadcasts a Neighbor_dead packet which consists of the Id of the dead node and TTL=x. The higher the value of x, the higher the broadcast of the information about the dead node. The value of x has to be set in such a way that it will not increase the congestion in the network. The authors target network faults caused by hardware failure especially node failures.

Fang and Dobson (2013) propose a fault detection scheme that detects data faults and reduces the energy required for fault detection. In this technique, a model that represents the behavior of the sensed data is maintained by the sink and sensor nodes. This model is used to verify the sensed data. If the sensed data is within certain error band, then it is accepted as non-faulty data else it is considered as faulty data. But this could also mean that the model is outdated, hence, a fault detection algorithm is used. Each node maintains a spatial matrix which stores the difference between the maximum and minimum data sensed by a node and its neighbors. A voting mechanism is used to validate a probably faulty node. When a sensed data does not match the statistical model, the node will send the data to its neighbors. Each of the nodes received the request will consult the spatial matrix and will send a Boolean decision. The final result will be decided by majority voting, based on decisions from the neighbors. The authors target the gain faults, additive faults, stuck-at faults (constant faults), and drift faults.

Sahoo and Khilar (2014b) propose a fault detection strategy based on the comparison of sensed data and residual energies among the neighboring nodes. The proposed fault detection scheme can detect permanent and transient faults at the sensor nodes and permanent faults at the sink. If x_i^t is the sensed measurement at time t by node i, then the spatial correlation which is the difference in the sensed data is assumed to be small else if at least one of them is faulty, it is assumed to be higher. This can be mathematically represented as

$$|x_i^t - x_j^t| = \begin{cases} \leq \delta_1 & j \text{ and } i \text{ are fault} - \text{free} \\ > \delta_1 & \text{atleast } i \text{ or } j \text{ is faulty} \end{cases} \quad (9)$$

Moreover, the residual energy estimations of the neighboring nodes are also compared to get a better fault detection. If $E_{(i,j)}^t$ is the estimate of j about the residual energy of node i at time t, then mathematically we can represent it as

$$|E_i^t - E_{(i,j)}^t| = \begin{cases} \leq \delta_2 & j \text{ and } i \text{ are fault} - \text{free} \\ > \delta_2 & \text{atleast } j \text{ or } i \text{ is faulty} \end{cases} \quad (10)$$

δ_1 and δ_2 are two thresholds that need to be defined depending upon the application. Therefore the fault status of the node i can be given by the following condition:

$$c_{(i,j)}^t = \begin{cases} 0, & |x_i^t - x_j^t| \leq \delta_2 \text{ and } |E_i^t - E_{(i,j)}^t| \leq \delta_2 \\ 1, & \text{Otherwise} \end{cases} \quad (11)$$

If $c_{(i,j)}^t$ is 0 then the nodes i and j are fault free else, at least, one of them is faulty. Each sensor node has a status vector that stores faults status of all nodes in the network. Initially, all the values in this vector are set to zero (all nodes are not faulty). In each round, until a certain predefined number of rounds, the sensor nodes send its sensed data and the residual energy to all its neighboring sensor nodes. Once the neighbors receive these they perform the conditional test provided in Eq. (11). If the condition fails then it sets the corresponding bit in the status vector as 1. After r rounds, the status vector is shared and updated between the neighbors. A majority voting is performed on the status register to find out the faults including transient faults. The authors target intermittent faults. Two cases of intermittent faults are handled by the authors. One when the faulty node sends sensed data similar to its neighbors in all rounds and the second case when the send sensed data deviates in all rounds from its neighbors.

Sahoo and Khilar (2014a) propose an extension to their already existing work (Sahoo and Khilar, 2014b) that we discussed before. The proposed scheme consists of three phases. The initial phase is the comparison phase in which each fault-free node compares its sensed data and residual energy with its neighbors to classify them as permanently faulty, transiently faulty, or non-faulty. This phase is the same phase as in Sahoo and Khilar (2014b) as discussed before. At the end of the first phase, the local decision needs to be broadcasted. The authors propose a new technique for flooding using spanning trees that avoid the message implosion problem (Chessa and Santi, 2002). This reduces the communication complexity and hence saves the power consumption by the nodes. The second phase is the creation of spanning tree that covers all the nodes. The last node is the dissemination phase in which the spanning tree is used to transmit the local decision information between the nodes.

A distributed fault detection algorithm to detect faults is proposed by Huang (2015). The technique uses the comparison between neighborhood nodes and the testing node self-decides the fault status of the nodes. The diagnosis takes place periodically to save energy and reduce congestion. The scheme consists of two phases. In the initial phase, each node performs a comparison with its neighbors. The comparison function C_{ij} which is the comparison between the node s_i and all its neighbor s_j is given by

$$C_{ij} = C(s_i, s_j) = \begin{cases} 0 & \text{if } |x_i - x_j| \leq \xi \\ 1 & \text{otherwise} \end{cases} \quad (12)$$

x_i and x_j indicates the sensed data by the node s_i and its neighbors s_j and ξ is an application dependent threshold. A threshold θ is defined and for each node, the comparison function is compared against this threshold. When $C_{ij} > \theta$ we set it as fault free else it is set as faulty. In

the second phase, the fault-free node that had passed the threshold test in the first phase is compared with its neighbor's comparison function (i.e. $C_{ij} = C_{ji}$). If they are equal then the node is finally set as fault free else it is set as faulty. The authors target data faults like gain fault, offset faults, and out-of bounds faults.

Mahapatro and Panda (2014) propose a fault detection algorithm for the permanent faults and transient faults. The fault detection for transient faults has been modeled as an optimization problem based on multi-objective swarm optimization (2LB-MOPSO) to find an optimal trade off between the detection accuracy and the period of fault checking. The permanent fault detection is based on timeout mechanism. Each sensor node has a neighbor table that stores the values from its neighbor. In each round, the sensor nodes send its data to the neighbors. If any node does not receive data from its neighbor within the timeout time T, it assumes that it has a hard fault. Moreover, the sensor nodes compare their sensed data with the neighbor's data and if the difference between the neighbors is more than certain threshold theta, it is considered to have soft faults. The above technique is used at certain distinct times to detect the intermittent faults. The authors detect both hardware and data faults. They also detect intermittent faults. Hardware faults such as hardware failure and battery depletion is detected by this technique.

Ghorbel et al. (2015) propose a fault detection technique called Distributed and Efficient One-class Outlier Detection Classifier (DEOODC) which is based on the Mahalanobis kernel. The authors propose the use of Kernel principal component analysis based Mahalanobis kernel for the detection of faults in WSN. This technique uses the Mahalanobis distance to calculate the mapping of data points in the feature space to separate the outliers from normal data distribution. In the proposed technique, only one class is used to train the model. There are mainly two phases, the training phase of the model and the testing phase of the model. During the training phase, the data is collected and is centered and normalized using mean and standard deviation. Principal Component Analysis is applied to this normalized data to get the eigenvector and the eigenvalues. Thereafter, the suitable Principal Components are selected. Then the dissimilarity score, *Dis*, is measured using the training data measurement score and the corresponding eigenvalues. The training data measurement Px_i is given by

$$Px_i = x_i * EV(i) \tag{13}$$

where x_i is the sensed data by the node s_i used for training and EV is the eigenvector of ith node. And the dissimilarity score, *Dis*, is given by

$$Dis = \sum \frac{Px_i^2}{\lambda^2} \tag{14}$$

where λ denotes the eigenvalue. Thereafter, the threshold vector is found using Automated Cluster Discovery Procedure (ACDT) (Xie et al., 2006) which will contain the thresholds that will be used to differentiate between various classes of data during the training phase. During the training phase, the real-time data is normalized and Dissimilarity score, Dis_{test}, is computed according to Eq. (14). The Dis_{test} value is compared against the threshold value in the threshold vector. If it is greater than the threshold, it is classified as an outlier else it is classified as normal. The authors target data faults such as out-of bounds faults, stuck-at faults, and gain faults.

Sharma and Sharma (2016) propose a reactive distributed fault detection scheme called rDFD, which considers the excess energy consumption due to the transfer of control messages. In the proposed scheme each node initially utilizes temporal correlation and if necessary later spatial correlation to identify faults in the network. A confidence level is assigned to each sensor node, to enhance the detection accuracy. The proposed scheme consists of four phases: (1) self-detection phase, (2) communication phase, (3) decision phase, and (4) reconfirmation phase. Initially, all the sensors are set to working Good (GD) status and the sensors save the past T sensor observations. The past T readings of the i-th node can be denoted by $History_i(T) = X_i^t, X_i^{t-1}, \ldots, X_i^{t-T-1}$. *PF*, *LG*, *FT*, and *GD* are the Fault Status ($Status_i$) of node s_i and they respectively indicate whether the node is probably faulty, probably good, a sure fault in the node, and that it is surely working. In the self-detection phase, each sensor node performs self-analysis to detect faults. Upon satisfying any one of the following conditions the nodes are assigned a $Status_i = PF$, which indicates that it is probably faulty:

1.

$$(|X_i^t - X_i^{t-1}| \geq (\theta + \Delta X_i)) \tag{15}$$

2.

$$\left| X_i^t - \frac{1}{|Neigh(s_i)|}\left(\sum \delta X_j^t\right) \right| \geq \theta_1 \tag{16}$$

where θ and θ_1 are two thresholds. δX_j^t indicates the negotiated reading of the neighbors due to the attenuation of signal with respect to distance for reading at time t. $|Neigh(s_i)|$ indicates the number of neighbors for node s_i. If the fault status of node s_i, $Status_i$, is equal to PF then the node calculates the variance of its data values as in Eq. (17) and goes to the communication phase:

$$\sigma_{(i,j)}^2 = \frac{\sum_{j=1}^{T} (X_i^{t-j} - \mu_{(i,T)})}{k} \tag{17}$$

In the communication phase, all the nodes that have a PF (Probably faulty status) send a message to the neighboring nodes. The neighboring nodes, s_j, find its neighbors, s_k, and request for the current sensed data from the nodes. When s_j receives the reading $X_k{}^t$ from s_k each s_j calculates δX_k^t and δX_i^t as

$$\delta X_k^t = \begin{cases} X_k^t\left(\dfrac{D \pm d_{jk}}{D}\right) & \text{if } (|X_i^t - X_i^{t-1}| \geq (\theta + \Delta X_j)) \\ X_k^t & \text{if } (|X_i^t - X_i^{t-1}| < (\theta + \Delta X_j)) \end{cases} \tag{18}$$

$$\delta X_i^t = \begin{cases} X_i^t\left(\dfrac{D \pm d_{ji}}{D}\right) & \text{if } (|X_i^t - X_i^{t-1}| \geq (\theta + \Delta X_i)) \\ X_i^t & \text{if } (|X_i^t - X_i^{t-1}| < (\theta + \Delta X_i)) \end{cases} \tag{19}$$

where D is the maximum distance from an event that will affect the signal and d is the distance from the node to the event.

Next the Correlation (C_{ij}) between s_k and its neighbors s_j is calculated as

$$C_{jk} = \begin{cases} 0, & \text{if } |X_j^t - X_k^t| \leq \theta_1 \\ 1, & \text{otherwise} \end{cases} \tag{20}$$

After that the probable fault ($Status_j$) and confidence of each node (C_{jk}) is set as

$$Status_j = \begin{cases} LG, & \text{if } \sum (1 - C_{jk}) \geq \left\lceil \dfrac{|Neigh(s_j)|}{2} \right\rceil \\ PF, & \text{Otherwise} \end{cases} \tag{21}$$

$$CONF_j = \begin{cases} \sum (1 - C_{jk}), & \text{if } \sum (1 - C_{jk}) \geq \left\lceil \dfrac{|Neigh(s_j)|}{2} \right\rceil \\ \sum (C_{jk}), & \text{Otherwise} \end{cases} \tag{22}$$

The C_{ij}, $Status_j$, and $CONF_j$ values are passed backed to the suspicious node s_i and the system moves on to decision phase. In the decision phase, the final fault status decision is made according to

$$Status_i = \begin{cases} GD, & \text{if } \sum (1 - C_{ij}) \geq \left\lceil \frac{|Neigh(s_i)|}{2} \right\rceil \\ FT, & \text{if } \sum (C_{ij}) \geq \left\lceil \frac{|Neigh(s_i)|}{2} \right\rceil \\ PF, & \text{Otherwise} \end{cases} \quad (23)$$

If the sensor node is still not sure after decision phase, it goes to the reconfirmation phase. In the reconfirmation phase, the confidence levels of each node are utilized to decide the final fault status of the node. The authors target detecting intermittent faults. Intermittent faults are the faults that are like transient faults but occur in random manner.

Mahapatro and Khilar (2014) propose an online distributed fault tolerant algorithm to prevent the erroneous data entering the network. In this technique, each node s_i sends out a heartbeat message. All the one-hop neighbors of s_i, s_j sort the node IDs of the nodes from which it received the message based on the time of message arrival. After that, it increments its heartbeat sequence number and the heartbeat of s_i is heart beat number and send a response back to the node s_i. After a certain time all the unreported nodes are marked as having a hardware fault. Once the node s_i receives the response, it compares its value with its neighbors value. If they are similar, then the node s_i is marked fault free else it is characterized as soft fault.

Chanak and Banerjee (2016) propose a fuzzy rule-based fault detection and classification scheme that detects faults in WSN. The proposed fault detection scheme increases the fault detection accuracy by overcoming the uncertainties in the WSN's environment. In the proposed technique, detections are carried out at the node as well as the sink. This technique mainly deals with faults that affect sensor circuits, receiver circuits, and the battery status of the nodes. The technique consists of two stages. The initial stage is the fault detection phase and the second phase is the fault classification phase in which the faults are classified. In the initial phase, faults such as transmitter faults, receiver faults, sensor faults, and battery failures are detected. The transmitter faults are detected by the sink, whereas the rest of the faults are detected by the node itself. In this scheme, the hardware is classified into various states depending on its hardware fault. These states are the normal node, traffic node, end node, and dead node. The fuzzy system classifies the faulty nodes into each of these states. To detect transmitter faults, every sensor node sends a heartbeat message of size 200 bytes to the sink at precise time intervals. The sink replies with a heartbeat OK message of size 200 bytes. The functional efficiency of the transmitter is calculated as a ratio of total heartbeat messages send to the total time spent by the sensor network. If the functional efficiency is low then the sink assumes that the transmitter is faulty else it is in good condition. The sensor circuit fault is detected by the sensor itself. The nodes exchange sensed data and the sensing difference between the neighboring nodes are calculated. If the sensing difference between any two nodes is less than a certain threshold value then we reset the sensing difference between these neighbors as 0, else the value in sensing difference assumes as such. The ratio of the sum of all the sensing difference between the neighbors to a total number of neighbors is called as the average sensing information which is represented using a Fuzzy logic linguistic variable. If the linguistic variable is low then the sensor node is faulty, if it is medium then a fault will occur after some period of time, and if the value is high the sensor circuit is assumed non-faulty. Hence, the sensor node fault is detected by the node itself. Similarly for battery faults, the remaining battery is represented by a linguistic variable. If the linguistic variable is low then battery faults occur, if it is medium then the battery will fail in a specific amount of time and if it is high the battery is in good condition. The receiver fault status is also determined by the node. The node calculates the receiver circuit efficiency which is defined as the ratio of the number of heartbeat OK messages received by the receiver to the total time of WSN. This status is also stored in the linguistic variable. If the linguistic variable is low then the receiver is faulty, if it is medium then it will fail after specific time, and if it is high the receiver circuit is in good condition. In the second phase of the proposed scheme, the hardware conditions of the nodes are measured by fuzzy logic rules. The condition of the receiver, sensor and battery is given as an input to the fuzzy system which is converted into linguistic variables by fuzzifier. The inference engine makes a decision on the fault status of the nodes based on the rules defined in the fuzzy rule base. The defuzzifier then generates non-fuzzy control that classifies the nodes as normal, dead, end or traffic nodes.

4.3. Hybrid approaches

Wang and Chen (2013) proposed a fault and event detection algorithm based on similarity matrix. All nodes in the network are arranged into disjoint clusters. There will be a cluster head for each cluster that will act as a data sink. When an event occurs the nodes are supposed to send its neighbors data to cluster head. After that, these values will be compared to nodes value by the cluster head to create the similarity or trust matrix. Each element in the trust matrix indicates the similarity or trust between any two selected sensor nodes which are neighbors to each other. Each element in the trust matrix for any two neighboring nodes is calculated as $s_{ij} = (x_i x_j)/x_i^2 + x_j^2 - x_i x_j$. Next, the indirect similarity among all nodes in the network is computed to determine whether an error has occurred. Now for each node s, if $trust[s][v] > trust[s][v] \Upsilon w[s][u]$ then $trust[s][v] = trust[s][u] \Upsilon w[u][v]$ where $v \in$ non-adjacent nodes of s, $trust$ is the trust matrix, u denotes an intermediate node between s and v, and $w[u][v]$ denotes the distance between u and v. Moreover, Y is defined as

$$x \Upsilon y = \begin{cases} \min(x, y) & x > t \& y > t \\ x * y & \text{otherwise} \end{cases} \quad (24)$$

where t is the threshold of trust. The trust value of each pair of nodes lies in the range [0, 1]. Higher the similarity between nodes, the closer the trust value is to one. For a given node s_i, we analyze the trust values between s_i and its neighbors s_j and the trust values are divided into two sets containing neighbors with higher and lower similarity values. Each cluster head sends these two sets to the base station. The base station compares the sets received from various cluster heads and takes a final decision on fault status of the nodes. The authors consider data faults such as stuck-at faults, out-of bounds fault, and gain faults.

Nitesh and Jana (2015) propose a distributed algorithm for the fault detection in the upper tier of a two-tier wireless sensor network. In a two-tiered WSN, the sensor nodes are clustered and each cluster has a cluster head, known as relay nodes, that monitors the sensor nodes. This technique uses the neighbors sensor values to create a neighboring table. The relay nodes after entering the details of the node, the fault status of the node is set to *non-faulty*. The sensor nodes send pulse signals periodically to the relay node. After each pulse time, the relay node assumes that all the sensor nodes are faulty and updates the fault status as *faulty*. The sensor nodes send a pulse message periodically and announces its existence and updates its status to *fault free*. To manage lost pulses scenarios, a time window, T_ω, is used where the decision is made based on the previous x times. Relay nodes check the status of the neighboring node before communicating with sensor nodes. Relay nodes only communicate if $\sum_{t=1}^{t} T_w(n) \leq \theta$. The authors consider hardware faults, especially the permanent faults in transceiver units.

Titouna et al. (2015a) address the problem of fault detection in WSN by presenting a Hybrid Hierarchical fusion based outlier detection technique for WSN. The proposed technique consists of two levels of detection. The first level of detection occurs inside the sensor nodes and the second level of detection is performed by the cluster heads. In the first level, the initial decision regarding the fault status is made by the sensor itself. The sensors use Naïve Bayes classifier (Zhang, 2004) to detect the outlier in the first step. Non-faulty sensed data is generally considered to be in a range $[x, y]$. This range can be further split into

intervals, $I = \{i_1, i_2, \ldots, i_n\}$. Each of these intervals will represent a class of our system. Initially, we infer the classifier using maximum posteriori (MAP) (Hill et al., 2009; Mitchell et al., 1997), and the inferred map along with Bayes rule can be represented by

$$i_{MAP} = \underset{ij \in C}{\operatorname{argmax}} \frac{P(H_o|i_j)P(H_n|i_j)P(x_n|i_j)P(i_j)}{P(H_o, H_n, x_n)} \tag{25}$$

where $P(i_j)$ is the probability of sensed data falling in one of the intervals (class), $P(H_o|i_j)$ is the probability of the last sensed data of the same sensor node falling in some class, given sensed data falling in some other class, $P(H_n|i_j)$ is the probability of the last sensed data (history) of the neighbor's sensor node falling in some class, given sensed data falling in some other class, and $P(x_n|i_j)$ is the probability of neighbor's sensed data falling in some class, given sensed data falling in some other class.

Next, each node compares the class of the sensed data with the class derived from Eq. (25). If they are equal then the node is considered as nonfaulty else it is considered as faulty. If the data is nonfaulty then it sets *report*=0 else it sets *report*=1. *report* indicates the decision taken by the sensor node in level 1. The sensed data and the associated report value is sent by the sensor nodes to the cluster heads for further decisions on the fault status of the node. The cluster heads maintain two tables called the Normal table and the Anomaly table. The sensed data at time t, x_i^t, is stored in the Normal table if its associated report value is zero else the sensed data along with the node Id is stored in the Anomaly table. Next, the data stored in both the tables are compared with each other using a counter, *cp*. If $x_q{}^t$ be the value sensed by node q in the Anomaly table and if $x_{n1}^t, x_{n2}^t, \ldots, x_{nm}{}^t$ be the node values sensed by nodes $n_1, n_2, n_3, \ldots, n_n$ in the Normal table, then it should satisfy the condition

$$|x_q^t - x_{ni}^t| \geq \alpha_1 \quad 1 \leq i \leq m \tag{26}$$

where α_1 is a threshold. For each node when the condition is satisfied the counter, *cp*, is incremented. The counter is then compared with another threshold, α_2, which represents the degree of similarity. If $cp > \alpha_2$ then the node is not considered as faulty. The number of classes into which the range of values is partitioned is also an important parameter that decides the quality of this technique. The authors target gain faults, out-of bound faults, and stuck-at faults.

Similar to Titouna et al. (2015a), Titouna et al. (2015b) propose another hybrid hierarchical technique based on the Naïve Bayes classifier for fault detection in WSN. Similar to the previously proposed technique, this technique also consists of two levels of detection. In the first level, the sensors self-detect the fault and compute Joint probabilities. Whereas, in the second phase these joint probabilities are send to the cluster heads to make final decision on the fault status of the nodes. Initially in the first stage, the sensor nodes compute the conditional probability and the Joint Probability Table (JPT). Two parameters are considered as the parents for a sensor node N_i, its remaining energy level $E_i{}^t$ and the data sensed $x_i{}^t$. The conditional probability can be calculated by each node:

$$P_i^t(N_i|EL_i^t) = \frac{P_i^t(EL_i^t|N_i)P(N_i)}{P(EL_i^t)} \tag{27}$$

$$P_i^t(N_i|x_i^t) = \frac{P_i^t(x_i^t|N_i)P(N_i)}{P(x_i^t)} \tag{28}$$

where Eq. (27) represents the conditional probability of the sensor node N_i given the remaining energy level $EL_i{}^t$ at time t, and Eq. (28) represents the conditional probability of the sensor node N_i given the sensed data $x_i{}^t$ at time t.

Next the node computes the joint probability distribution. The joint probability distribution combines all the parameters and is given by

$$PJ_i^t(N_i|EL_i^t x_i^t) = P_i^t(N_i|EL_i^t|)P_i^t(N_i|x_i^t)P_i^t(EL_i^t)P_i^t(x_i^t) \tag{29}$$

where $PJ_i{}^t$ is the joint probability distribution of the sensor node N_i at time t. It should not exceed a certain threshold ϑ. If this condition is met, the corresponding sensor node N_i is possibly faulty (PF), otherwise, it may be possibly normal (PN). This local decision is transmitted to the cluster head along with the corresponding joint probability. Next the joint probability at time $t+1$ is also calculated by the sensor nodes and they are transmitted to the cluster heads. The send sensed data, and the joint probabilities at time t and $t+1$, is stored in the joint probability table at the cluster heads. Next it compares the joint probabilities at time t with $t+1$ by using the following condition:

$$|PJ^t - PJ^{t+1}| > \vartheta_2 \tag{30}$$

where ϑ_2 is a threshold.

If the above condition is satisfied then the sensor node is faulty and is blacklisted.

Wu et al. (2007) propose a two layered fault detection technique based on majority- voting technique. The decision mechanism is divided into two stages. The first stage takes place at the sensor node and the second stage takes place at a centralized fusion center to which all the sensor nodes are connected. In the first stage, each sensor node independently decides locally the fault status of the data and sends its binary decision to the fusion center that takes a further decision based on the data received from all the nodes. The fusion center processes the data at a time step t. Hence, all the initial decisions from the sensor nodes up to time t, from all the nodes are stored at the fusion center. The fusion center consists of a record table that stores the information of the nodes including the initial fault status from the local decision phase. The fusion center calculates the ratio of faulty initial status to all the status of nodes that has been sending to the fusion center by the ith node, denoted by $R_i{}^t$. $R_i{}^t$ is also called the rate of decision. When the rate of decision for any node varies highly from other nodes then that node has a higher probability of being faulty. The entire rate value is divided into q equal intervals. If p_i is a single interval after the division then it is given as:

$$\text{range of } p_i = \left[\frac{i-1}{q}, \frac{i}{q}\right) \tag{31}$$

where $i = 1, 2, \ldots, q$.

The rate of each sensor nodes is placed in the above interval and the intervals grouped together. A majority-voting is performed to find out which group has the maximum rate. The nodes that have values in the smallest rate groups are considered to be faulty. The authors investigate stuck-at faults and random faults.

Kaur and Sharma (2010) propose an agreement based fault detection scheme to detect the failure of Cluster Heads in a clustered WSN. Each cluster member independently detects CH failure and they employ a distributed agreement protocol to reach an agreement on the failure of the cluster heads. Each of the cluster members has a status vector. Each bit of the status vector indicates the fault status decision of the cluster head made by each member of the cluster. Initially, all the bits in this vector are set to zero. The cluster head of each cluster sends a hello message to its cluster members. The cluster members that do not receive the message set the corresponding status bit to 1 and locally decide that the CH has failed and broadcast the status vector. The other cluster member that receives the message integrates the status vector with their copy. At the end of a TDMA schedule, the cluster members reach an agreement about the failure of the cluster head. If all the status vector bits are set to 1, then the cluster head is faulty. The authors investigate the hardware failure of cluster heads in this technique.

Nguyen et al. (2013) discuss a hybrid fault detection and classification scheme based on neighborhood technique and time series analysis. The proposed detection technique is a two phase detection scheme that uses a combination of neighborhood voting technique and the ARMA (Autoregressive Moving Average) model for time series data analysis. In the first step, the a sensor node s_i collects data from all of its

neighbors, s_j. The authors then calculate the median of the data gathered from the neighbors. Let x_j denote the calculated median. Next, the sensor node s_i finds the difference between its reading x_i and the mean of its neighbor's x_j. If it is below a certain threshold τ, then the value is said to be correct else it is assumed to be faulty. The above-discussed phase is based on the spatial correlation among the sensor node. The second step relies on the temporal correlation between the readings of a sensor node. In the second phase, the ARMA model of the readings is formed, and the observed value is compared with the predicted value. If the difference between them is greater than a threshold δ, it is tagged as faulty. The authors investigate hardware faults such as hardware failure, drift faults, and data faults such as random faults, and gain faults.

5. Comparison

In this section, we compare various fault detection techniques that were discussed in the previous section. We discuss the advantages and shortcomings of current techniques and provide a comparison table. We also present a quantitative analysis of statistic based neighborhood techniques, where we analyze the fault detection accuracy and false positive ratio. We also determine the quality of the detection technique using Matthewson Correlation Coefficient.

5.1. Qualitative comparison

Tables 1 and 2 list out the major advantages and disadvantages of fault detection algorithms and Table 3 is a comparison of various fault detection algorithm.

Centralized fault detection algorithm (Panda et al., 2014; Warriach and Tei, 2013; Jin et al., 2015; Lau et al., 2014) has a higher packet transmission rate which normally results in network congestion. The collection and analysis of data from various nodes is an energy depleting process. The Hidden Markov chain algorithm (Warriach and Tei, 2013) and Neural network algorithm (Obst, 2014) are computationally intensive and depend on the quality of the training data provided to train these algorithms. Algorithms based on the transmission time of the packets (Lau et al., 2014) do not provide accurate detection when the network traffic is high. When the network traffic is high it increases the false positive rate. Neighborhood algorithms (Saihi et al., 2013; Nitesh and Jana, 2015; Panda and Khilar, 2015, 2014; Feng et al., 2014) are heavily dependent upon the node degree. They do not perform efficiently at lower node degrees. There are two techniques that are suggested to increase the efficiency for neighborhood-based techniques. One is to increase the density of sensor nodes so that the degree increases or alternatively increase the transmission power. But increasing density is expensive and increasing transmission range is not energy efficient. Techniques based on mean (Panda and Khilar, 2014; Panda et al., 2014) are not robust. A single highly skewed value is enough to mislead the algorithm. Fault detection based on the median by Panda and Khilar (2015) is a robust technique that reduces the degree required for efficient detection to a minimum of ten nodes. But the degree is still higher. If the data is skewed in only one direction, then the median also gets skewed and hence we need better techniques with better Gaussian efficiency. Weighted Neighborhood techniques (Feng et al., 2014) are dependent on parameters, which are difficult to optimize. The state graph algorithm proposed by Banerjee et al. (2014) gives freedom in the selection of the threshold unlike the mean and median based technique. Moreover, the technique provides a complete fault detection including the hardware faults and data outliers. Furthermore, Banerjee et al. (2014) provide replacement of faulty nodes. Algorithms using multi-tiered WSN (Wang and Chen, 2013; Nitesh and Jana, 2015) have a higher time

Table 1
Advantages of fault detection techniques.

Techniques	Advantages
Panda et al. (2014)	Good fault alarm rate (nearly zero)
Warriach and Tei (2013)	Can label and classify the errors. Robust against security attacks
Jin et al. (2015)	Passive algorithm. Less energy usage. Detects the failure of sensor node
Lau et al. (2014)	Does not require any extra transmission of information. Hence energy is preserved
Wang and Chen (2013)	Differentiates Faults and Events
Feng et al. (2014)	Enhances detection accuracy and fault alarm rate
Panda and Khilar (2015)	Low power usage due to reduction in transmission. Useful for even smaller degree WSN. Highly robust. Lower Message complexity. Low time latency
Saihi et al. (2013)	A very good False positive ratio of nearly zero
Panda and Khilar (2014)	Detects Byzantine faults and soft faults
Yuan et al. (2015)	Good detection accuracy possible with lower degrees of nodes
Nitesh and Jana (2015)	The Message complexity is linear
Obst (2014)	Higher Detection Rate. Spatio-temporal technique
Xu et al. (2014)	Detects transient faults and permanent Faults. Considers spatio-temporal values. Low energy usage. Reduces communications. Fault correction
Yang et al. (2015)	Fastest algorithm to detect errors
Banerjee et al. (2014)	Exploits node in fullest extend before detecting it as faulty. Replaces a faulty node with healthy node
Mo et al. (2015)	Passive algorithm. Hence saves energy
Yuvaraja and Sabrigiriraj (2015)	Detects node failure and network failures. Repairs the network on detection of fault
Titouna et al. (2015a)	Comprehensive detection of faults. Considers spatio temporal values
Titouna et al. (2015b)	Considers spatio-temporal values for fault detection. Clustered structure decreases data transfer which reduces the power consumption
Wu et al. (2007)	Detects permanent and transient faults. Lesser communication than most distributed schemes
Kaur and Sharma (2010)	Energy saving during the replacement after the detection of faults
Sahoo and Khilar (2014b)	Detects both intermittent, transient, and permanent faults
Fang and Dobson (2013)	Low false positive rate
Sahoo and Khilar (2014a)	Saves energy by reducing the broadcasted packets. Spanning tree reduces the network congestion. Detects both intermittent and permanent faults
Huang (2015)	Periodically active detection scheme decreases energy consumption and congestion
Mahapatro and Panda (2014)	Detects intermittent errors as well as permanent errors. Uses swarm optimization to decide the optimal time period in which the detection has to be performed. Decreases the power consumption and throughput in the network
Ghorbel et al. (2015)	Lower classification time. Lower memory requirements as compared other PCA based techniques
Nguyen et al. (2013)	Can detect both transient and permanent fault. Can classify the faults according to the type of faults
Kamal et al. (2014)	Accurate than passive techniques but does not have the network overhead of general active techniques
Yang et al. (2014)	A separate channel eradicates the network congestion
Guo et al. (2014)	A model indifferent fault detection technique
Sharma and Sharma (2016)	Energy efficient. Reduces the communication in the network which reduces congestion in the network
Mahapatro and Khilar (2014)	Online fault detection that reduces energy and message overheads. Detects both hardware and software faults
Abid et al. (2015)	Higher Detection accuracy with less False positive ratio
Chanak and Banerjee (2016)	Increases the reusability of nodes. Overcomes uncertainties in environment

Table 2
Disadvantages of fault detection techniques.

Techniques	Disadvantages
Panda et al. (2014)	Higher Traffic and communication latency. It Requires a higher node degree for accurate detection
Warriach and Tei (2013)	Computationally Intensive. Detection depends upon the accuracy of training set
Jin et al. (2015)	Cannot find out the exact number of faulty nodes
Lau et al. (2014)	Does not work properly in congested networks. The false positive rate is quadrupled at higher traffic
Wang and Chen (2013)	Border nodes in clusters might mislead the fault detection
Feng et al. (2014)	FAR ratio not good enough. Parameter selection is difficult. Topology dependent
Panda and Khilar (2015)	Does not detect intermittent, transient and Byzantine faults
Saihi et al. (2013)	Requires WSN with high degree and nodes. Only efficient when faulty sensor less than half the total sensor
Panda and Khilar (2014)	Difficulty in deciding the parameters
Yuan et al. (2015)	Probabilities a per-requisite for calculation of posterior fault probability
Nitesh and Jana (2015)	Increases time latency in fault detection
Obst (2014)	Computationally intensive
Xu et al. (2014)	Difficulty in deciding the parameters
Yang et al. (2015)	Increases congestion in network. It increases the processing at the sink which depletes the battery
Banerjee et al. (2014)	Parameter decision is difficult. Based on mean which is highly skewed on single large error
Mo et al. (2015)	Training time is more
Yuvaraja and Sabrigiriraj (2015)	Random broadcasting of packets results in packets not reaching some nodes. Large number of data transmitted increases power consumption
Titouna et al. (2015a)	Higher detection time due to two layered detection scheme. Initial fault detection rate is less. The rate increases with time
Titouna et al. (2015b)	Two layered detection scheme increases the detection time. Parameter decision is difficult
Wu et al. (2007)	Does not provide a first stage fault detection scheme. Only discusses the detection in the second stage. Use of majority-voting might fail when majority of the nodes fail
Kaur and Sharma (2010)	Only detects hard faults. Broadcasts by all cluster member nodes result in energy overhead
Sahoo and Khilar (2014b)	Parameter decision is difficult. Detection accuracy decreases with low node degrees. Increase in energy consumption and congestion due to broadcast of large number of packets
Fang and Dobson (2013)	Tradeoff between energy and detection rate need to balance with the error bound
Sahoo and Khilar (2014a)	Detection accuracy decreases with low node degrees. Parameter decision is difficult
Huang (2015)	Lower fault detection accuracy and higher false positive rate. Parameter decision is difficult
Mahapatro and Panda (2014)	Slower fault detection rate
Ghorbel et al. (2015)	Kernel PCA takes more time than normal PCA which slows down the training
Nguyen et al. (2013)	Requires periodic calculation of ARMA model. Higher number of faulty nodes leads to higher false alarm
Kamal et al. (2014)	Can only detect hardware errors and network errors such as node failures
Yang et al. (2014)	Extra wireless transmission results in extra energy consumption
Guo et al. (2014)	Higher false negative rate than the optimal rate
Sharma and Sharma (2016)	Need to select more than two thresholds which requires extra statistical analysis
Mahapatro and Khilar (2014)	Detection latency for transient faults are higher
Abid et al. (2015)	Higher fault detection latency. Higher network congestion due to centralized topology
Chanak and Banerjee (2016)	Cannot cope up with changing nature of WSN. Does not balance energy usage. More time for disconnected sensors

latency as the data has to travel through multiple layers of topology. The multi-tiered algorithms with two layered detection scheme such as Titouna et al. (2015a,b), Wu et al. (2007), and Nguyen et al. (2013) increases the detection accuracy and reduces false positives but the latency of detection increases.

The Passive algorithm proposed by Jin et al. (2015) reduces the energy consumption but does not completely detect the total number of fault nodes. It only monitors the health of the node. Cloud-based algorithm (Yang et al., 2015) decreases the time latency but the clustering according to real-time model is done by the base station before being send to the cloud. This depletes the energy of the intermediate base station between the cloud and the sensor nodes. Yang et al. (2014), Kamal et al. (2014), Kaur and Sharma (2010), and Yuvaraja and Sabrigiriraj (2015) are hardware and network fault detection schemes. Kamal et al. (2014) discuss a passive network fault diagnostic scheme that detects the network failures. This technique saves the energy of the network by piggy backing the information needed for fault detection through the normal sensed data but the technique cannot find the link failure when the node is idle. Moreover the detection accuracy decreases when the control messages are lost due to poor quality of network. Yang et al. (2014) propose the uses of external power measurement to decide the internal health of the nodes. This technique uses the power consumed by the sensor nodes to decide the type of failure. High noise can easily interfere this technique. Guo et al. (2014) discuss a fault detection algorithm that takes the distance of the node from an event for detecting faults. One major disadvantage of this technique is that the false alarm rate increases drastically when multiple events occur in the vicinity of a node. Guo et al. (2014) create a list of blacklisted node that it detects as faulty in order of fault likelihood.

5.2. Quantitative comparison

In this section, we make a quantitative analysis of the techniques that have been selected from qualitative analysis on the basis of following criteria:

- Not computationally intensive for WSN.
- Ease in assigning an optimal threshold parameter.
- Better performer qualitatively.

Based on these criteria, we have selected the following statistical and neighborhood fault detection strategies for quantitative analysis.

- Median absolute based deviation technique (MADN) (Panda and Khilar, 2015).
- Centralized Mean (MeanC) (Panda et al., 2014).
- Distributed Mean (MeanD) (Panda and Khilar, 2012).
- Energy-Efficient Algorithm (MV) (Panda and Khilar, 2014).
- Trust Matrix (Trust) (Wang and Chen, 2013).
- Error Function (EF) (Saihi et al., 2013).
- State-graph model (S-Graph) (Banerjee et al., 2014).

We also selected the following fault detection techniques based on soft computing and cloud.

- Hidden Markov Model (HMM) (Warriach and Tei, 2013; Warriach et al., 2012).
- Spatially Organized Distributed Echo State Network (SODSEN) (Obst, 2009, 2014).
- Naïve Bayes classifier (Bayes) (Titouna et al., 2015a).
- Cloud based fault detection (Cloud) (Yang et al., 2015).

Table 3
Comparison of fault detection techniques.

Technique	Classification	Approach	Detection approach	Topology dependent	Threshold based	Robust against attacks	Fault detection	
							Permanent	Transient
Panda et al. (2014)	Centralized	Statistic	Active	Yes	No	Yes	Yes	No
Warriach and Tei (2013)	Centralized	Machine learning	Active	No	No	Yes	Yes	Yes
Jin et al. (2015)	Centralized	Statistics	Passive	No	No	No	Yes	Yes
Lau et al. (2014)	Centralized	Machine learning	Active	No	Yes	Yes	Yes	Yes
Wang and Chen (2013)	Distributed	Hybrid	Active	No	Yes	No	Yes	No
Feng et al. (2014)	Distributed	Weighted voting	Active	Yes	Yes	No	Yes	No
Panda and Khilar (2015)	Distributed	Self detection	Active	Yes	Yes	Yes	Yes	No
Saihi et al. (2013)	Distributed	Statistics	Active	Yes	No	No	Yes	No
Panda and Khilar (2014)	Distributed	Self detection	Active	Yes	Yes	No	Yes	No
Yuan et al. (2015)	Distributed	Statistic	Active	Yes	Yes	No	Yes	No
Nitesh and Jana (2015)	Distributed	Hybrid	Active	No	Yes	No	Yes	No
Obst (2014)	Distributed	Supervised learning	Active	No	Yes	No	Yes	Yes
Xu et al. (2014)	Distributed	Self detection	Active	Yes	Yes	No	Yes	Yes
Yang et al. (2015)	Distributed	Clustering	Active	No	No	No	Yes	No
Banerjee et al. (2014)	Distributed	Statistic	Active	Yes	Yes	No	Yes	No
Mo et al. (2015)	Centralized	Machine learning	Passive	No	No	Yes	Yes	Yes
Yuvaraja and Sabrigiriraj (2015)	Distributed	Neighbor	Active	No	No	Yes	Yes	Yes
Titouna et al. (2015a)	Hybrid	Machine learning	Active	No	Yes	Yes	Yes	Yes
Titouna et al. (2015b)	Hybrid	Machine learning	Active	No	Yes	Yes	Yes	Yes
Wu et al. (2007)	Hybrid	Majority-voting	Active	Yes	No	Yes	Yes	Yes
Kaur and Sharma (2010)	Hybrid	Neighbor	Active	Yes	No	No	Yes	No
Sahoo and Khilar (2014b)	Distributed	Neighbor	Active	No	Yes	No	Yes	Yes
Fang and Dobson (2013)	Distributed	Majority-voting	Active	No	Yes	No	No	Yes
Sahoo and Khilar (2014a)	Distributed	Neighbor	Active	No	Yes	No	Yes	Yes
Huang (2015)	Distributed	Self detection	Active	No	Yes	No	Yes	No
Mahapatro and Panda (2014)	Distributed	Neighbor	Active	No	Yes	No	Yes	Yes
Ghorbel et al. (2015)	Distributed	Machine learning	Active	No	Yes	No	Yes	Yes
Nguyen et al. (2013)	Hybrid	Majority-voting	Active	Yes	Yes	No	Yes	Yes
Kamal et al. (2014)	Centralized	Statistics	Active	No	No	No	Yes	Yes
Yang et al. (2014)	Centralized	Machine Learning	Active/passive	No	No	No	Yes	No
Guo et al. (2014)	Centralized	Statistics	Active	No	Yes	No	Yes	No
Sharma and Sharma (2016)	Distributed	Neighbor	Active	Yes	Yes	No	Yes	Yes
Mahapatro and Khilar (2014)	Distributed	Neighbor	Active	No	Yes	No	Yes	Yes
Abid et al. (2015)	Centralized	Machine learning	Active	No	No	No	Yes	Yes
Chanak and Banerjee (2016)	Distributed	Soft computing	Active	Yes	Yes	Yes	Yes	Yes

We designed and simulated the algorithms using the statistical tool R (R Core Team, 2015) and Octave (Eaton et al., 2009). We used a real life data set consisting of fifty-four sensor nodes deployed in the Intel Berkeley Research lab between February 28 and April 5, 2004 (http://db.csail.mit.edu/labdata/labdata.html). Out of the fifty-four sensors, we excluded two sensor readings due to inconsistencies and incomplete data. In this dataset, humidity, temperature, light and voltage values are collected once every 31 s. But we have only considered temperature data. They can be considered to be placed in a rectangular grid of size 40×30. The location of these sensors in the lab can be viewed in Fig. 6.

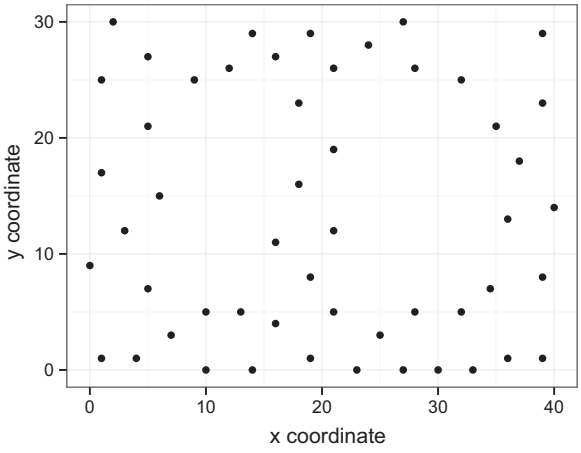


Fig. 6. Location of sensor nodes.

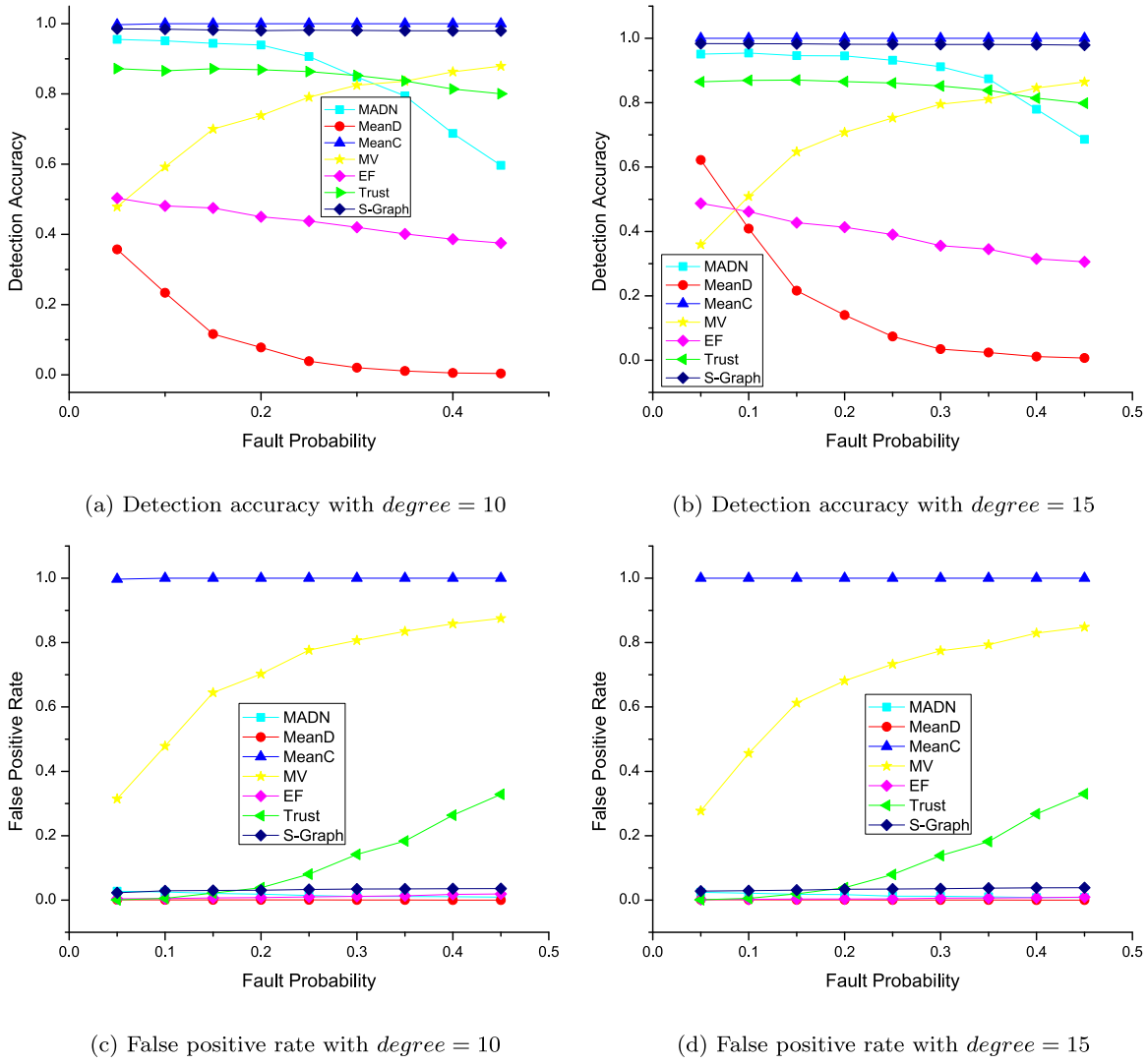
Each algorithm was run with average degrees, d=10 and d=15. To achieve this, the transmission ranges were fixed as 12 and 15.3 respectively. Random faults were introduced into random nodes with fault probability of 0.05–0.4, with a step size of 0.05. For the introduced fault probabilities, each fault detection technique was executed thousand times and the mean values were considered for calculations. The Detection Accuracy and Fault Positive Rate of the algorithms were calculated. The detection accuracy (DA) is defined mathematically as

$$DA = \frac{\text{Number of faulty node detected}}{\text{Total number of faulty node}} \tag{32}$$

False positive rate (FPR) is the proportion of normal nodes (non-faulty) that are reported as faulty, which is also known as the false alarm rate (FAR). It is the ratio of the non-faulty nodes detected as faulty to the total number of fault-free nodes. Eq. (33) gives the mathematical formula for FPR:

$$FPR = \frac{\text{Number of non} - \text{faulty node detected as faulty}}{\text{Total fault free nodes}} \tag{33}$$

The False positive rate (FPR) is also known as FAR (False alarm rate).



(a) Detection accuracy with $degree = 10$

(b) Detection accuracy with $degree = 15$

(c) False positive rate with $degree = 10$

(d) False positive rate with $degree = 15$

Fig. 7. Detection accuracy and false positive rate of selected fault detection techniques.

5.2.1. DA and FPR analysis

The DA and FPR of all the algorithms with average degrees, d=10 and d=15, have been plotted in Fig. 7a, b, c, and d respectively. As we can see from the figures the detection accuracy decreases as fault probability increases for most of the algorithms except MV and MeanC. MV and MeanC as shown in the graph have higher false positive rate with higher fault probabilities. This is due to majority voting step in MV technique. For any given fifteen neighbors, if seven was detected probably faulty in the first step, then they are reclassified as non-faulty in majority voting phase. This is because seven of the nodes will report a fault, but eight of them will report non-faulty and hence non-fault status is assigned to the nodes which result in higher fault alarm rate at higher error probability. The performance of Median (MADN) (Panda and Khilar, 2015) is better than Mean Centralized (MeanC), Mean Distributed (MeanD), Error Function (EF), and Energy efficient algorithm from panda (MV) (Panda and Khilar, 2014; Xu et al., 2014; Saihi et al., 2013). Panda et al. (2014) are poor in detecting the faulty nodes and have a very high false positive rate. The algorithms that utilized mean (Panda and Khilar, 2014; Xu et al., 2014; Saihi et al., 2013) for finding outlier fared poorly as the faults increased. Their false positive rate increases as the fault probability increases. The false positive rate of MeanC (Panda et al., 2014) and MeanD jumped to one at lower fault probability, which indicates that it is not an efficient and accurate technique. Since the mean is not a robust measurement, we can see a

Table 4
Average detection accuracy and false positive ratio for *degree*=10.

Technique	Average detection accuracy (%)	Average false positive rate (%)
MADN	84.69	1.6
MeanC	99.96	99.9
MeanD	9.60	0.4
MV	74.47	69.9
Trust	84.95	11.83
EF	43.67	1.05
S-Graph	98.15	3.17

Table 5
Average detection accuracy and false positive ratio for *degree*=15.

Technique	Average detection accuracy (%)	Average false positive rate (%)
MADN	88.66	1.45
MeanC	100	100
MeanD	17.07	0.04
MV	69.91	63.17
Trust	84.83	11.79
EF	38.89	0.48
S-Graph	98.15	3.41

large deviation. Comparatively, the DA for Panda and Khilar (2015) is better than other statistic and neighborhood-based techniques. Comparatively, the DA for Panda and Khilar (2015) is better than other statistic and neighborhood-based techniques. The Error function technique has comparatively better performance than the strategies that uses mean to detect faults. We have provided average DA and FA of analyzed techniques for degree=10 and degree=15 in Tables 4 and 5 respectively. Centralized techniques provide a better performance as the sink receives data from all the sensor nodes. However, we can observe that the MeanD technique has a lower false positive rate than MeanC. Both these techniques use the mean value of the sensed data to create a threshold. The major issue with mean is that it easily gets deviated when there are a few large data values. Moreover, the larger the number of greater data values, the larger the deviation of the mean. In the distributed MeanD technique the data sets with large values get distributed among different neighbors locally unlike the MeanC technique where all the outlier values are used to calculate the threshold for fault. When all the values are used for determining the threshold, it will cause a higher deviation of the mean when some sensors are faulty, and it hence has a higher effect on the z-value. If the mean and standard deviation were replaced by Median and Median absolute based deviation (MADN) in the centralized technique, it would produce a better performance than the distributed MADN technique. However, centralized schemes are avoided due to congestion in the network. Moreover, the fault threshold in the MeanC is dependent on the z-value whereas the distributed techniques such as MeanD and MADN use a threshold that is constant and do not change upon the modification in data values. The S-graph technique provides excellent performance, with a detection accuracy above 90% and a false positive rate of 3%. The threshold in this technique can be varied to improve the performance. This flexibility in selecting the threshold parameter is one of the reasons that it performs better than MeanC, MeanD, and MADN. The threshold values in MeanC, MeanD, and MADN are data dependent and hence affect the detection quality depending on the total quality of the sensed data. From the table, we can see that MeanC has 100% DA but also 100% false positive rate whereas trust technique has only 84.9% detection but a low false positive rate of 11%. Hence, in the next subsection we calculate the Matthews (1975) correlation coefficient to provide a better perspective on the tradeoff between the false positive rate and detection accuracy. These tests are used to rank the fault detection algorithms based on their detection performance.

While comparing machine learning and cloud-based techniques, the technique based on the Naïve Bayes algorithm outperforms the other techniques in terms of the Detection Accuracy. The HMM-based technique shows a detection accuracy of 93%. The DA and the average FPR of machine learning and cloud techniques is depicted in Fig. 8a and b respectively. The SODSEN technique's performance decreases once the faults were increased. This is due to the issues in training provided. Detection Accuracy for SODSEN decreases when the deviation between the prediction and the real value does not exceed the provided threshold. This happens due to improper training. Hence, training the network appropriately is necessary for good detection when using SODSEN. A proper training can increase the performance of this technique. The cloud-based technique is not a very robust technique as it is just based on a number of thresholds. The cloud-based fault detection algorithm discussed in Yang et al. (2015) considers out-of-bound faults in one direction only. This results in the decline of detection accuracy from 90% to 70%. This is the reason the performance of the cloud-based technique reduces suddenly. The performance of the cloud-based technique can be increased by adding an extra threshold for the lower bounds. Moreover, the false positive rate of the cloud-based technique is around 0.5%. HMM and Bayes have higher false positive rates than SODSEN and cloud-based technique, but the FAR is significantly lower than the statistical and neighborhood techniques.

5.2.2. Matthews correlation coefficient

Matthews (1975) correlation coefficient is used to test the accuracy of the fault detection techniques discussed in the previous sections. MCC is defined as

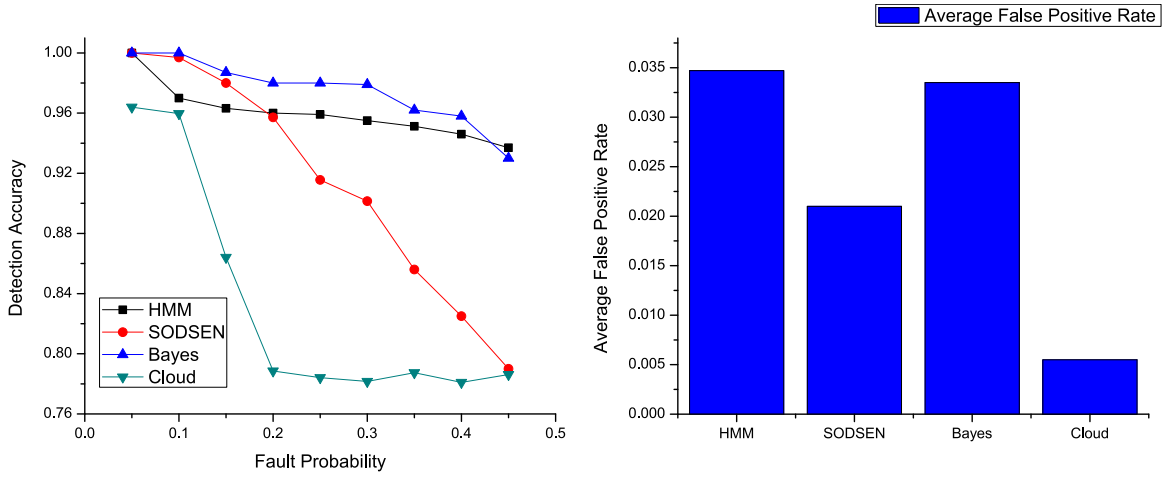
$$MCC = \frac{Tp \times Tn - Fp \times Fn}{\sqrt{(Tp + Fp)(Tp + Fn)(Tn + Fp)(Tn + Fn)}} \tag{34}$$

True Positive (Tp) is defined as the number of faulty nodes detected correctly. False Positive (Fp) is defined as the number of non-faulty nodes detected as faulty. False Negative (Fn) is defined as the number of faulty nodes detected as non-faulty. And True Negative (Tn) is defined as the number of non-faulty nodes defined as non-faulty.

An MCC of +1 represents ideal technique, 0 indicates similarity to random prediction, and a value of −1 indicates a discrepancy in detection technique. The closer the value is to +1, a very strong positive relationship exists between reality and test.

MCC was calculated for the discussed fault detection techniques and have been summarized in Tables 6 and 7.

Techniques with an MCC of +0.70 or higher is considered as an excellent technique. An MCC between +0.40 to +0.69 is considered



(a) Detection accuracy (b) Average False Positive rate

Fig. 8. Detection accuracy and false positive rate of selected Machine learning and cloud based fault detection techniques.

Table 6
MCC for selected statistical and neighborhood fault detection techniques.

Technique	Matthews correlation (MCC)	Rank
MADN	0.92	2
MeanD	0.43	5
MeanC	0	7
MV	0.20	6
EF	0.67	4
Trust	0.81	3
S-Graph	0.95	1

Table 7
MCC for selected soft computing and cloud fault detection techniques.

Technique	Matthews correlation (MCC)	Rank
HMM	0.89	2
SODSEN	0.46	3
Bayes	0.9	1
Cloud	0.41	4

as good technique and +0.30 to +0.39 is considered as moderate technique. MCC in the range +0.20 to +0.29 is classified as a weak relationship. Based on the test we can reach the conclusion that S-Graph is the best performer among the statistical and neighborhood techniques followed by MADN, which is followed by Trust Matrix technique and EF technique. MeanC and MV fare poorly due to higher false negatives. MeanD performs moderately but yet does not touch the threshold of good detection techniques. Among the soft computing and cloud techniques, Bayes have a better performance followed by HMM technique. Cloud and SODSEN produce nearly equal performance. We summarize the advantages and disadvantages of various techniques in Table 8.

6. Future research, issues, and challenges

In the previous section, we saw shortcomings from the analysis of fault detection techniques in wireless sensor networks. From this analysis we can summarize the areas that require more focus for future research.

1. Fault detection for mobile nodes and topology independence. The fault detection techniques discussed only deals with static node. Since most of the detection technique depends upon the topology of the network, these algorithms cannot be applied to the mobile nodes.
2. Dynamic error status detection. The error status of the sensors does not change when it is being determined. Whereas in real conditions, the errors occur dynamically.
3. Parameter selection. Selection of a proper parameter for the fault detection is an issue. Incorrect selection of algorithm leads to loss in detection accuracy.
4. Selection of training data for learning algorithm. Correct selection of training data can lead to higher detection accuracy.
5. Fault replacement, recovery, and fault tolerance. Most of the algorithms do not have a fault recovery scheme or fault replacement scheme.
6. Differentiate between event and error. Most of the algorithms do not differentiate between the occurrence of events and errors.
7. Robustness against security attacks. Most of the fault detection algorithms are susceptible to security attacks. For example, in all the techniques except the machine learning techniques the data send by the neighboring nodes can be intercepted and false information can be sent to the relay node or decision-making node. There is no security mechanism to protect the nodes against security attacks.

Table 8
Summary of fault detection techniques in WSN.

Reference	Major contributions	Fault types
Panda et al. (2014)	Centralized algorithm based on *Z*-score test	Random faults, incorrect computation
Warriach and Tei (2013)	Hidden Markov model to detect, identify and classify faults	System fault, data fault, Gain fault, Stuck at fault
Jin et al. (2015)	An approach based on Auto-regressive model and Kuiper test that monitors the health of nodes	Determines health of node
Lau et al. (2014)	Centralized Technique based on transmission time and Naïve Bayes	System faults and Hard faults
Wang and Chen (2013)	Trust Matrix based fault detection Identify Events from Faults	Permanent fault, Data Fault
Feng et al. (2014)	Neighborhood algorithm based on weight	Data fault, Permanent fault
Panda and Khilar (2015)	A modified three sigma test based on median to detect fault A distributed self-diagnosis algorithm where each sensor node diagnoses itself The message complexity is $O(N)$	Data fault, Permanent fault
Saihi et al. (2013)	Improved DFD algorithm based on error functions and Majority voting	Data fault, Permanent fault
Panda and Khilar (2014)	Distributed self fault detection algorithm which is energy efficient Message Complexity is $O(2N)$ Time latency is $2T_{out}$	Data fault, Permanent fault, Incorrect computation
Yuan et al. (2015)	Improved Bayesian algorithm for data fault detection	Data faults, Permanent faults
Nitesh and Jana (2015)	A distributed algorithm based on multi tiered network using time redundancy Message complexity $O(M)$, where M is the number of relay nodes	Hard faults, Transient faults, Permanent faults
Obst (2014)	Supervised learning based algorithm based on spatially organized distributed Echo State Network	Transient, Data faults
Xu et al. (2014)	A low energy fault detection scheme that takes into consideration the spatio-temporal features of the network Fault replacement with detection	Transient, Permanent, Data faults, Hard faults
Yang et al. (2015)	Time efficient strategy to check faults	Data Faults
Banerjee et al. (2014)	A strategy utilize the node to maximum extend before declaring it faulty	Permanent faults, Hard faults
Mo et al. (2015)	A strategy based on Time domain features of sensed data An unsupervised machine learning technique utilizing Gabor transform and Self organized map	Transient, Permanent, Data fault, Gain fault
Yuvaraja and Sabrigiriraj (2015)	A scheme of fault detection based on Agent queries and query path A fault repairing scheme named LeDiR	Hard faults, Permanent faults
Titouna et al. (2015a)	A two-tiered hybrid scheme based on Naïve Bayes and sensor fusion	Data Faults, Transient, Permanent
Titouna et al. (2015b)	A two-tiered fault detection scheme based on conditional probability and Naïve Bayes	Data Faults, Transient, Permanent
Wu et al. (2007)	A two-tiered fault detection scheme based on sensor fusion and majority voting	Data Faults, Transient, Permanent, Random faults
Kaur and Sharma (2010)	A two-tiered hardware (cluster head/gateway) fault detection using distributed agreement among cluster members	Hard faults
Sahoo and Khilar (2014b)	A distributed fault detection scheme for transient and permanent faults based on	Data faults,Transient faults, Permanent faults

(continued on next page)

Table 8 *(continued)*

Reference	Major contributions	Fault types
	neighborhood coordination	
Fang and Dobson (2013)	A distributed statistical technique based on majority-voting for detecting data faults in sensed data	Data faults
Sahoo and Khilar (2014a)	A distributed fault detection scheme for transient and permanent faults that uses spanning tree for broadcasting control data An improvement to their previous work (Sahoo and Khilar, 2014b)	Data faults,Transient faults, Permanent faults
Huang (2015)	A distributed self-detecting fault detection algorithm based on neighborhood sensors	Data faults
Mahapatro and Panda (2014)	A distributed algorithm based on neighborhood comparison for fault detection in WSN Optimization of transient fault detection period using multi objective swarm optimization	Data faults,Transient faults, Permanent faults
Ghorbel et al. (2015)	A Distributed and efficient one-class outlier detection classifier based on kernel based PCA and Mahalanobis distance	Data faults
Nguyen et al. (2013)	A Hybrid decentralized fault detection algorithm based on neighborhood technique and time series analysis	Data faults, Offset fault, Gain fault, Random faults
Kamal et al. (2014)	A packet tagging server side analysis centralized fault detection algorithm	Hard faults, Network faults
Yang et al. (2014)	A novel power diagnostic tool that measures energy consumption to determine the faults An active and a passive diagnostic schemes for the fault diagnosis	Hard faults, Network faults
Guo et al. (2014)	A novel fault detection algorithm called FIND that does not require event injection or a particular sensing model	Data faults
Sharma and Sharma (2016)	A reactive fault detection technique based on the spatio-temporal correlation	Data faults, Transient faults, Permanent faults
Mahapatro and Khilar (2014)	An online distributed fault tolerant algorithm for permanent and transient fault detection	Data faults, Transient faults, Permanent faults, Hardware faults
Abid et al. (2015)	A centralized fault detection scheme based on KNN and Euclidean distance	Data faults
Chanak and Banerjee (2016)	A fuzzy logic based fault detection and classification scheme	Hardware faults

7. Conclusion

In this paper, we have reviewed state-of-art fault detection techniques in WSN and provided an updated technique based taxonomy to categorize them. The taxonomy encompasses all the latest techniques till date. Based on our proposed taxonomy, we have provided a qualitative and quantitative comparison of these techniques. This survey is the first of its kind to provide a quantitative and qualitative comparisons in this area of research to the best of our knowledge. We provided a comparison table that lists out the key characteristics of various techniques and qualitatively evaluated each technique. Furthermore, we have summarized various techniques along with its advantages and disadvantages. From the qualitative analysis, six techniques were selected for quantitative tests. We provide a quantitative analysis and calculate MCC of the selected techniques to find the best technique. The techniques were ranked based on performance. We reached a conclusion that MADN (Panda and Khilar, 2015) and trust matrix (Wang and Chen, 2013) are the best techniques at error probability lesser and greater than 50% respectively. We have outlined prospective future research challenges and issues for fault detection in WSN. The drawbacks of existing fault detection strategies in WSN calls for developing techniques that takes into account mobility, dynamic error status, parameter selection, fault replacement and recovery, and robustness against attacks.

References

Abid, A., Kachouri, A., Guiloufi, A.B.F., Mahfoudhi, A., Nasri, N., Abid, M., 2015. Centralized knn anomaly detector for WSN. In: 2015 12th International Multi-Conference on Systems, Signals Devices (SSD), pp. 1–4. http://dx.doi.org/10.1109/SSD.2015.7348091

Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E., 2002. Wireless sensor networks: a survey. Comput. Netw. 38 (4), 393–422. http://dx.doi.org/10.1016/S1389-1286(01)00302-4.

Andel, Jiri, 1976. Autoregressive series with random parameters. Math. Operationsforschung Stat. 7 (5), 735–741.

Anisi, M., Abdullah, A., Razak, S., 2013. Energy-efficient and reliable data delivery in wireless sensor networks. Wirel. Netw. 19 (4), 495–505. http://dx.doi.org/10.1007/s11276-012-0480-x.

Balzano, L., Nowak, R., 2007. Blind calibration of sensor networks. In: Proceedings of the 6th International Conference on Information Processing in Sensor Networks. IPSN-'07, ACM, Cambridge, Massachusetts, USA, pp. 79-88, http://dx.doi.org/10.1145/1236360.1236372

Banerjee, I., Chanak, P., Sikdar, B. K., Rahaman, H., 2011. DFDNM: A Distributed Fault Detection and Node Management Scheme for Wireless Sensor Network. In: Advances in Computing and Communications: First International Conference, ACC 2011, Kochi, India, July 22-24, 2011, Proceedings, Part III. Springer, pp. 68–81. http://dx.doi.org/10.1007/978-3-642-22720-2/7

Banerjee, I., Datta, A., Pal, S., Chatterjee, S., Samanta, T., 2014. A novel fault detection and replacement scheme in WSN. In: Thampi, S.M., Abraham, A., Pal, S.K., Rodriguez, J.M.C. (Eds.), Recent Advances in Intelligent Informatics: Proceedings of the Second International Symposium on Intelligent Informatics (ISI'13), August 23-24 2013, Mysore, India, Advances in Intelligent Systems and Computing, vol. 235. Springer International Publishing, pp. 303–310. http://dx.doi.org/10.1007/978-3-319-01778-5_31

Book reviews, 1988. J. Am. Stat. Assoc. 83 (403) 902–926. http://dx.doi.org/10.1080/01621459.1988.10478680.

Buonadonna, P., Gay, D., Hellerstein, J.M., Hong, W., Madden, S., 2005. Task: sensor network in a box. In: 2005 Proceedings of the Second European Workshop on Wireless Sensor Networks. IEEE, Istanbul, Turkey pp. 133–144. http://dx.doi.org/10.1109/EWSN.2005.1462005

Bychkovskiy, V., Megerian, S., Estrin, D., Potkonjak, M., 2003. A collaborative approach to in-place sensor calibration. In: Proceedings of the 2nd International Conference on Information Processing in Sensor Networks. IPSN'03. Springer, Palo Alto, CA, USA, pp. 301–316

Casey, K., Lim, A., Dozier, G., 2008. A sensor network architecture for tsunami detection and response. Int. J. Distrib. Sens. Netw. 4 (1), 28–43. http://dx.doi.org/10.1080/15501320701774675.

Chanak, P., Banerjee, I., 2016. Fuzzy rule-based faulty node classification and management scheme for large scale wireless sensor networks. Expert Syst. Appl. 45, 307–321 . http://dx.doi.org/10.1016/j.eswa.2015.09.040http://www.sciencedirect.com/science/article/pii/S0957417415006697.

Chen, J., Kher, S., Somani, A., 2006. Distributed fault detection of wireless sensor networks. In: Proceedings of the 2006 Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks, DIWANS'06. ACM, New York, NY, USA, pp. 65–72. http://dx.doi.org/10.1145/1160972.1160985.

Chessa, S., Santi, P., 2002. Crash faults identification in wireless sensor networks. Comput. Commun. 25 (14), 1273–1282.

Cong, F., Chen, J., Pan, Y., 2011. Kolmogorov-smirnov test for rolling bearing performance degradation assessment and prognosis. 1337–134J. Vib. Control 17 (9). http://dx.doi.org/10.1177/1077546310384003.

Dean, J., Ghemawat, S., 2008. Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (1) 107–113. http://dx.doi.org/10.1145/1327452.1327492.

Dean, J., Ghemawat, S., 2010. Mapreduce: a flexible data processing tool, Commun. ACM 53 (1) (2010) 72–77. http://dx.doi.org/10.1145/1629175.1629198.

Eaton, J.W., Bateman, D., Hauberg, S., 2009. GNU Octave Version 3.0.1 Manual: A High-level Interactive Language for Numerical Computations. CreateSpace Independent Publishing Platform, ISBN 1441413006. URL ⟨http://www.gnu.org/software/octave/doc/interpreter⟩

Fang, L., Dobson, S., 2013. Unifying sensor fault detection with energy conservation. In: Self-Organizing Systems: 7th IFIP TC 6 International Workshop, IWSOS 2013, Palma de Mallorca, Spain, May 9–10, Revised Selected Papers. Springer, Berlin, Heidelberg, 2014, pp. 176–181. http://dx.doi.org/10.1007/978-3-642-54140-7_18.

Feng, Z., Fu, J.Q., Wang, Y., 2014. Weighted distributed fault detection for wireless sensor networks based on the distance. In: 2014 33rd Chinese Control Conference (CCC). IEEE, Nanjing, China, pp. 322–326. http://dx.doi.org/10.1109/ChiCC.2014.6896642

Franke, J., 1985. A Levinson–Durbin recursion for autoregressive-moving average processes. Biometrika 72 (3), 573–581. http://dx.doi.org/10.1093/biomet/72.3.573.

Ghorbel, O., Jmal, M.W., Abid, M., Snoussi, H., 2015. Distributed and efficient one-class outliers detection classifier in wireless sensors networks. In: Wired/Wireless Internet Communications: 13th International Conference, WWIC 2015, Malaga, Spain, May 25–27. Revised Selected Papers. Springer International Publishing, Cham, 2015, pp. 259–273. http://dx.doi.org/10.1007/978-3-319-22572-2_19.

Guo, S., Zhang, H., Zhong, Z., Chen, J., Cao, Q., He, T., 2014. Detecting faulty nodes with data errors for wireless sensor networks. ACM Trans. Sens. Netw. 10 (3) 40–27. http://doi.acm.org/10.1145/2594773.

Hill, D.J., Minsker, B.S., Amir, E., 2009. Real-time Bayesian anomaly detection in streaming environmental data. Water Resour. Res. 45 (4). http://dx.doi.org/10.1029/2008WR006956, w00D28.

Huang, X., 2014. Distributed fault diagnosis of wireless sensor network. In: Advances in Wireless Sensor Networks: The 8th China Conference, CWSN 2014, Xi'an, China, October 31–November 2. Revised Selected Papers. Springer, Berlin, Heidelberg, 2015, pp. 275–283. http://dx.doi.org/10.1007/978-3-662-46981-1_26.

Intel Lab Data. ⟨http://db.csail.mit.edu/labdata/labdata.html⟩).

Jin, X., Chow, T., Sun, Y., Shan, J., Lau, B., 2015. Kuiper test and autoregressive model-based approach for wireless sensor network fault diagnosis. Wirel. Netw. 21 (3), 829–839. http://dx.doi.org/10.1007/s11276-014-0820-0.

Jurdak, R., Wang, X. R., Obst, O., Valencia, P., 2011. Wireless Sensor Network Anomalies: Diagnosis and Detection Strategies. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 309–325. URL http://dx.doi.org/10.1007/978-3-642-17931-012

Kamal, A.R.M., Bleakley, C.J., Dobson, S., 2014. Failure detection in wireless sensor networks: a sequence-based dynamic approach. ACM Trans. Sens. Netw. 10 (2), 35:1–35:29. http://dx.doi.org/10.1145/2530526.

Kar, C., Mohanty, A., 2004. Application of {KS} test in ball bearing fault diagnosis. J. Sound Vib. 269 (1), 439–454. http://dx.doi.org/10.1016/S0022-460X(03)00380-8.

Kaur, A., Sharma, T.P., 2010. AFDEP: agreement based CH failure detection and election protocol for a WSN. In: Information and Communication Technologies: International Conference, ICT 2010, Kochi, Kerala, India, September 7–9, 2010. Proceedings. Springer, Berlin, Heidelberg, pp. 249–257. doi:10.1007/978-3-642-15766-0_36

Kim, D.-J., Prabhakaran, B., 2011. Motion fault detection and isolation in body sensor networks. Pervasive Mob. Comput. 7 (6) 727–745, (the Ninth Annual {IEEE} International Conference on Pervasive Computing and Communications (PerCom 2011)). http://dx.doi.org/10.1016/j.pmcj.2011.09.006.

Kohonen, T., 1998. The self-organizing map. Neurocomputing 21 (1–3), 1–6. http://dx.doi.org/10.1016/S0925-2312(98)00030-7.

Kutten, S., Peleg, D., 1995. Fault-local distributed mending (extended abstract). In: Proceedings of the Fourteenth Annual ACM Symposium on Principles of Distributed Computing, PODC '95. ACM, New York, NY, USA, pp. 20–27. http://doi.acm.org/10.1145/224964.224967.

Kutten, S., Peleg, D., 1999. Fault-local distributed mending. J. Algorithms 30 (1), 144–165. http://dx.doi.org/10.1006/jagm.1998.0972.

Lam, C., 2010. Hadoop in Action, 1st edition. Manning Publications Co., Greenwich, CT, USA,

Lau, B.C., Ma, E.W., Chow, T.W., 2014. Probabilistic fault detector for wireless sensor network. Expert Syst. Appl. 41 (8), 3703–3711.

Lee, P.M., 2012. Bayesian Statistics: An Introduction 4th Edition. Wiley Publishing.

Leys, C., Ley, C., Klein, O., Bernard, P., Licata, L., 2013. Detecting outliers: do not use standard deviation around the mean, use absolute deviation around the median. J. Exp. Soc. Psychol. 49 (4), 764–766.

Lindley, D., 1972. Bayesian Statistics: A Review. Society for Industrial and Applied Mathematics, Ch. Bayesian Statistics a Review, 1–74. http://dx.doi.org/10.1137/1.9781611970654. ch1.

Liu, W., Zhang, Y., Lou, W., Fang, Y., 2006. A robust and energy-efficient data dissemination framework for wireless sensor networks. Wirel. Netw. 12 (4), 465–479. http://dx.doi.org/10.1007/s11276-006-6546-x.

Louter, A.S., Koerts, J., 1970. On the Kuiper test for normality with mean and variance unknown. Stat. Neerlandica 24 (2), 83–87. http://dx.doi.org/10.1111/j.1467-9574.1970.tb00110.x.

Mahapatro, A., Khilar, P., 2013. Fault diagnosis in wireless sensor networks: a survey. IEEE Commun. Surv. Tutor. 15 (4), 2000–2026. http://dx.doi.org/10.1109/SURV.2013.030713.00062.

Mahapatro, A., Khilar, P., 2014. Online fault diagnosis of wireless sensor networks. Open Comput. Sci. 4 (1), 30–44. http://dx.doi.org/10.2478/s13537-014-0203-8.

Mahapatro, A., Panda, A.K., 2014. Choice of detection parameters on fault detection in wireless sensor networks: a multiobjective optimization approach. Wirel. Pers. Commun. 78 (1), 649–669. http://dx.doi.org/10.1007/s11277-014-1776-1.

Maronna, R., Martin, D., Yohai, V., 2006. Robust Statistics. John Wiley & Sons, Chichester, ISBN.

Massey Jr., F.J., 1951. The Kolmogorov–Smirnov test for goodness of fit. J. Am. Stat. Assoc. 46 (253) 68–78. http://dx.doi.org/10.1080/01621459.1951.10500769.

Matthews, B., 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochim. Biophys. Acta (BBA) – Protein Struct. 405 (2), 442–451. http://dx.doi.org/10.1016/0005-2795(75)90109-9.

Michie, D., Spiegelhalter, D.J., Taylor, C., 1994. Machine Learning, Neural and Statistical Classification.

Mitchell, T.M., et al., 1997. Machine Learning.

Mo, L., Li, J., Wang, G., Chen, L., 2015. Passive diagnosis for WSNs using time domain features of sensing data. Int. J. Distrib. Sens. Netw. 501, 590430.

Nguyen, T.A., Bucur, D., Aiello, M., Tei, K., 2013. Applying time series analysis and neighbourhood voting in a decentralised approach for fault detection and classification in WSNs. In: Proceedings of the Fourth Symposium on Information and Communication Technology, SoICT '13. ACM, New York, NY, USA, pp. 234–241. http://dx.doi.org/10.1145/2542050.2542080.

Ni, K., Pottie, G., 2012. Sensor network data fault detection with maximum a posteriori selection and Bayesian modeling. ACM Trans. Sens. Netw. 8 (3), 23:1–23:21. http://dx.doi.org/10.1145/2240092.2240097.

Nitesh, K., Jana, P., 2015. Dfda: a distributed fault detection algorithm in two tier wireless sensor networks. In: Satapathy, S.C., Biswal, B.N., Udgata, S.K., Mandal, J. K. (Eds.), Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014, Bhubaneswar, Odisha, India. Advances in Intelligent Systems and Computing, vol. 328. Springer International Publishing, pp. 739–746

Obst, O., 2009. Poster abstract: distributed fault detection using a recurrent neural network. In: International Conference on Information Processing in Sensor Networks, 2009. IPSN 2009, pp. 373–374.

Obst, O., 2014. Distributed fault detection in sensor networks using a recurrent neural network. Neural Process. Lett. 40 (3), 261–273. http://dx.doi.org/10.1007/s11063-013-9327-4.

Panda, M., Khilar, P., 2012. Distributed soft fault detection algorithm in wireless sensor networks using statistical test. In: 2012 2nd IEEE International Conference on Parallel Distributed and Grid Computing (PDGC), pp. 195–198. http://dx.doi.org/10.1109/PDGC.2012.6449816.

Panda, M., Khilar, P., 2014. Energy efficient distributed fault identification algorithm in wireless sensor networks. J. Comput. Netw. Commun.. http://dx.doi.org/10.1155/2014/323754.

Panda, M., Khilar, P., 2015. Distributed self fault diagnosis algorithm for large scale wireless sensor networks using modified three sigma edit test. Ad Hoc Netw. Part A 25, 170–184. http://dx.doi.org/10.1016/j.adhoc.2014.10.006.

Panda, R.R., Gouda, B.S., Panigrahi, T., 2014. Efficient fault node detection algorithm for wireless sensor networks. In: 2014 International Conference on High Performance Computing and Applications (ICHPCA). IEEE, Bhubaneswar, Odisha, India, pp. 1–5

Pantazis, N., Nikolidakis, S., Vergados, D., 2013. Energy-efficient routing protocols in wireless sensor networks: a survey. IEEE Commun. Surv. Tutor. 15 (2), 551–591. http://dx.doi.org/10.1109/SURV.2012.062612.00084.

Peatman, J.G., 1947. Descriptive and Sampling Statistics. Harper.

Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., 1990. Numerical Recipes.

R Core Team, 2015. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. URL ⟨http://www.R-project.org/⟩

Rabiner, L., Juang, B., 1986. An introduction to hidden Markov models. IEEE ASSP Mag. 3 (1), 4–16. http://dx.doi.org/10.1109/MASSP.1986.1165342.

Ramanathan, N., Balzano, L., Burt, M., Estrin, D., Harmon, T., Harvey, C., Jay, J., Kohler, E., Rothenberg, S., Srivastava, M., 2006. Rapid deployment with confidence: calibration and fault detection in environmental sensor networks. Center for Embedded Network Sensing.

Ramesh, M.V., 2014. Design, development, and deployment of a wireless sensor network for detection of landslides. Ad Hoc Netw. Part A 13, 2–18. http://dx.doi.org/10.1016/j.adhoc.2012.09.002, ((1) Special issue: Wireless technologies for humanitarian relief & (2) Special issue: Models and algorithms for wireless mesh networks).

Rosberg, Z., Liu, R.P., Dinh, T.L., Dong, Y.F., Jha, S., 2010. Statistical reliability for energy efficient data transport in wireless sensor networks. Wirel. Netw. 16 (7), 1913–1927. http://dx.doi.org/10.1007/s11276-009-0235-5.

Sahoo, M.N., Khilar, P.M., 2014a. Diagnosis of wireless sensor networks in presence of permanent and intermittent faults. Wirel. Pers. Commun. 78 (2), 1571–1591. http://dx.doi.org/10.1007/s11277-014-1836-6.

Sahoo, M.N., Khilar, P.M., 2014b. Distributed diagnosis of permanent and intermittent faults in wireless sensor networks. In: Advanced Computing, Networking and Informatics—Volume 2: Wireless Networks and Security Proceedings of the Second International Conference on Advanced Computing, Networking and Informatics (ICA CNI-2014). Springer International Publishing, Cham, pp. 133–141. http://dx.doi.org/10.1007/978-3-319-07350-7_15.

Saihi, M., Boussaid, B., Zouinkhi, A., Abdelkrim, M.N., 2013. Decentralized fault detection in wireless sensor network based on function error. In: 2013 10th International Multi-Conference on Systems, Signals & Devices (SSD). IEEE, Hammamet, Tunisia, pp. 1–5

Sanchez, E., Shibata, T., Zadeh, L.A., 1997. Genetic Algorithms and Fuzzy Logic Systems: Soft Computing Perspectives 7. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Sharma, K.P., Sharma, T.P., 2016. rdfd, reactive distributed fault detection in wireless sensor networks. Wirel. Netw., 1–16. http://dx.doi.org/10.1007/s11276-016-1207-1.

Szewczyk, R., Polastre, J., Mainwaring, A., Culler, D., 2004. Lessons from a sensor network expedition. In: Karl, H., Wolisz, A., Willig, (Eds.), Wireless Sensor Networks: First European Workshop, EWSN 2004, Berlin, Germany, January 19-21, 2004. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 307–322. http://dx.doi.org/10.1007/978-3-540-24606-0_21

Titouna, C., Aliouat, M., Gueroui, M., 2015a. Outlier detection approach using Bayes classifiers in wireless sensor networks. Wirel. Pers. Commun. 85 (3), 1009–1023. http://dx.doi.org/10.1007/s11277-015-2822-3.

Titouna, C., Aliouat, M., Gueroui, M., 2015b. Fds: Fault detection scheme for wireless sensor networks. Wirel. Pers. Commun. 86 (2), 549–562. http://dx.doi.org/10.1007/s11277-015-2944-7.

Tolle, G., Polastre, J., Szewczyk, R., Culler, D., Turner, N., Tu, K., Burgess, S., Dawson, T., Buonadonna, P., Gay, D., Hong, W., 2005. A macroscope in the redwoods. In: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems. SenSys '05. ACM, San Diego, California, USA, pp. 51–63. http://dx.doi.org/10.1145/1098918.1098925

Varshney, U., 2004. Vehicular mobile commerce. Computer 37 (12), 116–118. http://

dx.doi.org/10.1109/MC.2004.261.
Wang, N., Chen, Y.X., 2013. A fault-event detection model using trust matrix in WSN. Sens. Transducers J. 158 (11), 190–194.
Warriach, E.U., Tei, K., 2013. Fault detection in wireless sensor networks: a machine learning approach. In: 2013 IEEE 16th International Conference on Computational Science and Engineering (CSE). IEEE, Sydney, NSW, Australia, pp. 758–765. hhttp://dx.doi.org/10.1109/CSE.2013.116
Warriach, E.U., Aiello, M., Tei, K., 2012. A machine learning approach for identifying and classifying faults in wireless sensor network. In: 2012 IEEE 15th International Conference on Computational Science and Engineering (CSE), pp. 618–625. http://dx.doi.org/10.1109/ICCSE.2012.90.
Wu, J.Y., Duh, D.R., Wang, T.Y., Chang, L.Y., 2007. Fast and simple on-line sensor fault detection scheme for wireless sensor networks. In: Embedded and Ubiquitous Computing: International Conference, EUC 2007, Taipei, Taiwan, December 17–20, 2007. Proceedings. Springer, Berlin, Heidelberg, pp. 444–455. http://dx.doi.org/10.1007/978-3-540-77092-3_39.
Xie, Z., Quirino, T., Shyu, M.L., Chen, S.C., Chang, L., 2006. Unpcc: a novel unsupervised classification scheme for network intrusion detection. In: 18th IEEE International Conference on Tools with Artificial Intelligence, 2006. ICTAI'06. IEEE, Arlington, VA, USA, p. 743–750. http://dx.doi.org/10.1109/ICTAI.2006.115
Xu, X., Geng, W., Yang, G., Bessis, N., Norrington, P., 2014. LEDfd: a low energy consumption distributed fault detection algorithm for wireless sensor networks. Int. J. Distrib. Sens. Netw. 2014, 10. http://dx.doi.org/10.1155/2014/714530.
Yang, Y., Su, L., Khan, M., Lemay, M., Abdelzaher, T., Han, J., 2014. Power-based diagnosis of node silence in remote high-end sensing systems. ACM Trans. Sens. Netw. 11 (2), 33:1–33:33. http://dx.doi.org/10.1145/2661639.
Yang, C., Liu, C., Zhang, X., Nepal, S., Chen, J., 2015. A time efficient approach for detecting errors in big sensor data on cloud. IEEE Trans. Parallel Distrib. Syst. 26 (2), 329–339. http://dx.doi.org/10.1109/TPDS.2013.2295810.
Yu, M., Mokhtar, H., Merabti, M., 2007. Fault management in wireless sensor networks. IEEE Wirel. Commun. 14 (6), 13–19. http://dx.doi.org/10.1109/MWC.2007.4407222.
Yuan, H., Zhao, X., Yu, L., 2015. A distributed Bayesian algorithm for data fault detection in wireless sensor networks. In: 2015 International Conference on Information Networking (ICOIN). IEEE, Cambodia, pp. 63–68. http://dx.doi.org/10.1109/ICOIN.2015.7057858
Yuvaraja, M., Sabrigiriraj, M., 2015. Fault detection and recovery scheme for routing and lifetime enhancement in wsn. Wirel. Netw., 1–11. http://dx.doi.org/10.1007/s11276-015-1141-7.
Zhang, H., 2004. The optimality of Naive Bayes. AA 1 (2), 3.