Design of Algorithm for Fault Detection in Wireless Sensor Networks (WSNs)

Deepak Yadav

Department of Information Technology

IIEST, Shibpur

Supervisor

Dr. Tuhina Samanta

In partial fulfillment of the requirements for the degree of

Master of Technology in Information and Communication Engineering

May 3rd, 2019

Acknowledgements

First and foremost, I take this opportunity to express my sincere thankfulness and deep regard to Dr. Tuhina Samanta, for the impeccable guidance, nurturing and constant encouragement that she had provided me during my post-graduate studies. Words seem insufficient to utter my gratitude to her for her supervision in my dissertation work. Working under her was an extremely knowledgeable experience for a young researcher like me. I would like to convey my gratitude to the department of Information Technology and all its staffs for allowing me to perform my research here. Sincere thankfulness to Dr. Hafizur Rahman, the Head of Department, Information Technology, for his continued supervision, encouragement and affection. Ramanujan Central Library must be credited for being a wonderful source of information throughout the past two years. I shall forever remain indebted to my parents, teachers and friends for supporting me at every stage of my life. It is their constant encouragement and support that has helped me throughout my academic career and especially during the research work.

To my family and friends

Abstract

A key issue in the wireless sensor network applications is how to accurately detect the fault status of a node when it is working in a harsh environment. The wrong detection of nodes status can cause a lot of the damage, when it is used for critical applications. Using distributed self-fault diagnosis (DSFD) method, faults in wireless sensor networks (WSNs) can be easily detected. In this method, each sensor node collects its neighbourhood sensor node data and uses the statistical-based method for detecting its own fault status. We propose a distributed fault detection method based on statistical Q_n scale estimator. The proposed method is implemented using python language and found that Q_n scale estimator shows better false alarm rate in comparison to the other statistical methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Wireless Sensor Networks

Wireless Sensor Networks (WSNs) shown in figure 1.1 consists of thousands of tiny sensor nodes having limited battery and processing power[3]. They are placed underwater for monitoring of water quality such as pH level, conductivity and contamination level of water etc. Underground applications consist of monitoring of soil quality such as humidity, nutrient level etc. There installation on the land having many applications like monitoring of forest fire, bridge vibration etc.

The sensor nodes collect the data and transmitted to its base station via. router nodes for further processing. Each wireless sensor nodes consists of a sensing unit, transmission unit, CPU and power unit shown in figure 1.2. Any unit failure causes the fault in WSNs, which can be corrected only by replacing the faulty unit, this type of fault is called the hard fault[4]. The main constraint of WSNs is the energy source, it is operated with the limited energy source, once the energy is drained out, the sensor nodes is no longer any part of the network. The replacement of the energy source is not possible in all types of applications. There are the many causes, which can damage the wireless sensor networks, such as failure of any subsystems causes a hard fault which cant be repaired without replacing of that subsystems, another type of fault is a soft fault which is due to the inaccurate sensor reading, software malfunctioning etc.

1.2 Motivations

There are several types of fault diagnosis techniques are available. They can be broadly classified into three parts: 1. Centralized 2. Distributed and, 3. Hybrid approach[4]. In a centralized approach, all the nodes are sending their data to the base station for the detection of the fault. This method put lots of burden on the network and it is also more energy consuming method. The distributed

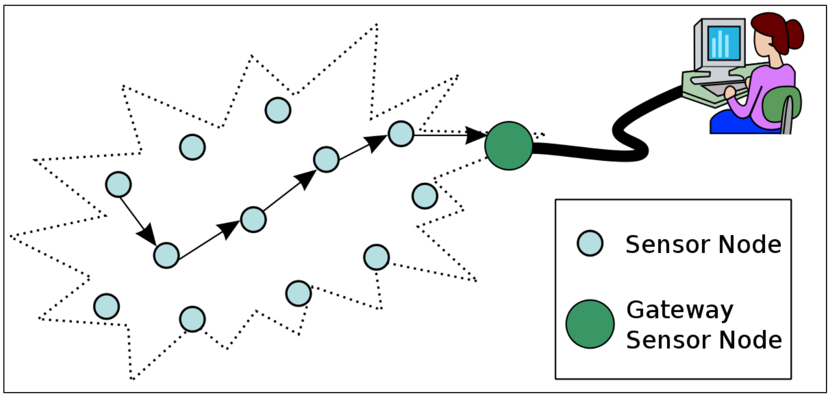


Figure 1.1: Architecture of WSNs [1]

approach divides the computation among all nodes so that energy consumption should be reduced. The hybrid approach is a combination of both methods[4].

The distributed approach further divided into sub-section neighbourhood based, statistic, probability, soft computing, self detection and cloud based. The main reason to choose statistical based fault detection technique is that it is easy to implement on power constraint WSNs.

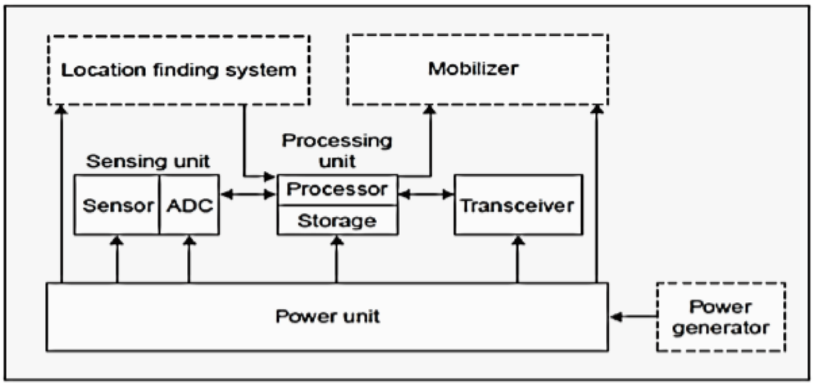


Figure 1.2: Block diagram of Sensor Node [2].

1.3 Objectives and Contributions

Wireless Sensor Networks (WSNs) are used for many critical applications such as health monitor [5], military purpose [6], fire detection and many more applications. The reliability of these applications are totally dependent on correctness of data received by WSNs. To maintain the reliability of these applications faulty data must be detected and eliminated before sending it to base station.

We designed an algorithm which is based on the statistical method which can easily detect the faulty node in the network and eliminate the faulty data. There are many algorithm based on statistical method already exists. I have done a comparison study between these statistical method based on different parameters [3], such as detection accuracy (DA), false alarm rate (FAR), false positive rate (FPR), total energy consumption (TEC) and running time (RT).

1.4 Overbased view of the Thesis

In this thesis we introduce a new fault detection technique which is based on statistical method. This chapter introduces WSNs and presents its limitation in full-fledged application of WSNs which leads to the motivation of our work. The remaining five chapters constitute the contributory part of the entire thesis, which are outlined below as follows:

- **Chapter 2:** This chapter highlights few previous works done on fault detection in the WSNs using statistical method.
- **Chapter 3:** In this chapter we represent the network model, energy model and fault model that have been regarded throughout the thesis.
- **Chapter 4:** In this chapter we proposed the distributed self-fault diagnosis (DSFD) method based on statistical method and also proposed several

Lemma which verify the correctness of the algorithm.

- **Chapter 5:** This chapter discuss the various simulation works performed on the algorithm. Simulation work is divided into two parts, first part is about software simulation which includes different test such as detection accuracy, false alarm rate, false positive rate, total energy consumption and running time. Second part is based on hardware simulation which is done using Arduino, XBee RF modules and sensors.

- **Chapter 6:** In the final chapter, we make some concluding remarks on the proposed methodologies. We touch upon few limitations of these proposed approaches.

Chapter 2

Related Works

The fault diagnosis algorithm is classified into three categories: Centralized, Distributed and Self-diagnosis approach[3]. In the centralized approach all the sensor nodes send their data to the central node for the detection of their fault status. The central node (base station), computes the fault status of all the nodes and broadcast the fault status to all nodes in the network. There are several disadvantages of this type of approach, the central node should be ultra-reliable, high computation capability and large storage are required. The broadcast of fault status required multihop communication, which depletes the energy of the network quickly. The failure of the base station causes unable to detect the fault in the nodes. There are the advantage of this type of approach is that the detection latency is very high.

These disadvantages are overcome in the distributed fault diagnosis approach [4], where each sensor nodes participating in the diagnosis process but the final fault status is decided by the base station. In this approach, each node acts as a tester as well as a testing node. Each node also tests the fault status of neighbours node and collect all the test result known as a syndrome and send it to the base station. The base station uses the syndrome analysis approach to detect the final fault status and sends the status to all nodes. This type of approach is suitable for unconstrained based networks.

The centralized and distributed approach is costly in terms of message complexity, leads to low network life due to more power consumption. The self-diagnosis approach [3] solves the problem, in this approach, each node checks their own fault status by collecting neighbours sensor nodes data. If the fault detected then that sensor node is not participating in the network. There are several advantages of this approach. This type of approach does not put communication, memory, bandwidth and energy overload on the network. There is no need of initiator node for diagnosis of entire network and no need to know the fault status of neighbour nodes (which is the essential condition in the distributed approach), which saves the time.

Lee and Choi [7] proposed neighbour co-ordination method in which sensor data compare with neighbour data at any time t and store it to the memory. This method is repeated for C times, in the final step each sensor analyzing data stored in the memory. The disadvantage of this method is that it collects the data C times (more energy loss during transmission in comparison to processing).

Liang et al.[8] proposed the statistical-based approaches which uses a weighted median based fault detection technique. In this technique, normalized data calculated which is equal to the ratio of the difference of sensed data and estimated data to the sensed data. This normalized value is compared with the threshold value, if it is greater than the threshold then it treated as the faulty node.

In three sigma edit test [3], each node collects data from the neighbour and send its data to the neighbour also and then calculate the fault status of its own node as well as neighbours node by using three sigma edit test rule.

Modified three sigma edit test rule proposed by [3], in this method, no need to send the fault status to the neighbour node. Therefore, the number of message exchange reduced. They replace the mean with median and standard deviation with normalized MAD.

Peter J. Rousseeuw and Christophe Croux [9] proposed the alternative of the median absolute deviation. They found that, there is still room for improvement in the median absolute deviation. They proposed S_n and Q_n scale estimator which can replace the median absolute deviation in term of efficiency.

Our proposed algorithm is based on the most robust scale estimator (Q_n scale estimator)[9]. There are several other scale estimators such as median absolute deviation (MAD), interquartile range (IQR), S_n scale estimator, Hodges Lehmann estimator, etc. are present but among them, Q_n scale estimator performance is more satisfactory. The Q_n scale estimator having Gaussian efficiency 82% [9], a more efficient estimator needs fewer observations than the less efficient one to achieve a given performance. Financial companies now using these estimators on daily basis in analysis of the behaviour of stocks. The detailed analysis of Q_n scale estimator is given in the subsequent section[9].

Chapter 3

System Model and Assumptions

The system model is consists of a network model, fault model and energy model. In the network model, the placement of sensor nodes and their way of communication are described, the fault model described how the sensor nodes get faulty during their operation and In the energy model, we discussed about the loss of energy during their transmission and reception.

3.1 Assumptions, Notations and Symbols

- The energy level of each node during installation are equal and nodes are homogeneous in nature.
- Each node can send packets consists of node ID and Data with its neighbours and can receive from its neighbours.
- If any nodes receive packets in which data part is missing from the packets then it considered that the sender node having faulty.
- All the nodes in the network is static and synchronized by the network protocol.
- Each node is powered with a battery source and energy loss during each packet transmission is the same for each node in the network.
- The communication link between node is assumed to be fault free.
- Each node can sense the data in a periodically manner and two nodes can communicate with each other using UDP/IP protocol.

Notation	Meaning
S	Set of all sensor nodes
s_i	i_{th} sensor node
k	k^{th} sample of data
N_{eg_i}	Set of neighbouring nodes of s_i
x_i(t)	Sensed data of s_i at time t
Nx_i	Neighbors node sensed data
NT_i	Neighboring table stored at s_i
FS_i	Fault status of s_i
T_r	Transmission range of a node
T_s	Sampling interval
k	Number of samples in a frame
med_i	median over neighboring nodes data Nx_i

Table 3.1: List of important notations and their meanings.

3.2 Network Model

All the sensor nodes are placed in a random manner in $L \times L$ area with maximum communication range is d meter in all the direction. It is assumed that the network model follows the unit disk model shown in figure 3.1. In this model a node can communicate within d meter radius. $S = s_i$, where $i \in [1, N]$ is the

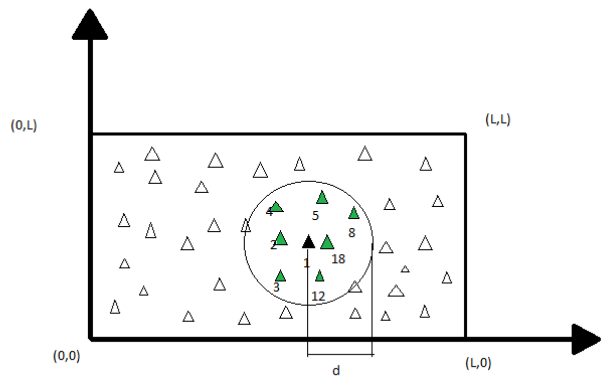


Figure 3.1: Unit disk model

set of all sensor nodes present in the network, where N is the number of sensor nodes present in the network shown in figure 3.2 and N_{eg_i} is the i_{th} sensor node neighbours. $x_i(t)_{t\in[1,k]}$ is the reading of s_i node at time t sec. $x_i(t)$ can be any quantity based on the application such as temperature, light, noise, humidity etc. IEEE 802.15.4 is used as a MAC layer protocol for communicating to neighbours node[3].

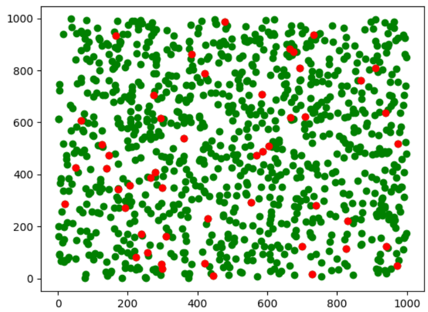


Figure 3.2: Network model.

3.3 Energy Model

Power consumption of a sensor node is a function of i) power to transmit 1 bit, ii) power to receive or decode 1 bit, and iii) power while being idle or sensing. Power consumption in idle mode or while sensing is very small, hence is ignored. The first order radio model is used for calculating power consumption. For receiving k_1 bits/sec, the power consumption (p_r) at a sensor node is

$$p_r = k_1\beta \tag{3.1}$$

Here β is a factor indicating the energy consumption per bit at the receiver circuit. The power p_t needed for transmitting k_2 bits/sec is

$$p_t = k_2(\alpha_1 + \alpha_2 d^\theta) \tag{3.2}$$

where α_1 is the energy consumption factor indicating the power consumed per bit by the transmitter circuit and $\alpha_2 d^\theta$ indicates the energy consumption on the amplifier (per bit), d being the physical distance between the transmitting and the receiving node and θ the path loss exponent (usually between 2 and 4, depending on the environment). Therefore the total energy consumption at a node per time unit is:

$$p = p_r + p_t \tag{3.3}$$

3.4 Fault Model

Each sensor nodes collects k samples of data periodically from the environment and store it on the local memory. The data can be temperature, pressure, humidity, light etc. During fault detection phase each node collects k_{th} sample of the neighbours sensed value (Nx_i), node id and store it into NT_i table shown in table 4.1. These sensed values (Nx_i) are used to calculate Q_n scale estimator.

$$FS_i = \begin{cases} faulty & \text{if } \forall |x_i(t)_{t\in[1,k]} - med_i| > 3 \times Q_n \\ fault-free & \text{if } 0 \leq \forall |x_i(t)_{t\in[1,k]} - med_i| \leq 3 \times Q_n \\ transientfault & \text{otherwise} \end{cases}$$

Where FS_i is the fault status of i_{th} node and $x_i(t)_{t\in[1,k]}$ is the k samples value of the s_i node.

Chapter 4

Proposed Fault Detection Method

4.1 Distributed Self-Fault Diagnosis Method

The distributed self-fault diagnosis method (DSFD) consists of two phases 1) initialization phase and 2) self-diagnosis phase. In the initialization phase, each node collects k number of samples from the environment within kT_s time interval. Where T_s is the sampling interval. These k samples of the sensor node are stored in the local memory of the sensor node. After storing the samples each node transmit a request message to it's neighbouring node for their sensed value (k_{th} sample). When node received data from their neighbouring node, it construct a neighbouring table (NT_i). Which is consists of node id and k_{th} sensed value of neighbouring nodes.

In the self-diagnosis phase, Q_n scale estimator is calculated with the help of sensed value (Nx_i) of neighbouring node. After the calculation of Q_n scale estimator each sample value is tested weather it lies between 0 and $3 \times Q_n$ or not. If each sample fall in this range then it is fault-free. If all the value lies beyond that range then it is treated as faulty node. If some value lies between range and some values lies beyond that range then it is treated as transient fault and ignored by the sensor node. The detailed description of the algorithm given in algorithm 1 and algorithm 2.

Node Id	Sensed value(Nx_i)
40	26.4
81	25.8
.	.
.	.
20	27.1

Table 4.1: Neighboring table NT_i.

Notation	Meaning
sort()	Function for sorting list
n	Number of neighbourhood nodes
$Q_n = \{\}$	One dimensional array
m	order statistics
Counter_one	counting the number of faulty status
Counter_zero	counting the number of fault-free status

Table 4.2: List of notations used in the algorithm 1 and 2.

Algorithm 1 DSFD algorithm: Initialization Phase

INPUT: Sample values of s_i node $x_i(1), x_i(2), x_i(3), ..., x_i(k)$.
OUTPUT: Neighbourhood table of s_i nodes.
Initialization Phase:
$S = \{s_1, s_2, \ldots, s_i, \ldots, s_N\}$, where S is the set of all nodes.
N_{eg_i} is i_{th} node neighbour
Step:1
Each sensor node $s_{i\in[1,N]}$ collects k samples of sensed value from the environment and store it into the local memory.
for $s_i \in S$ **do**
| $X_i \longleftarrow \{x_i(1), x_i(2), \ldots, x_i(k)\}$
end
Step:2
Store the X_i into the local memory of the sensor node s_i.
Step:3
Sensor node s_i store node id and k_{th} sensed value from the neighbours nodes N_{eg_i} into neighbouring table (NT_i). At the same time node s_i also send k_{th} sensed data and node id (nd_i) to its neighbourhood.
$NT_i = \{nd_i, Nx_i\}$
$Nx_i = \{x_j(k)\}_{j\in N_{eg_i}}$

Algorithm 2 DSFD algorithm: Self-diagnosis Phase

```
INPUT: Neighbours node data Nx_i
OUTPUT: Fault status (FS_i) of s_i nodes.
sort(Nx_i)
/*Step 1: calculation of median(med_i) of Nx_i*/
n = N_egi.length
if n%2 == 0 then
 | med_i={Nx_i[n/2]+ Nx_i[(n/2)+1]}/2
else
 | med_i=Nx_i[(n + 1)/2]
end
/*Step 2: Calculation of difference pairs (C_n * |x_i − x_j; i < j|)*/
index = 0
Q_n = {}
for i ← 1 to n do
 | for j ← i + 1 to n do
 | | Q_n[index] = C_n * |Nx_i[i] − Nx_i[j]|
 | | index + +
 | end
end
Sort(Q_n[index])
/*Step 3: Calculation of Q_n scale estimator*/
h = [n/2] + 1
m = h * (h − 1)/2
σ_Qn = Q_n[m]
/*Step 4: Calculation of the fault-status of s_i node*/
counter_zero = 0
counter_one = 0
for t ← 1 to k do
 | z_i = (x_i(t) − med_i)/σ_Qn
 | if 0 < z_i < 3 then
 | | counter_zero + +
 | else
 | | counter_one + +
 | end
end
if counter_zero == k then
 | s_i is fault-free.
else
 | if counter_one == k then
 | | s_i is faulty
 | else
 | | s_i having transient fault
 | end
end
```

4.1.1 Explanation:

Assume that the node s_i diagnosis itself using distributed self-fault diagnosis (DSFD) method. First, it collect k samples $(x_i(t)_{t\in[1,k]})$ from the environment. After the collection of k samples, it collects k_{th} sensed data (present value) and node id of it's neighbouring nodes and construct the neighbouring table NT_i, at the same time it also sends data to it's neighbourhood nodes N_{eg_i}. After construction of table NT_i, it detects the fault using statistical-based method, Step 1: calculate the median (med_i) of data set Nx_i, where Nx_i is the magnitude of sensed value from the neighbouring nodes. Step 2:, difference pair is calculated using $C_n * |N_{x_i}[i] - N_{x_i}[j]|; i < j$, where C_n is the constant factor and it's value is 2.219. $Q_n[index]$ is the difference between i^{th} and j^{th} element present in the Nx_i. where $i, j = |N_{eg_i}|.length$. Step 3: select the m^{th} element from the sorted $Q_n[index]$ array. Step: 4 there is two counter are created *counter_zero* and *counter_one*, *counter_zero* is incremented when z_i is lies between 0 and $3 \times \sigma_{Q_n}$, otherwise *counter_one* is incremented. To check whether s_i node is faulty or not, compare the counter value to the number of samples i.e k. if *counter_zero* is equal to the k then the node s_i is treated as the faultfree node, if *counter_one* is equal to the k then the node s_i is the faulty node. if both of the condition is not satisfied then it is assume that transient fault detected at the node s_i.

4.1.2 Analysis of statistical method

The sensor nodes collects k samples during kT_s time. These k samples are treated as the single frame of the data. Samples are given as: $x_i(1), x_i(2), x_i(3), \ldots, x_i(k)$. The k_{th} samples of all the neighbours $x_j(k)_{j\in N_{eg_i}}$ are collected and used for the fault detection. Before apply statistical method, it is assumed that these data set $x_j(k)_{j\in N_{eg_i}}$ are follows the normal distribution $N(A, \sigma_j^2)$. where A is actual value of sensor reading and σ_j is the standard deviation of the data set. Any sensor reading can be written as: $x_j(k) = A + \varepsilon_j(k)$, where A represent the actual data and $\varepsilon_j(k)$ is the fault in the data[3]. The fault status of the node is calculated using Q_n estimator. A good estimator having a high breakdown point and high Gaussian efficiency[10]. The breakdown point of Q_n estimator is 50% and Gaussian efficiency is 82%[10]. Q_n estimator is defined as: $\sigma_{Q_n} = C * (|x_i - x_j|; i < j)_k$, where C is the constant factor and k is the k_{th} order statistic and defined as $k = h * (h-1)/2$, where $h = [n/2] + 1$ and n is total number of difference pair, $|x_i - x_j|$. The main purpose to choose the Q_n estimator is that it attains 82% Gaussian efficiency in comparison to the other methods[10].

To distinguish between faulty and non-faulty node, z- score is calculated which is given as:

$$z = \frac{|x_i - median(x_i)|}{\sigma_{Q_n}(x)} \tag{4.1}$$

if the value of z is lies between 0 and 3 then it is treated as the fault-free node, otherwise it is treated as the faulty node. the 99.7% of the data lies within 3σ shown in figure 4.1.

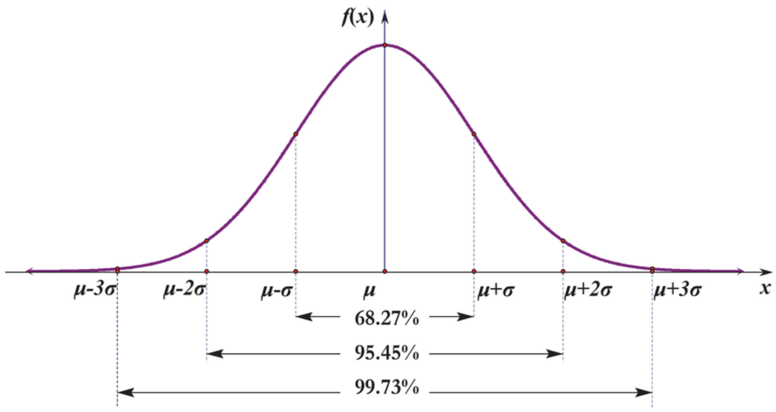


Figure 4.1: 3 sigma rule

4.2 Some Important Definitions

4.2.1 Definition 1.

Distance between any two sensor nodes s_i and s_j is $\frac{k}{\sqrt{P_r}}$, where P_r is the received power and k is a constant (depends upon the parameter setup by the transceiver system). In Friss free space propagation loss model, the received power P_r is computed as [3]:

$$P_r = P_t \times G_t \times Gr \times \frac{\lambda^2}{(4 \times \pi \times D)^2} \tag{4.2}$$

where P_r is the power received by the receiving antenna, P_t is the power transmitted by the transmitting antenna, G_t and G_r are the gain of transmitting and receiving anteena respectively, λ is the wavelength of the signal and D is the distance between transmitting and receiving anteena. On solving equation 4.2 we get:

$$D = \frac{k}{\sqrt{P_r}} \tag{4.3}$$

where k is a constant which is given as:

$$k = \sqrt{P_t \times G_t \times G_r \times \frac{\lambda^2}{(4 \times \pi)^2}} \tag{4.4}$$

4.2.2 Definition 2.

The estimated transmission time (ETT_i) for sensor node s_i is $\frac{dist(si,sj)}{c} + \tau_i$, where τ_i is the processing time. Estimated transmission time is the approximate time required by a sensor node s_i to transmit it's data to all its surrounding neighboring nodes Neg_i which are coming under it's transmission range T_r . The estimated transmission time ETT_i can be defined as [3]:

$$ETT_i = maxETT_{i,j} + \tau_i, \forall s_j \in Neg_i \tag{4.5}$$

where $ETT_{i;j}$ i.e. estimated transmission time between the sensors s_i and s_j and τ_i is the processing delay of s_i . The $ETT_{i;j}$ can be calculated as given in (4.6)

$$ETT_i = \frac{dist(si, sj)}{c} \tag{4.6}$$

where c is the speed of light.

4.2.3 Definition 3.

The detection accuracy (DA) of the proposed DSFD algorithm is defined as [3]:

$$DA = \frac{\text{number of faulty nodes detected as faulty}}{\text{the total number of faulty nodes present in the network}} \tag{4.7}$$

4.2.4 Definition 4.

The false alarm rate (FAR) of DSFD algorithm is defined as [3]:

$$FAR = \frac{\text{number of fault free nodes detected as faulty}}{\text{total number of fault free nodes present in the network}} \tag{4.8}$$

4.2.5 Definition 5.

The false positive rate (FPR) for the DSFD algorithm is defined as [3]:

$$FPR = \frac{\text{number of faulty nodes detected as fault free}}{\text{total number of faulty nodes present in the network}} \tag{4.9}$$

Chapter 5

Simulation Results and discussions

The performance of the proposed DSFD algorithm is evaluated and compared with other statistical methods in term of detection accuracy (DA), false alarm rate (FAR), false positive rate (FPR), total energy consumption (TEC) and running time (RT). All the algorithm are simulated using python language on 4 GB RAM and $i3 - 2350MCPU@2.30GHz \times 4$ core processor system. The sensor nodes are randomly placed over 1000×1000 m^2 area having transmission range chosen among 56 m and 74 m. Since the neighbourhood node degree is directly proportional to the transmission range (T_r) hence varying transmission range degree of neighbours can be increased for any sensor node.

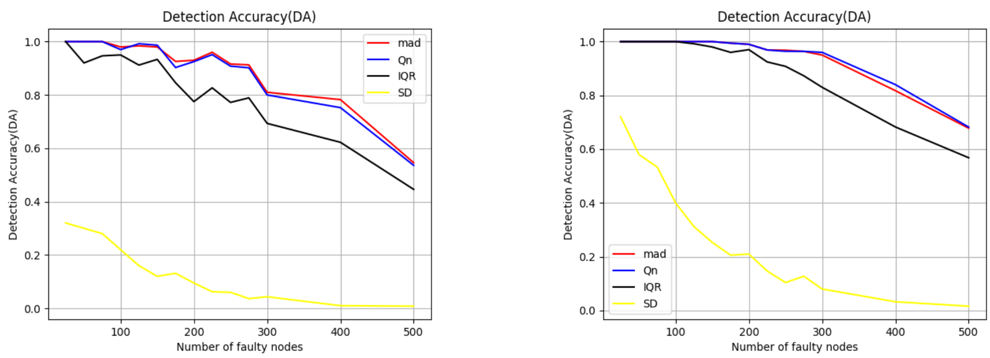
5.1 Software Simulation

In this section efficiency of the algorithm is evaluated based on detection accuracy, false alarm rate, false positive rate, total energy consumption and running time. The parameters set for simulation is given below:

1. Total number of the nodes (N) participating in the network is 1000.
2. Sensor nodes are randomly placed in $1000m \times 1000m$ area.
3. Transmission range (T_r) of the sensor nodes are chosen among $56m$ and $74m$.
4. Fault-free nodes values are randomly set between 24.4 to 26.3.
5. Faulty nodes injected in the network is varies between 25 to 500.
6. Parameters in energy consumption model are considered as: $\alpha_1 = 45nJ/bit$, $\alpha_2 = 0.001pJ/bit/m^4$, $\beta = 45nJ/bit$ and $\theta = 4$

5.1.1 Detection Accuracy (DA)

Detection accuracy (DA) is the ratio of number of faulty nodes detected as faulty to the total number of faulty nodes present in the network. From this definition, it is clear that more detection accuracy means more robustness of the algorithm. Detection accuracy unity means, all the faulty node detected correctly. This simulation is performed by randomly injecting faulty nodes into the network, during this simulation transmission range (T_r) are kept constant. The injected faulty nodes are increases from 25 to 500 at the interval of 25 faulty nodes per simulation. This simulation is again performed by varying the transmission range (T_r). The simulation result are shown in figure 5.1a and 5.1b.



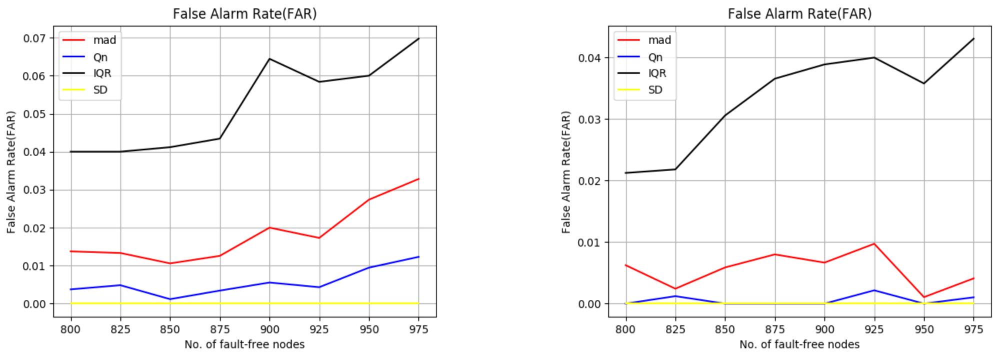
(a) At T_r equal to $56m$ (b) At T_r equal to $74m$

Figure 5.1: Detection accuracy.

From the above graph it is clear that detection accuracy of median absolute deviation (MAD) and Q_n scale estimator is more than the interquartile range (IQR) and standard deviation (SD). Detection accuracy drops sharply as the number of faulty nodes increases.

5.1.2 False Alarm Rate(FAR)

False alarm rate is defined as the ratio of the number of fault-free nodes detected as faulty to the total number of fault-free nodes present in the network. The false alarm rate (FAR) should be less. The false alarm rate of different statistical methods are shown in the figure 5.2a and 5.2b.



(a) At T_r equal to $56m$ (b) At T_r equal to $74m$

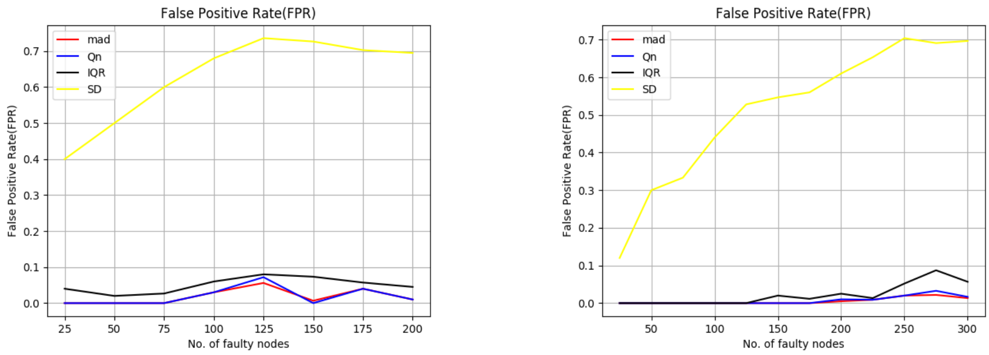
Figure 5.2: False alarm rate.

From the above graph, it is clear that the false alarm rate of the standard deviation (SD) is least followed by Q_n scale estimator, median absolute deviation (MAD) then interquartile range (IQR).

5.1.3 False Positive Rate(FPR)

Fault Positive Rate is defined as the ratio of the number of faulty nodes detected as faultfree to the total number of faulty nodes present in the network. From the above definition, it is clear that the false positive rate value should be less.

It is clear that the false positive rate value of the Q_n scale estimator and median absolute deviation (MAD) are almost the same and having the least value. The interquartile range (IQR) shown some medium value of FPR while standard deviation (SD) shown very poor value of FPR.



(a) At T_r equal to $56m$ (b) At T_r equal to $74m$

Figure 5.3: False positive rate.

5.1.4 Total Energy Consumption (TEC)

Total energy consumption graph is plotted between number of neighbourhood node and total energy used during transmission and reception of the signal. As much as the neighbourhood nodes increases, total energy consumption during transmission also increases. Energy consumption is already discussed in chapter 3.

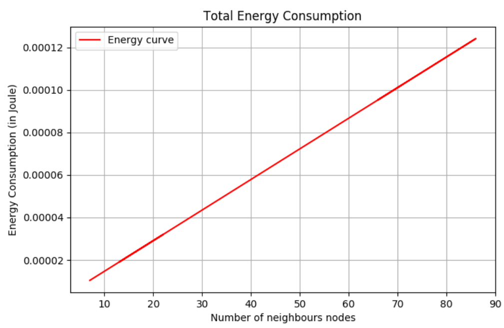


Figure 5.4: Total Energy Consumption.

5.1.5 Running Time

In this section running time of different algorithm is plotted. The running time is plotted between time taken by algorithm and number of neighbourhood nodes. From the above graph it is clear that the time taken by Q_n scale estimator

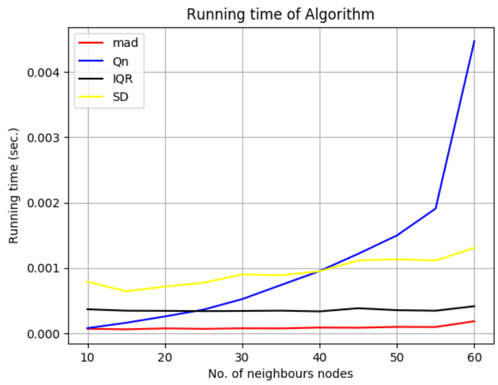


Figure 5.5: Running Time.

increases when the neighbourhood node degree increases. while median absolute deviation (MAD), interquartile range (IQR) and standard deviation (SD) takes less time in comparison to the Q_n scale estimator.

5.2 Hardware Simulation

5.2.1 Hardware Component

5.2.1.1 Arduino Microcontroller Board

Arduino is an electronic device which is open source and based on microcontroller. Arduino microcontroller board can be programmed using Arduino software (IDE). We have used Arduino Uno board (Figure 5.6a) which is designed with 8-bit ATmega328P microcontroller. It consists of 6 Analog input pins and 14 Digital I/O pins out of which 6 provide PWM output [11]. ATmega328P has EEPROM memory of 1 KB. Either an external power supply or USB connection can be used to power up.

5.2.1.2 XBee RF Modules

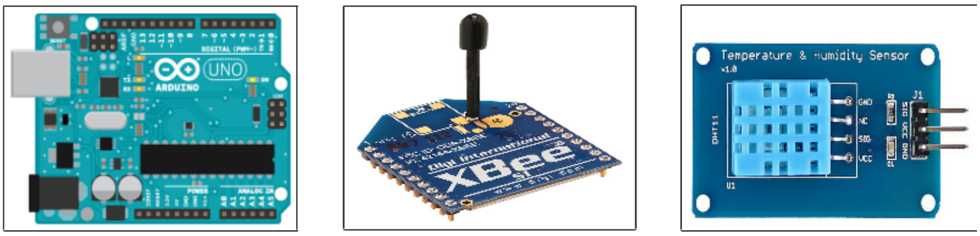
XBee is radio communication module fabricated by Digi. A microchip manufactured by Freescale is used by these radio modules to produce basic, standards-based point-to-point communications. XBee supports different variations of communication protocols including ZigBee. We have used XBee Series 1 module (Figure 5.6b) [12]. XBee modules are set up in API mode to transmit or receive data in form of frames.

We have used the XCTU software to configure the XBee modules. The XBee module can be configured into three types of devices: coordinator, router, and end device. Coordinator is responsible for forming the network, handing out addresses, and managing the other functions that define the network, secure it, and keep it healthy. Router can join existing networks, send information, receive information, and route information. End device can join networks and send and receive information. They dont act as messengers between any other devices [13].

The XBee module has support for transparent (AT) and application programming interface (API) modes [14]. With the AT mode, the XBee module behaves as a serial line replacement. With the API mode, all data entering and leaving the module is contained in frames. The API mode is required when the network needs to be formed into a multi-node mesh or tree topology. In this work we configure the XBee modules to operate in API mode.

5.2.1.3 Temperature and Humidity Sensor

We have employed DHT11 (Figure 5.6c) temperature and humidity sensor. The DHT11 features a temperature and humidity sensor complex with a calibrated



(a) Arduino Uno (b) Xbee Antenna (c) DHT11 Sensor

Figure 5.6: Hardware simulation.

digital signal output.

5.2.2 System Setup

The system consists of two modules: static nodes and Base Station. We have designed five static nodes each consists of an Arduino Uno board, an XBee RF module and a DHT11 temperature and humidity sensor. These static nodes act as end device of the wireless sensor network. Their responsibility is only to sense the temperature and humidity from their surroundings and store the sensed data in the EEPROM memory of the Arduino Uno board. These nodes are powered up via external power source and placed in the different locations.

The base station is a desktop computer. An Arduino Uno board along with an XBee RF module, which acts as a end device of the wireless sensor network, is connected to the base station shown in figure 5.7 .

5.2.3 Working Principle

We are considering five static nodes in the Wireless Sensor Network and they are placed close to each other. Static nodes sense the temperature and humidity from the surroundings and store the values in the EEPROM memory of their Arduino module. After storing the K samples value, each static nodes diagnosis itself before sending the data to the base station. Assume that a static node

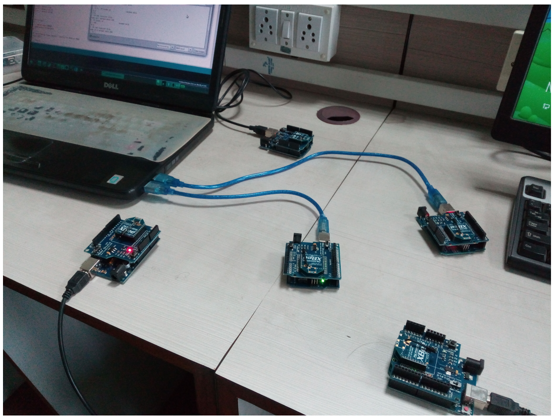


Figure 5.7: System setup.

tried to diagnosis itself, first it send the request message to its neighbours node, Once neighbours nodes receive request message, they create API frames from stored data in their EEPROM memory. The data frames sent from neighbours nodes are given in Figure 5.8. The data packet within the frame is of 11 bytes containing ID of the neighbour node, temperature indicator (01), temperature vale, humidity indicator (02) and humidity vale. On receiving data frames from neighbours nodes, node extract the data packet from the frame and store the information in the EEPROM memory of their Arduino modules.

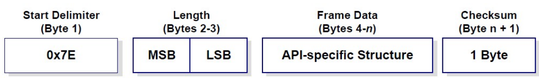


Figure 5.8: API frame structure.

After storing the data from neighbours, node is tried to diagnosis itself using statistical method. Using neighbours data node calculate the Q_n scale estimator and then compare the samples value to it. If all the sample of node lies in the range of estimator then node is fault-free. If all the values of sample outside the range then it is treated as faulty node. If some value between the range and some value are outside the range then it is treated as transient fault, which can be ignored by the node. This process is performed by all the nodes present in the network.

5.2.4 Experimental Result and Discussion

In this experiment, node 1 find out fault status by collecting neighbourhood node data shown in table 5.2 and at the same time, it also exchange sensed data to the neighbours. When node collects all the data from the neighbours shown in table 5.3, it applies the statistical method for fault detection.

5.2.4.1 Experiment: 1

Five static nodes namely node 1, node 2, node 3, node 4 and node 5 are placed randomly and each node sense the data from the environment periodically shown in table 5.2. The sensed data are stored in the EEPROM memory of the node shown in table 5.1. After collecting the sample into the EEPROM memory, node

Node id	sample 1	sample 2	sample 3	sample 4	sample 5
1	24.4	24.8	24.6	24.4	24.5

Table 5.1: Node 1 data.

1 collects the neighbour's node data shown in table 5.3.

Node id	sample 1	sample 2	sample 3	sample 4	sample 5
2	26.3	25.2	24.4	26.0	25.0
3	24.4	24.6	26.0	25.3	25.4
4	24.5	26.3	26.0	25.2	25.1
5	26.0	26.2	25.1	25.4	25.8

Table 5.2: Neighbours node data.

After that node 1 calculate the Q_n scale estimator and construct the fault status table shown in table 5.4. The zero value of fault status table shows that

Neighbours node id	Sensed data(Nx_i)
2	25.0
3	25.4
4	25.1
5	25.8

Table 5.3: Neighbouring table (NT_i).

all the sample of node 1 lies between 0 and $3 \times Q_n$. Since every sample gives positive status hence node 1 treated as fault-free.

	sample 1	sample 2	sample 3	sample 4	sample 5
FS	0	0	0	0	0

Table 5.4: Fault status of node 1.

5.2.4.2 Experiment: 2

This experiment is performed for the detection of transient fault in the sensor nodes. All the network setup are same as the previous experiment. Only the node 1 samples value are the different and it is assume that neighbourhood nodes are fault-free. node 1 samples value are shown in the table 5.5.

Node id	sample 1	sample 2	sample 3	sample 4	sample 5
1	100.3	24.8	84.6	24.4	74.5

Table 5.5: Node 1 data during transient fault.

In the initialization phase, node 1 collects k_th sample i.e 5_th sample of the neighbourhood nodes shown in table 5.2 and store into the neighbourhood table 5.3.

	sample 1	sample 2	sample 3	sample 4	sample 5
FS	1	0	1	0	1

Table 5.6: Fault status of node 1.

node 1 calculates Q_n scale estimator using table 5.3 and compute the fault status shown in the table 5.6. Since the fault status consists of zeros (0) and ones (1) hence the node 1 detected the transient fault.

5.2.4.3 Experiment: 3

The third experiment is based on the detection of faulty nodes in the network. All the setup are same as the previous experiment. In this experiment node 1 is set to be faulty, all the processes are same as the previous experiment. node 1 data during faulty state are shown in table 5.7.

Node id	sample 1	sample 2	sample 3	sample 4	sample 5
1	200.4	400.8	240.6	624.4	424.5

Table 5.7: Node 1 data during faulty state.

The status of all the samples are one(1), hence node 1 treated as faulty node shown in table 5.8.

	sample 1	sample 2	sample 3	sample 4	sample 5
FS	1	1	1	1	1

Table 5.8: Faulty node status.

Chapter 6

Conclusions

In this thesis, fault detection algorithm based on statistical method is proposed. The reason to choose statistical method is that it is easy to implement on the resource constraint wireless sensor network. There are different statistical method are available among which Q_n scale estimator are chosen for the fault detection because it having high detection accuracy (DA), least false positive rate (FPR) and minimum false alarm rate (FAR). This method is verified by the simulation using python language and found that its performance is better than the other statistical method. Other statistical methods used in this thesis are standard deviation (SD), interquartile range (IQR) and median absolute deviation (MAD). standard deviation (SD) not a robust method, because it is based on a mean hence a single large fault can influence the fault status of the other nodes. The interquartile range (IQR) is better than standard deviation because it is based on the median of data set which is not influenced by extremely high or extremely low value but it is not much efficient than median absolute deviation (MAD) and Q_n scale estimator. The Q_n estimator having higher Gaussian efficiency (82%), hence it is chosen for fault detection.

There are some disadvantages of this method, it is based on the normal distribution of the data. If data are not follows the normal distribution then this method fails to work.

References

[1] "https://en.wikipedia.org/wiki/file:wsn.svg," vi, 3

[2] "https://www.google.co.in," vi, 3

[3] M. Panda and P. M. Khilar, "Distributed self fault diagnosis algorithm for large scale wireless sensor networks using modified three sigma edit test," *Ad Hoc Networks*, vol. 25, pp. 170–184, 2015. 2, 4, 7, 8, 13, 20, 21, 22

[4] T. Muhammed and R. A. Shaikh, "An analysis of fault detection strategies in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 78, pp. 267–287, 2017. 2, 3, 7

[5] S. N. Madge, V. C. Prabhakaran, D. Shome, U. Kim, S. Honavar, and D. Selva, "Orbital tuberculosis: a review of the literature," *Orbit*, vol. 27, no. 4, pp. 267–277, 2008. 4

[6] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002. 4

[7] M.-H. Lee and Y.-H. Choi, "Fault detection of wireless sensor networks," *Computer Communications*, vol. 31, no. 14, pp. 3469–3475, 2008. 8

[8] J.-L. Gao, Y.-J. Xu, and X.-W. Li, "Weighted-median based distributed

fault detection for wireless sensor networks.," *Ruan Jian Xue Bao(Journal of Software)*, vol. 18, no. 5, pp. 1208–1217, 2007. 8

[9] P. J. Rousseeuw and C. Croux, "Alternatives to the median absolute deviation," *Journal of the American Statistical association*, vol. 88, no. 424, pp. 1273–1283, 1993. 8, 9

[10] M. Daszykowski, K. Kaczmarek, Y. Vander Heyden, and B. Walczak, "Robust statistics in data analysisa review: Basic concepts," *Chemometrics and intelligent laboratory systems*, vol. 85, no. 2, pp. 203–219, 2007. 20

[11] "http://datasheet.octopart.com/a000066-arduino-datasheet-38879526.pdf.," 28

[12] "https://www.digi.com/pdf/ds xbeemultipointmodules.pdf.," 29

[13] R. Faludi, "Building wireless sensor networks.," 29

[14] S. Ferdoush and X. Li, "Wireless sensor network system design using raspberry pi and arduino for environmental monitoring applications," *Procedia Computer Science*, vol. 34, no. 9, pp. 103–110, 2014. 29

[15] N. Peng, W. Zhang, H. Ling, Y. Zhang, and L. Zheng, "Fault-tolerant anomaly detection method in wireless sensor networks," *Information*, vol. 9, no. 9, p. 236, 2018.