Fault Discrimination in Wireless Sensor Networks

Nishikanta Sahu
Anurag Das

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

Fault Discrimination in Wireless Sensor Networks

Thesis submitted in

May 2013

to the department of

Computer Science and Engineering

of

National Institute of Technology Rourkela

in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Nishikanta Sahu

[Roll: 109CS0091]

Anurag Das

[Roll: 109CS0136]

with the supervision of

Prof. Manmath Narayan Sahoo

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

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

May 11, 2013

Certificate

This is to certify that the work in the thesis entitled ***Fault Discrimination in Wireless Sensor Networks*** by ***Nishikanta Sahu and Anurag Das*** is a record of an original work carried out under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Prof.Manmath Narayan Sahoo
Assistant Professor
Department of Computer Science and Engineering
NIT Rourkela

Acknowledgment

Dreams never turn to reality unless a lot of effort and hard work is put into it and no effort bears fruit in the absence of support and guidance. It takes a lot of effort to work our way through this goal and having someone to guide us and help us is always a blessing. We would like to take this opportunity to thank a few who were closely involved in completion and execution of this Project.

At the outset, we thank the Almighty God for making our endeavor a success. We would like to express our sincere thanks to the Administration of National Institute of Technology, Rourkela for providing excellent infrastructure and other facilities, which enabled us to sharpen our skills.

We express our sincere gratitude to our supervisor Prof. Manmath Narayan Sahoo for his constant support and valuable suggestions without which the successful completion of this project would not have been possible. We are highly indebted to him for making us learn both research and writing skills.

Further we are also thankful to all the faculty members and staff of Department of Computer Science and Engineering, National Institute of Technology, Rourkela for their constant help and support during the entire project work.

Last but not the least we would like to thank our family members for their constant motivation and our classmates who in one way or another have helped us in the successful completion of this work.

Nishikanta Sahu
Anurag Das

Abstract

In current times, one of the promising and interesting areas of research is Wireless Sensor Networks. A Wireless Sensor Network consists of spatially distributed sensors to monitor environmental and physical conditions such as temperature, sound, pressure etc. It is built of nodes where each node is connected to one or more sensors. They are used for Medical applications, Security monitoring, Structural monitoring and Traffic monitoring etc. The number of sensor nodes in a Wireless Sensor Network can vary in the range of hundreds to thousands. In this project work we propose a distributed algorithm for detection of faults in a Wireless Sensor Network and to classify the faulty nodes. In our algorithm the sensor nodes are classified as being *Fault Free*, *Transiently Faulty*, or *Intermittently Faulty*, considering the energy differences from its neighbors in different rounds of the algorithm run. We have shown the simulation results in the form of the output messages from the nodes depicting their health and also compared the results in form of graphs for different average node degrees and different number of rounds of our algorithm run.

Contents

List of Figures

List of Abbreviations

WSN: Wireless Sensor Network

RAM: Random Access Memory

ROM: Read Only Memory

P2P: Peer-to-Peer

OS: Operating System

I/O: Input and Output

ADC: Analog to Digital Converter

CMOS: Complementary Metal — Oxide — Semiconductor

FF: Fault Free

TF: Transiently Faulty

IF: Intermittently Faulty

–

Chapter 1

Introduction

1.1 Wireless Sensor Network

A wireless Sensor Network (hence forward referred to as WSN) is a collection of nodes deployed mostly in the range of hundreds to thousands connected to the same network. Each node has its own processing capability, memory, power source and sensors. It is described in detail in following sections. The nodes are designed in such a manner that they can communicate and organize themselves through the network. The nodes have a wide range of cost depending on the need. The concept of WSN is getting popular day by day. WSN is mostly used to monitor environment and physical conditions.

1.2 Structure of a WSN

The nodes in a WSN are connected in a Mesh Topology , Peer to Peer, Tree and Star. The size of a node is as small as a coin as depicted in the following figure.

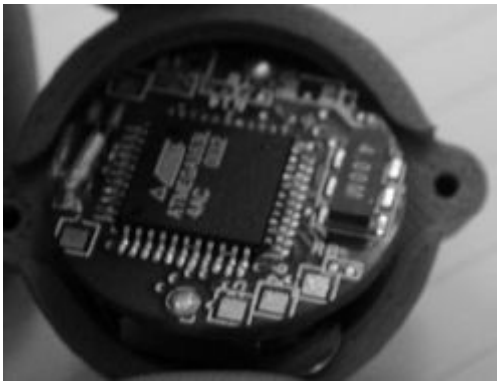

FIGURE 1.1: Structure of Node

The wireless sensor nodes do not communicate with a central node rather with its surrounding local nodes. There are many constraints associated with the designing of a node. The nodes have embedded processors which have to implement complex networking protocols with just a memory of some kilobytes. As the size of the device is smaller and so is the power source. The nodes include both hardware and an operating system such as TinyOS (an operating system for WSN). The most advanced hardware platform used now-a-days is single chip CMOS device. This WSN node consists of a microcontroller, transmitter, ADC, I/O ports, and memory.

1.2.1 Peer-to-Peer Topology

In this type of topology the nodes do not have any centralized controller rather communicate with each other that is its peers. Each node behaves as a client and server to its neighboring nodes.

FIGURE 1.2: Peer-to-Peer

1.2.2 Mesh Topology

In this type of topology the nodes can communicate with each other. The data to be sent to the desired destination from the source travels from one node to another by hopping. It is quite expensive to deploy this type of topology.

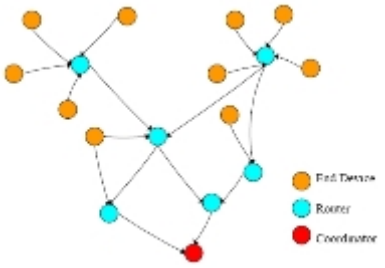

FIGURE 1.3: Mesh Topology

1.2.3 Star Topology

This type of topology has a centralized approach when it comes to nodes communicating with each other. Every message from a node must pass through a central hub or server before it reaches the destination. The central node acts as the server while others act as the clients.

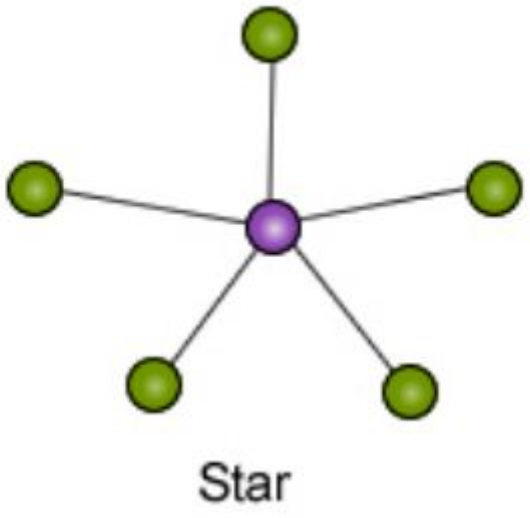

FIGURE 1.4: Star Topology

1.2.4 Tree Topology

The Tree topology can be considered as a hybrid of star and peer-to-peer topology. The central node acts as a root.

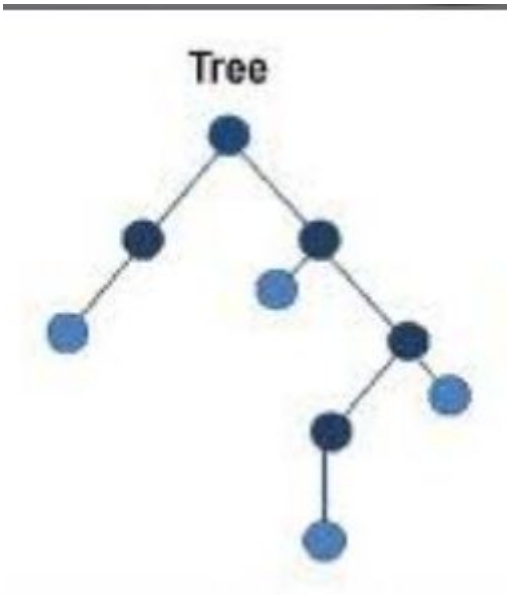

FIGURE 1.5: Tree Topology

1.3 Basic architecture of a WSN Node

1.3.1 RAM(Random Access Memory)

The RAM in a sensor node is used to store the current readings sensed. The data may be hampered in case of power supply disruption.

Figure 1.6: RAM

1.3.2 ROM(Read Only Memory)

The ROM is used to store the programs used in implementing the WSN.

Figure 1.7: ROM

1.3.3 Transmitters

The transmitters are used in half-duplex mode for both receiving and sending operations. It has 4 states

- IDLE
- SLEEP
- RECEIVE
- TRANSMIT

FIGURE 1.8: Transmitters

1.3.4 Power Supply

The basic requirement the power supply must provide maximum power in minimum size. As the batteries cannot be charged in an usual way, energy has to be obtained from other sources like photo voltaic cells, temperature gradient etc.

FIGURE 1.9: Power Supply

1.4 Applications of a WSN

1.4.1 Monitoring of Environment

A WSN is deployed to collect information in an environment. These are deployed for over a period ranging from months to years to analyze any trend in the weather and other aspects.

FIGURE 1.10: Monitoring Environment

1.4.2 Monitoring of Security

A WSN can be deployed in an area for military surveillance. The sensors are used to detect any unusual activity in the deployed region. If any such activity is sensed then it gets reported to the centre for taking any required action.

1.4.3 Used in Hospitals

The WSN can be used to monitor the health conditions of patients remotely there by reducing the use of more man power.

FIGURE 1.11: Monitoring Security

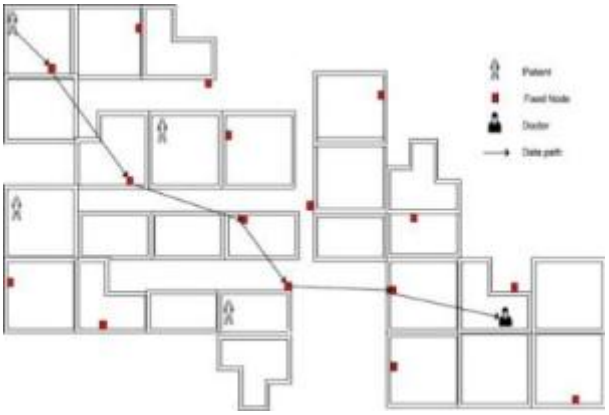

FIGURE 1.12: Applied in Hospital

1.4.4 Used in Tracking Inventory

WSNs can be used to track items in a store house or factory to prevent the loss of items. It is also used in tracking of parcels.

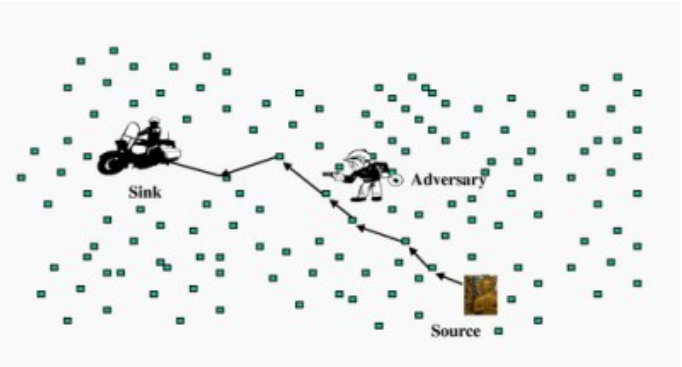

FIGURE 1.13: Tracking Inventory

1.4.5 Used in the field of Agriculture

WSNs can be deployed to monitor irrigation activities and fertilization of fields.

FIGURE 1.14: Applied in Agriculture

1.5 Faults

What are Faults??
Faults are a kind of a situation which leads to errors

1.5.1 Types of Faults

Based on persistence, faults are categorized into the following types:
Transient Fault: The transient faults occur very less frequently or rarely and gets removed without any intervention. These are mostly caused due to noise.
Intermittent Fault: The Intermittent Faults occur more frequently compared to transient faults. The nodes which are faulty sometimes behaves fault free and vice-versa.

Permanent Faults: The permanent fault is permanent for a node. The intermittently faulty node gradually becomes a permanent faulty node in due course of time.

1.5.2 Causes of Faults

Due to the fragile nature of sensor nodes and also because of the depletion of their limited power source, faults may occur. Due to harsh environments where nodes are being deployed, the nodes may receive and transmit incorrect sensor readings. In the WSNs the links are also prone to faults. Also when nodes are embedded or mobile nodes can sometimes go out of range of communication. Faults are also caused due to multi-hop communication as it takes several hops to deliver the data to sink. Failure of single intermediate node may lead to a total erroneous data. Congestion which occurs due to large number of nodes transmitting the same time may also lead to packet loss.

Chapter 2

Literature Review

The rise in growth of WSN in applications in various wireless environmental monitoring applications has forced to focus on the quality of the service. The main task now is to design better fault management approaches. The existing approaches have been divided into 3 phases. The decreasing cost of the electronic devices has been an important factor for its increase use in deployment to extract useful information from harsh environment. But there are some limitations on the hardware and software pertaining to the small size of the nodes like energy supply, memory, processing capabilities etc. In usual cases the nodes have irreplaceable power source with limited energy. The energy level gradually decreases over days leading to faults.

The nodes are also prone to faults because of the harsh environments in which they are deployed. This leads to situations like the node behaving arbitrarily or the node becomes inactive for some time. There are various existing approaches to manage faults in forms of architectures [1–3], protocols , detection algorithm[4–7] and detection decision fusion algorithm[8–13]. These fault management approaches have been divided into three phases such as Fault detection, fault diagnosis and fault recovery. Fault Detection has been divided into 2 approaches.

2.1 Centralized Approach

In a WSN the centralized approach is an usual solution to find the faulty nodes. A central node based on geographically or logically (it can be a base station [14–16], central controller, sink[17]) does the job of finding the faulty node and monitoring the nodes that misbehave or give arbitrary outputs. The assumptions is that the central node has unlimited amount of resources like power so that it can execute a wide range of fault management maintenance. It also assumes that the network lifetime can be extended if the complex management work and message passing is given to the central node. Active detection model is used by the central manager to the receive the states of the sensor nodes as well as the network. It is done by sending requests to the nodes at regular intervals and detects faulty nodes. Sympathy[17] is message flooding approach to collect event data states of the sensor nodes. Sympathy nodes in order to minimize the number of communication messages nodes must send and save energy. The sympathy node can transmit events after selection to the sympathy sink.

In another approach [16] the network topology information is appended into the routing update messages of the node. In this way the base station can construct the entire network topology. After the network topology is formed, the base station can now easily know the faulty nodes by using divide and conquer approach. But this model assumes that the base station can send messages directly to any node and each node has an unique identifiatcion number. In another approach[15] the base station uses marked packets. These marked packets conatin information about source and destination nodes. It uses the responses of the nodes to identify and locate the faulty nodes due to excessive packet drops or compromised data detection. In WinMS[3] approach the central manager prevents failure by comparing the current and historical states of the sensor nodes with the total network topology. It is a centralized approach.

The centralized approach in identifying the faulty nodes in a sensor network is efficient and accurate to some extent. But due to the resource constraints it is not easy to collect the node information at regular intervals. Due to the centralized approach there is transmission of packets to a single central node which leads to high volume of message traffic and energy depletes quickly. It happens not only to the central node but also to the nodes close to it. It is inefficient and expensive in case of large WSNs. The multi hops communication of this approach increases the response delay from the base station to the faults in a network.

2.2 Distributed Approach

It is a localized approach evenly distributing fault management in a WSN. A node makes some decisions before communicating with the central node. Information is delivered to central nodes when required in case of fault detection. Similar developments are self-detection and self-correction of faults in a node[18, 19] failure detection by neighboring nodes[4–7, 19] , WATCHDOG[20], Clustering[1, 21]

A self detection model was proposed by S Harte et al.[18] to observe the malfunctioning of the physical components of the sensor node. In the self-detection of node[22], the node just observes status of its sensors by comparing with pre-defined fault models.

In neighbor coordnination approach, neighbors coordinate with the neighbors to find any fault before communicating with the central node. This approach helps in reducing the network communication messages hence saves energy. In an decentralized approach[19] the sensor node can execute a localized fault detection algorithm. The nodes can also ask for diagnostic information from the neighboring nodes as well. Min et al[6] proposed an algorithm that can detect faults by comparing the sensor nodes with neighboring nodes and the nodes are suspicious if there is huge difference in the reading. It can be implemented on large size networks but the number of faulty nodes must be less. In another approach [4, 5] the accuracy of failure detection is addressed by 2 phases called clustering and distributed detection.

Clustering[23] is an important technique in WSNs. In an approach proposed by Ann T .Tai et al., [1]failure detection of nodes is done using clustering to achieve scalability , completeness, and accuracy. The entire network is divided into clusters and the fault management is divided among them. The cluster head detects the faults. When a failure is detected this information is propagated to all the clusters. In another approach proposed by Ruiz et al[21] fault detection is supported by MANNA[2]. Here the agents are executed in the cluster heads which has more resources than the usual nodes. Every node checks its energy value and sends the information to the agent during any state change. This information is used by manager to build a network energy model to detect potential failures in the network.

In another approach using clustering [24], the clusters are formed without any overlapping. Each node declares itself to be a cluster head. Clusters are formed but with the condition that every sensor should belong to a cluster without crossing the limit. All the nodes keeps on sending its sensed temperature value to its neighbors. Consider two nodes S_i and S_j. The temperature sensed by S_i is compared with another node S_j . This comparison is done for two time instants t_i and t_{i+1}. The status element C_{ij} is recorded for any deviation from the threshold value. C_{ij} can hold two values, 1 or 0. The status of the node such as Likely Good(LG) or Likely Fulty(LF) is calculated. It is calculated by comparing the value of C with its neighboring sensors. Then it is determined whether a node is good(GD) or Faulty (FT). The GD node is used to detect the faultiness of other nodes. If all the nodes are FT then the algorithm is stopped. This algorithm is repeated for every cluster. Every cluster head maintains the status of the nodes in the cluster along with the all the nodes in the network.

Distributed detection is a process where each node takes a decision on faults in a sensor network.

The major challenge in this approach is to obtain a better balance between fault detection accuracy and energy usage in the network. The metrics on which the fault detection techniques depend are:

- Precision
- Detection
- Communication cost
- The maximum limit of number of faulty nodes

Clouqueurs work[13], confirms that all the fusion nodes(manager nodes) in the network possess the same information about the network before deciding, as the faulty nodes may provide inconsistent data. Wang et al [15], adopts data aggregation and redundant data lessening methods of cluster to reduce the transmission power loss and achieve less computational time and memory at the fusion nodes.

In Self-Detection also called as Passive Detection, the sensor nodes monitor their residual energy at regular intervals to identify a potential failure. In this model the cause of sudden death of a WSN node is depletion of the battery.

When the battery loses its energy below a threshold, it is considered to be dead . The node in such a situation is called as a failing node. In such a situation the failing node sends a message informing to its cell manager that it is going to sleep mode due to depletion of its battery below the threshold[21]. This is an approach localized to the node itself and doesnt use much in-network communication thereby reducing the consumption of energy. The response delay of the management system towards the potential faulty sensor nodes is also reduced [23].

Active Detection is another efficient approach. The cell members are asked by their respective cell managers to reply to it with their updates. This method continues at regular intervals called as in-cell update cycle. The cell members sends a message which consists of its node id, location, and energy. Failing to the receipt of this update message from a node, the cell manager sends a message to that corresponding node to send its status[25]. If that node doesnt reply within a stipulated time then the cell manager declares the node to b a faulty and then informs all the cell members. The cell manager also uses Self-Detection approach to monitor the residual energy at regular intervals.

A node is classified as a low energy node and is about to sleep if its residual energy decreases to less than or equal to 20%. If a node has residual energy more than 50% then it is a potential candidate for a cell manager. But if a case arises when the cell manager has its energy below 20%, it informs its status to the cell members and the group manager as well so as to find a new cell manager. Every cell manager sends its status to the group manager at regular intervals which is also termed as out-cell update cycle. This out-cell update cycle unlike in-cell update cycle is less frequent. If the group manager does not get any update from the same cell manager consecutively second time then it informs to the cell members [25] that the respective cell manager is dead and faulty. The group manager performs in a similar manner. If the base station doesnt receive any acknowledgement from the group member it declares it to be faulty and dead. It then informs to the cell managers in that group.

The fault diagnosis stage aims at detecting the causes of a fault occurring in the network. The accuracy and correctness of fault detection is studied thoroughly in[2, 6, 10, 13]. Most of the fault diagnosis techniques such as in [8, 18]concentrate on hardware component malfunctioning, assuming that softwares are fault tolerant. Farinaz et al[18] assume two fault models, one for sensors producing binary outputs and the other for sensors producing analog or contiguous output. Thomas

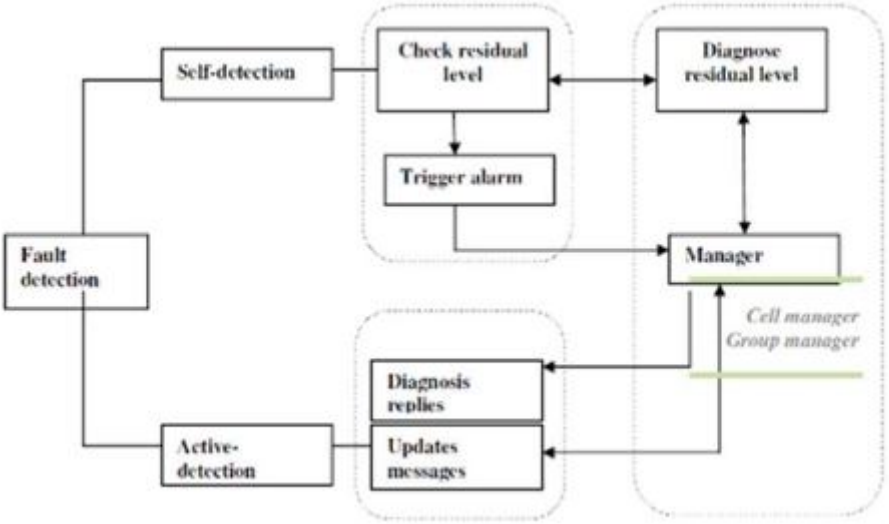

FIGURE 2.1: Fault Detection Mechanism

et al[13], assumed faulty nodes are always due to bitter environmental conditions. Ming Ding et al.[7], model the events by real numbers such as the sensor readings which can be specified by fault tolerance requirements for various sensor applications.

The fault recovery stage aims at restructuring and reconfiguring of the faulty network such as the faulty nodes do no more hamper to the network performance. The most simple and straight forward approach followed in WSN is isolation of faulty nodes. In Marti et al[4], when a node detects a faulty neighbour then it discards it and chooses a new one for routing. Staddon et al[17] proposed two method to resume network paths from the faulty nodes that gets detected in each routing update epoch. Some of the proactive method includes WinMs[14], where the central maneger identifies a network region of weak health by comparing it with current network state. Koushanfar et al[18] suggested a heterogeneous back up scheme where they assumed that the application programs or the operating system can adapt to match the available hardware. This design throws an insight towards the future where a node functionality needs to be updated because of occurance of a fault.

In another approach, the dead nodes can be made to work again or these gaps can be filled in by using mobile nodes. For a cell manager getting dead, it can appoint a secondary cell manager before-hand to act as its back up. If the cell manager fails then a message is passed to all the cell members and the secondary cell manager as well. The secondary cell manager now behaves as a cell manager

and the previous cell manager acts as a normal node in its low energy state. The new cell manager now receives updates from all the cell members and it also identifies a new secondary cell manager [21]. Consider the example given below. The cell member 1 is the acting cell manager now. The cell member 3 is chosen as the secondary cell manager. The energy of the member 1 starts depleting and it reduces to below 20%. The cell manager 1 sends a message to all the cell members invoking the fault recovery method. Now the cell member 3 acts as the cell manager now. If the secondary node has its energy depleted as well then all the cell members exchange messages between them. The node with residual energy more than 50% is chosen as the new cell manager.

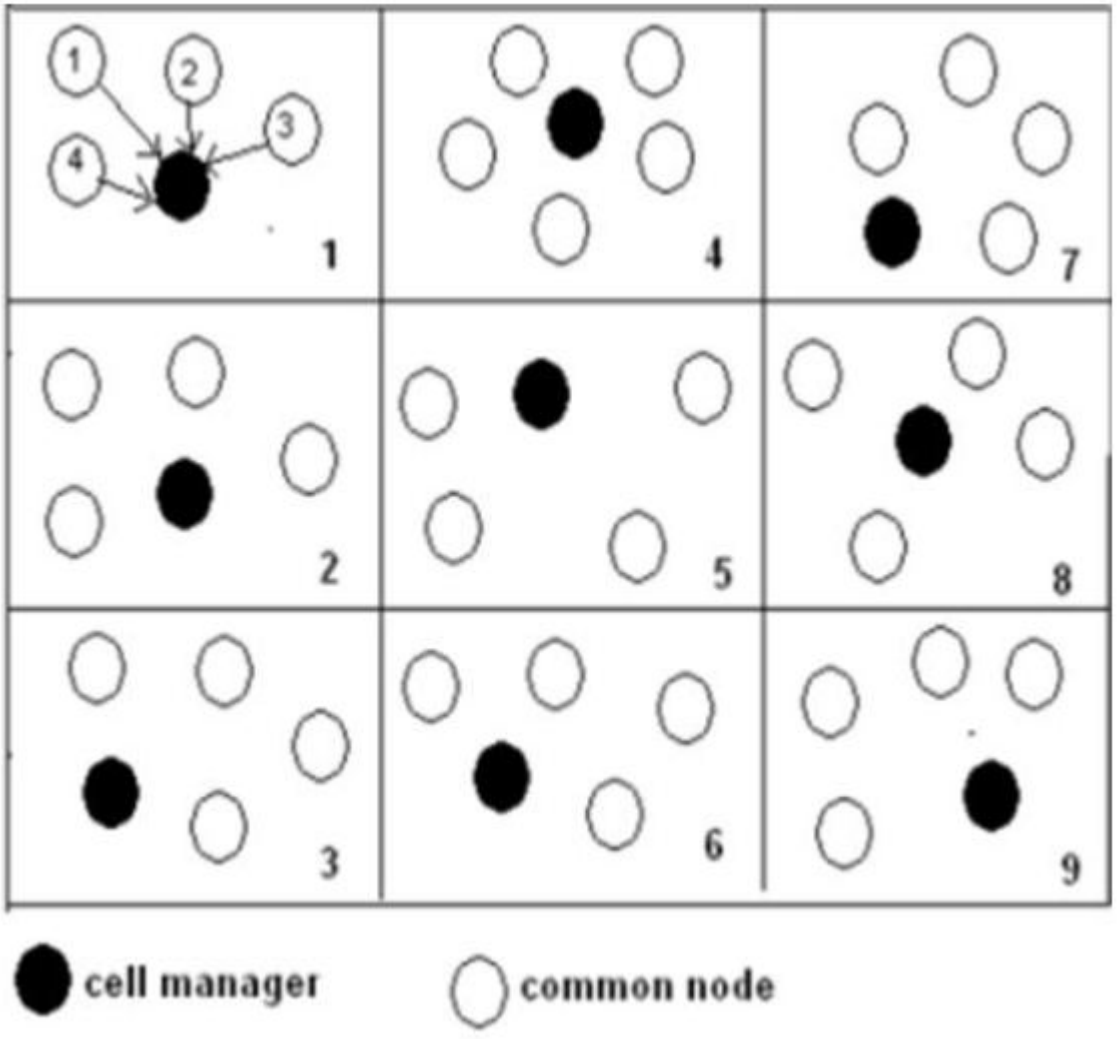

FIGURE 2.2: Fault Recovery

The approach we have focused on is a probabilistic approach to diagnose intermittent faults in a Wireless Sensor Network[26]. The algorithm proposed in this

paper is a distributed one run by each node. Each node compares the energy value of its neighbors and maintains a vector according to the energy difference. At the end of each round the vector is analyzed to judge if the node was faulty or fault free for that round and another vector b is maintained to record the finding of each round. At the end of rounds a node determines if it is intermittently faulty or not by verifying it vector b. Though this approach successfully detects intermittent faults but it does not classify the faults further and behaves harsh with the nodes which are faulty for only a few instance of time.

Chapter 3

Motivation

Discrimination of transient from intermittent or permanent faults is crucial as a sensor node with transient fault does not necessarily imply that the sensor node should be isolated. Although the unstable environment might warrant a temporary shutdown. We were motivated for classification of faulty nodes as discrimination between transient and intermittent faults solves many key problems. It leads to effective bandwidth utilization. By isolating intermittent faults, the traffic generated by the intermittently faulty nodes is restricted. Then it helps in effective energy utilization. The depletion of sensor node battery energy in forwarding the erroneous data generated by intermittent faults can be avoided. It also helps in better network coverage and connectivity. Isolation of fault-free nodes with transient faults will reduce the available sensor nodes in the network thus impacting network coverage and connectivity.

3.1 Issues in the existing algorithms

The existing algorithms that we have studied detects nodes as intermittently faulty nodes or fault free nodes in the Wireless Sensor Network. Labeling nodes as intermittently faulty that shows fault for only some instances is very harsh on them. They work and may use to work as fault free for a long period of time henceforth. So in our algorithm we detect faults in each round and classify the faulty nodes into transiently faulty and intermittently faulty according to the frequency of their occurrences.

An intermittently faulty node is one that malfunctions for some instances and otherwise shows correct sensor readings and results. These malfunctioning instances are very irregular and do not show any pattern. So the nodes have to be isolated from the network for its proper functioning.

A transiently faulty node is one that malfunctions only for a very few instances of its healthy lifetime and hence do not cause a serious threat to the working of the network. In the worst case, it may cause a temporary shutdown of the system. So these nodes need not be isolated from the system and only the readings during which time they were found faulty need to be discarded during the observation.

Chapter 4

Proposed Mechanism

4.1 Network Model

It is assumed that in the Wireless Sensor Network, the sensor nodes are randomly deployed in the area under surveillance. The area is very dense and all the sensors have a common range of transmission. And the sensor nodes located in this range of transmission of a particular node are called its neighbors. There might be a fault occurring in any of the sensor nodes at a particular instant of time. That node at that instance goes out of service.

As we are carrying out a kind of voting among the sensors such that even if a single neighbor predicts it to be fault free then the node is labeled fault free for that instance of time. So we assume that each node at least has 2 neighbors. Since the area under surveillance is very dense so this condition can be obtained very easily.

4.2 Communication Model

In the proposed system, communication is between only the neighbor sensor nodes. A sensor node is considered as neighbor of another sensor node if it lies within the transmission range of the first said sensor node. So the set of nodes that a sensor node n can communicate is given as
E(s)={ n_i such that n_i lies within the transmission range on n}

We have assumed a full duplex mode of communication. So the communication is bidirectional between nodes. So if there is a communication link between n_i and n_j then there is also a communication link between node n_j and n_i
$l_{n_i,n_j} \Leftrightarrow l_{n_j,n_i}$

4.3 Energy Consumption Model

Energy is consumed in a node for reading data from the environment, processing it, transmitting it to its neighbors and receiving data and control signals etc. The energy consumption in our proposed algorithm follows the equation:

$$P = P_0 c(x/d_0) \tag{4.1}$$

where,
P= Power used,
P_0 = Power used to transmit a data packet to a distance d_0,
x = Distance travelled by the packet and
c = Network constant

4.4 Fault Model

We assume that fault can occur at any level of the sensor network such as hardware, physical layer, middleware and system software. In this project we have focused on hardware level faults only and have assumed that all the application and system softwares are tolerant to faults.

The hardware components of the sensor nodes are categorized into two groups. The first group consists of the following components:

- Computation Engine
- Storage Subsystem
- Power Supply Infrastructure

The second group consists of the following components:

- Sensors
- Actuators

The components present in the first group are very reliable because they follow heterogeneous BISR fault tolerant schemes. But the second group is more prone to malfunctioning. Since targeted level of fault tolerance will be provided by the first group[27], only the second group of faults are considered that includes three types of faults:

- Calibration Systematic Error
- Random Noise Error
- Complete Malfunctioning

We have assumed it in the project that nodes are still capable of receiving, sending and processing even if they are faulty

Now considering the type of faults that our algorithm detects. it detects both hard faults and soft faults. Hard faults are considered as permanent faults that always produce errors when they are fully exercised[28]. Soft faults are temporary faults that only temporarily affect the system. In the proposed algorithm we have detected transient faults which are in the category of soft faults and intermittent faults which are categorized as hard faults.

Another categorization of faults are into static and dynamic faults. Static faults are those which are caused due to only a single kind of malfunctioning and are persistent. Whereas dynamic faults are caused due to combined malfunctioning of different kinds and are dynamically introduced in the system while it is running. Our proposed algorithm only focusses on the faults that are static in nature. No faults are dynamically introduced in the system.

4.5 Definitions

n: Total number of sensor nodes.
S: Set of all the sensor nodes.
n_i: Any particular node or we say the ith sensor node in the network.
N(n_i): Set of the neighbors of node n$_i$.
E_{in}: Initial energy value of each node.
E_u: Energy usage in each round of the algorithm run.
E_i: Current energy value of the i^{th} node. E_j: Current energy value of the j^{th} node which is a neighbor of i^{th} node.
Δ E: Energy difference between i^{th} and j^{th} node.
θ_e: A predefined energy threshold value.
k: The numbers of rounds of the algorithm run.
v: A vector in the node to store the votes of the neighbors.
b: A vector in the node to store the labels for each round of the algorithm run.
f: Frequency of faults. It is ratio between number of rounds in which a fault was detected to the total number of rounds
θ_f: A predefined fault frequency threshold

A sensor node is considered as neighbor of another sensor node if it lies within the transmission range of the first said sensor node. Each sensor node sends its energy value to each of its neighbors in each round of the algorithm. Comparisons are carried out between the energies of the sensor nodes. If the difference is within a predefined threshold then the node is voted a fault free by its neighbor and if the difference is above the predefined threshold then it is voted as a faulty node. The votes of all the neighbors are considered in a way as if all the neighbors say the node is faulty then only the node is labeled faulty for that round of the algorithm run. The algorithm is run for a particular number of rounds to have faith in the results obtained.

The results found in each round i.e the labels of the sensor nodes in each round is considered to finally determine if the node is fault free, transiently faulty or intermittently faulty.

4.6 The System

We have considered the System Graph S of the Wireless Sensor Network as an undirected graph. A miniature model is shown below: The System model consists

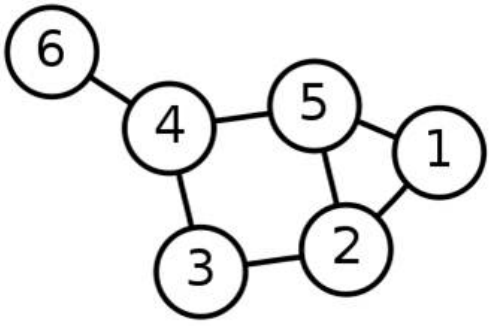

FIGURE 4.1: System Graph

of vertices and edges.
Vertices:
V(s):Nodes of the Network
Edges:
E(s)={ (n_i, n_j) such that $n_i, n_j \in$ V(s) and there is a communication path between n_i and n_j}

Testing Graph:
We have considered the testing graph T as
$V(T_s) = V(s)$
$E(T_s) = \{(n_i, n_j)$ such that $n_i, n_i \in V(s)$ and the node n_i compares the estimated and observed remaining energy value of node n_j and vice-versa $\}$
For each edge $(n_i,n_j) \in E(T_s)$
Label=0: If the comparison outcome of observed & estimated remaining energy value of nodes n_i and n_j is within threshold.
Label=1: If the comparison outcome is above threshold.

4.7 Proposed Algorithm

Data: Energy Value of each node for each round of Algorithm run
Number of rounds k
Result: Health of Each Node
$E_i = E_{in}$;
while *rounds* $\leq k$ **do**
  $E_i = E_i$ - E_u ;
  while *each sensor node is visited* **do**
    while *energy value of each neighbor is received* **do**
      if $(E_i - E_j) \leq \theta_e$ **then**
        Set V[i][j] = 0;
      else
        Set V[i][j] = 1;
      end
    end
    if *at least one of the elements of v[i] is 0* **then**
      Set b[i][k] = 0;
    else
      Set b[i][k] = 1;
    end
  end
end
Calculate the Fault Frequency f;
if *f=0* **then**
  Print: Fault Free
else
  if $f \leq \theta_f$ **then**
    Print: Transiently Faulty
  else
    Print: Intermittently Faulty
  end
end

Algorithm 1: Proposed Fault Classification Algorithm

4.8 Analytical Study of the Algorithm

Here we present an analytical study of our proposed algorithm. We verify our algorithm for its correctness and completeness.

4.8.1 Proof of Correctness

A correctness proof is a formal mathematical argument that an algorithm meets its specification, which means that it always produces the correct output for any permitted input.

We write down below, informal arguments giving an outline of the proof.

First we consider a Fault Free node n_p:

Step 1

$E_p = E_{in}$;

Step 2:

After each round of the algorithm run the energy will become:

$E_p = E_p - E_u$

Step 3:

In each round E_p will be compared with the energy of all the neighbors of n_p.
Its energy value is found to be within the threshold when compared with the energies of the maximum neighbors.

Step 4:

The vector V[p] is majorly assigned 0s.

Step 5:

The vector b[p][k] is assigned 0 as the majority of elements in V[p] are 0.

Step 6:

At the end of all the rounds of the algorithm the vector b[p] entirely consist of 0s

Step 7:

The fault frequency f is calculated to be 0 as all the elements in b[p] is 0.

Step 8:

Since f=0 so Fault Free is printed. Since n_p is detected as "Fault Free", the algorithm is correct in detecting a fault free node.

Now we consider a Transiently Faulty node n_q:

Step 1:

$E_q = E_{in}$;

Step 2:

After each round of the algorithm run the energy will become:

$E_q = E_q - E_u$

Step 3: In each round E_q will be compared with the energy of all the neighbors of n_q. Its energy value is found to be within the threshold when compared with the energies of the maximum neighbors at instance x and out of threshold when compared with the energies of all its neighbors at instance y.

Step 4:

The vector V[q] is majorly assigned 0s in instance x but all the elements of V[q] are assigned 1 in instance y.

Step 5:

The vector b[q][k] is assigned 0 in instance x and 1 in instance y.

Step 6:

At the end of all the rounds of the algorithm the vector b[q] majorly consists of 0s when the instance was x but also contains some 1s when the instance was y

Step 7:

The fault frequency f is calculated considering the elements in b[q].

Step 8:

Since it is found f $\leq \theta_f$ so Transiently Faulty is printed.

Since n_q is detected as "Transiently Faulty", the algorithm is correct in detecting a transiently faulty node.

Now we consider an Intermittently Faulty node n_r:

Step 1:

$E_r = E_{in}$;

Step 2:

After each round of the algorithm run the energy will become:

$E_r = E_r - E_u$

Step 3:

In each round E_r will be compared with the energy of all the neighbors of n_r. Its energy value is found to be within the threshold when compared with the energies of the maximum neighbors at instance x and out of threshold when compared with the energies of all its neighbors at instance y.

Step 4:

The vector V[r] is majorly assigned 0s in instance x but all the elements of V[r] are assigned 1 in instance y.

Step 5:

The vector b[r][k] is assigned 0 in instance x and 1 in instance y.

Step 6:

At the end of all the rounds of the algorithm the vector b[r] consists of 0s when the instance was x but majorly contains 1s when the instance was y

Step 7:

The fault frequency f is calculated considering the elements in b[r].

Step 8:

Since it is found f $> \theta_f$ so Intermittently Faulty is printed. Since n_r is detected as "Intermittently Faulty", the algorithm is correct in detecting an intermittently faulty node.

So we found that the algorithm correctly finds all types of faults that we have considered in the project. So the correctness of the algorithm is proved.

4.8.2 Proof of Completeness

A completeness proof is a formal mathematical argument that an algorithm covers all the valid input values and produces output for them, which means that it always produces some output for any permitted input.

We write down below, informal arguments giving an outline of the proof.

Any node n_i that is considered, its b vector is analyzed. Its frequency of faults f is calculated. Now depending upon the calculated f, it is assigned to one of the class Fault Free abbreviated as ff, Transiently Faulty abbreviated as tf and Intermittently Faulty abbreviated as if. No node is left unassigned a class or assigned to multiple classes as we have considered hard lined and mutually exclusive conditions.
So we have:
$N_{ff} + N_{tf} + N_{if} = \mathrm{N}$
$S_{ff} \cup S_{tf} \cup S_{if} = S_n$
$S_{ff} \cap S_{tf} = \phi$
$S_{tf} \cap S_{if} = \phi$
$S_{ff} \cap S_{if} = \phi$
Where S is the set of nodes.
So from the above the completeness of the proposed algorithm is proved.

Chapter 5

Simulations and Results

5.1 The Setup

The simulation set up used is Intel Dual Core Processor with 2.10GHz Clock speed and Memory of 4 GB. The algorithm was simulated in OMNeT++ Version 4.2.2 Network simulator.
The Wireless Sensor Network used in the simulation is shown below:

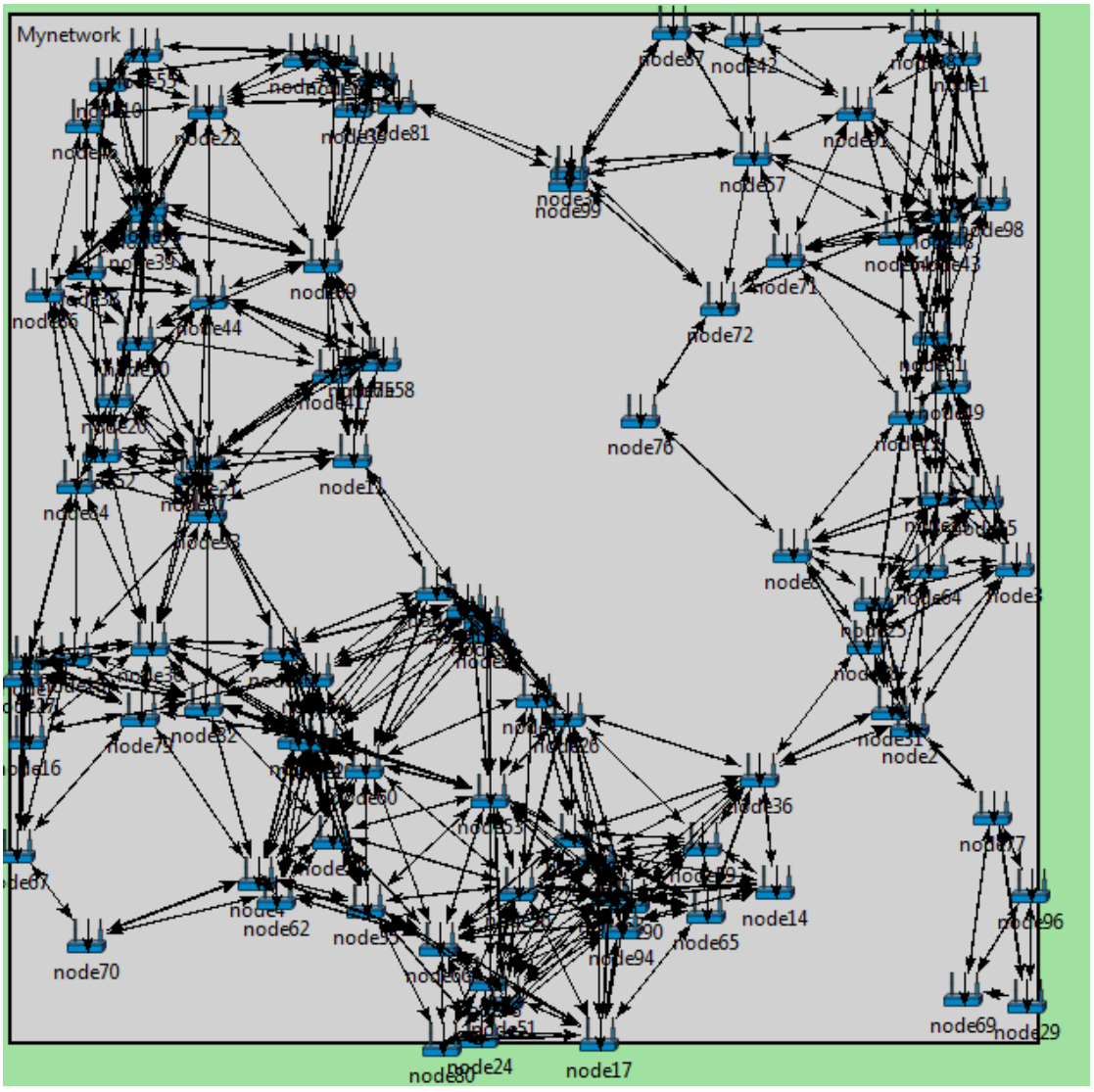

FIGURE 5.1: Simulation Network

5.2 Simulation Snapshot

The Simulation output trace from the OMNeT++ simulator is as shown below:

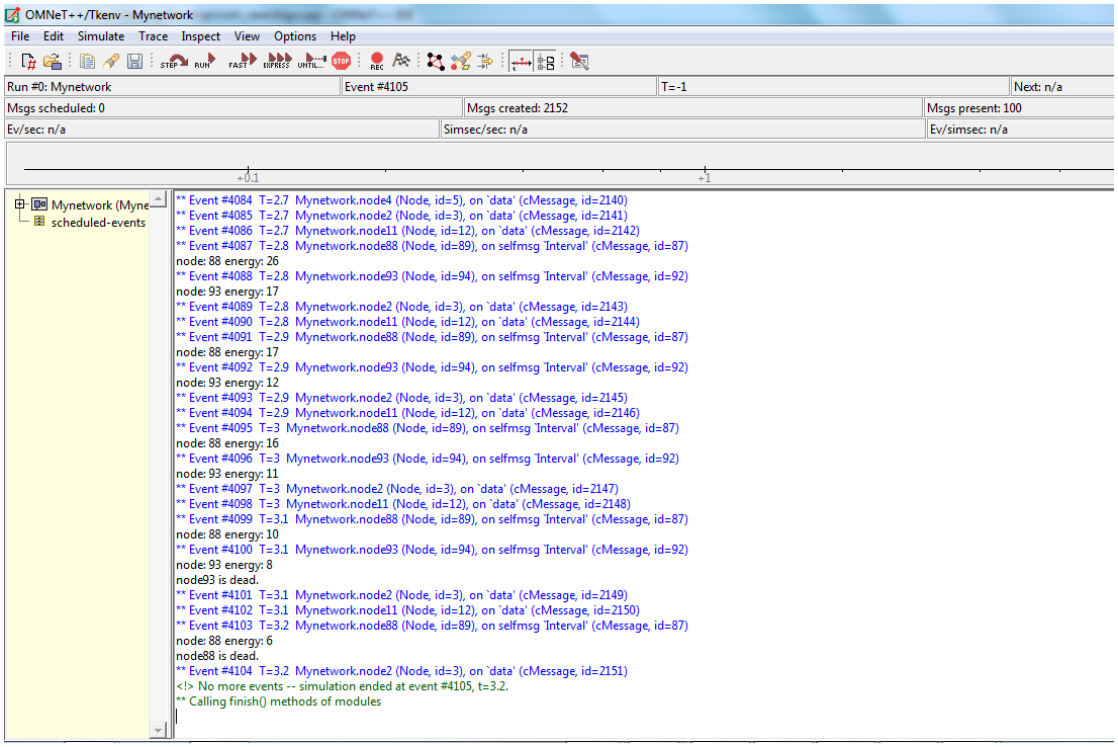

FIGURE 5.2: Omnet Output Window

5.3 Outputs

```
The algorithm Classified the given nodes as:

Node 1 : Intermittently Faulty  Node 2 : Fault Free
Node 3 : Intermittently Faulty  Node 4 : Intermittently Faulty
Node 5 : Intermittently Faulty  Node 6 : Intermittently Faulty
Node 7 : Fault Free             Node 8 : Fault Free
Node 9 : Fault Free             Node 10 : Intermittently Faulty
Node 11 : Fault Free            Node 12 : Fault Free
Node 13 : Intermittently Faulty Node 14 : Fault Free
Node 15 : Fault Free            Node 16 : Fault Free
Node 17 : Fault Free            Node 18 : Fault Free
Node 19 : Fault Free            Node 20 : Fault Free
Node 21 : Intermittently Faulty Node 22 : Intermittently Faulty
Node 23 : Intermittently Faulty Node 24 : Intermittently Faulty
Node 25 : Intermittently Faulty
```

FIGURE 5.3: Output of Existing Algorithm for 25 nodes

```
The algorithm Classified the given nodes as:

Node 1 : Intermittently Faulty  Node 2 : Intermittently Faulty  Node 3 : Fault Free
Node 4 : Fault Free             Node 5 : Intermittently Faulty  Node 6 : Intermittently Faulty
Node 7 : Intermittently Faulty  Node 8 : Intermittently Faulty  Node 9 : Fault Free
Node 10 : Intermittently Faulty Node 11 : Fault Free            Node 12 : Fault Free
Node 13 : Fault Free            Node 14 : Fault Free            Node 15 : Intermittently Faulty
Node 16 : Fault Free            Node 17 : Fault Free            Node 18 : Fault Free
Node 19 : Fault Free            Node 20 : Fault Free            Node 21 : Fault Free
Node 22 : Fault Free            Node 23 : Fault Free            Node 24 : Fault Free
Node 25 : Fault Free            Node 26 : Fault Free            Node 27 : Fault Free
Node 28 : Fault Free            Node 29 : Fault Free            Node 30 : Intermittently Faulty
Node 31 : Fault Free            Node 32 : Fault Free            Node 33 : Fault Free
Node 34 : Fault Free            Node 35 : Fault Free            Node 36 : Fault Free
Node 37 : Fault Free            Node 38 : Fault Free            Node 39 : Fault Free
Node 40 : Fault Free            Node 41 : Fault Free            Node 42 : Fault Free
Node 43 : Fault Free            Node 44 : Fault Free            Node 45 : Fault Free
Node 46 : Fault Free            Node 47 : Fault Free            Node 48 : Fault Free
Node 49 : Fault Free            Node 50 : Intermittently Faulty Node 51 : Fault Free
Node 52 : Fault Free            Node 53 : Fault Free            Node 54 : Fault Free
Node 55 : Fault Free            Node 56 : Fault Free            Node 57 : Fault Free
Node 58 : Fault Free            Node 59 : Fault Free            Node 60 : Intermittently Faulty
Node 61 : Intermittently Faulty Node 62 : Fault Free            Node 63 : Fault Free
Node 64 : Fault Free            Node 65 : Fault Free            Node 66 : Fault Free
Node 67 : Fault Free            Node 68 : Fault Free            Node 69 : Fault Free
Node 70 : Intermittently Faulty Node 71 : Intermittently Faulty Node 72 : Fault Free
Node 73 : Fault Free            Node 74 : Fault Free            Node 75 : Fault Free
Node 76 : Fault Free            Node 77 : Fault Free            Node 78 : Fault Free
Node 79 : Fault Free            Node 80 : Fault Free            Node 81 : Fault Free
Node 82 : Fault Free            Node 83 : Fault Free            Node 84 : Fault Free
Node 85 : Fault Free            Node 86 : Fault Free            Node 87 : Fault Free
Node 88 : Fault Free            Node 89 : Fault Free            Node 90 : Fault Free
Node 91 : Intermittently Faulty Node 92 : Intermittently Faulty Node 93 : Intermittently Faulty
Node 94 : Intermittently Faulty Node 95 : Fault Free            Node 96 : Fault Free
Node 97 : Intermittently Faulty Node 98 : Intermittently Faulty Node 99 : Fault Free
Node 100 : Intermittently Faulty
```

FIGURE 5.4: Output of Existing Algorithm for 100 nodes

```
The algorithm Classified the given nodes as:

Node 1 : Intermittently Faulty  Node 2 : Transiently Faulty
Node 3 : Intermittently Faulty  Node 4 : Transiently Faulty
Node 5 : Intermittently Faulty  Node 6 : Intermittently Faulty
Node 7 : Fault Free             Node 8 : Transiently Faulty
Node 9 : Transiently Faulty     Node 10 : Transiently Faulty
Node 11 : Transiently Faulty    Node 12 : Transiently Faulty
Node 13 : Transiently Faulty    Node 14 : Fault Free
Node 15 : Transiently Faulty    Node 16 : Transiently Faulty
Node 17 : Transiently Faulty    Node 18 : Fault Free
Node 19 : Fault Free            Node 20 : Transiently Faulty
Node 21 : Transiently Faulty    Node 22 : Intermittently Faulty
Node 23 : Transiently Faulty    Node 24 : Intermittently Faulty
Node 25 : Intermittently Faulty
```

FIGURE 5.5: Output of Proposed Algorithm for 25 nodes

```
The algorithm Classified the given nodes as:

Node 1 : Intermittently Faulty  Node 2 : Intermittently Faulty  Node 3 : Transiently Faulty
Node 4 : Transiently Faulty     Node 5 : Transiently Faulty     Node 6 : Intermittently Faulty
Node 7 : Intermittently Faulty  Node 8 : Transiently Faulty     Node 9 : Transiently Faulty
Node 10 : Intermittently Faulty Node 11 : Fault Free            Node 12 : Transiently Faulty
Node 13 : Transiently Faulty    Node 14 : Fault Free            Node 15 : Transiently Faulty
Node 16 : Transiently Faulty    Node 17 : Transiently Faulty    Node 18 : Fault Free
Node 19 : Fault Free            Node 20 : Transiently Faulty    Node 21 : Transiently Faulty
Node 22 : Transiently Faulty    Node 23 : Fault Free            Node 24 : Transiently Faulty
Node 25 : Fault Free            Node 26 : Fault Free            Node 27 : Transiently Faulty
Node 28 : Transiently Faulty    Node 29 : Transiently Faulty    Node 30 : Transiently Faulty
Node 31 : Transiently Faulty    Node 32 : Transiently Faulty    Node 33 : Fault Free
Node 34 : Transiently Faulty    Node 35 : Fault Free            Node 36 : Fault Free
Node 37 : Fault Free            Node 38 : Transiently Faulty    Node 39 : Fault Free
Node 40 : Fault Free            Node 41 : Transiently Faulty    Node 42 : Fault Free
Node 43 : Fault Free            Node 44 : Transiently Faulty    Node 45 : Fault Free
Node 46 : Fault Free            Node 47 : Transiently Faulty    Node 48 : Fault Free
Node 49 : Transiently Faulty    Node 50 : Intermittently Faulty Node 51 : Transiently Faulty
Node 52 : Fault Free            Node 53 : Fault Free            Node 54 : Fault Free
Node 55 : Fault Free            Node 56 : Fault Free            Node 57 : Transiently Faulty
Node 58 : Fault Free            Node 59 : Fault Free            Node 60 : Transiently Faulty
Node 61 : Transiently Faulty    Node 62 : Fault Free            Node 63 : Fault Free
Node 64 : Transiently Faulty    Node 65 : Fault Free            Node 66 : Fault Free
Node 67 : Fault Free            Node 68 : Fault Free            Node 69 : Transiently Faulty
Node 70 : Intermittently Faulty Node 71 : Transiently Faulty    Node 72 : Fault Free
Node 73 : Transiently Faulty    Node 74 : Fault Free            Node 75 : Transiently Faulty
Node 76 : Fault Free            Node 77 : Fault Free            Node 78 : Fault Free
Node 79 : Transiently Faulty    Node 80 : Transiently Faulty    Node 81 : Fault Free
Node 82 : Fault Free            Node 83 : Fault Free            Node 84 : Transiently Faulty
Node 85 : Fault Free            Node 86 : Transiently Faulty    Node 87 : Transiently Faulty
Node 88 : Transiently Faulty    Node 89 : Transiently Faulty    Node 90 : Fault Free
Node 91 : Transiently Faulty    Node 92 : Transiently Faulty    Node 93 : Intermittently Faulty
Node 94 : Transiently Faulty    Node 95 : Transiently Faulty    Node 96 : Transiently Faulty
Node 97 : Transiently Faulty    Node 98 : Transiently Faulty    Node 99 : Transiently Faulty
Node 100 : Transiently Faulty
```

FIGURE 5.6: Output of Proposed algorithm for 100 nodes

5.4 Results and Analysis

We compared the number of fault free, transiently faulty and intermittently faulty nodes found with the number of rounds of the algorithm run and found the following results depicted in the graph for different node degrees i.e the average number of neighbors each node has.

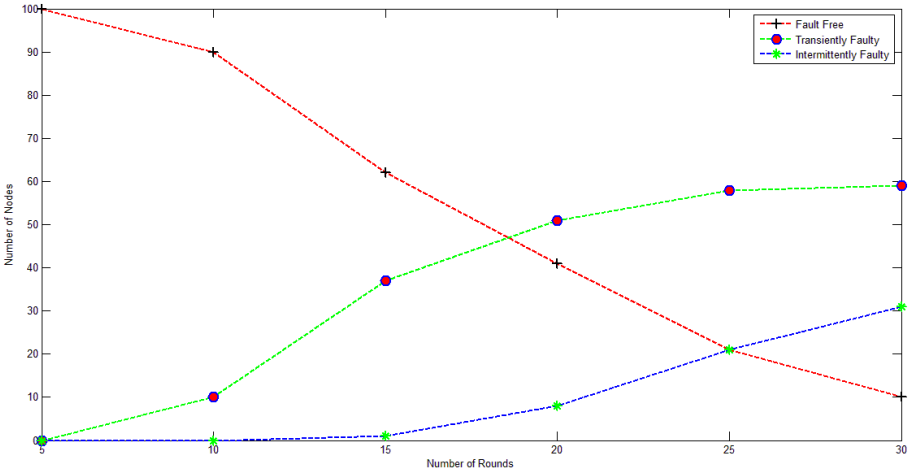

FIGURE 5.7: Number of Faulty Nodes Vs Number of Rounds (Average Node Degree 5)

It is observed from the output of our proposed algorithm that as the number of rounds of the algorithm run increases, more number of faulty nodes are detected. So the number of rounds of the algorithm run has to be considerable to have faith in the results of the algorithm. The output for simulation with increased node degree is shown below.

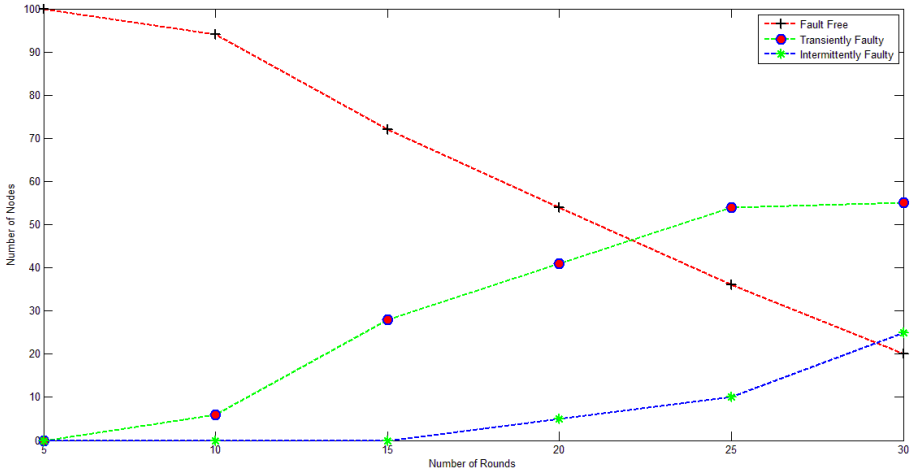

FIGURE 5.8: Number of Faulty Nodes Vs Number of Rounds (Average Node Degree 10)

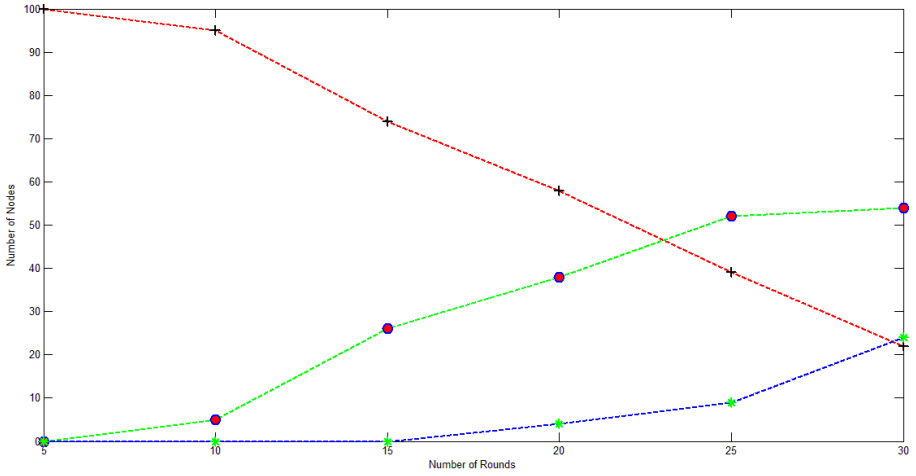

FIGURE 5.9: Number of Faulty Nodes Vs Number of Rounds (Average Node Degree 15)

It is also observed from the output of our proposed algorithm that as we increase the node degree that is the average number of neighbors of a node, the number of faulty nodes detected decrease as we are using a type of polling where even if any neighbor says the node is fault free then the node is judged to be fault free for that particular round. So we should not keep the average node degree very high for proper results.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

We proposed a distributed algorithm for detection of faults in a Wireless Sensor Network and the classification of the faulty nodes. In our algorithm the sensor node classifies itself as being fault free, transiently faulty, or intermittently faulty considering the energy differences from its neighbors. We have shown the simulation results in the form of the output messages from the nodes depicting their health and also compared the results in form of graphs for different average node degrees and different number of rounds of our algorithm run. By the simulation results and comparisons we conclude that as the number of rounds in the algorithm increases, number of faults detected increases. Also we conclude that by increasing the node degree or we can say the average number of neighbors the number of faults detected decreases. So these two factors affect the accuracy of our algorithm.

6.2 Future Works

In future we intend to optimize the algorithm by focusing upon the energy dissipated by the sensor nodes of the Wireless Sensor Network. We will include the energy dissipation models used in different applications and carry out the simulations. We also intend to carry out the simulations in a Wireless Sensor network having more number of sensor nodes and a greater node degree.

Bibliography

[1] Ann T. Tai, Kam S. Tso, and William H. Sanders. Diversity-inspired clustering for self-healing manets: Motivation, protocol, and performability evaluation. *in Proceedings of the International Conference on Dependable Systems and Networks*, pages 547–556, 2010.

[2] Winnie Louis Lee, Amitava Datta, and Rachel Cardell-Oliver. Flexitp: A flexible-schedule-based tdma protocol for fault-tolerant and energy-efficient wireless sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 19(6):851–864, 2008.

[3] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. *Proceedings of the 7th annual international conference on Mobile computing and networking.*

[4] Chih fan Hsin and Mingyan Liu. Self-monitoring of wireless sensor networks. *Computer Communications*, 29(4):462–476, 2006.

[5] Kai Xing Min Ding, Dechang Chen and Xiuzhen Cheng. Localized fault-tolerant event boundary detection in sensor networks. *INFOCOM*, 1.

[6] Shubha Kher Jinran Chen and Arun Somani. Distributed fault detection of wireless sensor networks. *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, pages 65–72, 2006.

[7] Ming Dong Xuanwen Luo and Yinlun Huang. On distributed fault-tolerant detection in wireless sensor networks. *IEEE Trans. Computers*, 55(1):58–70, 2006.

[8] Ming Dong Xuanwen Luo and Yinlun Huang. Optimal fault-tolerance event detection in wireless sensor networks. *IEEE Transactions on Computers.*

[9] Ruixin Niu, Pramod K. Varshney, and Qi Cheng. Distributed detection in a large wireless sensor network. *Eurasip Journal on Wireless Communications and Networking*, 7(4):380–394, 2006.

[10] Ruixin Niu, Pramod K. Varshney, Michael Moore, and Dale Klamer. Decision fusion in a wireless sensor network with a large number of sensors. *In Fusion*, pages 21–27, 2004.

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

[12] Pramod K.Varshney Po-Ning Chen Tsang-Yi Wang, Yunghsiang S.Han. *IEEE Journal on Selected Areas in Communications*.

[13] Rohan Bhindwale Sapon Tanachaiwiwat, Pinalkumar Dave and Ahmed Helmy. Secure locations: routing on trust and isolating compromised sensors in location-aware sensor networks. *SenSys*, pages 324–325, 2003.

[14] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. *6th International Conference on Mobile Computing and Networking*, pages 255–265, 2000.

[15] Dirk Balfanz Jessica Staddon and Glenn Durfee. Efficient tracing of failed nodes in sensor networks. *WSNA*, pages 122–130, 2002.

[16] Nithya Ramanathan, Kevin K. Chang, Rahul Kapur, Lewis Girod, Eddie Kohler, and Deborah Estrin. Sympathy for the sensor network debugger. *SenSys*, pages 255–267, 2005.

[17] M. Potkonjak F Koushanfar and A Sangiovanni Vincentell. Fault tolerance techniques for wireless ad hoc sensor networks. *Proceedings of IEEE, Sensors, 2002.*, 2:1491–1496, 2002.

[18] C. Hartung A. Sheth and R. Han. A decentralized fault diagnosis system for wireless sensor networks. *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, 2005*, pages 3 pp.–194, 2005.

[19] Leonardo B Oliveira Wong Hao Chi José Marcos S Nogueira Antonio A. F. Loureiro Linnyer Beatrys Ruiz, Isabela G Siqueira. Fault management in

event-driven wireless sensor networks. *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 149–156, 2004.

[20] Mihaela Cardei and Jie Wu. Energy-efficient coverage problems in wireless ad-hoc sensor networks. *IN ACM WORKSHOP ON WIRELESS SECURITY*, 29(4), February 2006.

[21] H. Mokhtar M. Yu and M. Merabti. A survey on fault management in wireless sensor network. *8th Annual PostGraduate Symposium on The Convergence of Telecommunications, Networking and Broadcasting*, 2007.

[22] A. Rahman S. Harte and K.M. Razeeb. Fault tolerance in sensor networks using self-diagnosing sensor nodes. *The IEE International Workshop on (Ref. No. 2005/11059) Intelligent Environments, 2005.*, pages 7–12, 2005. ISSN 0537-9989.

[23] K.Lai S. Marti, T. J. Giuli and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 255–265, 2000.

[24] Neelam Banerjee and P.M.Khilar. Distributed intermittent fault diagnosis in wireless sensor networks using clustering. *2010 First International Conference on Integrated Intelligent Computing*, pages 264–269, 2010.

[25] M. Potkonjak F. Koushanfar and A. Sangiovanni-Vincentelli. Fault tolerance techniques in wireless ad-hoc sensor networks. *UC Berkeley technical reports*, 2002.

[26] Pabitra Mohan Khilar and S.Mahapatra. Intermittent fault diagnosis in wireless sensor networks. *10th International Conference on Information Technology*, pages 145–147, 2007.

[27] M. Potkonjak F. Koushanfar and A. Sangiovanni-Vincentelli. Fault tolerance techniques in wireless sensor networks. *Handbook of Sensor Networks*, 2004.

[28] A. Dahbura M. Barborak and M. Malek. Wireless sensor network-management system, an adaptive policy-based management for wireless sensor networks. *ACM Computing Surveys*, 25(2), 2010.