A Novel Fault Diagnosis Technique in Wireless Sensor Networks

Anas Abu Taleb, J. Mathew and D.K. Pradhan
Department of Computer Science
University of Bristol
Bristol, UK
{abutaleb, jimson, pardhan}@cs.bris.ac.uk

Taskin Kocak
Department of Computer Engineering
Bahcesehir University
Istanbul, Turkey
taskin.kocak@bahcesehir.edu.tr

***Abstract*—In sensor networks, performance and reliability depend on the fault tolerance scheme used in the system. With increased network size traditional fault tolerant techniques have proven inadequate. Further, identifying and isolating the fault is one of the key steps towards reliable network design. Towards this, we propose two new algorithms to detect and substitute faulty nodes at different levels in the network. In the proposed approach, the network is divided into zones which are having a master for each zone. Moreover, the masters of the zones are connected in a De Bruijn graph based network. When a fault occurs, the masters are checked, tested. After that, the sensor nodes in the suspected zone are tested. Our fault model assumes communication, processing and sensing faults caused by hardware failures in a node. We analyzed the performance of the first algorithm according to the number of messages it needs to diagnose faulty nodes. In addition, the performance of a 4-node De Bruijn graph was also studied by measuring the end-to-end delay. Finally, the performance of the second algorithm was studied by measuring the fault detection accuracy.**

Keywords- ***Wireless Sensor Networks, Fault Tolerance, Fault Diagnosis, De Bruijin Graph***

I. INTRODUCTION

The advances in wireless communication and electronics made it possible to develop low-cost sensor nodes, which can be deployed easily in specific areas in order to accomplish a specific mission by forming a wireless sensor network (WSN). It might be difficult or dangerous for humans to enter these areas because nodes in this type of networks are expected to operate in inhospitable environments [2]. Therefore, sensor nodes are expected to operate for periods ranging from days to years without any human intervention. There is a tremendous need for fault tolerant WSNs because, sensor nodes are subject to various types of failures and faults such as communication, processing and sensing faults.

A sensor network must be capable of identifying and replacing the faulty nodes in order to make sure that the network's quality-of-service (QoS) is maintained. Identifying faulty sensor nodes is not an easy task as it is difficult and time consuming for the base station to keep the information about all the sensor nodes in the network. When addressing fault tolerance in WSNs, three types of node failures must be taken into account. First, when the sensor node is faulty and not providing data. Second, when the node processes data erroneously. Third, occurs when we have an active node that is providing incorrect data.

In this paper, we propose a new technique consisting of two algorithms to identify faults occurring at different levels or places in the network, i.e. faults that occur at the zones masters and the sensor nodes associated with the zones masters. The proposed technique divides the network into disjoint zones while having a master for each zone. When a fault occurs, the first algorithm is triggered to test the masters. The technique will not trigger the second algorithm unless all the masters are diagnosed fault free by the first algorithm. Thus, when the second algorithm is triggered, the master of the suspected zone is responsible for identifying the suspected faulty nodes. As a result, the master will start searching for sleeping nodes to wake up and depending on the reading it gets from the suspected and awakened nodes, the master can decide whether the suspected nodes are faulty or not, moreover, it can decide on which node to switch off. A preliminary version of this paper is published in [1], in which a technique to detect faulty sensor nodes was presented.

The paper is organized as follows. In section II the related work is reviewed. Then the concept of De Bruijn graph is discussed and explained in section III. In section IV, the network architecture and the fault model are defined. In section V the proposed technique is described. Section VI describes the simulator used and illustrates the simulated scenarios. Also, we use an example of a potential chemical spill to describe various concepts. The simulation results are also reported in this section. Finally the paper is concluded in section VII.

II. RELATED WORK

Several works have addressed the problem of how to deal with faults occurring in wireless sensor networks in order to achieve fault tolerance [3][4][5]. These researches consider the faults that result from sensor nodes failures, which affect the network connectivity and coverage. The research proposed in [3], makes use of redundancy and uses a technique to decide on which nodes to keep active and on which to put in a sleep mode. The technique aims to provide the sensor field with the best possible coverage. In addition, it maintains network connectivity to route information. When an active node fails it is substituted by one of the sleeping nodes. However, other researchers have addressed the problem of having active nodes that provide incorrect data which results in making inappropriate decisions. The research proposed in [4] focused on such issues and proposed a mechanism to detect and diagnose data in consistency failures in wireless sensor networks.

The mechanism proposed in [4] uses two disjoint paths to send the sensed data to a static sink. After the sink receives both copies, it will compare them to check if they match. If the two copies match, both the data and the paths are considered to be fault free otherwise, a third disjoint path will be established. Then, the sensor node will send three copies on the three disjoint paths to the sink. The sink will compare these copies and decides on the faulty path. Finally, a diagnosis routine will be executed to identify the faulty node within the faulty path.

Another research has taken fault tolerance in account, so that to achieve fault tolerance the sensor network is partitioned into distinct clusters and the node that has the highest energy level is selected to be the cluster head where only cluster heads are allowed to communicate with the base station [5]. Therefore, they introduced a two-phase fault tolerant approach which consists of detection and recovery where the status of the cluster heads is checked periodically. Sensors associated with a faulty cluster head are recovered by joining them to another cluster [5].

The research described in [6] proposed a scheme based on multi-path routing combined with channel coding to achieve fault tolerance. It uses a fuzzy logic based algorithm that is energy and mobility aware to select multiple paths. When selecting the paths, the algorithm takes the remaining energy, mobility and the distance to the destination into account. Another research has proposed a design for a system to diagnose the roots of faults occurring in wireless sensor networks. The authors have proposed an algorithm to diagnose the cause of faults in which the behavior of sensor nodes in monitored locally. The diagnosis procedure will be triggered when a node detects a strange behavior [7]. In [8], a general framework to achieve fault tolerance in wireless sensor networks was proposed. The framework is based on a learning and refinement module which provides adaptive and self-configurable solutions.

A localized algorithm for fault detection to identify faulty sensors that is based on having neighbor sensor nodes testing each other was proposed in [9]. In [10], an efficient algorithm to trace failed nodes in sensor networks was proposed. In addition, they demonstrate that if the network topology is conveyed efficiently to the base station, it allows tracing the failed entities quickly with moderate communication overhead.

In [11], the authors proposed fault tolerant algorithms to detect the region of an event in wireless sensor networks. Also, they assume that nodes report a binary decision to indicate the presence of an event or not and considered a byzantine behavior for the faulty nodes, which means that the faulty nodes will be providing arbitrary values. Hence, they proposed a randomized decision scheme and a threshold decision scheme which a sensor node can use to decide on which binary decision to send by comparing the decision it has with the decisions of its neighbors

In [12], a fault map was constructed using a fault estimation model. In order to build the fault map, sensor nodes are required to send additional information that can be used by the fault estimation model. Furthermore, a cluster based algorithm to estimate faults in wireless sensor networks was proposed. In [13], a target detection model for sensor networks was proposed. In addition, two algorithms to facilitate fault tolerant decision making were presented. The first algorithm is based on collecting the actual readings from the neighboring nodes. In the second algorithm, the sensor node obtains the decisions made by the other neighboring nodes to take a final decision.

A distributed cluster based fault tolerant algorithm was proposed in [14]. The cluster head sends a small packet to indicate that it is still alive. Hence, a sensor node in the same cluster listens to the transmissions of its neighbors and to that of the cluster head. When a sensor node does not receive the short packets sent by the cluster head, it will trigger fault detection. Depending on the number of nodes that have not heard from the cluster head, it can be decided whether the cluster head is faulty, as the faulty node can be a member of the cluster and not the cluster head itself. If the cluster head was faulty, the cluster members will select a new cluster head. The authors in [15] apply error correcting codes to achieve fault tolerance. As a result, a distributed fault tolerant classification approach was proposed. The approach proposed is base of fault tolerant fusion rules that are used to obtain local decision rules at every sensor. In addition, the authors proposed two algorithms that can be used to find good code matrices to be used by the classification approach.

The work proposed in this paper differs from that presented by other researches in two aspects. First, the mechanism according to which sleeping nodes are activated to test active node. Second, the reading of neighboring nodes i.e. nodes covering the same terrain, are needed and compared only when the network is suspected to contain faulty nodes.

Moreover, we compare the performance of our work to the performance of the work presented in [11] because both techniques make use of neighboring nodes to detect a fault. In addition, no restriction on the number of neighboring nodes is imposed. Also, both techniques make use of threshold in their operation.

III. De Bruijn graph

Part of the work proposed in this paper is based on constructing a De Bruijn graph based network at the zones masters level. This graph has interesting properties that make it important to investigate its use in WSNs. The degree of this graph is bounded, which means the degree of the network remains fixed even when the network size increases. In addition, this graph has interesting properties such as small diameter, high connectivity and easy routing. Furthermore, De Bruijn graph contains some important networks such as ring. Regarding fault tolerance and extensibility, these graphs maintain a good level of fault tolerance and self-diagnosability. For instance, in the presence of a single fault in the network, it takes four additional hops to detour around the faulty node and the control information needed to do so can be integrated locally between the faulty node's neighbors. Also, De Bruijn graph is extensible in two methods that are described in [18].

As a result, it will be interesting to investigate the used of De Bruijn graphs in sensor networks in order to increase the fault tolerance capabilities. In other words, if some nodes in the network were deployed according to De Bruijn graph, the network will have the ability to tolerate the presence of faulty nodes in the network and remain functional. In this work, the zone masters are assumed to be connected according a De Bruijn graph. The network assumed to be working if a zone master fails and the rest of the nodes in network will remain functional and the fault free zone master are capable of communicating with each other. Thus, accomplishing the network mission until the problem in the network is resolved.

The De Bruijn graph denoted as DB(r, k) has $N = r^k$ nodes with diameter k and degree 2r. This corresponds to the state graph of a shift register of length k using r-ary digits. A shift register changes a state by shifting in a digit in the state number in one side, and then shifting out one digit from the other side. If we represent a node by $I = (i_{K-1} i_{K-2}, \ldots, i_1 i_0)$ where $i \in 0,1,\ldots,(r-1)$, $0 \le j \le (k-1)$, then its neighbors are represented by $i_{k-2} i_{k-3}, \ldots, i_0 p$ and $p i_{k-1} i_{k-2}, \ldots, i_1$, where $p = 0,1,\ldots,(r-1)$. The DB(2, k), which is called binary De Bruijn graph, can be obtained as follows. If we represent a node I by a k-bit binary number, say, $I = i_{k-1} i_{k-2}, \ldots, i_1 i_0$, then its neighbors can be presented as $i_{k-2}, \ldots, i_1 i_0 0$, $i_{k-2}, \ldots, i_1 i_0 1$, $0 i_{k-1} i_{k-2}, \ldots, i_1$, and $1\, i_{k-1} i_{k-2}, \ldots, i_0$.

IV. NETWORK ARCHITECTURE AND FAULT MODEL

We assume that the network is densely deployed and consists of heterogeneous nodes; which means that in addition to the ordinary sensor nodes, the network consists of some nodes that are more energy rich than others. The energy rich nodes are placed or deployed in a way that guarantees them to form a De Bruijn based network, while the rest of the nodes are deployed randomly. Also, the network has most of the nodes awake and a small number of nodes are in a sleep status. The nodes are fully static and the network is divided into four zones. In each zone the active node with the highest energy level will be chosen to be the zone master, for example in the shaded zone in Fig.1 the zone master is node 9, where the dark dots are the active nodes. After that, the master acts as a data sink and will be responsible for identifying faulty nodes in its zone while the remaining nodes in a zone can only send the sensed data to their master.

After being elected as zones maters for their zones, the zone masters communicate among themselves. In other words, each zone master knows the neighboring zone masters in the neighboring zones. Hence, a De Bruijn graph based network, consisting of the zone masters only, is constructed. This graph has interesting properties that assist in increasing fault tolerance capabilities of the network. Figure 2 shows the DB(2,2) De Bruijn Graph.

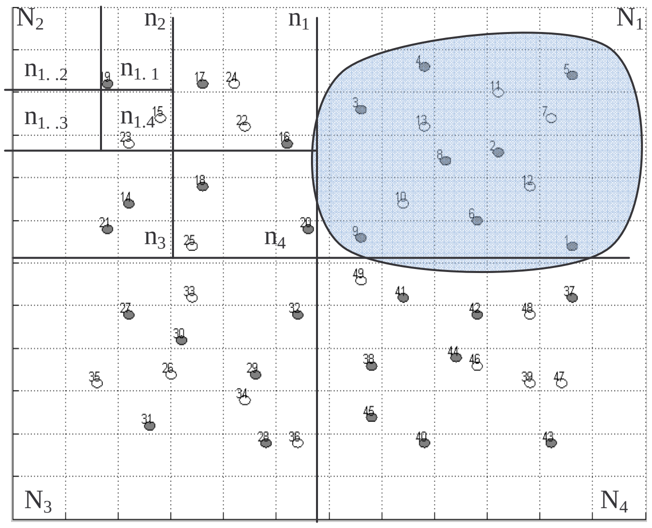

Figure 1. The Network Architecture.

After constructing the De Bruijn based network, when a zone is suspected to contain faulty nodes, the master of that zone will be tested and diagnosed using the distributed fault diagnosis algorithm described in section V. If the zone master is faulty, it will be substituted by one of its neighboring sleeping nodes, as a result, the De Bruijn based network will not need to be constructed again. On the other hand, if the master was fault free, the technique proceeds to test the nodes in that zone, as described before, until the faulty node is identified. When a sensor node is suspected to be faulty, the master activates some of the sleeping nodes to check the correctness of that node and to substitute it when the suspected node is identified as faulty. Figure 1 illustrates the network architecture where the main zones are the big squares denoted by N_1, N_2, N_3 and N_4. Also, the division process is illustrated in Fig.1 where N_2 is divided into four subzones denoted by n_1, n_2, n_3 and n_4. Furthermore, n_2 is divided into $n_{2.1}$, $n_{2.2}$, $n_{2.3}$ and $n_{2.4}$, thus node 19 is suspected to be faulty which is in $n_{1.1}$.

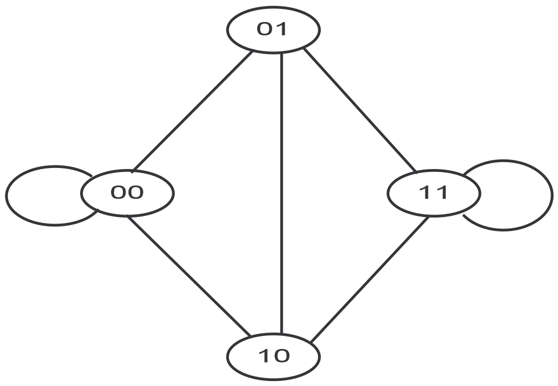

Figure 2. DB(2,2) Binary De Bruijn Graph.

A sensor node is considered to be faulty, if it reports a value that deviates from the expected one [16]. In this work, we restrict our attention to permanent faults, resulting from sensing and communication faults caused by hardware failures are considered at the sensor nodes level. At the zone master level, permanent faults caused by communication and processing fault are considered. Processing faults are taken into account at the zone master level because, the zone masters have to calculate their zones throughput periodically. As a result, if a master is suffering from processing failure the calculation acquired will be misleading.

For communication faults, we propose to measure the throughput (T) of a zone or a subzone and compare the calculated value of throughput to predefined thresholds of the tested zone (τ) or subzone (τ_{sub}). As a result, the presence of a communication fault is detected if $T < \tau_{sub}$. On the other hand, we detect the presence of a sensing fault by measuring the discrepancy between the readings reported by the sensor nodes involved in the test. Discrepancy between sensors readings is used because we consider that the sensor nodes report the actual values rather than binary decisions [17].

V. PROPOSED IDENTIFICATION TECHNIQUE

Because WSNs are deployed in inhospitable environments, sensor nodes are prone to faults such as communication, sensing and processing faults. As a result, the node that suffers from a communication fault will not be reporting data to its zone master in the same rate as a non-faulty one does. This will result in the throughput of that zone to decrease. On the other hand, nodes that suffer from a sensing fault will be providing data frequently to the zone master, but the data reported will be erroneous. Also, the nodes may suffer from processing faults at the zone master level. Hence, the aggregated or fused data at the faulty master will be erroneous and affects the decision made to detect the faulty nodes. Thus, there is a need to detect these faults and eliminate their effect.

Therefore, we consider dividing the network into zones, to allow faster identification and location of faults occurring in a zone since the master is responsible for a small number of nodes. In addition, the master can keep track of the data sent to it by the members of its zone more efficiently. In addition, the approach starts by testing the master nodes in the first stage to avoid testing individual nodes in the zones when the master is faulty.

A. Overview of The Proposed Approach

The technique is based on periodically calculating the throughput of the four zones. Each zone master will calculate the throughput of its zone and will compare it to a predefined threshold; if it is less than the threshold, the distributed diagnosis algorithm will be triggered to test the masters. If a zone master was diagnosed as faulty, it will be replaced and the technique will not proceed to test the sensor nodes in that zone. However, if all the masters were diagnosed as fault free, the technique proceeds to test the sensor nodes in the zone that has provided low throughput.

As a result, the zone master will start dividing its zone virtually into quadrants. After that, the zone master will calculate the throughput of each quadrant and will compare it to another threshold. If the throughput of one of the quadrants is less than a threshold, the zone master will divide that quadrant for another four quadrants. The zone master will keep dividing the zone virtually and calculating the throughput until it reaches a quadrant that contains only one node. As a result, it can identify that the node enclosed in that quadrant is the suspect that is causing the throughput to be low. After identifying the suspect node, the zone master will start searching for sleeping nodes that are near to the suspect to wake them up to test the suspect node. Note that a zone or a subzone is divided by calculating its center, after that, it will be divided into four subzones that have equal size.

In addition, when a sensor node reports data to the zone master, the data will be compared to the node status and the data ranges values stored in the master. If the data reported deviates from the stored values, the master will start dividing the zone virtually until a suspect is identified. After that, it will start searching for neighboring sleeping nodes to activate in order to test the suspect.

The proposed technique has the following distinctive feature; first the distributed fault diagnosis algorithm that is used to diagnose faults at the zones masters level does not use a central node to trigger and carry out the diagnosis process. The second feature is the way in which faulty nodes associated with the zone master are identified or pinpointed by dividing the zone into quadrants. The third feature is the mechanism used to make sure that the suspect node is faulty which is conceptually similar to our previous work on the multi-processor environment [19]. In the Roll-forward Checkpointing Scheme, two copies of the same task will be run on two different processing modules while having a pool of spare processing modules. At every checkpoint, the state of the two processing modules is compared, if they mismatch, the state of the last checkpoint on which the state of the two processing modules has matched will be loaded into a spare processing module, while the other two processing modules continue the execution of the task beyond the checkpoint where a mismatch occurred. At the next checkpoint, the state of the spare processing module will be compared to the stored state of the other two processing modules. As a result, the processing module whose state disagrees with that of the spare will be the faulty one. After identifying the faulty processing module, the state of the non faulty processing module is copied to the faulty one to restore its state [19]. A similar scheme was applied in this work by activating one of the sleeping neighbors of the suspected node. Both nodes will sense their region simultaneously. After receiving the data, the sink compares the data sent by both nodes. If they match or were similar, the suspect node will be considered fault free and the activated neighbor goes to sleeping mode again. Otherwise, another sleeping neighbor is activated, and after the three nodes sense their region and send data to the sink, the sink can identify the faulty node using the mechanism mentioned above. If the faulty node was the originally active one, it is deactivated and one of the activated neighbors is selected to substitute it. On the other hand, if the faulty node was the first activated neighbor, it is flagged as faulty and is suspend from the network.

B. Design and Implementation

The technique is divided into the following phases:

1) Initialization Phase

The nodes are grouped into four zones depending on their positions and the active node with the highest energy level in each zone is chosen as the zone master. The zone master keeps track of the data sent to it from the other nodes in its zone. Also, it acts as a data sink for the nodes. This means that the ordinary nodes in the zone can only send data to the zone master which is responsible to forward it to the base station. After that, the zones masters will communicate with each other. Hence, each zone master knows its neighboring zone masters. As a result, a De Bruijn graph based network consisting of the masters only is constructed.

The zone master will be able to keep track of the data it received and of the nodes belonging to its zone by maintaining an information table and a registration table.

In the information table the zone master records the sender's ID of the received message, the packet length and the time stamp to indicate when the message was received. As a result, this table gives the master the ability to keep track of the data sent in its zone. An example is given in Table 1.

Table 1. Information table

NodeID	Message_length	Time_Stamp
8	4	0.501
1	4	0.504
6	4	1.002
2	4	1.003

The registration table is used by the master to keep track of the nodes inside its zone and their positions. In addition, it contains some entries that will give the master node the ability to divide the zone into quadrants or subzones when needed. In other words, XMax and YMax entries are used to know the coordinates of the zone. In addition, they are used by the master when there is a need to divide the zone into subzone. The Center attribute is calculated because it is used as a reference point when dividing a zone or a subzone. An example is given in Table 2.

In order to be able to detect the presence of a sensing fault, a third table, which is called *Grid table*, is maintained by each zone master. The zone master divides its zone virtually to a grid and stores the information in the grid table. Note that three binary values are used to indicate the status of a node because a node reports the value that corresponds to its original reading which depends on the node's proximity from an event. An example is given in Table 3.

Table 2. Registration Table

NodeID	XPosition	YPosition	XMax	YMax	Center
1	23	5	30	30	15
2	16	13	30	30	15
10	7	7	30	30	15
11	16	17	30	30	15

Table 3. Grid Table

Square-Number	Enclosed_nodes	Low	Medium	High
1	[9, 10]	0	0	1
4	3	0	1	0
7	[4, 13]	1	0	0
9	5	1	0	0

In the initialization phase, all the nodes will be providing low data values to indicate that no event was detected. After that, when an event occurs the value can be changed to medium or high based on the position of the node. A sensor node has three different values to choose from when it is about to send data to the master which are low, medium and high. For example, if the sensor node is in a place where there is a very high concentration of a chemical spill, it will send the value stored in the high field of the grid table to indicate that there is a high chemical spill in its region. As a result, the other nodes will choose to send low, medium or high data values depending on their positions and distance from that node.

In Fig.3, the pseudo code used in the initialization phase is illustrated. It can be observed that after the nodes are deployed, the network is divided into four zones and the nodes are allocated to the zones as mentioned before. In addition, each zone master will initialize its registration table, mentioned above, and will store the needed information about the nodes belonging to its zone. Finally, the grid table will be initialized, i.e. the zone will be virtually divided into a grid, and the node or nodes belonging to every square in the grid are identified.

```
For each zone
    Find the node with the highest energy level to be the
     zone_master
    Initialize Registration table for zone_master
   Set the ID of the new entry to the ID of the current node
   Get node position
```

```
    Calculate zone center
    Update zone registration table
    Initialize Grid table for zone_master
End Loop
For each zone Grid table
  Locate nodes to the grid squares according to their positions
End Loop
 For each zone_master
   Find the neighboring zone_masters
End Loop
```

Figure 3. Initialization Pseudo Code.

2) Failure Detection Phase

The zone master is responsible for checking the throughput of its zone periodically. This is done by calculating the throughput of the zone since the last time the throughput was checked until the current time. This process can be described as taking a snapshot of the information table depending on the specified period of time. Subsequently, the master compares the calculated value of the throughput to a threshold "τ". If the value of the calculated throughput is greater than τ, the master concludes that there is no communication fault in the zone. However, if the value of the calculated throughput is less than τ, the master assumes that there is a communication fault in the zone and initiates the failure detection phase.

The failure detection phase starts by testing the masters first because, the zone master might be suffering from a processing fault. The distributed De Bruijn based fault diagnosis algorithm is used to test the masters. The number of nodes in a De Bruijn based network is assumed to be equal to r^m, where r is a parameter that bounds the number of faults that can be diagnosed in each cluster and will be referred to as *base* parameter in this paper. The variable *m* is the radix-r representation of the node address e.g. $y_{m-1}y_{m-2}, ..., y_0$ is the radix-r representation of node y. Also, the number of faults that can be diagnosed is equal to r -1 [20]. In addition, we assume that nodes can test their neighbors only.

The algorithm is based on building directed tree structure for the De Bruijn based network. According to our previous work in [20], r different tree structures can be built where each one of them has a different root. In this paper, the base variable *r* is equal to 2 which mean we can diagnose only one fault and we can build two tree structures for the De Bruijn based network. Figures 4 and 5 illustrate two trees that can be built for a 4-node De Bruijn based network.

Consider Fig. 4, the following conditions are satisfied:

- The test tree must contain all the nodes in the cluster.
- The number of non leaf nodes is equal to r^{m-1}.
- The number of leaf nodes is $(r-1)r^{m-1}$.
- Any combination of r – 1 nodes must appear in at least one tree.

The algorithm is triggered at the zones masters level. As a result the first tree is built to test the zones masters. The test tree is traversed in an inorder fashion. According to Fig. 4 the root node, 0, initiates the process by sending a test packet to node 2. Then, node 2 checks if it is a leaf node. In this case, node 2 in a non leaf node, thus a test packet will be sent to its left child, node 1. This process continues until we reach a leaf node. When a leaf node, for example node 1, receives a test packet, it will execute the required computation for the test and send the result back to its parent, node 2. Node 2 compares the result received from node 1 with the expected or the predefined one. If a miss match occurs node 1 will be considered faulty and its status will be reported back to the root node that is responsible for sending it to the base station.

Note that, the algorithm will stop after finding the faulty node. Also, the faulty node can be detected only if it is a leaf node in the test tree shown in Fig. 4. However, if the faulty node is a non leaf node in the first tree, the algorithm cannot diagnose whether the non leaf node is faulty or there is a communication problem between that node and one of its children. As a result, when a non leaf node is suspected to be faulty, the algorithm will stop searching the tree shown in Fig. 4 and will construct the second test tree shown in Fig. 5. After constructing the second tree, the test packets will be passed in the same manner as mentioned before. The faulty node can be detected because; it is a leaf node in the second tree. After diagnosing the nodes at one level, the algorithm proceeds to test the nodes in the subsequent level.

The test packet sent to diagnose the nodes triggers the tested node to perform a specific computation whose result is known in advance. Therefore, if the tested node provides a value that deviates from the expected one it will be diagnosed as faulty.

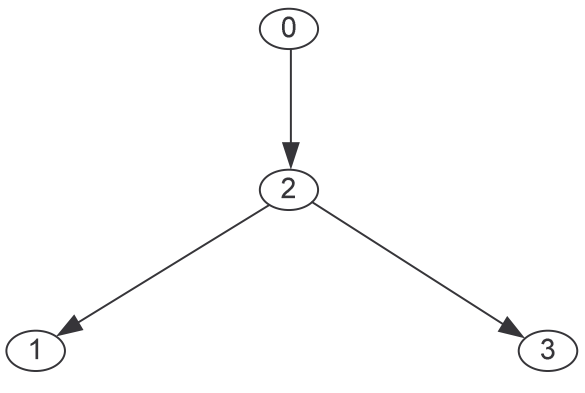

Figure 4.Diagnosis algorithm Tree A.

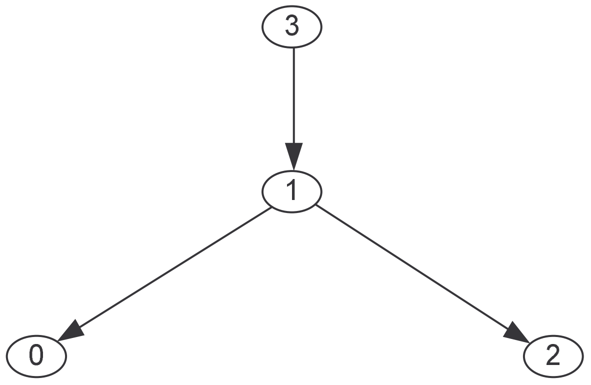

Figure 5.Diagnosis algorithm tree B.

If all master nodes were diagnosed fault free, our approach proceeds to test the sensor nodes associated with the zone master that has calculated low throughput.

Based on the information maintained in the registration table, the master, that has calculated a low value of throughput, starts dividing its zone into quadrants. In the first stage the zone master divides its zone into four quadrants or subzones. After that, it will calculate the throughput of each of the quadrants based on the same snapshot taken before. Furthermore, the throughput of each of the quadrants will be compared to another threshold "τ_{sub}"; if the calculated value of a quadrant's throughput is less than τ_{sub}, the quadrant will be further divided into another four quadrants because it is the most likely quadrant to contain the faulty node. The zone master will keep repeating the division process until it reaches to a quadrant that only contains a single node. Depending on the calculated throughput and the comparison with thresholds, the node causing the throughput to be lower than the threshold is identified and is considered as a suspect node that suffers from a communication fault.

In order to decide whether the suspect node is faulty, a technique that makes use of the redundancy in sensor networks is applied. In other words, based on the information stored in the registration table, the zone master will start searching for the nearest sleeping node to the suspect. After identifying such a node the master will wake it up so that it can start sensing. After a period of time the master will calculate the throughput of the suspect node and the node it woke up based on a new snapshot of the information table. If the difference between the two values of the throughput is larger than a threshold "Δ" and the throughput of the suspect is less than that of the awakened node, another sleeping node will be awaken in order to be able to decide whether the suspect is faulty or the awakened node i.e. having a third node sensing in the same area will help to solve the conflict.

After activating two nodes, which are the nearest to the suspect node, the suspect node and the other two nodes will start sensing. After a period of time the throughput of the three nodes will be calculated and compared to Δ, if the values of the throughput of the awakened nodes are similar and their differences with the value of the suspect node is large, the master can decide that the node that was suspected to be faulty is suffering form a communication fault and will be switched off and one of the awakened nodes that is nearer to the faulty node will be kept awake and the other one will go back to sleep.

The presence of a sensing fault is detected by comparing the data received from a node to its entry in the grid table. If the data reported is within the correct range and the node has a correct or a matching status, it will be considered fault free otherwise, the master will start dividing its zone virtually until it finds the suspected node, after that the same technique that was mentioned above to test the suspect node by waking sleeping nodes up is used.

```
    For each zone
       Initialize Information table for zone_master
    End Loop
    For Each zone_master
       If message destination = zone_master
          Update Information table of the zone_master
       End If
    End Loop
    Set period to 2
    Set time to the result of dividing current time by period
    Set threshold to 100
      Set decrement to 10
      Set i to 1 // this variable is used to control the access of the
       subzone array and make the process recursive
       If time = 0
          Calculate throughput for each zone until current time
          Set subthreshold to 50
          Set new_threshold to subthreshold
           For each zone
             If zone throughput < threshold
              Divide zone
              For each subzone
                 If zone throughput < threshold
                   Divide zone
                  For each subzone
                    If the number of nodes in the subzone >1
                    Calculate throughput of the subzone
                     If subzone throughput < subthreshold
                       Divide the subzone
                       Get division_array
// array where the subzones arrays are stored
                         While i <= 4
                               get number of nodes in subzone(i)
// the first subzone array in the division_array
                               Calculate subthroughput of subzone(i)
                               If (number of nodes > 1 and
                                (subthroughput < new_threshold – decrement))
                                      divide subzone
                                      set divison_array to new_array
 // replace the old division_array with a new division  array resulting
 //from the new division
                                      If new_threshold > 10
                                        Set new_threshold to new_thresold-
                                         decrement
```

```
                              End If
                            Else
                             If number of nodes in the subzone =1
                               Set suspect_node to node ID in the
                                   subzone
                                Find a neighboring sleeping node to
                                    wake up
                                Increment i by 1
                              End If
                           End If
                        End Loop
              End If
            Else
               If number of nodes in the subzone = 1
                   Set suspect_node to node ID in the subzone
                   Find a neighboring sleeping node to wake up
               End If
            End If
       End Loop
     End If
   End Loop
End If
Get current time
Set time to the result of dividing current time by period
If time = 0
  Calculate throughput of the three nodes until current time
  Compare the values and find the faulty node
End If
```

Figure 6. Communication Fault Identification Pseudo Code.

The pseudo code in Fig.6 illustrates how a faulty node is identified. Note that, the throughput is calculated according to equation (1).

$$T = (N * L) / P \qquad (1)$$

where T is the throughput to be calculated, N is the number of messages received by the master, L is the message length and P is the time period on which throughput is calculated.

```
For each entry in the grid table
  Find node ID that is equal to the message source
  If (status = high and data = high) or (status = medium and data =
  medium) or (status = low and data = low)
       Set suspect to 0
  Else
       Divide zone
       While i <= 4
           get number of nodes in subzone(i)
            If (number of nodes > 1
               Divide subzone
               Set divison_array to new_array
            Else
                Set suspect to nodeID
                Find a neighboring active node
                Incremenr i by 1
            End If
```

```
        End Loop
    End If
  End Loop
  Set sensing_threshold to 5
  Get current time
  Set time to the result of dividing current time by period
  If time = 0
     Get data provided by both nodes
     Compare the data of both nodes
     If difference in readings > sensing threshold
         Find a neighboring sleeping node and wake it up
     End If
  End If
  Get current time
  Set time to the result of dividing current time by period
  If time = 0
   Get data provided by the three nodes
   Compare the data of the three nodes and find the faulty node
  End If
```

Figure 7. Sensing Fault Identification Pseudo Code.

In Fig.7, the variable *status* is used to check that the data reported by the node is correct according to its status, while the variable *data* is used to check if the data reported is correct and is actually within the expected range according to the nodes status and the ranges stored in the zone master.

```
Calculate zone_center
For each node in the zone_master registration table
   Compare node position to the center
   Allocate node to a subzone according to its position
End loop
```

Figure 8. Division Procedure.

In Fig 8, the code that is used to divide the zone into subzones is illustrated. A zone is divided into subzones by calculating the zone center as the x and y axis values of the main zone which are stored in the registration table and according to the node position and the value of the zone center. Therefore, the node will be allocated to an array that represents each subzone. Note that the ID of the zone master will be known to this procedure from the code in Fig.3.

VI. SIMULATION

A. The Simulator

The simulator used to conduct the experiments is TrueTime 1.5 which is MATLAB/Simulink based. Its main feature is that it gives its users the ability to co-simulate the interaction between the continuous dynamics of the real world and the architecture of the computer [21], [22].

B. Simulation Scenarios

1) Faults Occurring At The Zone Master Level Only

In this scenario, faults were injected at the zone master level, i.e. sensor nodes associated with a faulty zone master were fault free and the zone master was suffering from processing fault, as a result throughput was calculated erroneously. Note that, only one fault was injected at the masters level as the algorithm can detect one fault only. In this scenario, the faulty master was in different level in the diagnosis tree; in one case it was a leaf node in tree A. In another case, it was a non leaf node in tree A. Thus, tree B was built in order to detect the fault in such case.

This scenario is proposed to show the ability of the algorithm to detect faults at the zone masters level occurring at different levels in the diagnosis tree.

2) *Faults Occurring In One Zone Only*

In this scenario, all the active nodes in the network will be providing data to their masters. However, only nodes belonging to one zone will be suffering from faults as a result, the technique to identify and locate faulty nodes will be initiated in that zone only. In addition, the faulty nodes in that zone will be in different positions within the zone, which means that when the division process is started the faulty nodes will be in different quadrants or subzone and each quadrant may contain more than one faulty node. Also, a chemical spill will occur and affect nodes in this zone only.

This scenario was proposed to show the ability our technique to divide more than one quadrant into different levels until the faulty node is identified and replaced. Figure 9 illustrates the described scenario.

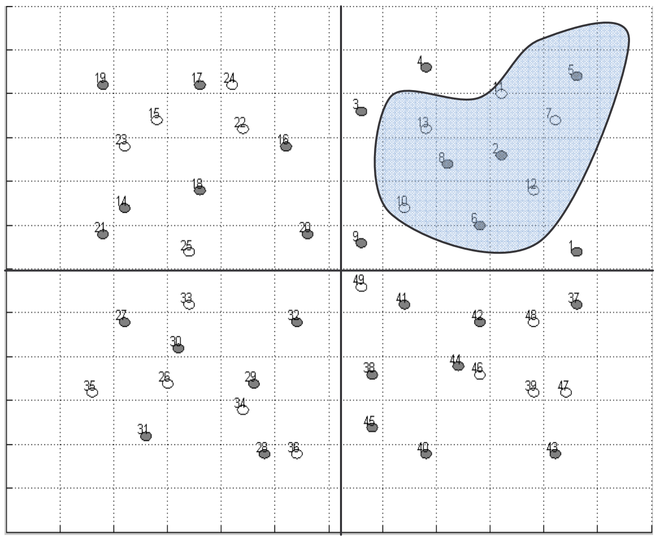

Figure 9. The Second Scenario.

Similar to the second scenario, all active nodes in the network will be providing data to their zone master but, the nodes that belong to two different zones will be faulty. The chemical spill will occur and will affect nodes in both zones.

This scenario is created to show the ability of our technique to locate and identify faulty nodes in different zones synchronously. The faulty nodes in this scenario might not be in the four quadrants of each zone when the division starts. In other words, after the zone is divided into quadrants, some quadrant may provide values of the throughput higher than the threshold mentioned in section V., while other will have throughput value lower than that threshold which indicates that there is a problem in that quadrant. Figure 10 illustrates the scenario.

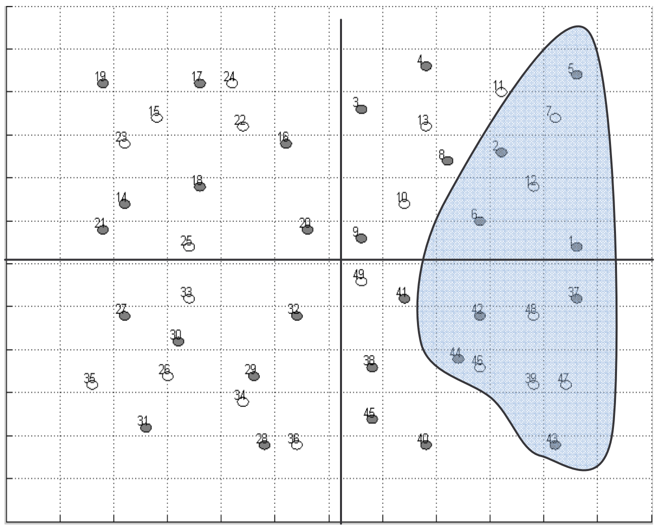

Figure 10. The Third Scenario.

C. *Simulation Results*

The scenarios studied were based on a network consisting of 50 nodes deployed randomly in 60x60 units region. The performance of the two fault diagnosis algorithms described in this paper was not compared because, they work at different levels in the network.

The distributed fault diagnosis algorithm used to detect fault zone masters is evaluated according to the number of messages required to detect the faulty master. Table 4 shows the number of messages required by the first algorithm depending on the level at which the faulty node occurs in the diagnosis tree.

Table 4. Number of Messages for the First Algorithm.

Case	Number of Messages
1	4
2	6
3	10
4	12

Cases 1 and 2 in table 4 represent the cases where the faulty nodes were the leaf nodes in the first test tree, while the remaining two cases are gained when we have to build the second test tree. It can be observed that the distributed diagnosis algorithm requires a small number of messages to be exchanged between the masters; because these master nodes are more rich in energy they can afford to send a small number of messages to accomplish the diagnosis process.

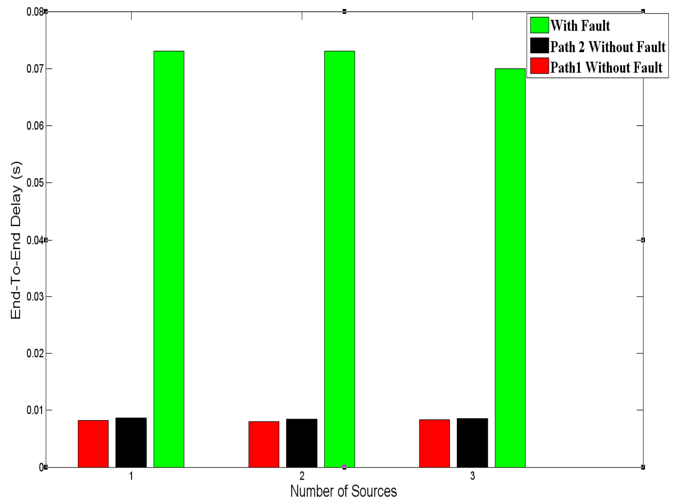

Figure 11.Comparison of the Performance of De Bruijn Network.

Figure 11 shows the performance, in terms of average end-to-end delay for 4-node De Bruijn based network. It can be observed that both paths have similar delay under fault free conditions. However, when a fault was injected, the nodes in the network had to switch between path 1 and path 2 to detour around the faulty node, which caused the average end-to-end delay to increase. Not that, these results were gained by selecting random source and random destinations and the result for each case was obtained by averaging the results of 10 runs.

In the simulation, sensor nodes were faulty nodes were randomly chosen and the technique was tested with the following number of faulty nodes 2, 4, 6 and 8. In addition, the performance of the technique presented in this paper was compared, in terms of detection accuracy, to that of the Randomized Decision Scheme (RDS) presented in [11], where the detection accuracy can be defined as the percentage of the number sensor nodes that are detected to be faulty by a technique to the total number of faulty nodes the WSN [9].

Figure 12 shows the detection accuracy with respect to the number of faulty nodes. From Fig 8, it can be observed that as the number of faulty nodes increases, the detection accuracy decreases. This can be regarded to the ratio of neighboring sleeping nodes to the suspected node because the studied technique depends on awaking two nodes for every suspect node. As a result it can be inferred that the higher the redundancy of the network, the better the performance of our technique. In some cases, when there are not enough sleeping nodes near the suspect, the technique will awake the first sleeping node that is the nearest to the suspect but, because of not having enough sleeping node, the awakened node could be a bit far from the suspect and may not be under the same conditions as a result, not providing similar readings.

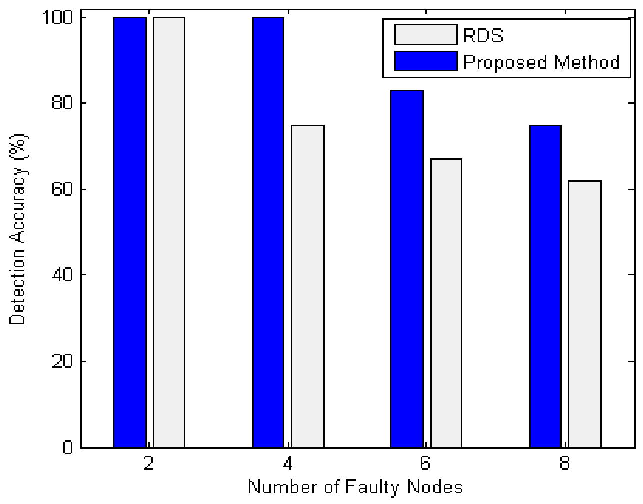

Figure 12. Comparison of Communication Faults Detection Accuracy.

Figure 13 shows the detection accuracy of our technique when having sensor nodes suffering from a sensing fault. It can be observed that when the number of faulty nodes was increased, the detection accuracy decreased because of not having enough fault free nodes near the faulty nodes.

Our technique has shown better performance than that of RDS, because in RDS the threshold value is selected randomly. As a result, in some cases the threshold value was suitable to help RDS detect fault node. However, in other cases this value was not suitable to be used in the detection which results in reducing the accuracy of fault detection in RDS.

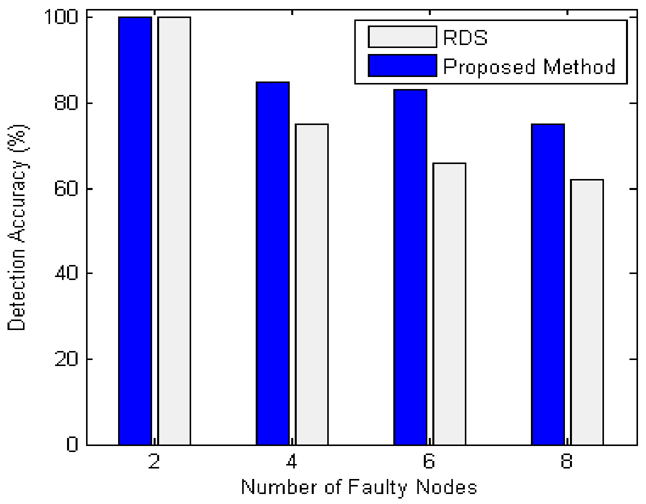

Figure 13. Comparison of Sensing Faults Detection Accuracy.

VII. CONCLUSION

In this paper, we proposed a new technique, consisting of two algorithms, to identify and substitute faulty nodes in wireless sensor networks. The proposed technique divides the network into four zones while having a master node for each zone. The first algorithm proposed in this paper is used to diagnose faulty zone masters. On the other hand, the second algorithm is used to test and substitute faulty sensor nodes, i.e. non master nodes, in the network.

The simulation has shown that the proposed technique does not require a lot of messages to be exchanged in order to detect the fault master node. Also, it has shown that because the master nodes are connected in a De Bruijn graph, the end-to-end delay is low under faulty and fault free conditions. Furthermore, the simulation has shown the ability of the technique to identify several faulty nodes in the same zone. Also, it has illustrated that the technique is capable of identifying more than one faulty node in more than one zone at the same time. Finally the algorithm is tested by simulating two different scenarios. Our results show that the detection accuracy was very high when the number of faulty nodes was small compared to the number of sleeping node.

Future work for this work may include studying the effect of the second algorithm on the energy consumption and the life time of the network. In addition, the effect of having the sensor nodes in a zone connected in a De Bruijn graph will be studied.

REFERENCES

[1] A.A.Taleb, D.K. Pradhan and T. Kocak, "A Technique to Identify and Substitute Faulty Nodes in Wireless Sensor Networks," Third International Conference on Sensor Technologies and Applications, 2009. SENSORCOMM '09., pp.346-351, 18-23 June 2009.

[2] A. Mainwaring, Culler, D. Culler, J. Polastre, R. Szewczyk and J. Anderson , "Wireless Sensor Networks for Habitat Monitoring," First ACM Workshop on Wireless Sensor Nerworks and Applications, Atlanta, Georgia, USA, Sept 2002.

[3] Y. Zou and K. Chakrabarty, "A Distributed Coverage and connectivity Centric Technique for Selecting Active Nodes in Wireless Sensor Networks," on IEEE Trans. on Computers, vol. 54, no. 8, pp. 978-991, Aug 2005.

[4] K. Ssu, C. Chou, H. C. Jiau and W. Hu,"Detection and diagnosis of data inconsistency failures in wireless sensor networks,"in The Int. Journal of Computer and Telecommunications Networking, vol. 50,no. 9, pp. 1247-1260, June 2006.

[5] G. Gupta and M. Younis, "Fault-tolerant clustering of wireless sensor networks," *IEEE Conf.* on *Wireless Communications and Networking, (WCNC).* pp.1579-1584, March 2003

[6] Q. Liang, "Fault-tolerant and energy efficient wireless sensor networks: a cross-layer approach," in Proceedings of IEEE Military Communications Conference (MILCOM '05), vol. 3, pp. 1862–1868, Atlatnic City, NJ, USA, October 2005.

[7] A. Sheth, C. Hartung and R. Han (1999). A decentralized fault diagnosis system for wireless sensor networks. In: Proceedings of the 2nd IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS) 2005. pp. 192–194.

[8] I. Saleh, H. El-Sayed and M. Eltoweissy, "A Fault Tolerance Management Framework for Wireless Sensor Networks," Innovations in Information Technology, 2006 , vol., no., pp.1-5, Nov. 2006.

[9] J. Chen, S. Kher, and A. Somani "Distributed Fault Detection of Wireless Sensor Networks". In Proc of the 2006 Workshop in Dependability issues in wireless ad hoc networks and sensor networks, L.A., California, USA, Sept. 2006.

[10] J. Staddon, D. Balfanz and G. Durfee, "Efficient tracing of failed nodes in sensor networks". In Proc of the 1st ACM international workshop on Wireless sensor networks and applications, Atlanta, Georgia, USA, Sept 2002.

[11] B. Krishnamachari, S. Iyengar, "Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," Computers, IEEE Transactions on , vol.53, no.3, pp. 241-250, March 2004.

[12] Yue-Shan Chang; Tong-Ying Juang; Chih-Jen Lo; Ming-Tsung Hsu; Jiun-Hua Huang, "Fault Estimation and Fault Map Construction on Cluster-based Wireless Sensor Network," . *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006*, vol.2, no., pp.14-19, 5-7 June 2006.

[13] T. Clouqueur, K.K. Saluja, P. Ramanathan, "Fault tolerance in collaborative sensor networks for target detection," *Computers, IEEE Transactions on* , vol.53, no.3, pp. 320-333, Mar 2004.

[14] Yongxuan Lai; Hong Chen, "Energy-Efficient Fault-Tolerant Mechanism for Clustered Wireless Sensor Networks," *Proceedings of 16th International Conference on Computer Communications and Networks, 2007. ICCCN 2007.*, vol., no., pp.272-277, 13-16 Aug. 2007.

[15] Tsang-Yi Wang; Han, Y.S.; P.K. Varshney, Po-Ning Chen, "Distributed fault-tolerant classification in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol.23, no.4, pp. 724-734, April 2005.

[16] S. Hwang and Y. Baek, "Fault Tolerant Time Synchronization for Wireless Sensor Networks," LNCS-3894, Springer, pp. 480–493, March 2006.

[17] F. Koushanfar; M, Potkonjak; A, Sangiovanni-Vincentell, "Fault tolerance techniques for wireless ad hoc sensor networks," Sensors, 2002. Proceedings of IEEE , vol.2, no., pp. 1491-1496 vol.2, 2002.

[18] M. R. Samatham and D. k. Pradhan, "THE De Bruijn Multiprocessor Networ: A versatile Parallel Processing and Sorting Network for VLSI," IEEE Trans. on Computers, Vol 38, No. 4, April 1989.

[19] D. K. Pradhan and N. H. Vaidya, "Roll-Forward Checkpointing Scheme: A Novel Fault-Tolerant Architecture," on IEEE Trans. On Computers, vol. 43, no. 10, pp. 1163-1174, Oct. 1994.

[20] D. K. Pradhan, and S. M Reddy, "A fault-tolerant communication architecture for distributed systems," IEEE Trans. on computers., pp. 863-870, 1982.

[21] M. Andersson, D. Henriksson, A. Cervin and K. Årzén, "Simulation of wireless networked control systems". In Proc of the 44th IEEE conference on Decision and control and European Control onference ECC, Seville, Spain 2005.

[22] TrueTime, Lund University. (2008, June 10) [Online]. Available: http:// www.control.lth.se/truetime/