REsilient Double WEighted TruST Based (REDWEST) WSN Using SAX

Aline S. Siranossian and Hoda W. Maalouf

Notre Dame University Louaize, Zouk Mosbeh, Lebanon
{asiranossian,hmaalouf}@ndu.edu.lb

Abstract. Wireless Sensor Networks (WSNs) are becoming the most widely used applications in monitoring environment and military operations. However, in such applications sensors are deployed in harsh environments and sometimes are inaccessible once deployed making them vulnerable to both physical and software attacks. Malicious nodes can send misleading data to the controller affecting monitoring results. Sophisticated security applications cannot be used to overcome this problem due to the limited power of the sensors. A new mechanism is needed which first identifies malicious nodes in an accurate manner and offers indispensible characteristics namely, resiliency and reliability to the WSN. In this paper, we develop a malicious and malfunctioning node detection scheme using a resilient double weighted trust evaluation technique in a hierarchical sensor network. Our system evaluates all sensor nodes, increases and decreases trust value accordingly and excludes nodes having under threshold trust values. The simulation results show that our approach is very efficient even in harsh environments.

Keywords: Wireless sensor networks, malicious node detection, weighted trust, resiliency.

1 Introduction

The field of Wireless Sensor Networks (WSNs) is now in a stage where serious applications of societal and economical importance are in reach. Examples such as landslide, forest fire and underground mines advocate the use of wireless sensing technology as a new scientific instrument for environmental monitoring under extreme conditions. In such applications, reliability, availability, and maintainability are indispensible characteristics.

When an environment needs to be monitored, a large number of sensor nodes are usually deployed in a random fashion. The main purpose of the sensor nodes in this case is to take measurements and to forward this data to the sink node where it is processed and necessary action is taken.

Being used in very critical applications, data has to be transmitted accurately. However, WSNs have limited capacity and energy resources and hence are likely to be influenced by unpredictable failures occurring in the harsh sensor field. So the system requires a routing protocol to deliver event packets from source nodes to sink

M. Zuniga and G. Dini (Eds): S-Cube 2013, LNICST 122, pp. 17–33, 2013.

nodes in a fault-tolerant and energy efficient way regardless of node failures and attacks such as, HELLO flooding attacks, sink hole attacks, black hole attacks, worm hole attacks, or DDoS attacks [1]. Sybil attacks are when a malicious node behaves as if it were a large number of nodes. In the worst case scenario, an attacker may generate an arbitrary number of node identities using only one identity [15]. In application layer, attackers may take control over nodes and make them send false data in a very intelligent manner to fool data aggregators and hence lead to an incorrect decision, facing a byzantine problem [14]. This is one of the worst attacks, which when solved can also solve many types of WSN node problems. Some solutions depending on trust value of the sensor are reported to detect these attacks so that the influence of the malicious node is minimized and finally removed from the network. However, all of these approaches assume that only sensor nodes that are placed at the lowest level in the hierarchical network are prone to attacks and failure. Forwarding nodes and access points are assumed to be trustful and won't be compromised. In reality, all sensor nodes have similar properties since they are situated in the same environment making them all equally prone to attacks and failures.

Since sensors have very limited resources (memory, storage and power) therefore, dimensionality reduction, code and task minimization are other indispensable factors to be considered. In fact, a sensor is a tiny device with only a small amount of memory and storage space for the code, so the overall code for detection, aggregation and security has to be small. Furthermore, the power consumption needed for transmission dominates processing energy consumption. Hence, communication should be minimized as much as possible. To meet these stringent bandwidth and power constraints, especially when considering real-time data monitoring, the high-dimensional sensor observation should be converted into low-dimensional data by carrying out local data dimensionality reduction.

Several techniques like, Discrete Wavelet Transform (DWT) and Discrete Fourier Transform (DFT) were used for dimensionality reduction. However, most of these techniques require lots of storage space. Recently, Lin and Keogh et al. [13] proposed the Symbolic Aggregate approximation (SAX), the first symbolic representation for time series that allows for dimensionality reduction and indexing with a lower-bounding distance measure, based on Piecewise Aggregate Approximation (PAA) and assumes normality of the resulting aggregated values. When using SAX, the data is first transformed into the PAA representation and then symbolized into a sequence of discrete strings. The symbolization region is determined by looking up in statistical tables since the time series represent a Gaussian distribution. Breakpoints are represented as a sorted list of numbers such that the area under a Gaussian curve from β_i to $\beta_{i-1} = \frac{1}{\alpha}$. These breakpoints are determined by statistical tables. All PAA coefficients that are below the smallest breakpoint are mapped to the symbol "a", all coefficients greater than or equal to the smallest and less than the second smallest breakpoint are mapped to the symbol "b", etc.[13].

In this paper, we will be using SAX with some modifications for data dimensionality, code and task reduction. Furthermore, by considering the real case of all nodes being prone to attack, we propose in this paper a dual weighted trust scheme

for detecting and removing compromised nodes. Whenever a controller node is detected as malicious, the network will perform modifications by assigning a new controller so that the system will not be affected and continue to provide valid reports even under harsh environmental conditions.

The rest of the paper is organized as follows: Section two summarizes previous work related to fault detection schemes. Section 3 explains the proposed dimensionality reduction scheme. Section 4 describes the network topology to be used throughout the paper. Our resilient double trust based scheme is presented in section 5 and the experimental tests and results are shown in section 6. Finally, section 7 concludes the paper.

2 Related Work

The goal of fault detection is to verify that the services being provided are functioning properly, and in some cases to predict if they will continue to function properly in the near future. Fault detection techniques are classified as: self-diagnosis where the node itself can identify faults in its components, group- detection where several nodes monitor the behavior of another node, and hierarchical detection. The approach used in [3] which performs diagnosis based on accelerometers to determine if the node suffers from an impact that could lead to hardware malfunctions, the approaches used in [4], [5] and [12] which use voltage and signal strength anomaly and the approach used by [10] which use localization anomaly are all self-diagnosis techniques. Some of the drawbacks of these techniques are the incapability of sudden crash failures and the reliability on single node in decision making which can be already compromised.

The approaches used by Iyengar in [7] and Cheng *et al*. in [6], which are based on the idea that sensors from the same region should have similar values unless a node is at the boundary to calculate the probability of the node being faulty, and the approach used by Loureiro *et al.* in [11] which is based on nodes reading sensors signal strength measured by neighboring nodes and comparing its compatibility with the node's geographical position to detect malfunction are group-detection techniques. Group detection schemes are applicable. However, they have several drawbacks. They require large overhead needed for transmitting data which is a problem both for sending and processing, they are not energy efficient and the use of encryption is often impracticable, since this would hamper other nodes observing the contents of messages [8].

Hierarchical detection techniques use data aggregation techniques in their scheme. In [9], the authors proposed mechanism which uses a hierarchical network topology where cluster heads monitor ordinary nodes, and the base station monitors the cluster heads. To perform the monitoring, the base station and the cluster heads constantly ping those nodes that still have battery power left and that are under their direct supervision. If a node does not respond, it is marked as a failure. Lately a special type of attack where the compromised nodes behave normally but report false readings to lead to an incorrect decision has been investigated by Atakli. Et al. [1] this is a straightforward hierarchical detection approach and incurs less overhead since there is

no expensive calculation involved. They proposed the scheme of weighted trust evaluation (WTE) to detect malicious nodes. The weights of nodes are updated after each cycle by reflecting the ratio of the number of incorrectly reporting nodes to the total number of nodes. However, as explained by [2] the aggregated result of their scheme, calculated by the forwarding node cannot reflect the real situation, and the update of weight value cannot reflect change of credibility of the node itself. So they proposed a weighted-trust application (WTA) scheme. The weight of each sensor node in this scheme is updated based on the behavior of the node itself, making the node's weight value more accurate and misdetection ratio distinctly lower. OH et *al.* in [14] found that both schemes proposed by Atakeli *et al's* and Ju *et al,* are likely to detect malicious nodes by sacrificing some normal nodes. The loss of normal nodes might be problematic due to the resulting lack of network connectivity and sensing coverage. In addition, faults are only partially taken into account in detecting malicious nodes. They proposed a dual weighted trust evaluation scheme (DWE) in an environment where noise, natural faults and malicious nodes coexist. Each sensor node is assigned two trust values. The trust values are increased or decreased depending on the reading and aggregation result at the forwarding node. An efficient updating policy is used to keep mis-detection rate low while achieving high malicious node detection rate [14].

3 Dimensionality Reduction

Depending on the application, each sensor node will be equipped with a special type of sensor. In general, the sensor data can be divided into three categories: normal (sensor is unharmed and the condition is normal, e.g. no fire), critical (sensor is unharmed while the condition is critical, e.g. fire) and abnormal (sensor is compromised, malfunctioned or dead). Even though the data is divided into three regions, each region may include a large number of data points. It is assumed that each sensor node knows its location, which will be sent to the parent node each time a symbol is sent. We will first normalize these data points making the normal value assigned equal to zero. In addition to dimensionality reduction purpose, we will be using the symbols from SAX to determine the deviation of a sensor from the normal. So, we proposed a new symbol conversion scheme by performing some modifications on SAX's look up table. SAX considers only positive values. However, in our case sensor readings can deviate from the normal from both sides (higher or lower) and should be penalized in the same manner. We have proposed a new look up table to perform the needed task. Table 1 is our proposed generalized look up table where the user is able to specify the complexity of the calculation. Increasing the number of breakpoints increases the number of levels (symbols). Although this would increase the accuracy of the system but it will increase the required discretization time.

For example, if we consider three regions in the table 1, the normalized sensor readings between [-0.43 and 0.43] will be converted to symbol "b" and the rest to symbol "a". However, if SAX is used in this scenario, then the symbols would be "a" if the value is less than -0.43, "b" from [-0.43 and 0.43] and "c" if greater than 0.43.

Table 1. Digit to symbol conversion table

Region / β	2	3	4	5	6	7	8	9	10
∞	a	b	b	c	c	d	d	e	e
-1.28									
-1.22								d	d
-1.15							c		
-1.07				b	b	c		c	c
-0.97							b		
-0.84			a			b		b	b
-0.76				a	a		a		
-0.67		a				a		a	a
-0.57			a				a		
-0.52				b	a	b		b	b
-0.43							b		
-0.32			b		b	c		d	d
-0.25		b		c	c	d	d	e	e
-0.18									
-0.14									
0	a								
0.14									
0.18									
0.25									
0.32									
0.43									
0.52									
0.57									
0.67	a								
0.76									
0.84									
0.97		b							
1.07									
1.15									
1.22									
1.28									
∞									

4 Network Topology

Our proposed system will have a four-layer architectural design, consisting of four types of sensor nodes: Simple Sensor Nodes (SS), Cluster Nodes (CL), Base Station (BS) nodes and the Sink Node (SN). SS nodes communicate directly with their CL nodes which in turn send their data to their BS parents, which finally send their data to the sink node. We shall assume that the SN has no limitations and is not vulnerable to any attacks. It receives the obtained readings, saves them for future use and takes the appropriate action in severe cases.

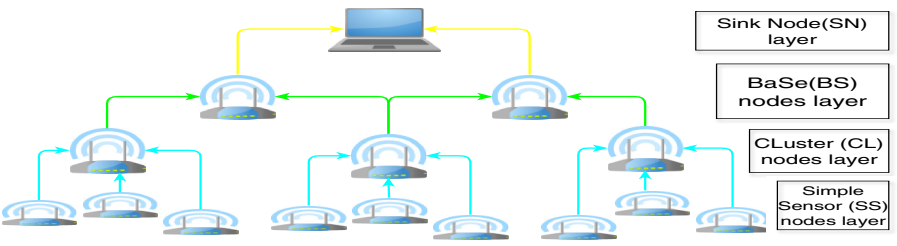

Fig. 1. Architecture of REDWEST

Based on the four-layered architecture, the deployed sensors must be divided into these types depending on their positions. At launch, sensor nodes are randomly distributed on a given terrain which is divided into a $l * l$ predefined grid by the user. Each grid will have one CL node. Several neighboring nodes (depending on the grid dimension) will have one common BS node. So at first all sensors are assumed to be SS nodes. In order to select the CL and BS nodes, the system accomplishes the following steps:

1. Determines the nearest sensor node to the intersection of the neighboring grids and designates it as BS. Each sensor node designated as BS will use its higher transmission capabilities to be able to communicate with all its children. This process is performed by the SN.

2. Determines the nearest sensor node from the center of the grid and designates it as a CL node. Each sensor node designated as CL will also use its higher transmission capabilities as well. This process is performed by the BSs.

It is assumed that this process is only performed using security measures. In this way, the location of BS and CL children is provided securely. This is important so that parent nodes can detect Sybil attacks by the number of children they have and the mismatch in the position information sent by each sensor when transmitting its data.

Whenever a CL or a BS node has consumed its power, or is detected as malicious, Redwest will be able to find that due to its ability to find malicious and malfunctioning sensors and replace it by using the above conditions.

5 REDWEST

5.1 Proposed Algorithm

- **Simple Sensor (SS) Node Layer:** Sensor nodes (SSs), will read the data sensed by the sensor, perform the conversion from digit to symbol using our proposed SAX algorithm and send the data and its position through its antenna.

- **Cluster Control (CL) Node Layer:** After receiving the data from its children SS nodes, the CL node will validate the position information and find the letter which has the maximum occurrence and designate it as the normal value. Then, it will calculate the deviation of each node from the normal and penalize those nodes by decreasing their weight. In addition to deviation from normal, REDWEST considers the performance system as another important factor in the evaluation process. If an SS node sends five **consecutive** correct values with respect to the normal, the CL node will increase the weight of that SS node. Having the weight of each sensor, data aggregation will be performed by multiplying the data sent by their weight and finding the average. In this way, sensors being suspected as malicious will have less impact on the system and sensors that were giving wrong results in one occasion will have the chance to be considered as an important factor in the system. After aggregating the data, the CL node will send the result to the BS node.

- **Base Station (BS) Node Layer:** As we go up in the hierarchy, the number of children decreases, meaning that the received data will be reduced making the impact of a single node higher since the influence caused by an erroneous sensor will be higher. So we proposed to take firmer actions by performing two types of weight calculation schemes. In the first scheme, the algorithm adopted by the CLs utilizes harsher conditions: the tolerance of accepting wrong readings will be changed. In the second scheme, BSs will compare the average of data of sensors CL_{x1} found at a certain distance from the edge with its neighboring CL_{x2}. Since the data sensed at the adjacent edges should be the same, then wrong results sent by two adjacent CLs would cause further decrease or increase in their weight.

5.2 Simulation Program and Adopted Formulas

The Symbol representations of each sensor in addition to its own reading will be collected by the CL node. If an SS node fails to send an accredited symbol or simply does not send any data due to battery failure or physical/software damage, the CL will consider its letter grade to be the last letter in the range. With the number of readings matching the number of children and the location sent by the sensor matching the one in its table, the CL node will determine the total count of each letter and designate the letter having the highest count as the normal value in the grid. In Table 2 the list of the used symbolic notations are given and explained.

Table 2. Symbolic Notations

Symbol	*Meaning*
E	Aggregation result
U_n	SS sensor node's output. E.g. temperature reading
U_n'	Symbol value of SS sensor node's output. E.g. "a"
B_n	Indicates if SS node's reading matches the average value
S_{letter}	Count of sensors (with penalty) reading the value "letter"
W_n	Weight value of SS sensor n, which ranges from 0 to 1
V_n	Weight value of CL sensor n, which ranges from 0 to 1
D_n	Deviation of the sensor value from "letter" value
S	Number of regions selected by user
F_n	Number of "m" consecutive correct readings out of "n"
R_n	Number of wrong readings sent by a single SS node n
M	The most common letter (of all sensors in one grid) in a single round
M_{x1}	The most common letter (of sensors on the right of grid) in a single round
M_{x2}	The most common letter (of sensors on the left of grid) in a single round
M_{y1}	The most common letter (of sensors on the top of grid) in a single round
M_{y2}	The most common letter (of sensors on the bottom of grid) in a single round
θ	Positive penalty coefficient
γ	Negative penalty coefficient

In this paper, counting the number of occurrences of each letter is not performed using a primitive manner. Here also the idea of trust is used. This is performed to solve the Byzantine problem. Sensors which are detected as malicious (even if they are giving correct values on purpose) will not have influence on the counting phenomena. The count of sensors reading the "symbol" value is given by S_{letter} .Where S_{letter} represents, the sum of sensors whose quantized ($U_n{}'$) value of its output (U_n) matches the normal symbol value "letter" multiplied by the weight of the sensor. S_{letter} can be obtained using the following formula:

$$S_{letter} = \sum_{n=0}^{N}(B_n) * (W_n) \qquad \text{where, } B_n = \begin{cases} 1 & if\ U_n{}' = "letter" \\ 0 & otherwise \end{cases} \tag{1}$$

Consequently, if "a" and "b" are the two symbols used, then, S_a will give the number of sensors reading the symbol "a" and S_b will give the number of sensors reading symbol "b" taking into account their weight value. Having these values, CL will find the symbol having the highest *S* value and designate it as the most common letter *M*.

The CL node will now find out how much each sensor is deviated from the most common (normal) value, calculate the extent of irregularity, the number of consecutive successes and accordingly penalize each sensor. The updated penalty will be used in the next round.

We proposed to calculate the deviation from the normal value using the following formula:

$$D_n = \left| \frac{U_n{}' - M}{\frac{s}{2}} \right| \tag{2}$$

Where d_n is the deviation of each sensor in a single round and "*s*" is steps (region from table 1) selected by the user.

The main purpose behind this convention is adding the factor of error deviation to the penalty formula (eq.3) meaning that a sensor making a deviation δ from the normal will be penalized less than the sensor making an error (δ+λ).

This factor was not considered in previous work; however, we believe that sensors should be penalized depending on how much they are deviated from the average. A sensor that is slightly deviated due to a disaster in its area should not be penalized as much as a sensor giving a value with high deviation due to malfunction or intrusion. E.g. a fire can start near a sensor, so that sensor will read values slightly higher than neighboring sensors at round one. If this is the case, and a large penalty is given to that sensor then it will be considered as a faulty node where in fact it is not. In our system, the node will be penalized with a small factor and will be rewarded in the next round, since the average will tend to be that of a disaster state if fire spreads making more sensors to detect the phenomena.

The number of wrong readings (R_n) the sensor has made, is another factor to be considered in finding the penalty weight. This issue was considered in previous body of work. However, we believe that the number of wrong readings ought to have an

exponential impact on the weight factor. In fact, we selected the $\left(1-\frac{1}{e^{R_n}}\right)$ factor since it gives the desired performance. The system will not tolerate a sensor giving more than 5 wrong readings and will give harsh penalties to sensors making more than 2 consecutive wrong readings.

We also proposed upgrading the weight value whenever the sensor has been affected by a natural noise. This scheme was recently used by Oh et *al.* in [14]. However, we think that the increase should not be done directly every time the sensor output matches the normal value. We propose to increase the weight if the sensor was able to send a certain number of consecutive correct readings out of a predefined number (F_n). Selection of the parameter F_n has an effect on the detection accuracy. By default, it is set to five out of ten. Hence, each five consecutive readings within the ten readings will increase F_n by 1. After the ten consecutive readings, the number is reset. For stricter conditions, this value can be set to a firmer range such as eight correct readings out of ten.

Having the number of wrong readings (R_n), the deviation from average (D_n) and the number of five consecutive correct sensor readings obtained (F_n), the CL node will calculate the weight value of each sensor in a single round. The weight can be increased or decreased depending on the behavior of a single node. The weight value represents the sensor node's dependability. That is, the readings of a sensor node with a higher weight are more trustworthy and thus its readings will have higher influence in the aggregation process. Updating the values is important to reflect the correctness of the current readings in the future decision making process.

Updating the weights has two purposes. First, if a sensor node is compromised and is frequently sending its faulty readings that are inconsistent with the final decision, its weight is likely to be decreased. Second, if an abnormal reading was sent by the sensor on one occasion and later by resolving its problem became consistent, then the weight value has to be increased. This is reasonable since sensors with incorrect reading should have smaller impact on the final decision than those with correct readings.

Hence, summing up we propose the following equation to calculate the weight, where *j* indicates the present round:

$$(W_n)_j = \begin{cases} (F_n * \theta) + (W_n)_{j-1} - (D_n) * \gamma * \left(1-\frac{1}{e^{R_n}}\right) - H & if\ U_n{}' \neq M \\ W_n & elsewise \end{cases} \quad (3)$$

Where, $0 \leq W_n \leq 1$.

In equation 3, the number of wrong readings (R_n), with the selected exponential factor is deducted from the sensor's previous weight. This means that, our formula is also based on the behavior of the sensor node itself. This was selected so that the penalty can depend on the number of mistaken reports which will increase the penalty exponentially. To add the ability to do fine adjustments, we have included the negative penalty coefficient (γ). Increasing this coefficient value will decrease the weight more rapidly. The value of γ can vary between 0.1 and 1.

On the other hand, the number of consecutive readings (F_n) multiplied by the positive penalty coefficient θ is added to the previous sensor weight. The larger the value of θ is, the faster the increase of the weight value is when consecutive successes are achieved. The value of γ can vary between 0.1 and 1.

Finding the optimal values of θ and γ is essential in our mechanism since these parameters affect the detection time and accuracy of our proposed algorithm.

In (eq. 3), we notice that sensors having higher deviation will be penalized more. Based on updated weights, the CL node is able to detect a node as a malicious node if its weight is lower or equal to zero. Sensors indicated as malicious will be taken out of the system.

Moreover we have used in (eq. 3) the factor H to detect intruder nodes as well as Sybil and replication attacks, another factor H is added to (eq. 3), which is the validation factor. If the position of the sensor is not validated by its parent, a value of 1 will be assigned to H, otherwise H will be zero. Subtracting 1 in (eq. 3) leads to the removal of the sensor from the system directly. We assumed here that the probability of finding the exact position of a sensor by a malicious node is low, sensors do not have the ability of finding the position of their neighboring sensors and that the position information is forwarded to the BS and CL nodes in a secure way.

Next, the CL node will aggregate two values to be sent to the BS node. The normal value aggregated from all sensors of the grid and the normal side sensor's values aggregated from the sensors having a minimum distance (defined by the user) from the sides.

To get the aggregation of the side sensors, the CL will use the same equations as above but instead of considering all sensors in the grid, it will consider the sensors which are positioned at the edge of the grid. This step will generate the values of the most common letter in the different sides of the grid, namely M_{x1} on the right side, M_{x2} on the left side, M_{y1} on the top side and M_{y2} on the bottom side.

Now, if we need to have the exact reading values and not just the letter characters then the aggregation equation will become:

$$E = \sum_{n=1}^{N} W_n * U_n \tag{4}$$

Where E is the aggregation result, W_n is the weight ranging from 0 to 1 and U_n the sensor reading.

After receiving the most common letters M, M_{x1}, M_{x2},M_{y1}, M_{y2}, values from its children CLs, the BS node performs Aggregation based on the M values where each BS node will collect the data received from the four corners. Similarly, a BS node will find the most common letter N_n based on the weight V_n and the different N_n using previous formulas but with firmer conditions. Figure 2 summarizes the weight based aggregation system.

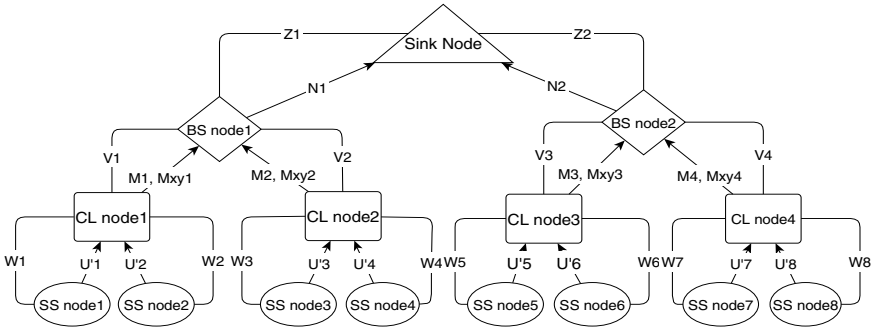

Fig. 2. Weight based aggregation of the hierarchical network REDWEST

The above steps are repeated whenever new information is to be reported to the sink node. The weight of each sensor is updated based on the correctness of the information. If the weight value of a sensor becomes less or equal to zero, it will be considered out of the system. If it happens to be a CL or a BS node, the system will designate that job to another sensor that has the necessary requirements by performing the steps explained earlier.

Sensor nodes whose weight value is less or equal to zero are excluded from the system; however, these nodes can join the aggregation process again if their weights increase to 1 by the user depending on the application.

6 Simulation Results

Several simulation experiments using Matlab[16] were conducted to evaluate the effectiveness and performance of REDWEST. In these simulations, we considered that a total of 900 temperature sensors were deployed in a forest which was divided to a $3 * 3$ grid. The number of letters chosen was 5. Faults (dead, malicious, and malfunctioning sensors) and critical situations were introduced. In the case of dead sensors, it was assumed that these sensors would remain dead during the selected 100 runs, where a run is the process of all sensor readings being sent to SN node. Malicious nodes were picked randomly with a probability of an occurrence set by the user. To make the simulation as close to reality as possible, we assumed that the probability of an already selected node to be picked again as malicious was higher in the next round. In the performed tests we have evaluated the effectiveness of our proposed formulas with respect to previously used similar schemes. Also, resiliency, endurance, performance and dynamism tests were performed as functions of different factors such as: the number of sensors deployed, the number of runs performed (endurance test), the number of malicious nodes deployed, the number of permanent faults deployed, positive penalty coefficient θ, negative penalty coefficient γ, the $\left(1 - \frac{1}{e^{R_n}}\right)$, H and F_n factors.

To begin with, we considered θ = 0.2 and γ = 0.8 since we have to be strict with sensors making mistakes and on the other hand not tolerant with the sensors giving

correct values after incorrect readings. Endurance of the system was measured by varying the number of reading instances (runs) from 100 to 1000 runs.

Two probability factors were generated: possibility of sensors to be damaged, malfunctioning and out of power denoted by P_{dead} , and possibility of sensor to be malicious, reading incorrect readings and under the influence of attack denoted by $P_{problematic}$. To consider very harsh environment, we took extreme bad conditions where the probability of dead sensors P_{dead} =0.10 and then the probability of problematic (damaged, having dead battery, created due to Sybil and malicious attacks) was increased. The system functioned error free until $P_{problematic}$ = 0.6.

From the first subplot (a) in Figure 3, it can be noticed that even with 10% dead sensor leading to 90 sensors in each grid with 60% of it not normal (malicious or

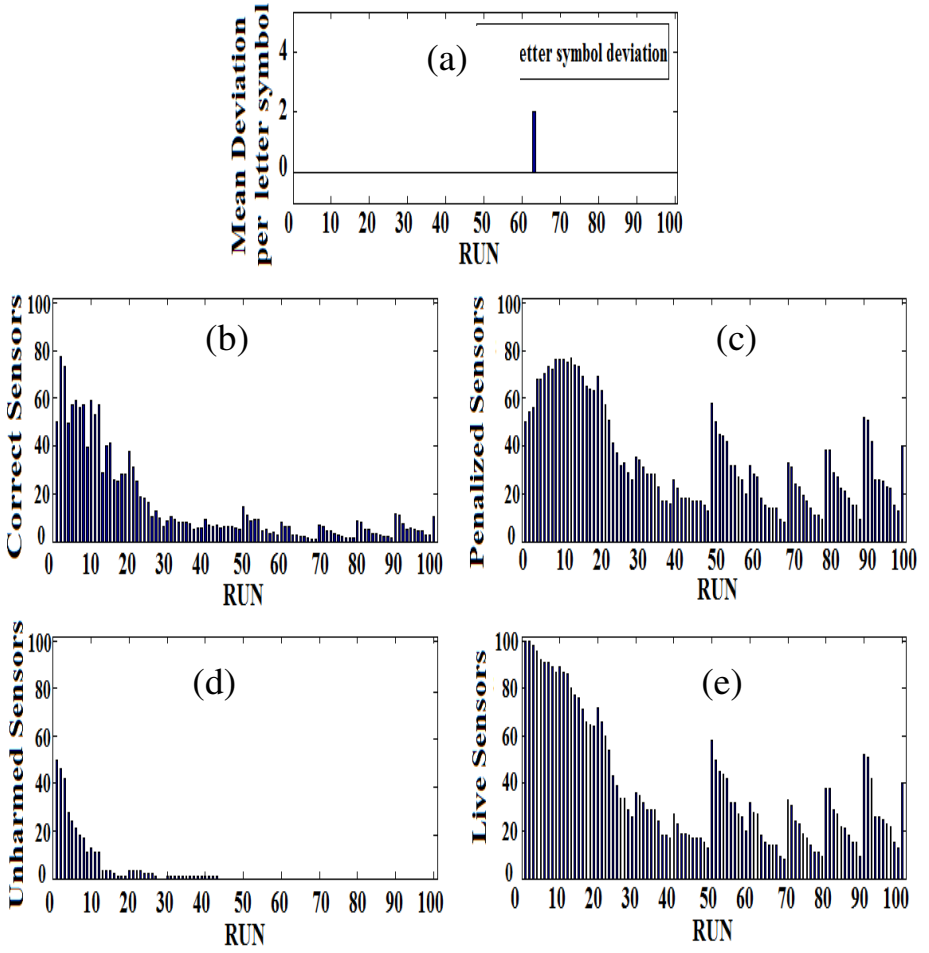

Fig. 3. General Outcome

malfunctioned), the system in the 100 runs gave only one mistaken output with a deviation of two letter grades. However, if we look at the remaining subplots, we can see that at that instance no unharmed sensors exist (d), less than 5 sensors were giving correct results (b), all sensors were penalized (c) and less than 20 sensors were alive (e). Moreover, even after the incorrect reading reported, the system was able to overcome this harsh situation due to our two way grading system. So we can say that the system is consistent, resilient, and was able to overcome our endurance test.

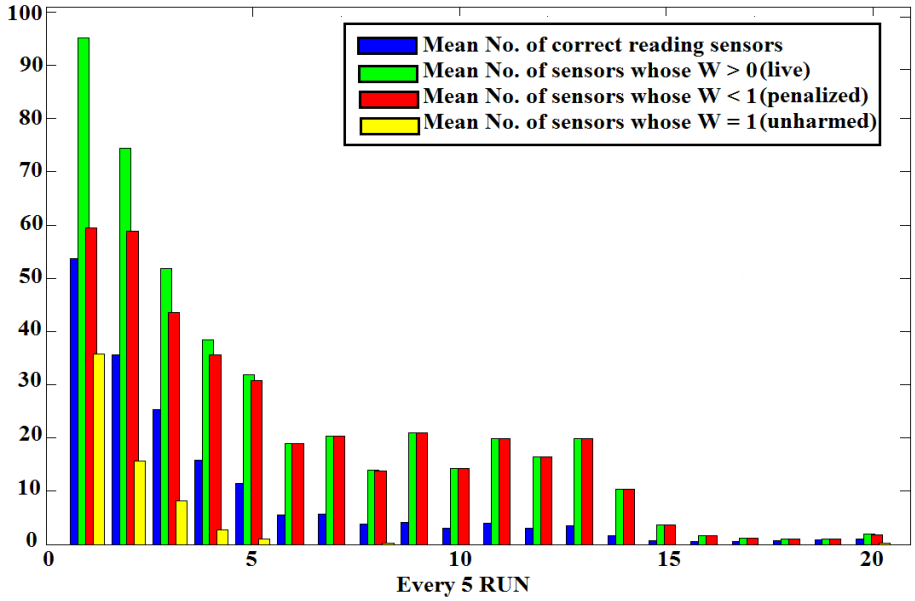

Fig. 4. Averages for every 5 run

Figure 4 magnifies what we previously noticed in figure 3. Here, instead of reading the result after each run, the average of every five runs was considered. We can see that after the 6th 5-Run step there are no remaining unharmed sensors, so all sensors on the terrain were malicious, dead or suspicious. It can also be noticed that in spite of having the number of correct readings most of the time less than half of the live sensors, the system was still able to give correct results (meaning correct temperature values). Furthermore, the system was able to revive itself by adding the non-malicious nodes to the system after they were temporarily removed due to erroneous readings. These come to substantiate what we previously already concluded previously.

Next, a comparison between our system and previous works that could be applied to our system is performed. Ju et *al's* system WTA was considered, since it is an improved version of WTE. Figure 5 presents:

- The average reading of all sensor nodes in the grid considering dead and malicious sensor nodes denoted by Averages. It should be noted that sensors

giving no values will be read as 2 (i.e. the value of "2" is considered as an infinite reading).

- The average reading of all sensor nodes in the grid using Ju et *al's* system, denoted by WTA averages.
- The average reading of all sensor nodes in the grid using our proposed system, denoted by REDWEST averages.
- The average reading of only sensor nodes in the grid which are giving correct values, denoted by Perfect Averages.

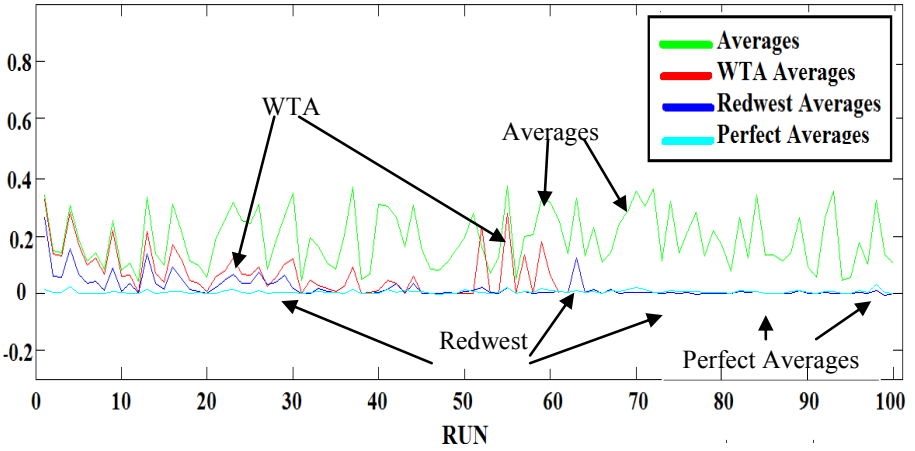

Fig. 5. Comparison Test

In comparison to Ju et *al's* system WTA, it can be noticed that REDWEST has passed the endurance test by at least 100 Runs while WTA was able to last until the 60th round. Moreover, if we further continue this comparison, we notice that REDWEST was too close to the perfect results, while WTA was more sensitive to errors.

Survival rate is an equally important factor especially when the system is adopted in battlefields or harsh environmental conditions. We have also tested the system with high rate of attacks for longer periods of time. Figure 6, shows that although the system was under high rate of malicious attacks, it was able to overcome it and gave correct answers.

Numerically speaking if 60% of the 90 sensors are malicious at every run then the system will collapse after eight runs at extreme conditions. REDWEST on the other hand is functioning perfectly until the 120th run even when all the sensors are damaged. The output was wrong only when none of the sensors were giving correct results, which is very normal. If we compare it with WTA, we can notice how REDWEST's lifetime and endurance is high. In fact, it gave near perfect results except in situations where none of the sensors were functioning correctly, while WTA stopped functioning after 60 Runs. Finally, in order to find the optimal values of the positive penalty coefficient (θ) and the negative penalty coefficient (γ) we considered ratio ∂.

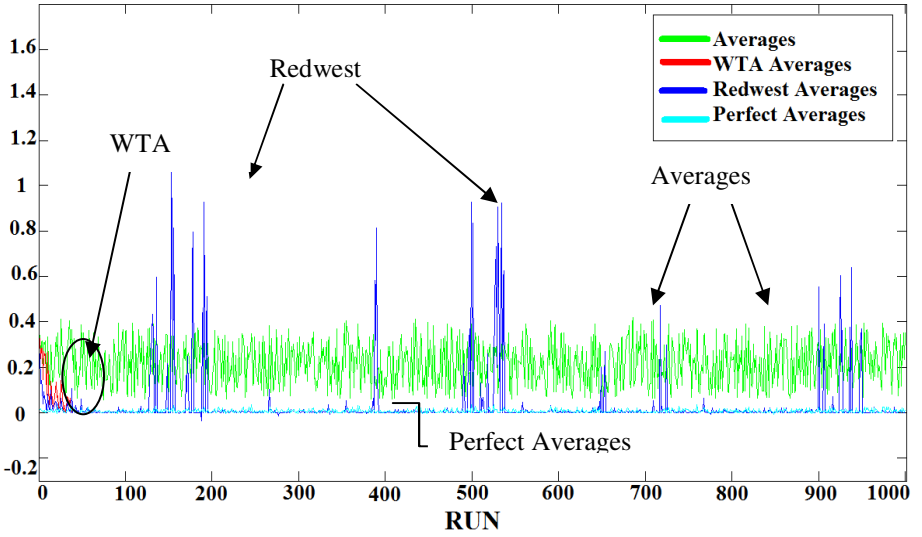

Fig. 6. System's performance under long term stress

$$\partial = \frac{\text{system correctness}}{\text{mean}\left(\frac{\text{correct}}{\text{live}}\ \text{sensors}\right)} \tag{5}$$

After considering different combinations of θ and γ, the simulation results showed that θ= 0.2 γ=0.8 combination gives the best results. By taking γ=0.8 we are decreasing the weight of a wrong sensor rapidly. However, taking θ=0.2 means that we are increasing the weight of the correct sensor smoothly. In this way the system will have enough time to decide whether the sensor was malicious or was under the effect of thermal noise.

7 Conclusion

In this paper, we proposed a novel dual weighted trust evaluation based scheme to detect compromised or misbehaved nodes in hierarchical WSNs. Trust values of sensor nodes are used as weights decided by the parent node to reflect the correctness of a sensor node's reports in decision-making procedures. The weights are updated in such a way that normal nodes with weights equal to 1 will retain their values, while those with weights less than one will be put in testing phase. If five consecutive correct values are recorded, then the trust value is increased. On the other hand, malicious nodes behaving differently from normal nodes gradually lose their weights and nodes having weight value equal to zero are excluded from the system.

In this paper, a modified SAX was used in order to minimize the transmitted data and to increase the system accuracy. Several equations were also proposed to test and calculate the different coefficients of the proposed algorithm.

As possible future work, we propose to add energy level to our weight formula hence solving the problems caused directly by selfish nodes. In this way, sensors having high power will be more trusted especially in the case of CL and BS nodes. Furthermore, additional aspects can be added to detect any source of replication leading to Sybil attacks; and to minimize extra security procedures used by security measures which consume several resources like energy and storage.

References

1. Atakli, I.M., Hu, H., Chen, Y., Ku, W.-S., Su, Z.: Malicious Node Detection in Wireless Sensor Networks using Weighted Trust Evaluation. In: The Symposium on Simulation of Systems Security (SSSS 2008), Ottawa, Canada, April 14-17 (2008)
2. Ju, L., Li, H., Liu, Y., Xue, W., Li, K., Chi, Z.: An Improved Intrusion Detection Scheme based on Weighted Trust Evaluation. In: The IEEE 2010 Proceedings of the 5th International Conference on Ubiquitous Information Technologies and Applications, CUTE (2010)
3. Harte, S., Rahman., A.: Fault Tolerance in Sensor Networks Using Self-Diagnosing Sensor Nodes. In: The IEE International Workshop on Intelligent Environment, pp. 7–12 (June 2005)
4. Benini, L., Castelli, G., Macii, A., Macii, E., Poncino, M., Scarsi, R.: A Discrete-Time Battery Model for High-Level Power Estimation. In: Proceeding of the Design, Automation and Test in Europe Conference and Exhibition 2000, pp. 35–39 (2000)
5. Rakhmatov, D., Vrudhula, S.B.: Time-to-Failure Estimation for Batteries in Portable Electronic Systems. In: Proceedings of the 2001 International Symposium on Low Power Electronics and Design, pp. 88–91 (2001)
6. Ding, M., Chen, D., Xing, K., Cheng, X.: Localized fault-tolerant event boundary detection in sensor networks. In: INFOCOM (2005)
7. Krishnamachari, B., Iyengar, S.: Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks. IEEE Transactions on Computers 53, 241–250 (2004)
8. Marti, S., Giuli, T.J., Lai, K., Baker, M.: Mitigating Routing Misbehavior in Mobile Ad hoc Networks. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, pp. 255–265 (2000)
9. Ruiz, L.B., Wong, H.C., Siqueira, I.G., Marcos, J., Nogueira, S., Loureiro, A.A.F.: Fault Management in Event-driven Wireless Sensor Networks. In: Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 149–156 (June 2004)
10. Du, W., Fang, L., Ning, P.: LAD: Localization Anomaly Detection for Wireless Sensor Networks. In: 19th International Parallel and Distributed Processing Symposium (IPDPS 2005), Denver, Colorado, USA, April 3-8 (2005)
11. Junior, W., Figueriredo, T., Wong, H.-C., Loureiro, A.: Malicious Node Detection in Wireless Sensor Networks. In: 18th International Parallel and Distributed Processing Symposium (IPDPS 2004), Santa Fe, New Mexico, USA, April 26-30 (2004)

12. Szewczyk, R., Mainwaring, A., Polastre, J., Anderson, J., Culler, D.: An analysis of a large scale habitat monitoring application. In: SenSys 2004 Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, pp. 214–226. ACM Press (2004)
13. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (2003)
14. Hyun, O.S., Hong, C.O., Hwa Choi, Y.: A Malicious and Malfunctioning Node Detection Scheme for Wireless Sensor Network. Wireless Sensor Network Scientific Research, SciRes Journal 4(3), 84–90 (2012)
15. Newsome, J., Shi, E., Song, D., Perrg, A.: The Sybil Attack in Sensor Networks: Analysis and Defenses. In: The Third International Symposium on Information Processing in Sensor Networks, pp. 259–268 (2004)
16. Matlab and Simulink for technical computing, `http://www.mathworks.com`