Increasing the Reliability of the Collected Data in Wireless Sensor Networks

Yaqoob J.Y. Al-Raisi
Sultan Qaboos University
Muscat, Sultanate of Oman
yalraisi@squ.edu.om

Nazar Alfadil
Fahad Bin Sultan University
Tabuk, Saudi Arabia
nfadel@fbsc.edu.sa

Sultan Aljahdali
Taif University
Taif, Saudi Arabia
aljahdali@tu.edu.sa

ABSTRACT

Wireless Sensor Networks are increasingly attracting many fields for monitoring and understanding the characteristics of different applications. Unfortunately, these characteristics also have an impact on the functionality of the network and, in some cases, cause it to deviate from its normal operation. Also, they cause a reduction in the quality and the quantity of the data collected by the network. This paper proposes a distributed monitoring performance algorithm which tracks these deviations and isolates those that degrade network functionality. The results obtained from the empirical and simulation experiments show that the algorithm achieves a high-level of detection reliability with resilience to both high packet loss and environmental changes.

Keywords: Wireless Sensor Networks, WSNs performance measurement, on-line fault detection, WSN outlier detection.

1. INTRODUCTION

DEVIATIONS of nodes from their normal operation in Wireless Sensor Networks (WSNs) are regular occurrences, not isolated events as in traditional networks. This is due to the goals in network design of reducing the cost of node manufacture and deployment, reducing the size of network nodes, and reducing their energy consumption. Such reductions directly affect sensor node resources and the flexibility of the protocol used; they also increase the impact of external and internal interference. This deviates network nodes operation from their normal behavior and causes reduction in the quality and the quantity of data collected by the network.

Operational deviations in the network and its nodes arise as a result of systematic or transient errors [1]. Systematic error is mainly caused by hardware faults, such as calibration errors after prolonged use, a reduction in operating power levels, and changes in operating conditions. This type of error continuously affects the operation of nodes until the problem is rectified. Transient errors, on the other hand, occur because of temporary external or internal circumstances, such as various random environmental effects, unstable characteristics of the hardware, software bugs, channel interface and multi-path effect. This type of error deviates nodes' operations until the effect disappears.

The effects of these deviations can be divided into two groups. The first reduces the quality of the data at a node and/or network level by deviating measurements from their actual values. This happens either by a constant drift value (i.e. they become biased), changes in the differences between a sensor measurement and the actual value (i.e. drift), or sensor measurements remaining constant regardless of changes in the actual value (i.e. complete failure). The second effect reduces the quantity of data collected by the network. This happens either by the node dropping packets, or by it constructing incorrect collaboration trees for data gathering and routing.

In some practical deployments, such as [2]-[4], an analysis of the collected network data showed a reduction in data quality which reached between 49% and 55% for that quantity of data. This reduction is caused by the deviations mentioned above and, in some cases, the network needed to be redeployed again to collect the required data because the available data were meaningless. The researchers' analysis for the data collected in these deployments expected an improvement of the deployed network's functionality of up to 51%, in terms of resource usage and the accuracy of the collected data, if a real-time monitoring tool is used to detect, report and isolate such deviations.

The work in this paper is motivated by the need to find a tool that will use a very low level of network resources but will, at the same time, detect deviations that affect the quality and quantity of data collected by the network before they have a high impact on the network's functionality. The paper proposes distributed performance network monitoring tools that detect these deviations and isolate the detected faulty nodes from the neighbourhood's functionality. The empirical and simulation experiments that were conducted showed a high level of performance of the proposed algorithm; it was also resilient to both high packet loss and environmental changes. Some of the most important results from the simulation experiments, which showed the ability of the algorithm in tracking spatial and temporal correlations between neighbour nodes, are discussed in this paper.

The remainder of this paper is organised as follows: Section 2 discusses related work in WSN functionality degradation detection; this is followed by an explanation of the algorithm approach. The fourth section discusses the results of the experiments at node level. Finally, the paper ends with a conclusion and suggestions for future work.

2. RELATED WORK

In data stream applications, such as WSNs, deviations in the data are detected by generating the residual of a monitored variable in terms of physical or analytical redundancy [5], [6]. Physical redundancy generates an estimate of the actual value of a quantity on the basis of the available redundant information. This is accomplished, by either statistical methods (such as descriptive statistics and inferential statistics), or by data fusion. The main advantage of this type of redundancy is that it is relatively easy to implement and provides a high degree of certainty. The reliability and performance of such methods mainly rely on measurement accuracy. Analytical redundancy methods, on the other hand, provide values other than direct measurements from the parameters and variables of interest using a process model such as Kalman filter, Parity relation, Principal Component Analysis (PCA) and Artificial Neural Networks (ANN). These methods are not easy to implement and they depend on the reliability of the process model.

Implementing most of the above-mentioned methods in WSNs, especially analytical methods, is either not possible, needs high spatial resource nodes, or has a high impact on the network's lifetime. As a result, researchers have tried to simplify the complexity of the above methods as a tradeoff with the method's detection accuracy or have used more than one method that increases detection accuracy.

Abdelrahman, in [10], used a fusion algorithm that utilises a Parzen estimator for calculating a probability distribution function (pdf) for measurements. This constructed pdf is based on weighted Gaussian functions whose parameters depend on the sensor's average noise level and its self-confidence. His proposed method needs relatively high resources in terms of the processing and memory required as it calculates the pdf of different neighbour node measurements and also calculates the intersecting areas of these pdfs and the intersection centroid in order to find the estimated true measurements. In addition, his algorithm needs to estimate the pdf standard deviation of each sensor node based on the standard deviation of the noise associated with an individual node. This makes his algorithm complex in term of the resources available in the existing sensor node platform which also needs a spatial resources node to implement it.

Elnahrawy, in [7], used a Bayesian approach that combines prior knowledge of sensor readings, the noise characteristics, and the observed noisy readings. This approach has proved to be robust in the detection of sensor node deviation but, because of its complexity and its model parameter calculations, the algorithm needs to be implemented on a spatial node with high resources. To reduce the complexity of the above-mentioned algorithm, Krishnamachari, in [11], proposed a distributed Bayesian algorithm for detecting and correcting these deviations. His algorithm was simple but it does not detect the gradual drift changes in the deviation of nodes. In addition, network coverage problems are not detected.

Bettencourt, in [12], proposed a distributed algorithm that would detect measurement errors and infer missing readings. His algorithm, compared with the Bayesian method, is faster in learning and has low memory storage. Unfortunately, his algorithm is not resilient to loss and drift.

From the perspective of packet losses, Ramanathan, in [13], proposed a tool (called Sympathy) that detects and debugs failures in Wireless Sensor Networks. The algorithm debugs low-level statistical changes in the network by drawing correlations between seemingly unrelated, distributed events; it also produces graphs that highlight these correlations. From this, the algorithm detects root-cause failures. Unfortunately, this tool needs to send and receive test packets to conform the detection of a failure. Moreover, it tests only the low level parameters to detect network deviations.

3. ALGORITHM APPROACH

In order to monitor the performance of WSNs, the Voting Median Based Algorithm for Approximate Performance Monitoring (VMBA) is proposed. This algorithm is a passive voting algorithm that collects its metrics directly from the application by utilising the overhearing that exists in a neighbourhood. The analysis of the proposed algorithm has been simplified such that it only requires the most recent readings of neighbour measurements at that time instant and, from it, a neighbourhood reference for the operation that has been deduced: i.e. neighbourhood measurements median. Then, the time instant data collected from that neighbourhood are compared with this reference by calculating the residual of the difference. After that, each residual is tested against a threshold whose accuracy depends on the rate of the propagation change of the monitored phenomenon at the end of the sensing range of the monitoring node. This simple approach reduces the complexity of the tracked method, reduces the memory that is required, and makes it possible to implement on existing sensor node platforms.

The algorithm is divided into four different modules; i.e. listening and filtering, data analysis and threshold test, decision and confidence control and warning packet exchange. In this section we give some definitions and then the VMBA functional algorithm presented.

3.1 Definitions

The notations used in the algorithm are listed below:

- S_i: Monitoring node used in VBAM algorithm.
- T: waiting time that depends on the reporting rate, node location, and network synchronization time.
- k: Number of neighbours.
- N(S_i): Set of S_i neighbour nodes; i.e. S_{i1}, S_{i2},…, S_{ik}.
- x_j^i: Measurement of node j received by monitoring node i.
- L_j^i: Loss counter at node j (by monitoring node i).
- C_L, C_M: Minimum and maximum limits of the sensor node that depends on its characteristics.
- D_j^i: Deviation detection counter of node j by monitoring node i.
- med_i: Neighbourhood sensed value median calculation made by monitoring node i.
- med_{i-1}: Previous neighbourhood median calculation.
- Δmed: Allowed change in the phenomenon characteristic that depends on the temporary and permanent precision of the application.
- M_i: Median deviation counter at monitoring node i.
- d_j: Deviation of node j from calculated median.
- R_i: Uncorrelated readings counter at each time interval.
- COV_j^i: Coverage problem counter of node j monitored by node i.
- N_i: Neighborhood malfunctions counter.
- Θ_M, Θ_C, Θ_d, Θ_w: Thresholds of median, coverage, distortion and monitoring window respectively whose values depend on the tolerance of the network protocol characteristics detected changes.
- ML_i: Median of L_j^i at the monitoring window size.

3.2 Algorithm Modules

The listening and filtering module is responsible for examining the validity of the received neighbour nodes measurements by filtering those readings beyond the range of the sensor's physical characteristics; as shown in the pseudo-code in Figure 1. The module then constructs neighbour readings tables and builds statistics in the loss table for neighbour readings.

This module is considered the most important module in the algorithm because it is concerned with the construction of tables that the algorithm depends on for its analysis.

1: Each S_i sense the phenomenon and wait for time T to receive N(S_i) readings
2: IF t > T THEN
3: For each un received x_j^i increment L_j^i;
4: IF $C_L > x_j^i > C_M$
5: Remove x_j^i from data set and increment D_j^i
6: Calculate med_i of the available S_i data set

Figure 1. Listening and filtering module pseudo-code

The second module; i.e. data analysis and threshold test module; tests the content of these tables. This is done by evaluating the data with regard to assigned dynamic or static limits calculated from a reference value or median. The proposed algorithm has followed a straightforward approach in calculating faulty deviations in sensor functionality. Its analysis assumes that true measurements of a phenomenon's characteristics, following a Gaussian pdf, centre on a calculated median of neighbourhood readings. Any deviation is controlled by the correlation expected at the end of the sensing range of a node, and the sensor nodes' measuring accuracy (i.e. the phenomenon's power dissipation model [14], where most of the physical processes monitored by a WSN are typically modeled as diffusion models with varying dispersion functions). This assumption is based on the fact that the standard deviation of the changes is specified by the network designed goals at the end of monitoring node sensing rang. Any sensor measurement that is not in this region is considered deviated to a degree equal to the ratio of the distance from the neighbourhood median value to the median value. This is because any external impact will affect all neighbours at the same time but to a different degree depending on theirlocation from the nodes and the position of the nodes from each other.

In addition, the second module tests the effect of any losses on the reliability of the collected data by calculating the degree of distortion in the neighbourhood data that has occurred because of its affect on the collected data accuracy and network functionality. This is done by calculating the ratio of the number of healthy readings to the total number readings as shown in Figure 2 step 8.

1: IF $|med_i - med_{i-1}| > \Delta med$
Increment M_i and let $med_i = med_{i-1}$
2: $d_j = |med_i - x_j^i|$

3: IF $d_j > \Theta_1$ and $|x_i^i - x_j^i| < \Theta_1$

4: Increment COV_j^i

5: ELSE increment R_i

6: IF $\frac{R_i}{k} > 40\%$

7: Increment N_i

8: IF $\frac{R_i}{k} * d_j > \Theta_1$

9: Increment D_j^i

Figure 2. Data analysis and threshold test module pseudo-code

The third module; i.e. Decision confidence control module; is concerned with tracking changes in the health of neighbour nodes in an assigned time window. This is set depending on the characteristics of the network application and the required response detection time. If exceeded, a request is sent to module four in order to send a detection message to the sink identifying the suspected node number, the type of fault, the number of times it has been detected and the effect of the detection on the neighbourhood data and communication. The function of this module is shown in Figure 3.

When module four receives a send request, it checks its neighbours warning exchange memory to ensure that none of the neighbour nodes have reported the same fault in that monitoring window period. If none of the neighbours have so reported, it sends a message or it cancels the request. In addition, this module tests warning messages received from its neighbours with statistics from module three.

1: Calculate ML_i

2: IF $ML_i > 60\%$

3: Send to module 4 a request to send an inefficient power consumption warning message

4: IF $M_i > \Theta_M$

5: Send to module 4 a request to send a neighbourhood malfunction due to losses warning message

6: IF $COV_j^i > \Theta_C$

7: Send to module 4 a request to send to detecting node j a coverage problem message

8: IF distortion > Θ_d & median of $L_j^i > 60\%$

9: Send to module 4 a request to send a degrade detection in network functionality message

10: IF $D_j^i > \Theta_w$

11: Send to module 4 a request to send a detection of node j malfunction message

Figure 3. Decision confidence control module pseudo-code

If the suspected node flags up a counter indication smaller than a threshold (that is, below the 30% set for the experiments), a message will be released indicating *'NO-FAUL-EVIDENCE'* regarding the received warning message. On the other hand, if the threshold is higher or equal to the threshold, then the node cancels any similar warning message request from module three during that monitoring period. This is to ensure the reliability of the warning message detection and to correct any incorrect detection that may occur because of losses or other network circumstances. Moreover, module four reduces the algorithm warning packets released by checking if any of its neighbours sent the same message at that time interval. If it been sent the algorithm will discard module three requests as shown in Figure 4 part 3.

4. PERFORMANCE EVALUATION

VMBA algorithm performance can be evaluated for eight different aspects: deviation detection in single and multi-hop levels, algorithm detection threshold, algorithm detection confidence, algorithm spatial and temporary change tracking, the impact of packet losses on algorithm analysis, resource usage at node and network levels, the impact of algorithm programming location in protocol stack, and algorithm released warning messages. In this paper, we considered the algorithm spatial and temporary changes tracking for neighbourhood sensor nodes at node level.

1: Receiving neighbour warning
   a) Check received warning with the same module 3 counter of reported node.
   b) IF module 3 counter < 30%
   c) Release *''NO-EVIDENCE-OF-FAULT'* message
   d) ELSE flag the stop sending of the same message from the node at this monitoring time.

2: Receiving module 3 request
   a) Test stop flag of received request warning
   b) IF flag = 1 discard message
   c) IF send message repeated 3 times send *'FAULT_MESSAGE_STOP'* message and flag stop fault counter.
   d) ELSE send the requested message by module 3.

3: Testing warning packet release
   a) IF detected fault returns to normal reset the same fault counters, send *'FAULT_CLEAR'* message and recalculate protocol tables.
   b) IF step 2 and 3-a alternate for the same

fault three times in a predefined monitoring window, the module send s an *'TOPOLOGY_UNSTABLEe'* warning message to report the detection and flags a permanent fault counter to stop reporting the same fault.

c) By the end of the predefined period reset all counters.

Figure 4. Warning packet exchange module pseudo-code

4.1 Data set

To evaluate the performance of spatial-temporal data deviation algorithm tracking, and test the effect of removing the confirmed faulty node on the accuracy of the neighbourhood collected data , real world data sets were used such as the Intel Lab data set [15]. This data set consist of temperature, humidity and light intensity measurements, and was collected by 54 sensor nodes deployed in the Intel lab from February 28th until April 5th 2004 for 720 hours with a scheduled communication approach used with a waking period of four seconds and a 13% duty cycle. The deployment goal of this network was to test the behaviour of a sensor network with different conditions of battery power depletion, traffic generation, and multi-hop aspects. As a result of this, the data set collected from this experiment had a lot of missing data, noise, and failed sensors, especially when the battery levels were low at the end of the experiment.

4.2 Performance Evaluation Metrics

Four metrics were chosen to analyse the results of the experiments, as shown in Table 1. The first metric is the residual value of deviated neighbours, which is the difference between the neighbourhood median and the data at a given time instance. This metric computes the diversity level of individual readings from other neighbourhood nodes. In addition, it shows the behaviour of the fault. The second metric was the weighted residual metrics: i.e. the difference between a reading and the median calculated at a time instance multiplied by the ratio of similar correlated readings compared to the total number of readings at the same time instance. This metric shows the weight of each deviation on the neighbourhood node collected data. The third metric was network performance, which is the ratio of healthy readings as opposed to the total number of nodes in the neighbourhood. This metric computes the effect of losses on the network's functionality.

4.3 Experimental Results

Figure 5 shows the functionality of the network when the data sets employed Matlab as a tool in simulating the proposed algorithm functionality on node 1 (as monitoring node), without removing suspected nodes. The figure shows fluctuation in the accuracy of the collected data and the network's performance as a result of the residual impact of deviated data on the neighbourhood data accuracy, as shown in Figure 5.4. This fluctuation continues up to the time where it becomes very heavy due to the effect of losses and the preponderance of unhealthy readings as they become the majority; as shown in Figure 5.1. Afterwards, this heavy fluctuation becomes constant when the number of permanently deviated nodes is greater than the healthy nodes at the end of the experiment (i.e. from event 680000 upwards). Normally, this heavy fluctuation does not occur in WSNs due to the redundancy that schedules the function of nodes, which makes the probability of failure occurring at the same time low. Even if this happens, it can be detected by the dramatic change in data accuracy and the increase in the weighted residual which moves up to a constant level for a long period.

The figure also shows that even there is heavy fluctuation in neighbour readings due to losses. The calculated median that the algorithm depends on in its analysis does not deviate until permanently faulty nodes become the majority (as shown in Figure 5.3). This is because the algorithm compares the difference between the new calculated median and the old stored median values with the application's permitted degree of change. This is used as a filter to remove median values that deviate from the normal as a result of the neighbourhood measurements loss impact.

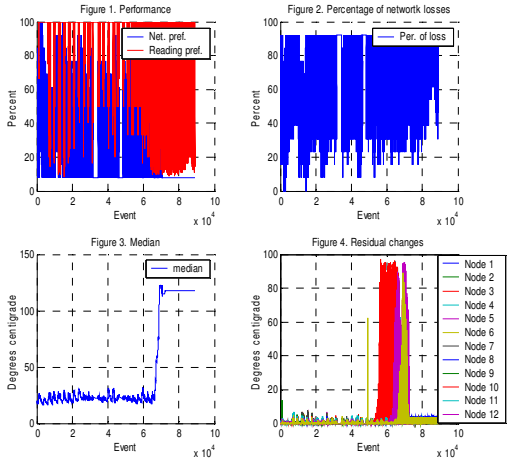

Figure 5. Threshold based on constant value

Figure 6 illustrates the proposed algorithm's calculation of the weighted residual for individual node measurements with respect to the neighbourhood median. If this figure is compared with Figure 5.4, almost the same changes in detection can be seen but with different residual values; these depend on the number of deviated readings at each time interval.

If the algorithm is allowed to isolate faulty deviated nodes in a specified monitoring window, the neighbourhood's performance is improved but with any new deviated node making a higher impact, as shown in Figure 7.1. This

happens because of the increase in the impact of the residual on the collected data as a result of reducing the number of data samples at each time interval. On the other hand, not removing the deviated reading affects the accuracy of the collected data for the period it occurs, as shown in Figure 5.1.

Figure 8 shows the proposed algorithm's weighted residual calculation when it is allowed to isolate faulty deviated readings. The figure shows clearly the low number of deviated nodes and the degree of their effect if compared with Figure 6.4.

Figures 9 and 10 illustrate a comparison between the accuracy of readings and network performance with and without removing faulty nodes. Figure 9 shows that removing the deviated nodes improved the accuracy of the data but Figure 10 shows a reduction in the network's performance due to this removal because of the reduction in the number of nodes in the neighbourhood (network performance depends on connectivity and collected data).

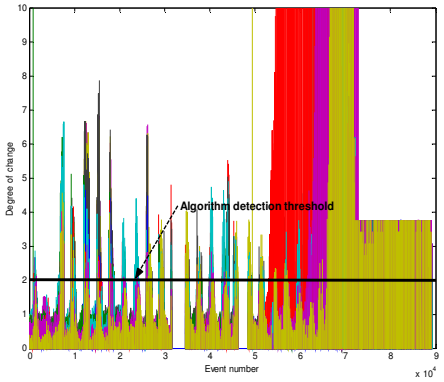

Figure 6. Threshold based on constant value

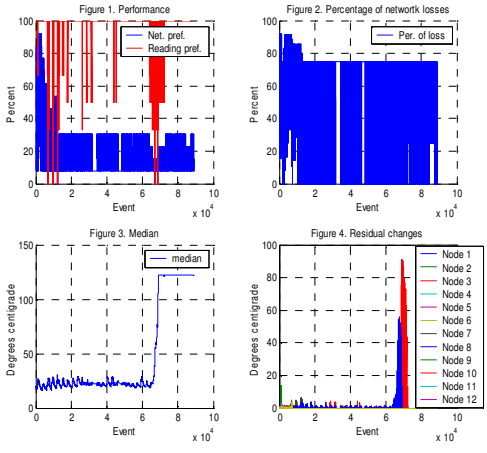

Figure 7. Threshold based on constant value

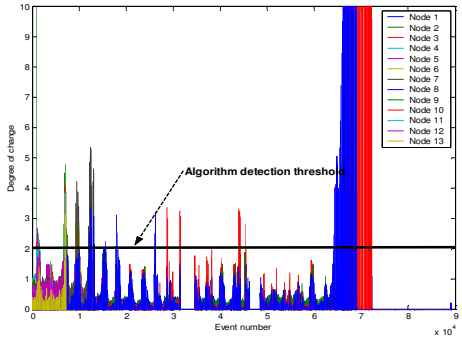

Figure 8. Threshold based on constant value

To check the effect of selecting neighbours on the algorithm's operation, 8 nodes were selected to be neighbours of Node 1 instead of 11, depending on the high correlation with Node 1 readings. These experiments showed a slight improvement in data accuracy and in network performance.

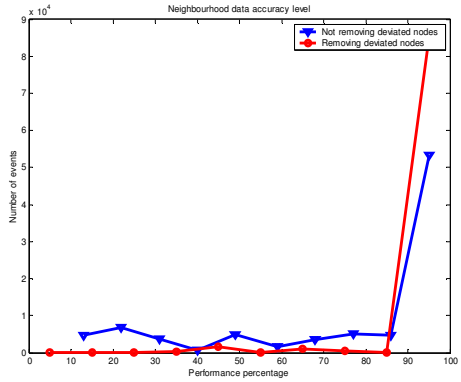

Figure 9. Reading performance

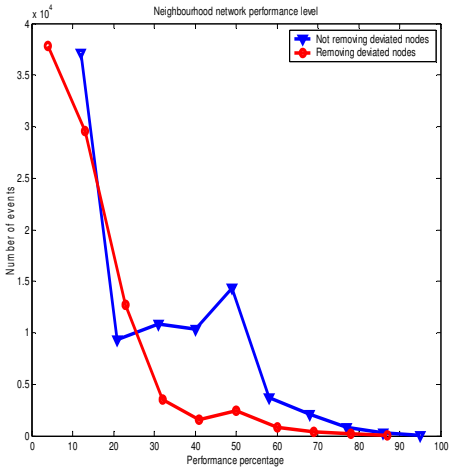

Figure 10. Network performance

The above experiments were repeated so that the analysis could be carried out on Node 2 acting as the monitoring

node. Table 2 shows the detection interval and the number of messages sent by Nodes 1 and 2 before isolating faulty nodes. As can be seen from the table, some of the nodes detected faulty nodes at the same event while others detected them at different times. The table shows that more algorithm warning messages were sent by Node 2. This was as a result of the different neighbour packet losses each node faced.

4.4 Validating the Algorithm's Detection Using Statistical Methods

Since there is no ground truth for the measured phenomenon, statistical methods were used to check the algorithm's detection of the location of faults. This was done by using the Box-Whisker method [16] (i.e. a box plot) which quantifies changes in the measurements of neighbour sensor nodes. With this method, the box represents the middle of the data while the median is the line around it at a range known as the inter quartile range.

The analysis shows that 97% of the faults detected by the proposed algorithm lie within the same outlier regions as those detected by the Box-Whisker method. The algorithm detected 108133 changes of value for all nodes, and conformed 83891 to be faulty deviations for a data set with 65% losses. Other 3% were measurements deviations which happened at the same time instance and have the same residual weight.

5. CONCLUSION AND FITURE WORK

A distributed performance algorithm is proposed here. This algorithm enables each sensor node in a sensor network to detect the Wireless Sensor Network's performance in a distributed method. It sends a warning packet to the sink reporting any detection of degradation. It tracks changes in the status of the nodes and compares them with a reference deduced from the neighbourhood.

Simulation experiments showed that 97% of outliers detected by the Box-Whisker method were detected by the algorithm. These experiments showed a similar level of detection of deviations by neighbour nodes that used the algorithm, with slight changes in detection time due to losses that each node faced in receiving its neighbours' measurements. The experiment also showed that if a detected faulty node was isolated from the neighbourhood functionality, there will be a high impact of any new measurement deviation on neighbourhood collected data accuracy.

Numerous aspects can be considered in the future in order to extend this work and improve the algorithm's functionality, such as checking the impact of the mobility of sensor nodes on the algorithm's functionality.

6. REFRENCES

[1] Kevin Ni, Nithya Ramanathan, Mohamed Nabil Hajj Chehade,"Sensor network data fault types ", ACM Transactions on Sensor Networks (TOSN), Volume 5 , Issue 3 (May 2009)

[2] Institute of Information and Control, Hangzhou Dianzi University,"A New method for Node Fault Detection in Wireless Sensor Networks", Sensors 2009, ISSN 1424-8220.

[3] J.C. Wallis, C.L. Borgman, M.S. Mayernik, A. Pepe, N. Ramanathan, and M. Hansen," Know the Sensor: Trust, Data Quality, and Data integrity in Scientific Digital Libraries", *In Procs. 11th European Conference on Research and Advanced Technology for Digital Libraries.* 2007. Budapest, Hungary.

[4] Z. Yonggang, "*Measurement and Monitoring in Wireless Sensor Networks*," PhD Thesis, Computer Science Department, University of Southern California, USA, June. 2004.

[5] Caimu Tang and Cauligi S. Raghavendra, "*Correlation Analysis and Applications in Wireless Microsensor Networks*," in *Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS 2004),* 2004, pp. 184-193.

[6] G. Indranil, V. Robbert Renesse and P. Kenneth Birman, "*Scalable Fault-tolerant Aggregation in Large Process Groups,*" in *The 2001* International Conference on Dependable Systems and Networks, 2001, pp. 433-442.

[7] C. T., S. K. and R. P., "*Fault Tolerance in Collaborative Sensor Networks for Target Detection,"* IEEE Transactions on Computers, vol. 53, pp. 320-333, March 2004.

[8] C. M. Vuran, B. O. Akan and F. I. Akyildiz, "*Spatio-Temporal Correlation: Theory and Applications for Wireless Sensor Networks,*" Computer Networks Journal (Elsevier), vol. 45, pp. 245-261, June. 2004.

[9] H. Song and C. Edward, "*Continuous Residual Energy Monitoring in Wireless Sensor Networks*," in International Symposium on Parallel and Distributed Processing and Applications (ISPA 2004), 2004, pp.
169-177.

[10] Linnyer Beatrys Ruiz, Isabela G. Siqueria and Leonardo B. Oliveira, "*Fault Management in Event-driven Wireless Sensor Networks*," in MSWiM'04, October 4-6, Venezia, Italy, 2004.

[11] O. Akan B. and I. Akyildiz F., "*Event-to-sink Reliable Transport in Wireless Sensor Networks*," Networking, IEEE/ACM Transactions on, vol. 13, pp. 1003-1016, Oct. 2005.

[12] M. Ding, D. Chen, K. Xing and X. Cheng, "*Localized Fault-tolerant Event Boundary Detection in Sensor Networks*," in IEEE INFOCOM 2005, 2005, pp. 902-913.

[13] K. Bhaskar and S. S. Iyengar, "*Distributes Bayesian Algorithms for Fult-tolerant Event Region Detection in Wireless Sensor Networks*," IEEE Transaction on Computers, vol. 53, pp. 421-250, March. 2004.

[14] Kevin Ni, Nithya Ramanathan, Mohamed Nabil Hajj Chehade,"Sensor network data fault types ", ACM Transactions on Sensor Networks (TOSN), Volume 5 , Issue 3 (May 2009)

[15] Intel Lab "*Intel Lab Experiment Data Set*," March 2007, http://berkeley.intel-research.net/labdata/.
[16] K. Ni, N. Ramanathan, M. Chehade, L. Balzano, S. Nair, S. Zahedi, G. Pottie, M. Hansen, and M. Srivastav,"Sensor Network Data Faulty Types", *Transactions on Sensor Networks. 2008.*

Table 1. Metrics calculated during the experiments

Metric	Formula used
Weighted residual values of deviated neighbours	$\frac{\lvert Node\ measurements - Median\rvert}{Median} * \left(1 - \frac{Number\ of\ neighborhood\ measurements\ agreed}{Total\ number\ of\ readings}\right) * 100$
Network performance	$\frac{Number\ of\ healthy\ readings}{Total\ Number\ of\ neighbours}$
Readings performance	$\frac{Number\ of\ healthy\ readings}{Total\ Number\ of\ readings}$

Table 2. Event number of removed detected faulty nodes

	1	**3**	**4**	**33**	**35**	**37**	**39**
Node1	--	66240	60960	15840	53760	62880	66720
Time's	--	17	15	15	37	20	18
Node2	--	66240	60960	18240	24000	24000	66720
Time's	--	13	6	29	21	16	24

Table 3. Detection comparisons between the proposed algorithm and the Box-Whisker method

	Detection Method	
	Algorithm	**Box- Whisker Method**
Detected faulty data	108133 Conform 83891	86246
Not faulty data	325639	323284