Wireless Sensor Networks Performance Measurements and Monitoring

Yaqoob J. AL-Raeisi
Electronic and Electrical Engineering
Loughborough University
Loughborough, UK
elyjya@lboro.ac.uk

Nazar Elfadil
Computer Engineering Department
Fahad Bin Sultan University
Tabuk, Saudi Arabia
nfadel@fbsc.edu.sa

Abstract— **Although the special characteristics of Wireless Sensor Networks (WSNs) help in reducing the cost of sensor nodes manufacturing and deployment, they added new challenges that directly affect the network functionality. This is increases the probability of network functionality deviation from its norm operation and affects its' collected data accuracy. Moreover, these challenges reduce the network lifetime. To insure the stability of WSN functionality at acceptable level during its operation, this paper proposes a new performance-monitoring algorithm. The proposed algorithm tracks network operation and isolate deviated network nodes before they have a high impact on network collected data accuracy or network lifetime. The purposed algorithm functionality was tested on a Berkeley (Crossbow) Mica2 sensor Motes testbed with the TinyOS '*Surge*' multihop application. The experiment results show that the algorithm achieved 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.

I. INTRODUCTION

Nodes in Wireless Sensor Networks (WSNs) are prone to temporary and permanent deviations from their norm. Their communications are not immune to any internal or external conditions because of the cheap manufacturing process used to produce them, the material they are made from, and the harsh environment they directly work in. In addition, nodes deviations increase because of the used operating system (i.e. event-driven nature), the limited use of fault-tolerant and diagnosis techniques, and used wireless communication.

Operational deviations in WSNs and their 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 affects of the above mentioned deviations on the network can be divided into three 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 affect 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. The last group affect network lifetime. This happened by constructing unideal collaboration functions between network nodes in network communication and data gathering.

Several researchers analysis of real deployed small and medum WSN; such as Nethya in [2], expected the overall network functionality improve to 51% if a real time monitoring tools used. But, such monitoring tools opposed by many network challenges that reduce its functionality and flexibility such as limited finite energy and communication resources, the network's adaptive nature to frequent changes in connectivity, link failure and node status, and many others.

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. It proposes a distributed performance measurement algorithm used at nodes and utilizes the characteristics of wireless communication. This algorithm uses both passive monitoring and active monitoring in its operations to reduce in network traffic generation, resource usage, and extend monitoring effect throughout the network.

The remainder of this paper is organized into 5 sections, as follows. In section 2 related works in WSNs functionality

978-0-7695-3737-5/09 $25.00 © 2009 IEEE
DOI 10.1109/UIC-ATC.2009.18

degradation detection is discussed, followed by explanation of algorithm approach. The fourth section explains the practical algorithm implementation on the TinyOS Surge multi-hop application. Then it discusses experimental results for a single hop. Finally, it ends with conclusions and future work ideas.

II. RELATED WORK

Researchers solved permanent and temporary node deviations in WSNs by proposing several data clearance, fault-tolerance, diagnosis, and performance measurement techniques.

Data cleaning techniques work at a high network level and consider reading impacts from a deviated sensor on multi-sensor aggregation/fusion such as in [3],[4]. Such research proposed several methods that isolate deviated readings by tracking or predicting correlation between neighbour nodes measurements. Most of this research used complex methods or models that need a high resource usage to detect and predict sensor measurements. Moreover, these techniques rectify deviated data after detecting them without checking their cause and their impact on network functionality.

Fault-tolerance techniques are important in embedded networks where it is difficult to access them physically. The advantage of these techniques is their ability to address all network levels; such as circuit level, logical level, memory level, program level and system level; but due to WSNs scare recourses these techniques have a limited usage. In general WSNs Fault-tolerant techniques detect faults in fusion and aggregation operation, network deployment and collaboration, coverage and connectivity, energy consumption, network detection, and many others [5]-[9]. Faults are detected using logical decision predicates computed in individual sensors [8], faulty node detection [6], or event region and event boundary detection [9]. These methods detect metrics either at high or low network level without relating them to each other and without checking their impact on network functionality. The main problem of these techniques is the impact of deviation on network functionality and collected data accuracy before it is detected.

Diagnosis techniques use passive or active monitoring to trace, visualize, simulate and debug historical network log files in real and non real time as discussed in [10]. These techniques are used to detect faults at high or low network levels after testing their cause for example Nithya at [2] proposed a debugging system that debugs low network level statistical changes by drawing correlations between seemingly unrelated, distributed events and produces graphs that highlight those correlations. Most of these diagnosis techniques are complex and use iteration tests for their detection. These techniques assume a minimal cost associated with continuously transmitting of debug information to centralized or distributed monitor nodes and use the screw pack technique.

Finally, performance techniques are similar to diagnosis techniques but without iteration tests and screw pack techniques. Unfortunately there is little literature and research on systematic measurement and monitoring in wireless sensor networks. Yonggang in [11] studied the effect of low network level and impact on network stability and network processing. He studied the effect of the environment conditions, traffic load, network dynamic, collaboration behavior, and constraint recourse on packet delivery performance using empirical experiments and simulations. Although packet delivery is important in wireless communication and can predict network performance, it can give wrong indications of network performance level due to collaboration behavior, and measurement redundancy which makes a network able to tolerate a certain degree of changes. Also, Zaho proposed an energy map aggregation based approach that sends messages recording significant energy level drops to the sink. Although energy consumption is very important in WSNs and all network levels affected by it, several researchers such as [12] showed in their analysis that there can be a sudden drop in node and network functionality which is not possible to detect by measuring voltage level. This drop causes network instability due to sudden route change, and more energy consumption due to usage of non-optimal routes to the destination.

III. ALGORITHM APPROCH

To solve the above discussed technique drawbacks, the Voting Median Base Algorithm for Approximate performance Measurements of Wireless Sensor Networks (VMBA) is proposed. Basically this is a passive voting algorithm that detects faults affecting the quality and quantity of WSN collected data. This is done in four 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 is presented.

A. *Definesions*

The notations used in the algorithm are listed below:

- S_i: Monitoring node using VBAM algorithm.
- T: waiting time.
- k: Number of neighbours.
- N(S_i): Set of S_i neighbour.
- x_j^i : Measurement from node j received by monitoring node i.
- L_j^i: Loss counter for node j (by monitoring node i).
- C_L , C_M : Minimum and maximum limits of the sensor node measurements.
- D_j^i : Deviation detection counter of node j by monitoring node i.

- med_i : median value at node i.
- med_{i-1} : Old median value.
- Δmed : Maximum expected phenomenon variation.
- M_i : Median deviation counter at monitoring node i.
- d_j : Deviation of node j.
- R_i : Uncorrelated readings counter.
- COV_j^i : Coverage problem counter of node j monitored by node i.
- N_i : Neighborhood malfunctions counter.
- $\Theta_M, \Theta_C, \Theta_d, \Theta_w$: Thresholds of median, coverage, distortion and monitoring window respectively.
- ML_i : Median of neighbourhood loss L_j^i.

B. 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. In addition, it constructs neighbour reading tables and builds statistics in the loss table for neighbour readings. The 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 senses the phenomenon and wait for time T to receive N(S_i) readings
2: IF t > T THEN
3: For each unreceived 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 tests the content of these tables. This is done by evaluating the data with regard to assigned dynamic or static limits calculated from an estimated norm of value of neighbourhood operation. The proposed algorithm has followed a straightforward approach in calculating faulty deviations in node functionality. Its analysis assumes that true measurements of a phenomenon's, following a Gaussian pdf, centre on a calculated median of neighbourhood operation. Any deviation is controlled by the sensor nodes' measuring accuracy and the correlation expected at the end of the sensing range of the monitoring node (i.e. the phenomenon's power dissipation model [8]). Moreover, the second module tests the effect of any losses on the reliability of the collected data. This is done by calculating the degree of distortion in the neighbourhood gathered data that has occurred because of random packets loss. This is simply done by calculating the ratio of the number of healthy readings to the total number of readings for that neighbourhood 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 warning message to the sink identifying the suspect 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 otherwise it cancels the request. In addition, this module tests warning messages received from its neighbours with statistics from module three. If the suspected node fault counter is less than a threshold (that is, below the 30% set for the experiments), a message will be released indicating *'NO-FAULT-EVIDENCE'* regarding the received warning message. On the other hand, if this counter is higher than 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 wrong 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 had been sent the algorithm will discard module three requests as shown in Figure 4 part 3.

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 $> \Theta_d$ & 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

IV. PERFORMANCE EVALUATION

VMBA algorithm performance can be evaluated on 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.

A. *Algorithm Programming in Protocol Stacks*

The algorithm was implemented on Berkeley (Crossbow) Mica2 sensor motes in a testbed and programmed in nesC on the TinyOS operation system. This is done by building the proposed algorithm on the TinyOS multi-hop routing protocol. The TinyOS multi-hop protocol consists of MultiHopEngineM which provides the over all packet movement logic for multi-hop functionality; and MultiHopLEPSM which is. used to provide the link estimation and parent selection mechanism components. These two TinyOS components were modified by added different functions of the proposed algorithm four modules as shown at Figure5.

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 stop reporting the fault 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 *'UNSTABLE_DETECTION'* 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

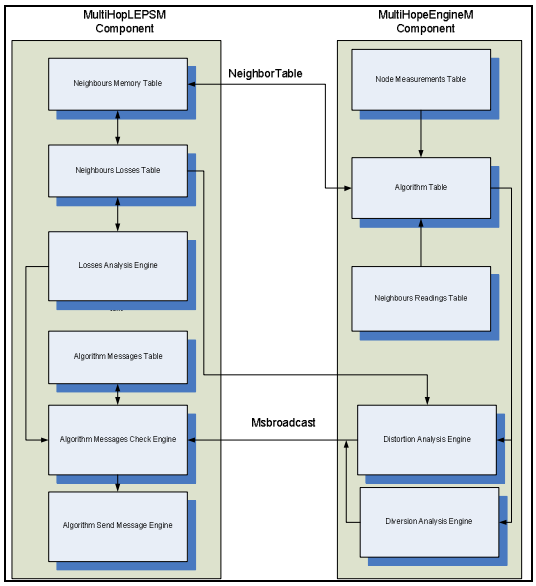

Figure 5. Functions Added to Multi-hop Components and Links between the Components

In order to send detected warning packets, a new packet type was constructed. This new packet carries the algorithm detection parameters; as shown in Figure 6. It has a total length of 20 bytes, the last 8 of them used for algorithm detection, while the first 12 follow the multi-hop protocol configuration. This is to route the released warning packet over the network.

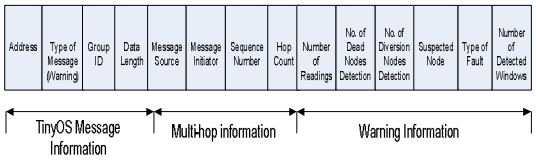

Figure 6: Algorithm Warning Message Packet.

At the algorithm detection part, the first byte carries the total number of readings; that is the number of neighbour nodes in addition to the monitoring node. The next two bytes carry the number of neighbors detected by the node as dead and diverted respectively. This is followed by a byte that carries the identification number of the detected faulty neighbour node. The byte after this carries a type of fault code; as shown in Table 1; and the final two bytes carry the number of time that the monitoring nodes detect the reported fault.

V. EXPEREMENTAL SETTING AND EVALUATION METRICS

Several experiments were conducted indoors at the High Speed Network Research Group Lab in Loughborough University to test the proposed algorithm's functionality in a real sensor network scenario. These experiments were conducted in the presence of other devices that are able to interfere with the sensor transmission and reduce the antennae performance; these offer experiments in a dynamic topology and in circumstances of high packet losses. These experiments were organised in a one-hop configuration to test the functionality of the algorithm under different scenarios such as packet losses. They were conducted with a number of nodes varying between 2 and 13, and at different distances from the sink and from each other. Nodes were measuring light intensity and were arranged in straight lines, in circular and in random formations (i.e. to have different loss percentages). They were programmed at the -10 dBm power level. A snooping node was added to the above described configuration to monitor all the packets that were exchanged, including the network routing packets, it was programmed at a 5 dBm power level.

Three metrics were chosen in order to analyse the results. The first was the average percentage of algorithm detected faults, which computes the ability of the algorithm to detect faults. The second metric computes the errors detected by the algorithm. The third metric defines the algorithm's errors in not detecting faults.

A. *Neighbourhood Performance*

As mentioned in the last section, the algorithm's warning packet contains in its fields the number of deviated nodes and the total number of nodes in the neighbourhood. This gives the expected network performance in terms of the accuracy of the collected readings and the nodes' connectivity. Figure 7 plots the measurements of the neighbourhood nodes, the detection of deviated nodes, and the degradation in the accuracy of the collected data. Furthermore, it shows that the accuracy of the collected data was degraded after inserting faults which depended on the change in the number of healthy nodes at each neighbourhood. The experiment showed that all the nodes in the neighborhood detected the same degrade of performance up to 50% deviated nodes, as shown in Figure 8. This plots the detection performance in the network of individual nodes received by the sink. When the number of deviated neighbours increased over 50%, the nodes were divided into two groups in terms of the same value of sent performance measurements (i.e. nodes 10,9,8,5 and 7,6,4,3,1) where each group of nodes were physically close to each other.

TABLE I. CODES OF DETECTED FAULTS IN ALGORITHM WARNING MESSAGES

Fault Type	Code
TOPOLOGY_UNSTABLE	0
FAULT_TYPE_DEVIATION	1
FAULT_TYPE_COMMUNICATION	2
FAULT_TYPE_COVERAGE	3
FAULT_TYPE_ENERGY_CONSUMPTION	4
NO_EVEDENCE_OF_FAULT	5
FAULT_MESSAGE_STOP	6
FAULT_TYPE_DIED	7
FAULT_CLEAR	8
NEIGHBORHOOD_MULFUNCTION	9
PROTOCOL_EFFECT	10

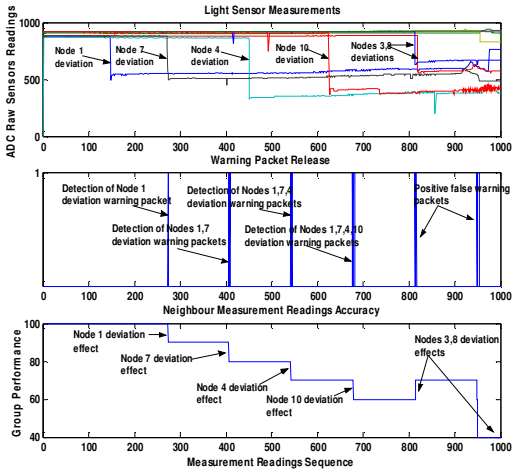

Figure 7. Network Performance Detection

B. *Effect of the Number of Faults on the Algorithm Functionality*

The total number of faulty neighbours affects the algorithm's calculations since it uses a simple median voting technique. Figure 9 depicts the percentage of the algorithm's detection, the positive and negative false detections by the algorithm versus the different percentages of faulty neighbour nodes. It shows that if deviated neighbour nodes number less

than 50%, the algorithm detected all faulty nodes. As the percentage of faulty neighbours increases above 50%, the algorithm detects only 20% of faulty nodes along with a sharp increase in negative and positive false detections.

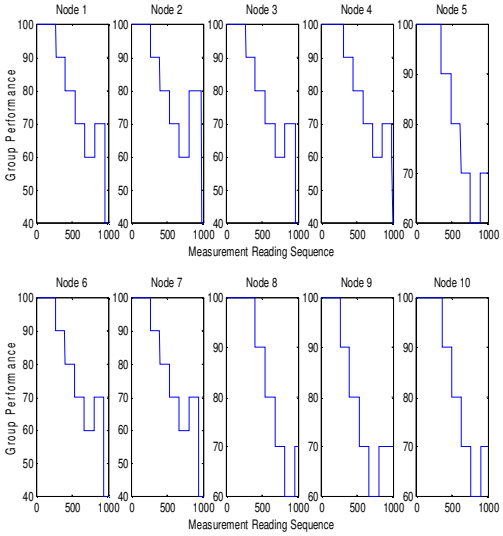

Figure 8. Network Performance Measurements with Nodes in a Group

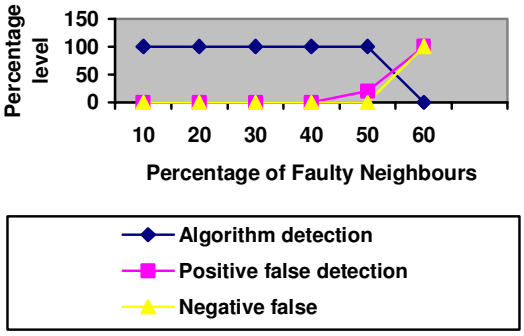

Figure 9. Algorithm Warning Message Exchange with Different Numbers of Faulty Neighbours

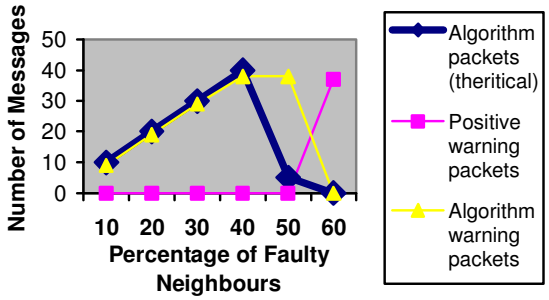

Figure 10. Number of Algorithm Warning Messages versus Number of Faulty Nodes in 10 Neighbour Scenarios

The number of messages released by the algorithm will increase linearly as the percentage of faulty deviated neighbour nodes increases, as shown in Figure 10.

VI. CONCLUSION AND FUTURE WORK

Authors proposed a distributed performance algorithm that enables each sensor node in a sensor network to detect the health of nodes in the neighbourhood and their collaborative functionality. The proposed algorithm was tested using the TinyOS *'Surge'* multi-hop application running on a Berkely Mica2 sensor nodes testbed. Empirical experiments showed that the VMBA algorithm detects 100% of faulty nodes when the number of faulty nodes was 50% or less.

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

REFERENCES

[1] D. Chen and P. Varshney K., "*QoS Support in Wireless Sensor Networks: A Survey,*" in 2004 International Conference on Wireless Networks (ICWN 2004), 2004, pp. 227-233.

[2] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler and D. Estrin, "*Sympathy for the Sensor Network Debugger,*" in The 3rd ACM Conf. Embedded Networked Sensor Systems (SenSys 2005), 2005, pp. 255-267.

[3] W. Yao-jung , M. Alice Agogine and G. Kai, "*Fuzzy Validation and Fusion for Wireless Sensor Networks,*" in ASMEinternational Mechanical Engineering Congress and RD&D Expo (IMECE2004),Anaheim, California*, USA,* 2004.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] F. Koushanfar, M. Potkonjak and A. Sangiovanni-Vincentelli, "*On-line Fault Detection of Sensor Measurements*," in *Sensors, 2003. Proceedings of IEEE,* 2003, pp. 974-979.

[9] X. Luo, M. Dong and Y. Huang, "*On Distributed Fault-tolerant Detection in Wireless Sensor Networks*," IEEE Transactions on Computers, vol. 55, pp. 58-70, June. 2006.

[10] C. Jaikaeo, C. Srisathapornphat and C. Shen, "Diagnosis of Sensor Networks," in Communications, 2001. ICC 2001. IEEE International Conference, 2001, pp. 1627-1632.

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

[12] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "*An analysis of a large scale habitat monitoring application*," in SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems. ACM Press, 2004, pp. 214-226.