1

Monitoring Wireless Sensor Network Performance by Tracking Node operational Deviation

Yaqoob J. Y. Al-raisi[1] and Nazar E. M. Adam[2]
[1]*HIS Department, Sultan Qaboos University Hospital, Oman*
[2]*Computer Engineering Department, Fahad Bin Sultan University Saudi Arabia*

1. Introduction

Wireless Sensor Network (WSN) is a very powerful tool that enables its users to closely monitor, understand and control application processes. It is different from traditional wired sensor networks in that its characteristics make it cheap to manufacture, implement and deploy. However, this tool is still at an early stage and many aspects need to be addressed in order to increase its reliability. One of these aspects is the degradation of network performance as a result of network nodes deviation. This may directly reduces the quality and the quantity of data collected by the network and may cause, in turn, the monitoring application to fail or the network lifetime to be reduced.

Deviations in sensor node operations arise as a result of systematic or/and transient errors (Elnahrawy, 2004). Systematic error is mainly caused by hardware faults, such as calibration error after prolonged use, a reduction in operating power levels, or a change in operating conditions; this type of error affects node operations continuously until the problem is rectified. Transient errors, on the other hand, occur as a result of temporary external or internal circumstances, such as various random environmental effects, unstable hardware, software bugs, channel interface, and multi-path effects. This type of error deviates node operations until the effect disappears.

These two types of error may directly and indirectly affect the quality and the quantity of data collected by the WSN. They directly affect sensor measurements and cause drift by a constant value (i.e. bias); they change the difference between a sensor measurement and the actual value, (i.e. drift); and can cause sensor measurements to remain constant, regardless of changes in the actual value, (i.e. complete failure). In addition, they affect the communication and exchange of packets by dropping them. On the other hand, the above-mentioned errors can have an indirect effect on the network's collaboration function, the construction of routing tables, the selection of the node reporting rate, and the selection of data gathering points. Analysis of the data collected by the network (in some practical deployments, such as (Ramanathan, 2004), (Tolle, 2005)), shows that these error reduces the

quality of network collected data by 49%; and in some cases, the network had to be redeployed in order to collect the data because of the failure of the monitored application. Analysis also indicate that a 51% overall improvement of WSN functionality can be expected, as well as an improvement in the quality of the collected data, if real-time monitoring tools are used.

2. Motivations

To detect and isolate operational deviations in WSNs researchers proposed 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 (Yao-jung, 2004). Such research proposes several methods that isolate deviated readings by tracking or predicting correlation between neighbour node measurements. Most of this research uses 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 which are difficult to access 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, energy event fault tolerance, reporting rate, network detection, and many others (Song, 2004, Linnyer, 2004, Bhaskar, 2004, Koushanfar, 2003, Luo, 2006). Faults are detected using logical decision predicates computed in individual sensors (Bhaskar, 2004), faulty node detection (Koushanfar, 2003), or event region and event boundary detection (Luo, 2006). 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 with 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 (Jaikaeo, 2001). These techniques are used to detect faults at high or low network levels after testing their cause. For example, Nithya at (Ramanathan, 2005) proposed a debugging system that debugs low network level statistical changes by drawing correlations between seemingly unrelated, distributed events and producing 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 send/receive test packets to conform the detection of a faultier.

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 (Yonggang, 2004) studied the effect of packet loss and their impact on network stability and network processing. He studied the effect of the environmental conditions, traffic load,

network dynamics, 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, Yonggang proposed an energy map aggregation based approach that sends messages recording significant energy level drops to the sink.
The work in this paper has been motivated by the need to find a tool that uses a very low level of network resources and detects deviations in the network's operations that affect the quality and quantity of the data that are collected before they seriously degrade the network's overall functionality and reduce its lifetime.

3. Project Methodology

3.1 Layout of manuscript

The layout of this paper is organised as follows: Section 2 includes a discussion of related work on functionality degradation detection in WSNs, followed by an explanation of the algorithm's approach. The fourth section explains the practical implementation of the algorithm in a TinyOS '*Surge*' multi-hop application; results of experiments at the network level are then discussed. Finally, the paper ends with a conclusion and suggestions for future work.

3.2 Algorithm Approach

In order to overcome the above-mentioned drawbacks, the Voting Median Base Algorithm for Approximate Performance Measurements of Wireless Sensor Networks (VMBA) algorithm is proposed. This algorithm is a passive voting algorithm that collects its metrics directly from the application by utilizing the overhearing which exists in the neighbourhood. The algorithm requires only readings of neighbours' measurements and does not rely on any information regarding global topology. This makes it scalable to any network deployment size. The proposed algorithm uses parameters found in nodes for other networking and application protocols which makes it much cheaper in terms of resource usage. It uses only the transceiver to send warning messages if there is a network performance degradation or when the node disagrees with the warning messages of neighbours.
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 is presented.

A. Listening and Filtering Module

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 Fig.1 . The module then constructs neighbour readings tables and builds statistics in the loss table for neighbour readings.

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

Fig. 1. VMBA Algorithm Module 1

B. Data analysis and Threshold Test Module

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, centred on the 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 (where most of the physical processes monitored by WSNs are typically modeled as diffusion models with varying dispersion functions). This assumption is based on the fact that random errors are normally distributed with a zero mean and standard deviation is equal to the specification of the goals designed for the nodes and the network. 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.

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

Fig. 2. VMBA Algorithm Module 2

In addition, the second module tests the effect of 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 Fig 2 step 8.

C. Decision Confidence Control Module

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 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 Fig 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 > 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

Fig. 3. VMBA Algorithm Module 3

D. Warning Packet Exchange Module

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. If the suspected node flags up a counter indication smaller than a threshold, a message will be released indicating

'NO_FAULT_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 is going to discard module three requests as shown in Fig. 4 part 3.

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.

Fig. 4. VMBA Algorithm Module 4

4. Performance Evaluation

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

4.1 Algorithm Programming in Protocol Stacks

The algorithm was implemented on a Berkeley (Crossbow) Mica2 sensor motes testbed that was programmed in nesC on 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 mechanisms. These two TinyOS components were modified by added different functions from the proposed algorithm modules as shown at Figure 5.

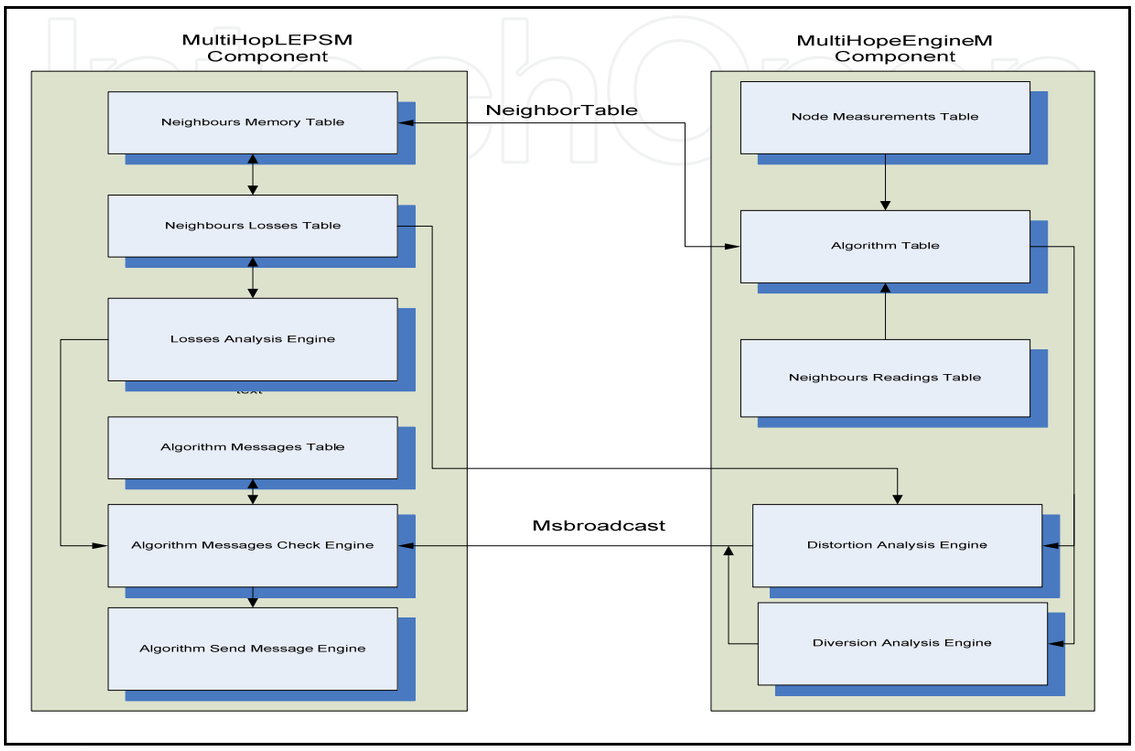

Fig. 5. Functions added to multi-hop components and links between the components

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

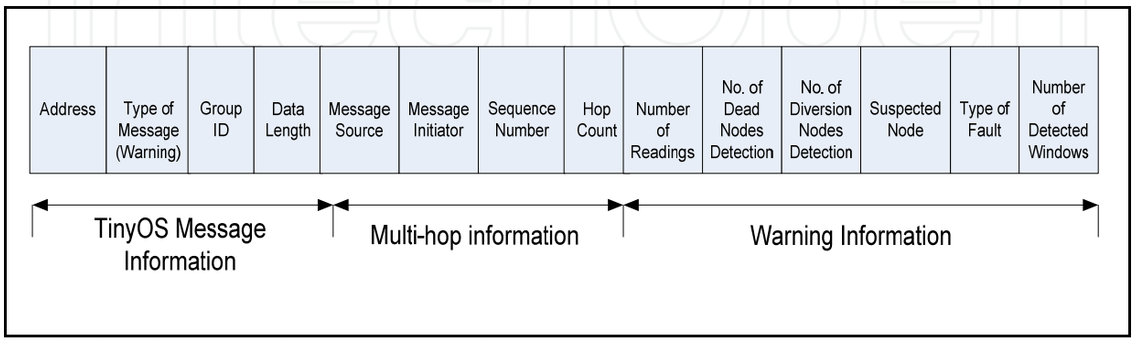

Fig. 6. Algorithm warning message packet

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 deviated respectively. This is followed by a byte that carries the identification number of the detected faulty neighbour node. The byte after this carries the type of fault codes; as shown in Table 1; and the final two bytes carry the number of times that the monitoring node detect the reported fault.

5. Experimental 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 real sensor network scenarios. 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. Some of these experiments were conducted to test the algorithm's functionality under multi-hop and highly dynamic topology configurations. These experiments used 13 Mica2 sensors, measuring temperature, distributed in an area of about 4mX5m. The nodes were programmed with the output power of -20 dBm and had top bent antennae to limit their communication range. In this configuration, the nodes were divided into two groups which overlapped in an area between them; thus, some of the nodes around the edge could not hear or communicate with each other (as shown in Figure 7). Moreover, this configuration forced the topology to be highly dynamic. This leads nodes to miss hearing each other and frequently change their multi-hop routing parents in the sink. These experiments used Mica2 nodes attached to a MIB510 programming board as a base station connected to a computer serial port. A snooping node was also added to the network setting with its power programmed to the maximum (i.e. 5dBm) in order to listen to communications among all the nodes within the network and to track packet exchanges in the multi-hop without increasing the usage of resources of the network's sensor nodes.

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_DEID	7
FAULT_CLEAR	8
NEIGHBORHOOD_MULFUNCTION	9
PROTOCOL_EFFECT	10

Table 1. Codes of detected faults in algorithm warning messages

The metrics used to evaluate the results were, firstly, the percentage of incorrectly released dead node warnings. This is the ratio of the number of false dead node detections released by the algorithm as opposed to the total number of packets released by the application. This indicates the impact of high network dynamics on the algorithm's incorrect detection. The second metric was the percentage of 'NO-FAULT-EVIDENCE' messages released by the algorithm, which is the ratio of the number of 'NO-FAULT-EVIDENCE' messages to the total number of packets released by the application. This also indicates the impact of high network dynamics but on neighbours' passive tests of incorrect detections.

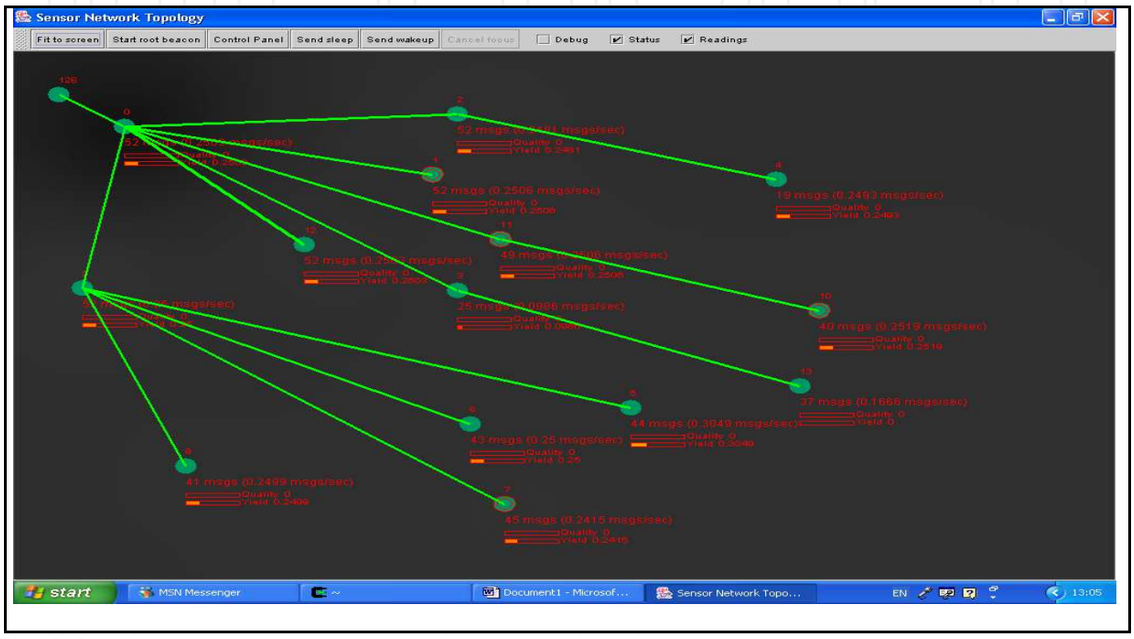

Fig. 7. Logical topology of the experiment at a time interval

These experiments tested the impact of the dead node window threshold, and monitoring window size on the algorithm's detection of dead nodes and the number of warning messages released by it in a highly dynamic network. The algorithm parameters that were tested, as shown in Table 2,and 3 were changed in different experiments to check their impact on the deductibility performance of the network and the exchange of warning packets.

Window Type	Small Monitoring window	Big Monitoring Window	Stop Reporting Window
Diversion	120 seconds (70% threshold)	480 seconds(8 minutes)	1920 seconds (32 minutes)
Distortion	60 seconds (84% loss threshold and larger than 25% accuracy of the two nodes)	240 seconds(4 minutes)	960 seconds (16 minutes)
Dead	60 seconds	240 seconds(4 minutes)	960 seconds (16 minutes)

Table 2. Sizes of monitoring windows in the experiments

Window	Small windows	Small window size	Size of Big window	Number of small window at the group	Total monitoring window size
1	Linear increased	240 seconds (4 minutes)	3 groups	4-8-12	48 minutes
2	Exponential increased			8-12-16	64 minutes
3				10-14-18	72 minutes
4				14-16-20	80 minutes

Table 3. Size of monitoring windows

5.1 Effect of Network Topology and Packet Losses on the Algorithm's Functionality

Figure 8 plots the relationship between the percentage of detected and 'No_Fault_Evidence' messages released from the algorithm for different application reporting rates. (Please note that reporting rates logs were used in the figure to plot these). The results of the experiments showed that at a 1 second reporting rate (a multi-hop protocol leads to congestion and an overflow of communication), a large amount of wrong suspected dead warnings occurred (around 3.2% of the total network packet exchange in the application). Furthermore, a large number of 'No_Fault_Evidence' replies were released from neighbour messages (i.e. around 0.5% of the total packets in the network application). Reducing the application's reporting rate to 2 seconds reduced the number of suspected dead messages; these decreased sharply to 0.5% of the total number of packets released by the network application. This happened alongside a reduction in 'No_Fault_Evidence' messages which reached around 0.01% of the total number of packets released. Thus, the number of suspected dead messages was reduced to almost 0% when the application's reporting rate was adjusted to 1 minute, along with a decrease in 'No_Fault_Evidence' messages released from neighbours. When the application's reporting rate was increased to 30 minutes, a sharp increase occurred in the number of suspected dead and 'No_Fault_Evidence' messages, as shown in the figure. Also, Figure 8 shows that, by increasing the application's reporting rate above 1 minute, the number of 'No_Fault_Evidence' messages increases so that it becomes higher than the number of suspected dead messages. This is as a result of the size of the monitoring windows and the highly dynamic network topology.

From these experiments, it can be concluded that dead node warnings will not disappear spatially in a monitored network when the network connections are highly dynamic. To reduce the number of wrong suspected dead messages, different window sizes and combinations were tested, as shown in Table 3. Figure 9 shows the relation between the percentage of correct, positive detected (wrong detection) by the algorithm, together with the negative false dead nodes for different sizes of large monitoring windows. The figure illustrates that, as the big monitoring window size increased, the confidence of the algorithm's detection of dead neighbour nodes increased, along with a decrease in the number of packets released by the algorithm. Although increasing window size will reduce the number of wrong messages, it also increases the response detection time and the probability of node failure occurring before releasing the warning message.

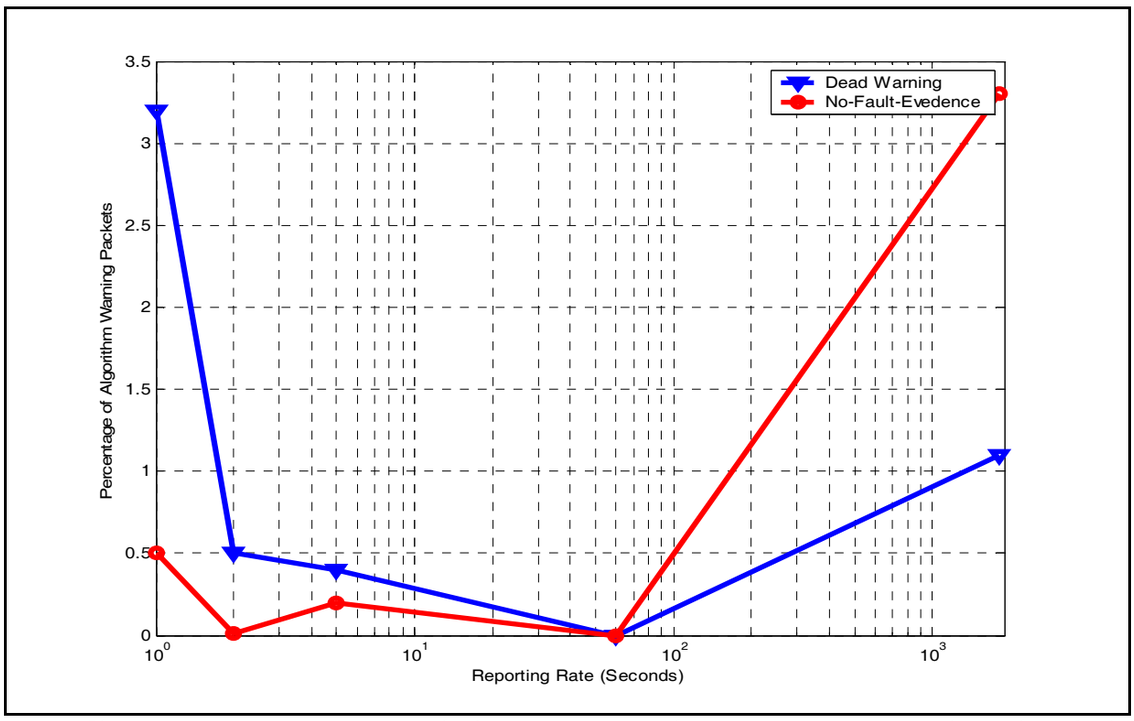

Fig. 8. Changing reporting rates with the percentage of warning messages released with the same window size

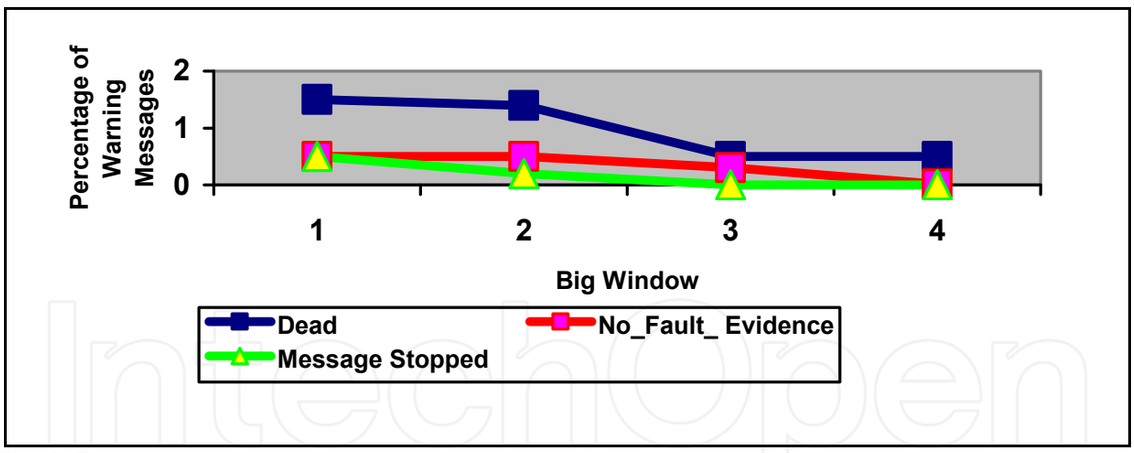

Fig. 9. Percentage of warning messages released for different window configurations

To solve this problem, the algorithm was programmed such that it would select the neighbours it would monitor; this selection depends on the amount of received packets. This configuration reduced the number of wrong packets reported by 80% and reduced 'No_Evidence_Fault' by 70%, as Figure 10 shows, but it also added additional complexity to algorithm's source code and its functionality. Moreover, there will be uncovered neighbour nodes in low density networks. In addition, the proposed algorithm was modified to send warning messages concerning the detection of connectivity problems between neighbour nodes. This makes the algorithm stop reporting a suspected node if the node is detected as

dead and if 3 clear dead messages are detected at the stop reporting monitoring window. Figure 10 plots comparisons between the percentages of the algorithm's released dead and no evidence messages in a neighbourhood with and without the modification covering connectivity problems. The figure shows that there is a reduction of 20% in the number of 'No_Fault_Evidence' messages as a result of a 34% reduction in the detection of dead packets.

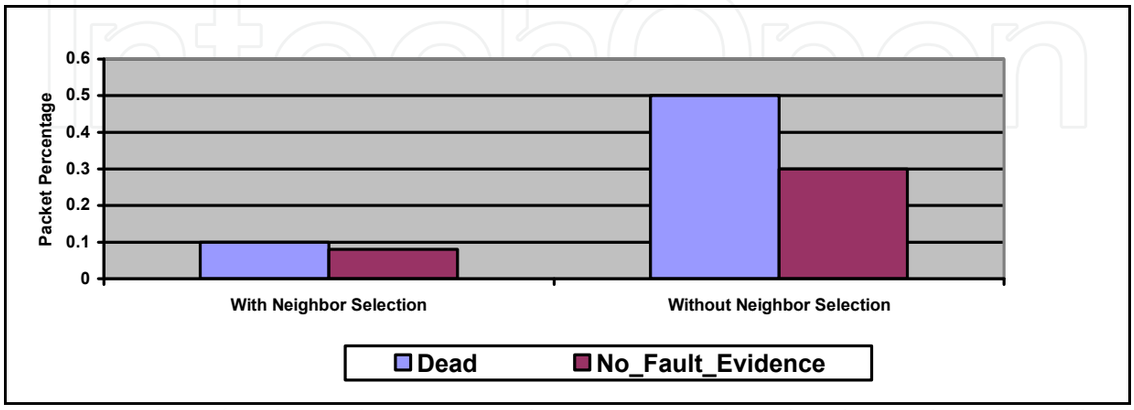

Fig. 10. Number of exchanged warning packets between selected and not selected neighbour nodes.

6. Conclusion and Future Work

We proposed a distributed performance algorithm that enables each sensor node at sensor network to detect the health of nodes at neighbourhood and their collaborative functionality. This algorithm sends a warning packet to the sink reporting any degradation detection.

The proposed algorithm tested using TinyOS 'Surge' multi-hop application on Berkely Mica2 sensor nodes testbed. These empirical experiments showed that the high loss in WSN causes proposed algorithm wrong detection of neighbour nodes aliveness and released more 'NO_EVIDENCE_FAULT' messages. This controlled by adjusting the monitoring window size and reduces the proposed algorithm wrong detection by 80% and the 'NO_EVIDENCE_FAULT' messages by 70%.

There are numerous aspects that 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. Also, it would be useful to study the impact of faulty data on individual WSN protocols and compare these results with the proposed approximate calculation that depends on the number of deviated nodes.

7. References

Elnahrawy Eiman and N. Badri, (2004). *Cleaning and Querying Noisy Sensors*, The First ACM Conference on Embedded Networked Sensor Systems (SenSys'03), pp. 78-87.

N. Ramanathan, T. Schoellhammer, D. Estrin, M. Hansen, T. Harmon, E. Kohler, and M. Srivastava,(2006). *The Final Frontier: Embedding Networked Sensors in the Soil, CENS Technical Report #68*, Center for Embedded Networked Sensing, UCLA, USA.

G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong, (2005). *A Macroscope in the Redwoods*, ACM Conference on Embedded Networked Sensor Systems (SenSys'05), pp. 51-63.

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

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

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

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

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

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

C. Jaikaeo, C. Srisathapornphat and C. Shen, (2001). *Diagnosis of Sensor Networks*, in Communications, 2001. ICC 2001. IEEE International Conference, pp. 1627-1632.

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

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

IntechOpen

Wireless Sensor Networks: Application-Centric Design
Edited by Yen Kheng Tan

ISBN 978-953-307-321-7
Hard cover, 492 pages
Publisher InTech
Published online 14, December, 2010
Published in print edition December, 2010

Over the past decade, there has been a prolific increase in the research, development and commercialisation of Wireless Sensor Networks (WSNs) and their associated technologies. WSNs have found application in a vast range of different domains, scenarios and disciplines. These have included healthcare, defence and security, environmental monitoring and building/structural health monitoring. However, as a result of the broad array of pertinent applications, WSN researchers have also realised the application specificity of the domain; it is incredibly difficult, if not impossible, to find an application-independent solution to most WSN problems. Hence, research into WSNs dictates the adoption of an application-centric design process. This book is not intended to be a comprehensive review of all WSN applications and deployments to date. Instead, it is a collection of state-of-the-art research papers discussing current applications and deployment experiences, but also the communication and data processing technologies that are fundamental in further developing solutions to applications. Whilst a common foundation is retained through all chapters, this book contains a broad array of often differing interpretations, configurations and limitations of WSNs, and this highlights the diversity of this ever-changing research area. The chapters have been categorised into three distinct sections: applications and case studies, communication and networking, and information and data processing. The readership of this book is intended to be postgraduate/postdoctoral researchers and professional engineers, though some of the chapters may be of relevance to interested masterâ€™s level students.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元

Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen

IntechOpen