A Self-Monitoring, Adaptive and Ressource EfficienT approach for Improving QoS in wireless sensor networks

Dima Hamdan [a,b], Oum-El-Kheir Aktouf[b], Ioannis Parissis[b], Bachar El Hassan [a], Abbas Hijazi[a], , Bassam Moslem[a]

a. LASTRE Laboratory, Lebanese University
Tripoli, Lebanon

b. LCIS Laboratory, INP- Grenoble University
Valence, France

***Abstract*— In Wireless Sensor Networks (WSNs), performance and reliability depend on the fault tolerance scheme used in the system. Fault diagnosis is an important part of fault tolerance. An effective diagnosis tool helps network administrators clearly monitor, manage, and troubleshoot the performance of the network. However, the design of online fault diagnosis is crucial in WSNs since many faults can easily happen and propagate. Besides, fault diagnosis put extra burden on the sensor node and it will also consume extra resources of the sensor nodes. Thus, in order to guarantee the network quality of service, it is essential for WSNs to be able to diagnosis faults efficiently. In this paper, we propose an adaptive and efficient approach for fault diagnosis in WSN called (SMART). SMART is a layer independent fault diagnosis service for WSNs. The presented service focuses on diagnosis two types of failures that are likely to happen in WSN deployments which are the node failure due to energy depletion, and the link failure due to poor connectivity with neighbors. From the design view, SMART provides to the application many tunable parameters that make it suitable for various deployment needs: energy-robustness-detection latency tradeoffs, tolerable packet loss, reports frequency etc. Simulation results prove that SMART is resource efficient while providing satisfactory detection and diagnosis accuracy**.

Keywords: wireless sensor networks; fault diagnosis; node failures; link failures;adaptive service; energy, qos tradeoffs

I. INTRODUCTION

Our work focuses on the issue of fault diagnosis in Wireless Sensor Networks (WSNs). A wireless sensor network (WSN) consists of a large number of sensor nodes spreading across a geographical area [1]. The main applications of WSNs include environment monitoring, security, area surveillance, manufacturing and industrial automation, etc. [2]. While WSNs have significant potential in a range of varied applications, they also pose many challenges in terms of fault diagnosis. Fault diagnosis in WSN is a crucial feature due to the diverse types of faults that may occur. Due to harsh environments and limited power supplies, sensor nodes are subject to faults in several layers of the system [3]. Figure 1 presents a layered classification of components in a WSN that can suffer faults. A fault in each layer of the system has the possibility to propagate to above levels. For example, a power failure of a node will cause the entire node to fail. If this node is on a routing path, the messages of other nodes relying on this routing path will not be delivered making an entire region of the network silent until the routing path is restored. Although the connectivity and functionalities of WSNs have been greatly improved by many efforts [4-7], sensor networks still suffer many network-related faults. Besides, WSN faults are usually difficult to detect and localize due to its improvisational nature and invisibility of internal running status. Therefore, the design of effective online WSN diagnosis tools, which can help network administrators monitor the network operational status and guide further improvement to or maintenance on the sensor network, has drawn significant attentions. In contrast, misdiagnosis and misapplication of a solution results not only in wasted time, energy, and communication bandwidth but also can exacerbate the problem, resulting in lower data throughput.

In this paper, we present a Self Monitoring, adaptive and Resource EfficienT service called SMART. SMART focuses on diagnosing the root causes of reduced sensor data throughput which is the first-order problem in WSN deployments. The root causes of data throughput reduction have been attributed in large part to either node failure, and/or wireless connectivity that exhibits irregular, asymmetric, and/or time-varying behavior [8].

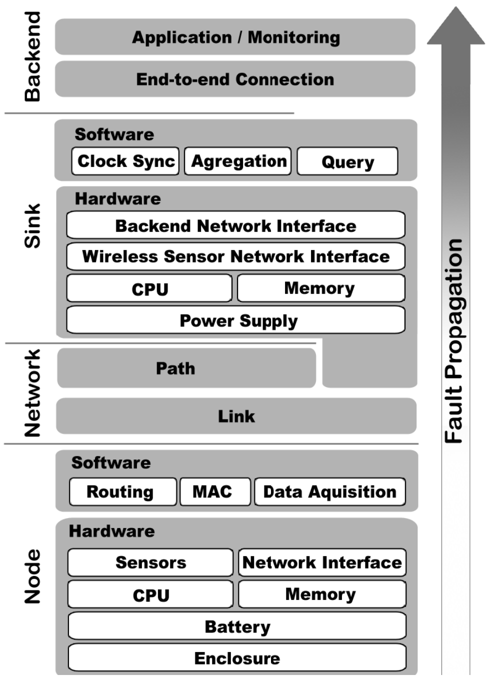


Figure 1: Fault classification and propagation [3]

SMART service tolerates some packet loss which is a deployment specific, but, when multiple neighboring nodes fail to communicate with a particular mote, a corroborated warning is sent to the base station. With power consumption in mind, SMART allows to decide the best tradeoffs between power consumption, robustness and detection latency to meet specific QoS application requirements.

The remainder of this paper is organized as follows. Section II provides an overview of the related works. Section III presents SMART service. Section IV details the simulation setup. Section V evaluates the SMART service. Section VI concludes the paper.

II. RELATED WORKS

Diagnosis of wireless sensor networks has been tackled from various perspectives. The first type aims at software debugging. Clairvoyant [9] is a GDB-like source-level tool that provides a suit of standard debugging commands. Declarative Tracepoint [10] supports a declarative, SQL-like language, allowing developers to insert a set of action-associated rules to applications at runtime. Our approach differs in that we do not aim at debugging the source code. The second type of diagnosis is sink-based [11-13]. For example, Sympathy [11] periodically collects metrics from sensor nodes to troubleshoot the data loss root-causes with a decision tree. PAD [12] leverages a packet marking strategy for constructing and maintaining the inference model. DID [13] links the process of diagnosis information collection to the root-cause deduction process to reduce the required information to be collected These centralized approaches incur huge communication overhead to the traffic sensitive sensor networks. Furthermore, some works [14, 15] proposed the passive monitoring of the network symptoms to infer the causes of the network failures. For example, LiveNet [14] and SNIF [15] overhear the network communication with a parallel capture network comprised of sniffer devices distributed over the network. Although these tools are transparent and lightweight in terms of energy consumption, they have no access to remote communication and nodes' internal state.

In order to bypass the limitations of the above approaches, we adopted the distributed fault diagnosis process over the nodes and actively inquiring the nodes about their status provides opportunities for significantly reducing the traffic generated by centralized methods and getting a complete picture about network status and nodes' internal state. Therefore, we try to make an efficient contribution in this field by introducing and analyzing a novel approach for faults diagnosis in WSNs. Our approach leverages the insights gained from existing distributed approaches [16-18] and uses a similar two-phase detection algorithm. However, our approach aims to provide the faults diagnosis service to a myriad of applications and scenarios. The presented approach SMART distinguishes by the ability to meet different QoS application requirements: power consumption, robustness and detection latency tradeoffs, tolerable packet loss, etc. Besides, SMART has the ability to adapt efficiently to the changes in the operating conditions such as the topological changes. SMART has been designed from a layered point of view, to be independent from other solutions, but at the same time easy to integrate with them. In our approach, the application has full control of the diagnosis service. The application can configure the service parameters, adjust service parameters at runtime activate/stop the service, and so on. Careful determination of the service parameter sets allows amazing results in terms of resource consumption and detection and diagnosis accuracy.

III. SMART SERVICE DESCRIPTION

SMART is designed as an independent layer under the local application layer (Figure 2) without assuming any prior knowledge of underlying routing and MAC protocols.

A. Failure detection phases

In order to reduce the false alarms rate, SMART uses a two-phase detection algorithm: the local detection phase and the consensus phase. In the local detection phase, each node monitors its neighbors (or a specific number of neighbors. Section) and waits for a certain period of time for a "heartbeat" message. To avoid the impact of transient failures, the period must be large enough to allow transmitting *L* messages from each selected neighbor. Thus, only when *L* messages are missed from a particular node, will that neighbor node be considered suspicious. In the consensus phase, the monitoring node corroborates its findings with other neighbors. This phase is particularly important to decrease the false alarm rates. Again, the period of the second phase must be large enough to allow exchanging *C* messages before deciding the final verdicts and sending them to the local application layer. In our approach, the values of *L* and *C* and other parameter values can be configured by the application layer.

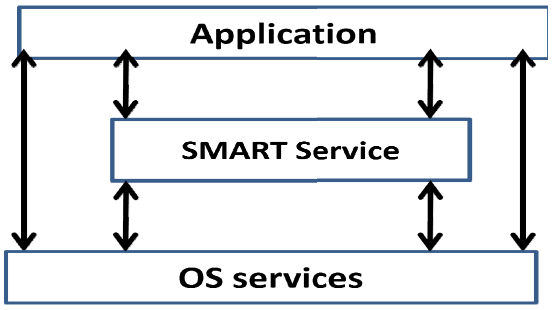


Figure 2: SMART service software architecture

B. *Duty cycle management*

In order to save energy consumption, duty-cycled operation is widely used in (WSNs), where each node periodically switches between sleeping mode and active mode [19]. At the beginning of each active period, a node turns on its radio and communicates with its neighbors. At the end of the active period, the node shut off its radio and processor for the remaining time (sleeping period) before the next duty cycle period. In our approach, all these timers are parameters of the service. They can be configured and adjusted at run-time in order to meet specific scenario requirements, particularly in terms of detection latency and energy consumption.

C. *Neighboring management*

In order to detect unresponsiveness nodes, each node listens to the messages protocol from a subset of its neighbors in the network. Therefore, each node must maintain a neighbor table where all needed information is stored and updated for each active period. Updating neighbor tables continuously makes the algorithm adaptive to network topology dynamics, such as nodes redeployment, nodes failures, nodes reconnection and so on. For each neighbor, a node stored the correspondent ID, the counters L and C, the remaining energy and the link quality. In our algorithm, a node is considered neighbor if the link quality is greater than a minimum threshold (LQ_{min}). The question here is: how many nodes should monitor any given node? What if we have a dense network or a sparse network? The minimum number of neighbors (N_{min}) must avoid unconnected nodes known as orphaned node. Monitoring other nodes in the radio neighborhood may provide more robustness against loss and topology reconfiguration. However, as the resource budget does not afford to monitor too many nodes, the maximum number of neighbors (N_{max}) must compromise to balance resources consumption and robustness degree. In our approach, the minimum and maximum numbers are configured initially by the application layer and then reconfigured at runtime. The reconfiguration is dynamic where the minimum link quality threshold is adjusted according to the neighbor table size and the actual number of neighbors.

D. *Messages Protocol*

Four types of messages are exchanged to monitor and discuss neighbor status: (1) Heartbeat message (H) broadcasted by each node to notify neighbor nodes it is alive, (2) Query message (Q) broadcasted by a node to inquire about the status of a suspicious node, (3) Reject message (R) broadcasted by a node to indicate that a suspicious node is still alive, (4) Acknowledge message (A) (unicast message)sent by a node to acknowledge the reception of Reject message about a suspicious node. SMART leverages the broadcasts of other periodic protocols that might already be running (e.g. routing advertisements, time synchronization beacons, *etc*.) and reuses them to update node neighbors' status.

E. *Messages Format*

The format of each message differs according to its role. The time to live (TTL) of the four types of message is equal to one.

1) *Hearbeat message (H)*

The size of the heartbeat message is 8 bytes (Figure 3). The message (H) is a broadcast message. It contains the following fields: the identifier of the sender node, the remaining energy level and the message counter (H). Upon receiving the message (H), the receiver node updates the entry table corresponding to the sender node (resetting the counter L, updating the energy level and link quality with the new values).

sender (2)	energy (2)	counter (4)

Figure 3: Format of the heartbeat message

2) *Query message (Q)*

The size of the query message is 10 bytes (Figure 4). The message (Q) is a broadcast message. It contains the following fields: the identifier of the sender node, the remaining energy level, the message counter and the identifier of the suspicious node. Upon receiving of the request message, the receiver sends a reject message (R) if it considers that the node in question is in a good state.

sender (2)	energy (2)	counter (4)	suspicious (2)

Figure 4: Format of the query message

3) *Reject message (R)*

The size of the reject message is 13 bytes (Figure 5). The message is a broadcast message. It contains the following fields: the identifier of the sender node, the remaining energy level of the sender node, the message counter, the identifier of the suspicious node, the current value of the counter L corresponding to the node and remaining energy level corresponds to the suspect node. Upon receiving the rejection message, the receiver node compares the counter value L included in the message stored in the table of neighbors then it saves the largest value between the two in order to reduce the latency of detection.

sender (2)	energy (2)	counter (4)	suspicious (2)
energy suspicious			

Figure 5: Format of the reject message

4) *Acknowledgement message (A)*

The size of the acknowledgement message is 10 bytes (Figure 6). The message (A) is a unicast message. It contains the following fields: the identifier of the sender node, the remaining energy level, the message counter and the identifier of the suspicious node. Upon reception of the acknowledgment message, the receiver node stops sending the reject messages concerning the suspicious node.

sender (2)	energy (2)	counter (4)	suspicious (2)

Figure 6: Format of the acknowledgment message

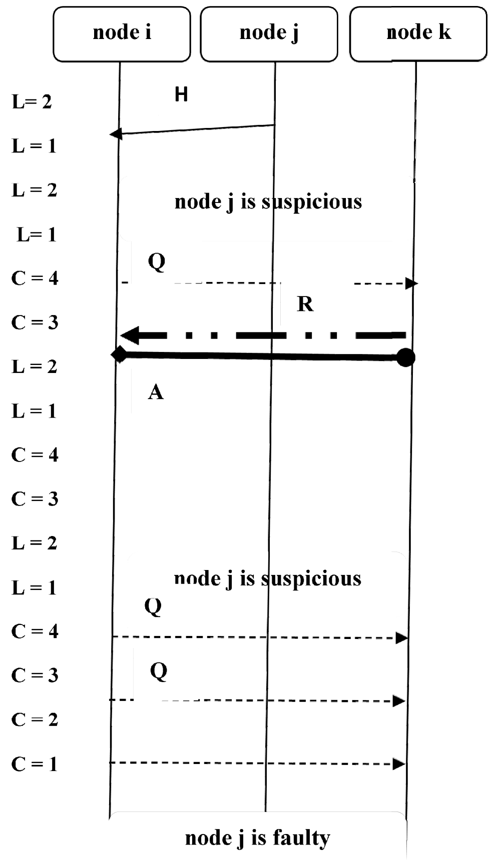


Figure 7: Fault detection process diagram

F. *Fault detection algorithm*

In the initial state, each node creates its neighbor table. Once this occurs, the local detection phase begins. In this phase, the counter *L* correspondent to each neighbor is decremented every active period that the node does not hear a message from a particular neighbor. Upon reception of H message, the counter L is reset. If the counter *L* related to a neighbor j reaches 0, the consensus phase begins. In this phase, a Q message is broadcasted to verify the node j status and the counter *C* is decremented every active period that the node does not hear a message from the neighbor j. In case of reception an R message the node j is considered alive (returning to local detection phase) and an A message is sent to the R message sender. If the counter *C* reaches 0, a final decision is made about node j status. The application is notified about the unresponsive node at the end of each reporting timer. Finally, when a report is delivered, the node j is removed from the monitoring node table (Figure 7). Note that a node is involved in different phases for each neighbor.

G. *Failures diagnosis*

The root causes reported by SMART service are: battery energy depletion, link failure, and unknown failure (Figure 8). Distinction among them is based on remaining energy information, and link quality indicators. The remaining energy is transmitted by each packet sent by a node. The link quality between two neighbor nodes is computed by a node upon receiving a packet from each of its neighbors. If the remaining energy is below a low battery threshold level, the first cause is the reported root cause. Similarly, if the link quality is below to a predefined link quality threshold, the link failure is the reported root cause.

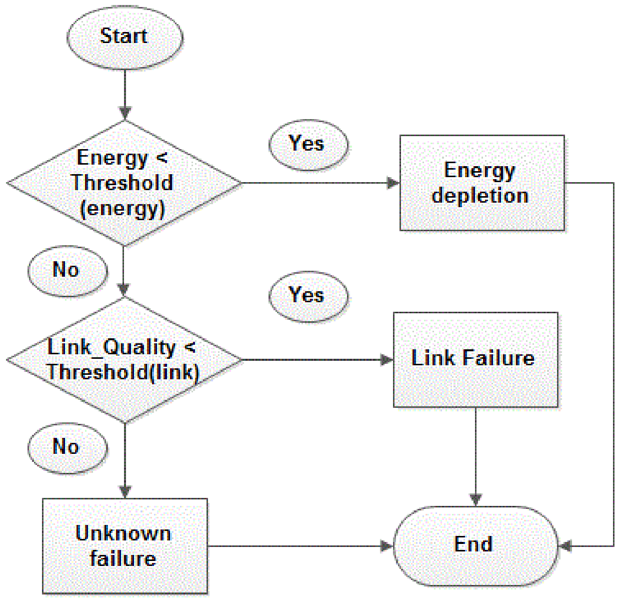


Figure 8: Fault diagnosis algorithm

H. Service Interfaces

SMART service provides several well defined interfaces to interact with the application

- Configuration command is provided to the application to determine and to adjust the service parameters (L, C, N_{min}, N_{max}, $Energy_{min}$, LQ_{min}, Duty-cycle timers, reporting timer).
- Start/Stop commands are provided to start/stop the diagnosis communication.
- Report is provided to get the report information.

Data Baud Rate	250 kbps
RAM	4 Kbytes
ROM	128 Kbyes
Radio TX mode	17.4 mA
Radio RX mode	19.7 mA
Radio Idle mode	20 µA
Supply voltage	2.7 V -3.3 V
Battery	2xAA batteries

Table I: Micaz platform features [22]

IV. Simulation Setup

To evaluate the performance of our approach, simulations are used because network faults and different deployment strategies are much easier to create and reproduce than in a real world deployment.

A. Simulator tool

We have decided to use the TinyOS 2.x [20]. TinyOS is a free and open source component-based operating system and platform targeting wireless sensor networks (WSNs). What motivates to use TinyOS is its portability to many hardware platforms and its programming structure based on components and interfaces which accommodate the diagnosis service layer structure. Then, the performance of our solution can be easily studied using the TOSSIM tool [21]. TOSSIM is a TinyOS simulation tool which simulates WSN physical and link layer features accurately. This allows validating the solution under realistic WSN deployment conditions. In the experiments we chose one of the most popular sensor platforms, Micaz (TABLE I) [22], which has been extensively used in both the research community and the industry Deployment.

B. Network models

We studied the performance of our approach based on two known deployment strategies [23]: pattern-based or grid deployment (Figure 9- (a)) Random deployment (Figure 5-(b)). When referring to a grid of size x we mean a grid containing x nodes each 10m apart. When showing different topologies/densities, we indicate that x number of nodes are placed in a 40m x 40m region according to that specific topology.

C. Energy consumption calculation model

Since network overhead consumes three times more energy than computations [24], only, the radio communication cost was considered in our study. We calculated energy consumption based on radio transmission, reception and idling periods, following the approach presented by Polastre et. al [25]. Let $Voltage$, $Current(tx/rx)$, $Time(tx/rx)$ denote the supply voltage, the current consumption in transmission or reception mode, time required to transmit or receive a data frame, respectively. The energy consumption required for transmission or reception a data frame was calculated as follows:

$$Energy\ (tx/rx) = Voltage * Current(tx/rx)\ * Time(tx/rx) \quad (1)$$

Where current is in Amperes, Voltage is in Volts, Time is in seconds and Energy in Joules. Values of the expected supply voltage, current consumption in transmission, reception mode were found in micaz platform datasheet (Table I). $Time(tx/rx)$ was calculated as follows:

$$Time(tx/rx) = \frac{Length\ of\ a\ data\ frame}{Data\ Baud\ rate} \quad (2)$$

The same formula defined in (1) was applied to calculate the energy consumed during the idle period. Let $Voltage$, $idlecurrent$, $idleTime$ denote the voltage supply, the current consumption in idle mode and the period of idle time, respectively, the energy consumed during an idle period was calculated as follows:

$$Energy\ (idle) = Voltage * idleCurrent * idleTime \quad (3)$$

Thus, Let T (n) denotes the total energy consumption at the nth duty cycle. T (n) was calculated as follows:

$$T(n) = \sum_1^n (Energy\ (tx) + Energy\ (rx) + \ Energy(idle) \quad (4)$$

Finally, Let R (n) denotes the remaining energy at nth duty cycle. R (n) was calculated as follows:

$$R(n) = initialEnergy - T(n) \quad (5)$$

Where initialEnergy denotes the initial energy level estimated from the platform datasheet [22].

D. Failure models

We studied the accuracy of SMART service by simulating two different failure models: isolated failures, and patterned failures. The first model captures independent node failures. The second model captures geographically correlated failures. Specifically, a patterned failure results in the failure of all nodes a circle of radius R. To simulate the isolated failures, randomly selected nodes were disconnected at a random time governed by a Bernoulli Process. Disconnection is simulated by deleting all links to the rest of the network in TOSSIM's radio model. Disconnected nodes were again reconnected after a specified time interval. In TOSSIM, nodes that reached a minimum energy level were halted in order to simulate insufficient node battery power. To simulate the patterned failures, the center and the radius of the faulty area were specified first, and then all nodes within that area were disconnected.

E. Parameters determination

N_{min}, N_{max} and LQ_{min} were chosen to be 4, 6 and -90 dB respectively. Based on theoretical analysis (distance between nodes, radio capabilities and the noise trace used in the simulation), these values allow avoiding the orphaned nodes. Duty-cycle and active timers were set to 1000ms and 100ms respectively. Active period is related to the maximum delay of the MAC layer (10 ms in our case) and duty-cycle period was fixed to let the nodes go to sleep for 900 ms. The reporting period (*Pr*), was selected to be equal to the duty-cycle period, so that the failures are reported as soon as they are detected. *L* and *C* were chosen to be greater than one in order to decrease the false alarm rates. Thus, only if *L* messages (not only one message) are missed from a particular neighbor, will that neighbor be considered suspicious. Similarly, *C* messages are required to consensually determine the status of the suspicious node. All the mentioned parameters can be reconfigured online in order to adjust to changing conditions. Reconfiguration process is done through sending a special message containing the new parameters values.

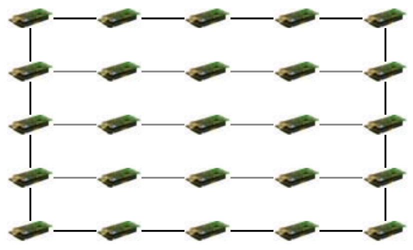
V. Performance Evaluation

A. Memory overhead

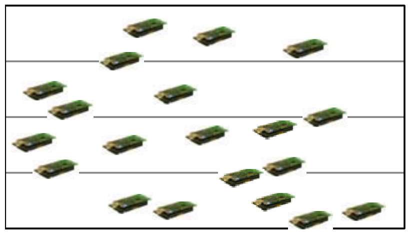
One main concern of our approach is memory cost. SMART resides on every node of the network. The required memory for TinyOS implementation on micaz platform is approximately 13 Kbytes in ROM and 1 Kbytes in RAM.

B. Energy overhead

Another important performance parameter is the energy consumption. Results shown in Figure 10 demonstrate that the presented service is lightweight in energy consumption where the average energy is around 200 mj in 8 minutes. In addition, the service scales well for different network sizes and densities. The energy consumption does not increase with network size or network density. This is because the approach only considers broadcasting information into one-node neighborhood (TTL=1). Besides, the number of neighbors a node can monitor is also upper-bounded, so that is a guarantee that the number of messages (and thus, the energy consumption) is kept upper bounded. However, the average energy consumption will increase as the duty cycle period decreases (Figure 11) leading to a decrease in the detection latency (Figure 12). The detection latency for the final judgment is (*L*+*C*)*duty-cycle period.



(a)



(b)

Figure 9: Deployment strategies in WSN (a) Grid (b) Random

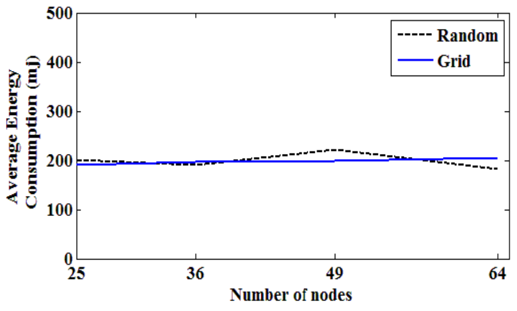


Figure 10: SMART service energy consumption

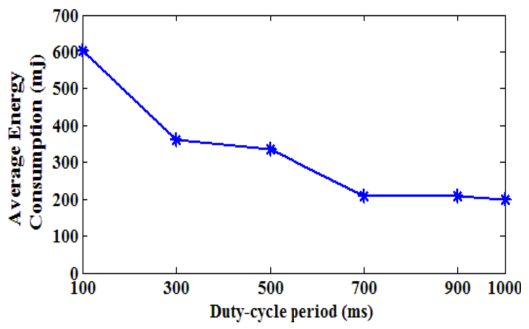


Figure 11: Impact the length of duty-cycle periods on energy consumption

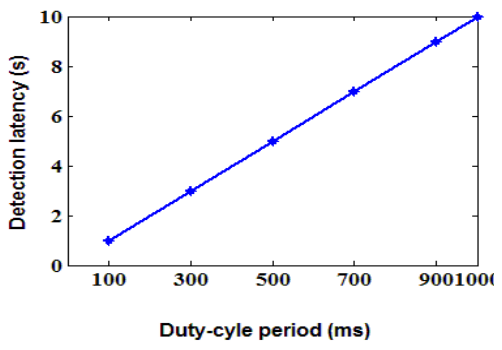


Figure 12: Impact the length of duty-cycle periods on detection latency

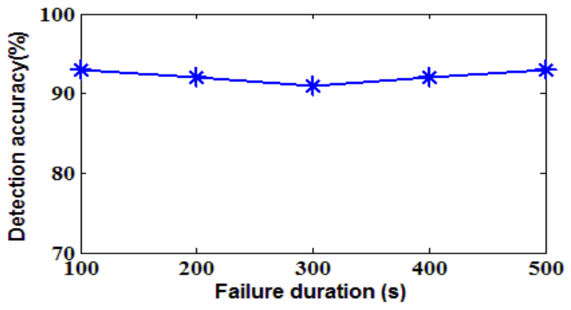


Figure 14: Detection accuracy for isolated failures with different failure durations.

C. *Detection and diagnosis accuracy*

Detection and diagnosis accuracy of the presented service was evaluated in different scenarios. In Figure 9, mean detection accuracy for isolated faults is shown to be always above 90% for different network sizes, network densities (Figure 13) and failure durations (Figure 14). Figure 15 depicts how the service performs in a 100 node grid in the presence of patterned failures. When having small faulty sensor groups, the service provides high detection accuracy but, as the faulty area increases, detection accuracy decreases. This is because live nodes can only monitor nodes at the perimeter of the faulty area, not nodes in the middle. Figure 16 shows the diagnosis accuracy for energy depletion node failure in a 25 node grid topology. SMART provides high diagnosis accuracy (around 95%) for energy depletion node failure. Figure 17 shows the diagnosis accuracy for link failure in a 25 node grid topology. SMART provides good diagnosis accuracy for link failure (around 70%)

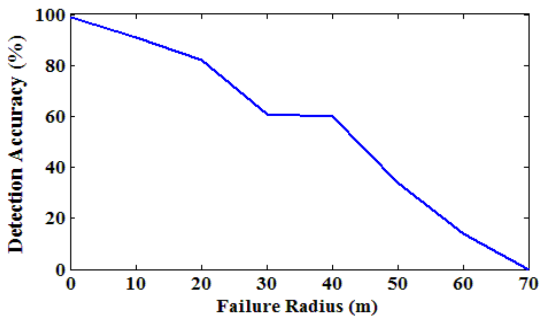


Figure 15 : Detection accuracy for patterned failures

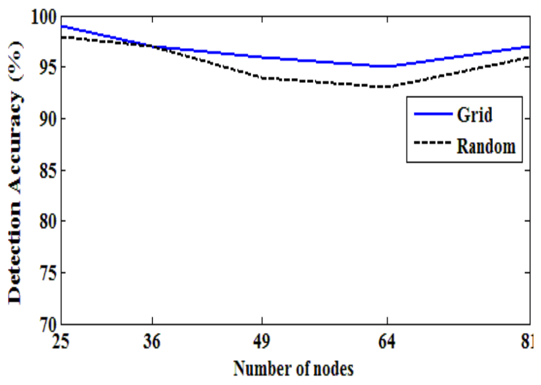


Figure 13: Detection accuracy for isolated failures with different deployment strategies

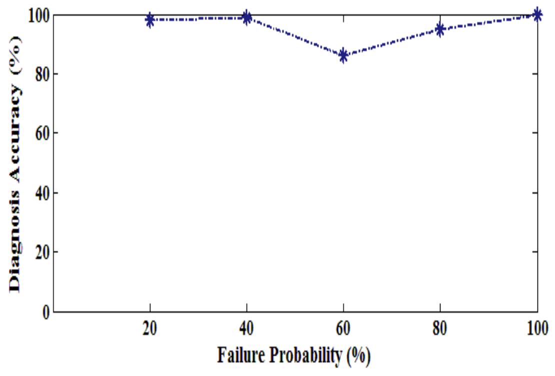


Figure 16: Diagnosis accuracy of energy depletion node failure

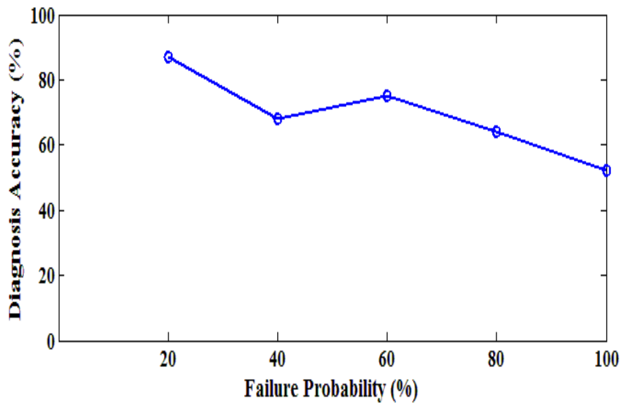


Figure 17: Diagnosis accuracy of link failure

VI. Conclusion and Future Work

To manage network faults, we developed a WSN service called SMART. SMART provides a fault diagnosis service to the application layer on each node, enabling the application to make distributed fault management decisions without burdening it with low-level detection algorithms. The service is also independent of Routing and MAC protocol layers. Beyond layer independence, SMART consumes trim resources in dense and sparse networks. It is a generous service which is suitable in a myriad of scenarios. Besides, SMART's has reconfiguration capability at run-time by using configure command through TinyOS interface. Simulation results demonstrated that the presented service is lightweight in terms of power consumption and memory overhead while providing satisfactory detection and diagnosis accuracy. Future works will focus on improving the detection accuracy of patterned failures as well as the diagnosis accuracy of the link failures.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Computer Networks 38 (2002) 393–422

[2] D. Culler, D. Estrin, M. Srivastava, Overview of sensor networks, Computer 37 (2004) 41–49.

[3] L. M. S. D. Souza, H. Vogt and M. Beigl, A survey on fault tolerance in wireless sensor networks, SAP Research. Karlsruhe University, Germany, 2007

[4] G. Hackmann, O. Chipara and C. Lu, "Robust Topology Control for Indoor Wireless Sensor Networks", In SenSys'08, November 2008.

[5] O. Chipara, G. Hackmann, C. Lu, W. Smart and G.C. Roman, "Practical Modeling and Prediction of Radio Coverage of Indoor Sensor Networks", In IPSN'10, April 2010.

[6] X. Mao, X. Li, X. Shen, F. Chen. "iLight: device-free passive tracking by wireless sensor networks". In Proceedings of the 7th International Conference on Embedded networked sensor systems (SenSys' 09), pp. 315–316.

[7] Xufei Mao, S. Tang, X. Li and Y. Sun. "MENs: Multi-user Emergency Navigation System Using Wireless Sensor Networks," in Ad Hoc & Sensor Wireless Networks, 2010.

[8] Anmol Sheth, Carl Hartung, Richard Han. A Decentralized Fault Diagnosis System for Wireless Sensor Networks. *2nd IEE International Conference on Mobile Ad Hoc and Sensor Systems. (MASS '05).* Washington, DC. September 2005

[9] J. Yang, M. L. Soffa, L. Selavo, et al., "Clairvoyant: A Comprehensive Source-Level Debugger for Wireless Sensor Networks," In Proc. of ACM SenSys, 2007.

[10] Q. Cao, T. Abdelzaher, J. Stankovic, et al., "Declarative Tracepoints: A Programmable and Application Independent Debugging system for Wireless Sensor Networks," In Proc. Of ACM SenSys, 2008.

[11] N. Ramanathan, K. Chang, L. Girod, et al., "Sympathy for the Sensor Network Debugger," In Proc. of ACM SenSys, 2005.

[12] K. Liu, M. Li, Y. Liu, et al., "Passive Diagnosis for Wireless Sensor Networks," In Proc. of ACM SenSys, 2008.

[13] Wei Gong, Kebin Liu, Yunhao Liu, Xibin Zhao, Ming Gu: "Directional diagnosis for wireless sensor networks". DCOSS 2011: 1-8

[14] B. Chen, G. Peterson, G. Mainland and M. Welsh, *LiveNet: Using "Passive Monitoring to Reconstruct Sensor Network Dynamics"*, In 4th IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS'08), Santorini Island, Greece, 2008

[15] M. Ringwald and K. Romer, *SNIF: A Comprehensive Tool for Passive Inspection of Sensor Networks*, In GI/ITG KuVS Fachgesprach Sensornetze, Aachen, Germany, 2007.

[16] S. Rost and H. Balakrishnan," Memento: A Health Monitoring System for Wireless Sensor Networks" , In 3rd Annual IEEE communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON' 06), Reston, VA, U.S.A., 2006.

[17] C. Hsin, M. Liu, "Self-monitoring of wireless sensor networks", Computer Communications 29 (2006) 462- 476.

[18] I. Urteaga, K. Barnhart, and Q. Han." Redflag a run-time distributed flexible, lightweight, and generic fault detection service for data-driven wireless sensor applications" . In IEEE PerCom, 2009.

[19] Shouwen Lai, "Duty-CycledWireless Sensor Networks: Wakeup Scheduling, Routing, and Broadcasting" thesis, 2010.

[20] TinyOS (2007), Url : http://www.tinyos.net/.

[21] P. Levis, N. Lee, M. Welsh, D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys2003), 2003.

[22] http://www.xbow.com/Products/Product/Wirelesspdf/MICAz Datasheet.pdf.

[23] Monica and Ajay K Sharma. Article:Comparative Study of Energy Consumption for Wireless Sensor Networks based on Random and Grid Deployment Strategies. International Journal of Computer Applications 6(1):28–35, September 2010. Published By Foundation of Computer Science

[24] H. Karl, A. Willis," Protocols and Architectures for Wireless Sensor Networks", John Wiley and Sons, 2005.

[25] J. Polastre, J. Hill, D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks", SenSys'04, Baltimore, Maryland, USA, November 3–5, 2004.