Intellectus
Intelligent Sensor Motes in Wireless Sensor Network

Tiziana Campana[1] and Gregory M. P. O'Hare[2]
[1]*CLARITY: Centre for Sensor Web Technologies, School of Computer Science and Informatics, Dublin, Ireland*
[2]*University College Dublin, Dublin, Ireland*

Keywords: Distributed Monitoring and Debugging, Distributed Intelligent Sensor Node, Connectivity Monitor.

Abstract: A diverse range of faults and errors can occur within a wireless sensor network (WSN), and it is difficult to predict and classify them, especially post-deployment within the environment. Current monitoring and debugging techniques prove deficient for systems which contain bugs characteristic of both distributed and embedded systems. The challenge that faces researchers is how to comprehensively address network, node and data level anomalies within WSNs through the creation, collection and aggregation of local state information while minimizing additional network traffic and node energy expenditure. This paper introduces Intellectus which seeks to develop sensor motes that are both self and environment aware. The sensor node relies on local information in order to monitor itself and that of its neighborhood, by adding a learning approach based upon perceived events and their associated frequency.

1 INTRODUCTION

Current monitoring and debugging techniques prove deficient for systems which contain bugs characteristic of both distributed and embedded systems. Such bugs can be difficult to track because they are often multi-causal, non-repeatable, timing-sensitive and have ephemeral triggers such as race conditions, decisions based on asynchronous changes in distributed states, or interactions with the physical environment. Existing debugging techniques for WSN fail to understand the entire range of WSN anomalies. Primarily they concentrate on data anomalies and often produce high traffic load within the network as a consequence of their monitoring activity. The challenge that faces researchers is how to comprehensively address network, node and data level anomalies within WSNs through the creation, collection and aggregation of local state information while minimizing additional network traffic and node energy. Intellectus seeks to imbue sensor motes such that they are both self-aware and environment-aware. The sensor node relies on local information in order to monitor itself and that of its neighborhood, by adding a learning approach based upon perceived events and their associated frequency. In Intellectus nodes maintain state information about self and their neighborhood. This is compiled by recording the occurrence of successful events. Maintaining local information results in a decrease in energy consumption and network overhead associated with sending debug information on the channel. Such periodic exchange of packets consumes more energy and creates additional overhead to the network. The node continuously listens and learns what it is happening within itself through event detection and analysis. This work is novel in that it adopts a self-detection methodology which is considered as a local computational process thus requiring less in-network communication and consequently conserving energy. This paper introduces Intellectus a scalabie and lightweight methodology for monitoring and debugging WSN. The remainder of the paper is structured as follows. Section 2 offers a pen sketch of the state of the art. Section 3 introduces the concept of monitoring and debugging in WSN post-deployment. Section 4 and 5 introduces Intellectus and its associated approach. Section 7 illustrates the approach through a case study and Section 8 concludes the paper.

2 RELATED WORK

Significant work on conventional network management and debugging tools exists (N. Ramanathan,

Campana T. and M. P. O'Hare G..
Intellectus - Intelligent Sensor Motes in Wireless Sensor Network.
DOI: 10.5220/0004268501250132
In *Proceedings of the 2nd International Conference on Sensor Networks* (SENSORNETS-2013), pages 125-132
ISBN: 978-989-8565-45-7

2005), (Min Ding, 2005), (Ann T Ta, 2004). One of the main challenges is determining how to incorporate network intelligence for detecting and localizing anomalies. The literature can be segmented into two broad approaches: *centralized* and *distributed*. The centralized approach, typically, necessitates access to more comprehensive network state information available at a back-end and, are thus, simpler to implement. The central node can easily become a single point of data traffic concentration in the network, as it is responsible for all the fault detection and fault management. This causes a high volume of message traffic and resulting energy depletion. The *distributed approach* provides more scalable and responsive anomaly detection. The *Distributed approach* (M. Yu, 2007), (Sa de Souza, 2009) fosters the concept of local decision-making, permitting a local node to make certain levels of decision before communicating with the central node. This approach is predicated on the belief that the more decisions an individual sensor can make, the less information needs to be delivered to the central node (Krishnamachari B, 2004), (N. Mehranbod, 2004), (Abhishek B., 2010), (Sa de Souza, 2009), (Andrew S. Tanenbaum, 2002). A total *distributed approach* includes *node self-detection*, *neighbor coordination*, and *clustering approaches*.*Node self-detection* schemes (Harte S., 2005), (Rakhmatov D., 2001), (M. Asim, 2010) identify possible failures by performing a self-diagnosis. The anomalies typically detected are binary data of abnormal sensor reading (Abhishek B., 2010). Failure detection via *neighbor coordination* (Min Ding, 2005), (C. Hsina, 2005) represents another example of fault management distribution. Nodes coordinate with their neighbors to detect and identify the network faults before consulting with the central node. This approach may be slow, and is error-prone and consequently may end up routing into a new neighbor that has also failed. *Clustering approaches* (Ann T Ta, 2004), decompose the entire network into different clusters and subsequently distributes fault management into each individual region. If a failure is detected, the local detected failure information can be propagated to all the clusters. Furthermore, random distribution and limited transmission range capability of common-node and cluster-heads provides no guarantee that every common-node can be connected to a cluster head. Within Intellectus sensor nodes are equipped with intelligence in order that they may undertake monitoring and debugging tasks. In Intellectus, a sensor recognizes an event, understands its internal and external behaviour, identifies its local states and advises the user-side of this local state.

3 MONITORING AND DEBUGGING WSNs

Monitoring and debugging WSNs in post-deployment demands the ability to monitor connectivity and topology change control. Connectivity is affected by changes in topology due to mobility, the failures of nodes, attacks and dynamic environmental factors. Connectivity information also helps in understanding how sensor networks operate. However, obtaining connectivity information efficiently within wireless sensor networks is inherently difficult. Firstly, the connectivity is unpredictable, and secondly as the connectivity link can vary over time, nodes need to send information to the central controller periodically, via multiple hops. The instability of the link between nodes leads to constant changes to the routing path. In addition to variable link qualities, nodes may dynamically fail and reboot. These bugs are difficult to diagnose because their only externally visible characteristic is that no data is seen at the sink, from one or more nodes. Neither of these situations would necessarily be detected by existing debugging and fault detection tools. Such tools do not address changing or newly created topologies that occur during the initial deployment of a sensor network. Individual nodes do not necessarily know anything about their local topology, and rarely know anything beyond their local topology. Such situations demand a new tool that can aid in the sighting and placement of nodes in an environment.

4 INTELLECTUS ARCHITECTURE

Intellectus from the Latin verb intellego (I understand, perceive) is a methodology which focuses on the sensor nodes ability to adapt throughout its life. Intellectus is a methodology that in part supports WSN Management. WSN management requires, (see figure 1):

- **Sensor Node Management** (Intelligent Sensor Node) - sensor node management requires planning, organizing and controlling of its own actions. Sensor motes study the observable actions involved in all phases of its life and catalogue the frequency of various events in order to construct a profile of its current and past life and how it is changing with respect to time and social context. At every instance, the sensor node gains new knowledge that can later be used in its state determination.

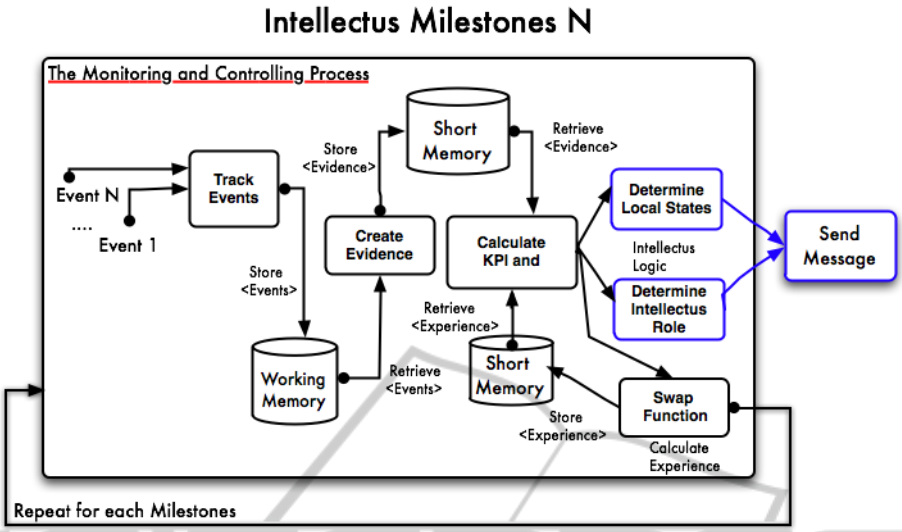

Figure 1: Overview of the Architecture of the entire Intellectus System.

- **Network Management** - necessitates the aggregation of sensor node local decisions as a basis for controlling and improving the overall network. The system will need a generalized global overview of the network as a whole and this will reside on the user-side where there is more capacity for computation.Global states are created from spatial and temporal correlation among decisions deriving from individual nodes. From this model the user will be empowered to deduce the overall health of the network.

This paper focus in Intellectus on a single node and presents a methodology by which the node controls its own actions and takes decisions based upon its local state.

5 THE MONITORING AND CONTROLLING PROCESS

5.1 Sensor Node Life Cycle Milestones

The life of each sensor node may be fragmented into discrete divisions called *milestones*. Each milestone represents a fragment of time where extra control is needed in order to effectively monitor the completion of activities. A sensor node life cycle is primarily a collection of sequential milestones. All milestones can be mapped to the following life cycle structure:

- Start of the milestone.
- Collect events in working-memory (self-memory system, see section 5.2).
- Execute Intellectus monitoring and controlling process group (see section 5).
- Close the milestone with a local decision and update of the message header accordingly.

Milestones form part of a generally sequential process designed to ensure proper control of the sensor node life cycle and the attainment of the desired monitoring and debugging outcomes. It is an iterative process, where only one milestone is executed at any given time and the execution for the subsequent milestone is carried out as work progresses on the current milestone and deliverables. This approach is useful in largely undefined, uncertain, or rapidly changing environment such wireless sensor network.

5.2 Intellectus Events and Track Events Process

Wireless Sensor Networks (WSNs) are fundamentally reactive systems. Reactive systems are computer systems that are mainly driven by events. Their progress as a computer system depends on events, which may occur unpredictably and unexpectedly at any instance, in almost any order, and at any rate. Indeed, the computations of sensor nodes are only performed as a reaction to certain events otherwise the node remains in a power conserving sleep mode. WSNs need to be able to react to a variety of events in an appropriate way. Since the handling of events is one of the primary tasks of a sensor node, Intellectus provides expressive abstractions for the specification of sensor events and their associated reactions.

```
Intellectus Events:=<Reboot>|<Internal>|<Network>;
```

In Intellects, events are categorized as either reboot, internal and network activities.

- **Reboot Events**: represent all events and states concerned with the reboot of a sensor node, for example the 3-tuple made of events (Boot, Radio Start, Serial Start).
- **Internal Events**: represent all events and states concerned with the internal activity of the sensor mote, like an internal timer that expires every few seconds, one such example would be the 2tuple of events (Timer fired, Read Temperature Done).
- **Network Events**: represent all events and states concerned with the network or external activity of the sensor motes. For the Collection Tree Protocol (CTP) this is the 3-tuple (Receive, Snoop Receive, Intercept Receive).

The Intellectus methodology adds to the event-action paradigm of the sensor mote the logic necessary to support its behavior. Intellectus adopts a discrete coding mechanism for such events, *IntellectusEventCode*. Its implementation is an enumeration variable comprising of integer values; all elements of the enumeration is associated with a specific event. The enumeration variable is the implementation of the IntellectusEventCode, which enables the labeling of discrete events.

```
1    enum {
2      FSM_BOOT = 0x00,
3      FSM_RADIO_CONTROL  = 0x01,
4      FSM_EVENT_n =  0x03,
6      ....
7    };
```

An event is characterised by its frequency, order and type. By keeping track of events coded with *IntellectusEventCode* and their associated frequency, it is possible to create prior knowledge about the ”life” of the sensor nodes. This knowledge takes the form of specific patterns that help in the design of the sensor node classifier. At each milestone each node tracks its events and their associated order of occurrence. *Evidence* represents the frequency of events in the current milestone while *experience* is the frequency of past events (in the immediately preceeding milestone). From the track events process (see Figure 1) a given node is able to summarize the current and past local states and understand its role within the network (see section 5.3). The sensor node knowledge base for storing and retrieving information includes the self-memory system comprising of:

- **Working Memory** used to collect and make available internal, reboot and network events. It offers a buffer for the collection of on-line events in each milestone.
- **Short-memory**, from short-memory Intellectus calculates current observations (evidence) and past observations (experience) of the immediately preceeding milestone (see section 5.3).

5.3 Intellectus Role

The role of a given sensor within a network is a key factor, which, can effect its local state and strongly influence its behavior. Two adjacent nodes can play very differing roles within a network and yet be exposed to the same environment but interpret and react to it in different ways. Within the Intellectus methodology each sensor node can take one of three distinct roles at a given instance: root, router and leaf. A node is aware of its role and changes to this role during the network life. The Intellectus adopts a hierarchical organization conforming to a tree like data structure. The roles can be characterized as follows:

- **Root Node**: receives data via messages from the collection tree using a **Receive interface**. .
- **Router Node**: the router node is an inner node of the tree, it is any node of a tree that has child nodes. The router node utilizes an **Intercept interface** to receive and update a packet. A collection service signal Intercept event is generated when it receives a packet for forwarding.
- **Leaf Node**: the leaf node is an external or leaf node, it is any node that does not have children and thus is, any node that does not use both the **Receive interface** and the **Intercept interface**.

Each Intellectus node, at a given instance, has one clear role based on interface and event activity. Of course during the network deployment a sensor node may change its role in the network reflecting topology change.

5.4 Key Performance Indicator KPI Calculation

Change control is the process of reviewing all changes in node behaviors (between current and the immediately proceeding past state), controlling the nature of these changes, and reflecting such changes as updates to the state of node. Intellectus utilizes Key Performance Indicators (KPI), internal node indicators, as a means of providing effective and efficient mechanisms by which to control and manage changes in the pattern of events. KPIs assist in identifying, documenting and controlling changes in the life cycle of a sensor node. And accomplish two main objectives:

- Establish a method by which to consistently identify changes to the past life cycle of the node,

and to assess the value and effectiveness of those changes.

- Provide mechanisms for the continuous validation of the life cycle of a node by considering the impact of each change.

For each milestone and each node a set of KPIs associated with given events will be calculated. The set selected are simple to implement and yet sufficient, and not needlessly burdensome in terms of memory usage, computational overhead and associated energy depletion, and support effectively the necessary determination of local state (see section 5.5). Intellectus Internal events are those events and states concerning the internal activity of a sensor node, like an internal timer that expires every few seconds. **The KPI of the internal event group** is calculated as follows: MI (Mean Internal), MID (Mean Internal Deviation) and IDR (Internal Deviation Relative). The *Internal* is a summation of Internal events (e.g. Timer and Read Temperature) during the past periods of time (past milestones). In general, *P(Timer)* is the probability of occurrence of *Timer* event in a milestone.

$$Internal_n = \sum_{i=1}^{n} (P(Timer)_i + P(ReadT)_i) \quad (1)$$

The *MI* is an average of internal events during the node 's life cycle. Past periods are obtained as a summation of past milestones.

$$MI_n = \frac{\sum_{i=1}^{n} Internal_i}{\sum_{i=1}^{n} timeSlot_i} \quad (2)$$

MID shows how much variation or "dispersion" exists from the average MI (or expected value). A low deviation indicates that the data points tend to be very close to the mean, whereas high deviation indicates that the internal events deviate significantly from the mean. Changes to the normal node behavior invariably manifests itself by high MID values and this will be used to check its own behavior and in formulating decisions regarding local states.

$$MID = (\sum_{i=1}^{n} Internal_i) - (\sum_{i=1}^{n-1} MI_i) \quad (3)$$

IDR is the difference between *Internal* in two consecutive milestones. The variation between *Internal evidence* in two consecutive milestones.

$$IDR = (\sum_{i=1}^{n} Internal_i) - (\sum_{i=1}^{n-1} Internal_i) \quad (4)$$

While *MID* shows a variance between current internal events and all internal events in past milestones, in contrast *IDR* shows a variance between current internal events and the previous milestone.

Network group events capture network activity of the sensor motes, akinned to the interfaces (e.g. Receive or Intercept) of Collection Tree Protocol (CTP). **The KPI of network group** are divided into two categories based upon the role of the sensor within the network:

- **Router Role**: uses the *Intercept interface* to capture the frequency of *Intercept* event. Thereafter the node calculates: MNI (Mean Network Intercept), MNID (Mean Network Intercept Deviation) and NIDR (Network Intercept Deviation Relative) calculated in an analogous manner for *Internal* event group but with *Intercept* event frequency.
- **Root Role**: uses *Receive interface* to capture the frequency of *Receive* events. The node thereafter calculates: MNR (Mean Network Receive), MNRD (Mean Network Receive Deviation) and NRDR (Network Receive Deviation Relative) calculated as for *Internal* event but with *Receive* event frequency.

Internal and Network KPI analysis examines the difference between what was done in the past and what was executed, progressive elaboration is made and details of the change are discovered over time.

5.5 Determine Local States

The Intellecus methodology is based on the simple state that only sensor nodes with their local information can provide the correct monitoring or debugging information to the sink. Each node can determine and distinguish between six local decisions based on evidence, experience, role and KPIs. A high level characterization of the rules used to determine such states is as follows:

- **Reboot State**; the node is rebooted. In the working-memory the reboot event appears:

```
Reboot:= (Frequency of Reboot Events )>0;
```

- **High Internal and Low Network Activity State**; the node has low network activity but it is not isolated from the network. The MID (see section 5.4) increases but there are still network events in the sensor node:

```
High Internal:= IDR>0 and MID>0 and
(Evidence: Frequency of Network Events>0;
```

- **Isolated State**; the node has no network events. It is isolated from the rest of network.

```
Isolate:= IDR>0 and MID>0 and
(Evidence: Frequency of Network Events)==0;
```

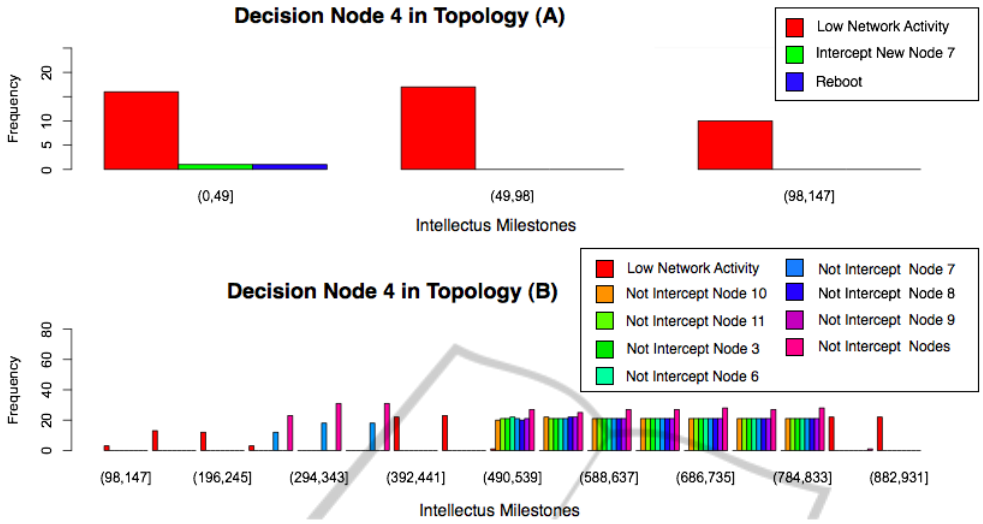

Figure 2: Node 4 - Dynamic Topology.

- **Not Receiving (or Intercepting) State**; the node no longer does receives (or intercepts) messages from its sub-tree nodes. In the working-memory, the network events have frequency zero in the current milestone.

```
Role Root, Look Evidence:
NotReceive:=(Frequency of Receive Event)==0;

Role Router, Look Evidence:
NotInterc:=(Frequency of Intercept Event)==0;
```

- **Good Network Activity State**; the message received (or intercepted) increase. The NRDR (or NIDR) increasing (see section 5.4).

```
Role Root, Look Evidence:
Good:=(Frequency of Receive Event)!=0 and
NRDR>0;

Role Router, Look Evidence:
Good:=(Frequency of Intercept Event)!=0 and
NIDR> 0;
```

- **Not Receive (or Intercept) Messages from Node X**; the node does not receive (or intercept) messages from a specific node. The NRDR (or NIDR) decreases (see section 5.4) and the frequency of messages received (or intercepted) from a specific node is zero.

```
Role Root, Look Evidence:
Not Receive ID:= NRDR< 0 and MNRD <0;
Network ID:=(Search for Node ID with
Frequency of Messages Received)==0;

Role Router, Look Evidence:
Not Intercept ID:=  NIDR<0 and MNID<0;
Network ID:=(Search for Node ID with
Frequency of Messages Intercepted)==0;
```

Intellectus incorporates spatial correlation across local states associated with given nodes in determining potential faults and global network states. Using the spatial correlation information from multiple nodes can result in a higher-fidelity model or better estimates from sensor local state, and hence, more accurate and robust fault detection.

5.6 Closing Process

A milestone is generally concluded and formally closed with two deliverables: a) an update of the permanent memory with a new state and role of the sensor node, b) an update of the header message of the next message to be sent to the user-side thus communicating efficiently its local state. Detailed description as to how the user-side fuses inputs from the collective of sensors in order to extract information pertaining to the overall wellness of the network is beyond the scope of this paper.

6 CASE STUDY

The Intellectus methodology has been evaluated through the use of the TOSSIM simulator (Philip Levis, 2003). In order to evaluate network change, faults are injected into a number of topologies. By way of illustration we will consider topology A, comprising of 12 nodes. This represents the *ground truth* which helps us better understand the performance of Intellectus. Each node executes a simple data-collection application of TinyOS. The test was run for 3.5 *min*. Each node engages in delta reporting

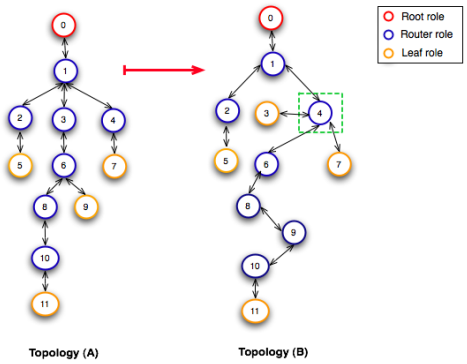

Figure 3: Dynamic Topology I.

whereby it will only report to the central controller what has changed in its local states. Figure (2) shows the local states of node 4. In topology (A) in the initial configuration, node 4 is a router node and intercepts only one node (node 7). A topology change is subsequently dynamically injected into the network resulting in topology (B). In the new topology, Node 4 becomes a router of seven nodes (nodes 3, 6, 7, 8, 9, 10 and 11). In topology (B), dynamic connectivity results in an increase in network activity. In fact, the node registers an increase in network activity as new links are created. Caused by changes in topology, such links are not stable and disconnected. Node 4 in topology (B) fails to intercept the "Intercept New Nodes 3,6,7,8,9,10 and 11" states due to local link changes that create discontinuity in intercepting such messages. During the unstable links prior to milestones [442-539] , node 4 reported the local states "Not Intercepted Nodes 3, 6,7,8,9,10 and 11". In the successive milestones post [442-539], the node 4 determines its local state to be 'Good network activity'. Correlations of local states associated with different nodes are used in computing the change within the network and thereafter global network states. Spatial and temporal correlations can provide a global view of the network and thus assist with overall network monitoring.

7 CONCLUSIONS

Failures are inevitable in wireless sensor networks due to inhospitable environments and unattended deployment. Therefore, it is necessary that network failures are detected in advance and appropriate measures are taken to sustain network operation. Intellectus provides a framework through which to address network, node and data level anomalies, future work will access the Intellectus algorithm in a broader range of topology change scenarios.

ACKNOWLEDGEMENTS

This work is supported by Science Foundation Ireland (SFI) under grant 07/CE/1147 and IRCSET under a Ph.D Scholarship.

REFERENCES

N. Ramanathan, K. Chang, R. Kapur, L. Kapur, E. Kohler, D. Estrin, *Sympathy for the sensor network debugger*,In SenSys, 2005

Kebin Liu, Minglu Li, Mo Li, Zhongwen Guo, Yunhao Liu, *Passive Diagnosis for Wireless Sensor Networks*,In SenSys, 2008

Abhishek B. Sharma, Leana Golubchik, and Ramesh Govindan, *Sensor Faults: Detection Methods and Prevalence in Real-World Datasets*, in ACM Transactions on Sensor Networks (TOSN), 2010

M.Yu, H.Mokhtar, and M.Merabti *A Survey On Fault Management In Wireless Sensor Networks*, In Proceedings of the 8th Annual PostGraduate Symposium on The Convergence of Telecommunications, Networking and Broadcasting Liverpool, UK, 2007

Luciana Moreira Sa de Souza, Harald Vogt, Michael Beigl, *A Survey on Fault Tolerance in Wireless Sensor Networks*, In PhD Dissertation, 2009

M. Asim, Hala Mokhtar, Madjid Merabti, *A Self-Managing Fault Management Mechanism for Wireless Sensor Networks*, In International Journal of Wireless and Mobile Networks (IJWMN) Vol.2, No.4, November 2010

B. Khelifa, H. Haffaf , Merabti Madjid, and David Llewellyn-Jones, *Monitoring Connectivity in Wireless Sensor Networks*, In International Journal of Future Generation Communication and Networking Vol. 2, No. 2, June, 2009

Min Ding, Dechang Chen, Kai Xing, Xiuzhen Chen, *Localized fault-tolerant event boundary detection in sensor networks*, INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, Vol. 2 (2005), pp. 902-913

Harte S., Rahman A., Razeeb K.M., *Fault tolerance in sensor networks using self-diagnosing sensor nodes*, Intelligent Environments, 2005. The IEE International Workshop on (Ref. No. 2005/11059)

C. Hsina, Mingyan Liub, *Self-monitoring of wireless sensor networks*, Computer Communications, vol.29, pp.462478, 2005

Andrew S. Tanenbaum, Maarten Van Steen, *Distributed Systems: Principles and Paradigms (2nd Edition)*, Prentice Hall, 2002

Krishnamachari B., Iyengar S., *Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks*, IEEE Transactions on Computers, 53:241-250, March 2004

Rakhmatov D., Vrudhula S.B.K., *Time-to-failure estimation for batteries in portable electronic systems*, in Proceedings of 2001 international Symposium on

Low Power Electronics and Design, pages 88-91, 2001

N. Mehranbod, Masoud Soroush, Chanin Panjapornpon, *A method of sensor fault detection and identification*, Journal of Process Control 15, pages 321-339, 2004

Ann T. Tai and Kam S. Tso and William H. Sanders, *Cluster-Based Failure Detection Service for Large-Scale Ad Hoc Wireless Network Applications,* in Proceedings of the International Conference on Dependable Systems and Networks (DSN, page.805–814, 2004

Philip Levis, Nelson Lee, Matt Welsh, and David Culler, *TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications*, in Proc. of the ACM SENSYS, pages. 126-137, 2003