International Conference on Automation, Control, Engineering and Computer Science (ACECS'14)
Proceedings - Copyright IPCO-2014, pp.95-101
ISSN 2356-5608

A Real Time Centralized Fault Detection Application in Wireless Sensor Networks

Marwa SAIHI
MACS (Research Unit of Modeling, Analysis and Control Systems)
National School of Engineers of Gabes
Gabes,Tunisia
Saihimarwa@yahoo.fr

Boussaid BOUMEDYEN
MACS (Research Unit of Modeling, Analysis and Control Systems)
National School of Engineers of Gabes
Gabes,Tunisia
Dr.boumedyen.boussaid@ieee.org

Ahmed ZOUINKHI
MACS (Research Unit of Modeling, Analysis and Control Systems)
National School of Engineers of Gabes
Gabes,Tunisia
Ahmed.zouinkhi@enig.rnu.tn

Mohamed Naceur ABDLKRIM
MACS (Research Unit of Modeling, Analysis and Control Systems)
National School of Engineers of Gabes
Gabes,Tunisia
Naceur.abdelkrim@enig.rnu.tn

Abstract— **Wireless sensor networks (WSNs) are usually composed of autonomous nodes. Each networked node constructs its own neighbor table, routing table, and schedules internal tasks on its own. However, this process may be slow, and is error-prone because the constrained sensor nodes cannot be expected to constantly police all their neighbors and consequently may end up routing into a new neighbor that has also failed. Centralized fault detection based on clustering approach has become an emerging technology for building scalable and energy balanced applications for WSNs. In our work, we try to apply the Distributed Fault Detection (DFD) algorithm in a True Time simulator based on clustering model where the cluster head or the sink node detects the suspicious nodes by exchanging heartbeat messages in active manner. By analyzing the collected heartbeat information, the cluster head finally identifies failed nodes according to a pre-defined failure detection rule..** *(Abstract)*

Key Words: Wireless Sensor Networks, Fault Detection, Centralized Approach, Decentralized Approach, Real Time Application

I. INTRODUCTION

Recent increasing growth of interest in WSNs has provided us a new paradigm to design wireless environmental monitoring applications in the 21st century. However, the nature of these applications and network operational environment has also put strong impact on sensor network systems to maintain high service quality [5]. As one of the key technologies involved in WSNs, node fault detection is indispensable in most WSN applications because Sensor faults may reduce the quality of monitoring and, if remaining undetected, might cause significant economic loss due to inaccurate or missing sensor data required for structural assessment and life-cycle management of the monitored structure [6]. The problem of distributed decision fusion in wireless sensor networks has received considerable attention recently because of many important applications. In WSN, sensor nodes are prone to damage. Therefore, the decision fusion rules employed in WSN, need to be fault tolerant [3]. However, most of the proposed applications, such as environment monitoring, are non-real-time [1]. For other real-time related applications, such as ZigBee, the built-in control functions are limited. For example, ZigBee mesh network is designed mainly for office automation. It provides ways to efficiently manage building energy consumption as well as fire alarm systems, which entails real-time responses. However, the characteristics of a wireless sensor network make it unsuitable for real-time applications. First of all, each sensor has its own task set and task scheduling. It is difficult to have all the sensors to cooperate to support a network wide real-time application. Secondly, the reduce resource which makes each node hard to provide a sophisticated cooperative mechanisms. Thirdly, the dynamic nature of a wireless network may make an existing schedule invalid [1]. We classify the existing failure detection approaches in WSNs into two types: centralized and decentralized approach [7]. We will discuss the advantages and inconvenient of centralized and decentralized approaches. We argue that centralized approach is simple and practical in real world. Our argument is further supported by the simulation results.

International Conference on Automation, Control, Engineering and Computer Science (ACECS'14)
Proceedings - Copyright IPCO-2014, pp.95-101
ISSN 2356-5608

The remainder of the paper is organized as follows: In section 1, we describe a Wireless Sensor Network. Section 2 presents two different fault detection approaches in a Wireless Sensor Network and compares both approaches. Section 3 defines the real-time tasks, presents how to model a real-time network application in the DFD approach and gives simulation results. The simulation and interpretation results are gathered in section 4.

II. Wireless Sensor Networks

Wireless sensor networks are emerging applications of pervasive computing, consisting of many small, low- power, and intelligent sensor nodes (or motes) and one or more base stations. Sensor nodes gather information in diverse settings including natural ecosystems, battlefields, and man made environments [8] and send the information to one or more base stations. Sensor nodes work under severe resource constraints such as limited battery power, computing power, memory, wireless bandwidth, and communication capability, while the base station has more computational, energy and communication resources. The base station acts as a gateway between sensor nodes and the end user. Sensor network applications use a data-centric approach that views a network as a distributed system consisting of many autonomously cooperating sensor nodes [8], any of which may have a role in routing, data gathering, or data processing. Every node will communicate through other nodes in a sensor network to produce information-rich results (e.g., temperature and soil-moisture in a certain region of the network). Furthermore, intermediate nodes can perform data aggregation and caching that is useful to reduce communication overheads [5]. Sensor network applications can be categorized according to its operational paradigm: data gathering and event-driven. The data gathering application requires sensor nodes to periodically report their data to the base station. In the event-driven application, nodes only send data when an event of interest occurs. In our work we treat a ZigBee wireless network. Figure 1 shows the topology models of ZigBee networks [9]. There can be two types of devices in a ZigBee network: Reduced Function Device (RFD) which can not relay data and Full Function Device (FFD) which forwards data from a collection of RFDs. Routers and Coordinators are FFDs. Coordinator manages routers and devices. Sometimes routers are mainline powered and connected to the base station by wire line. Figure 1 shows three basic topologies in ZigBee. In a star topology, sensors are connected to a central router/coordinator. In a cluster tree topology, routers form a tree. Sensors connect to tree nodes. In a mesh topology, stars and cluster trees are connected via their routers. We discuss in this paper sensor networks in the cluster topology.

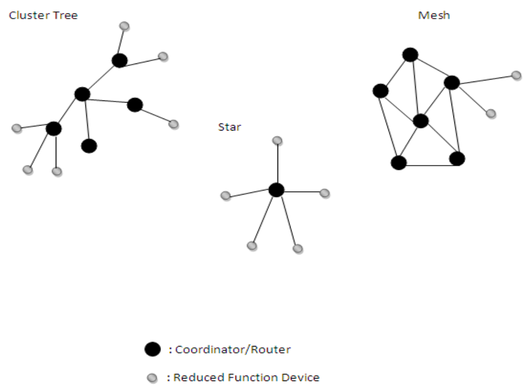


Fig.1: Zigbee topology models

III. Fault Detection in WSNs

Traditionally, a key technique towards fault detection and isolation in distributed systems is the multiplication, i.e. the redundant installation of hardware components such as sensors, data acquisition units or computers (“physical redundancy”). For example, for measuring one single parameter of interest, multiple sensors are physically deployed. To make a decision whether one of the observed sensors is faulty, the outputs of the redundant sensors are compared using decision rules that are commonly based on simple majority voting logics [5]. However, physical redundancy involves substantial penalties in cost and maintainability because multiple hardware components must be installed in the monitored structure. Moreover, voting assumes independent faults, and sensors operating in the same environment can hardly be considered independent. Overcoming these problems, the concept of “analytical redundancy” has emerged, fostered by the rapid advancements in computer science and information technology [12]. Instead of physically installing multiple sensors for measuring one single parameter, analytical redundancy takes advantage of the redundant information inherent in the observed SHM (Structural Healthy Monitoring) system and utilizes the coherences and relationships between the sensors regularly installed [13]. Analytical redundancy, when applied for fault detection and isolation in wireless SHM systems, has tremendous potential to reduce system costs and power consumption of wireless sensor nodes while substantially increasing availability, reliability, safety and maintainability of the SHM system. For each observed sensor, virtual sensor outputs representing non-faulty operation are predicted based on measured outputs of correlated sensors and on a priori knowledge about the system. Comparing actual and virtual sensor outputs, residuals are generated for each sensor. The residuals, reflecting inconsistencies between the actual sensor behavior and the model-based, virtual sensor behavior, serve as the basis for decision making with respect to potential sensor faults. Physical redundancy often uses simple voting logics to determine faulty sensors however analytical redundancy employs mathematical models of the observed decentralized (SHM) system for mapping the inherent redundancy contained in the system.

A. Decentralized Fault Detection Approach

In decentralized diagnostic, nodes monitor behavior locally. When abnormal behavior is detected, nodes execute an in-network diagnostic procedure that is local in nature, i.e. nodes query their neighbors for diagnostic information [4], each wireless sensor node is capable of autonomously detecting and isolating faults of its sensors based on information received from neighboring sensor nodes, while efficiently using the limited computing resources. Sensors are installed in the monitored structure to continuously measure structural and environmental parameters such as acceleration and temperature data. Each sensor is connected to a wireless sensor node designed to autonomously collect data from the sensors, to locally aggregate the sensor data and – in order to assemble a global picture about the structural condition – to communicate with other sensor nodes and with an Internet-enabled local computer placed near the structure. The local computer is primarily deployed to process and to store the sensor data and to enable further (remote) data processing [5]. In case of potential structural anomalies detected from the sensor data, alerts are autonomously generated by the local computer and sent to the human individuals involved.

B. Centralized Fault Detection Approach

In this section we compare centralized and decentralized control of wireless sensor networks. By centralized control we mean a node does not generate its own schedule; rather, it executes a schedule generated by and downloaded from a central scheduler such as the base station. The node simply collects communication statistics and forwards them to the central scheduler. On the other hand, by distributed control we mean a node is autonomous. It schedules its own tasks and data processing. It also processes requests from its neighbors and the host. In a centralized network, the central controller generates routing paths and distributes them to each node; in a distributed network, each node builds its own routing information by talking with each other [15]. We define a real-time application as a function on the base station with an input set and an output set. In centralized control, the schedule is generated offline by the central manager; the networks nodes runs the schedule as simple as using a lookup table. In decentralized control, the schedule in each node is usually generated online, which normally takes more memory space and execution time. Centralized approach has advantage in generating routing paths. By taking into account all possible links and their signal strengths, the central controller could drive the best routing table for each node based on the load, number of hops, signal strength, and more importantly, deadline requirements [20]. All the nodes need to find their neighbors and measure the signal strength with the neighbors and pass the information to the central manager because it would be difficult if each node forms its own knowledge of the network by itself [10]. A good path may be favored by all data transmissions and the nodes on the popular path will exhaust their battery before other nodes. Another advantage of centralized scheduling is collision avoidance. In random channel access scheme, a node having data to transmit first listens on the channel, if the channel is clear, it starts transmitting. If the channel is occupied, the node has to back off and retry later. This mechanism works very well when network traffic is low. However, once many nodes try to transmit at the same time, there would be lots of collisions which may lead to miss deadlines. As for centralized scheduling, a time slot is exclusively used by one transmission. As a summary we note that centralized control is more efficient because it reduces the scheduling computation in individual nodes, which in turn reduces the cost of the sensors and increases the battery life. Both real-time and non-real-time applications can benefit in this aspect [11].

IV. SIMULATION AND INTERPRETATION RESULTS

A. Simulation Environment : Real Time "True Time 2.0"

A real-time task T is a 3-tuple $\{C, D, P\}$, where C is the execution time, D is the relative deadline, and P is the period [10]. At the beginning of each period P, the task T requests an execution of length C that should be finished within D time units. Each request is called a job. Normally, we have $D \leq P$. The task T fails if any of its jobs misses its deadline [16]. A real-time task set S is a set of n real-time tasks $T1, T2, \ldots Tn$. A task set S is schedulable by a scheduling policy on a single processor if no job of any task will miss deadline under the control of this policy [21]. S is feasible if S is schedulable by at least one scheduling policy. There are also other real-time task models. For example, a task may have fixed initial start time. In some simple versions a task's deadline equals its period. For continuous real- time applications, a task is usually repetitive, hence the period P in the task definition. P is sometimes also called the minimum separation time to model tasks whose jobs are not exactly periodic. Many schedule ability results apply even if P is the minimum separation time. Research on real-time task scheduling on a single processor is considered mature now. Another well known scheduling policy, such as Earliest-Deadline-First (EDF) is presented in [11-12]. In centralized case, node D will receive exact data routing schedule from the central manager. Fixed time slots will be allocated every specified period to route data from application T. The central manager will also schedule node *D* to route data for other applications and send its own data. When not servicing the above schedules, *D* is free to handle its internal tasks. In centralized case, *D* will not cause any deadline misses for *T*.

B. Application of DFD algorithm on Real Time

B1. Description of DFD algorithm

The basic idea of DFD is to have each node make a decision on faults; a sensor node can execute a localized diagnosis algorithm in steps to identify the causes of a fault, it can also query diagnostic information from its neighbors. Although this algorithm works for large size of sensor networks, the probability of sensor faults needs to be small. If half of the sensor neighbors are faulty and the number of neighbors is even, the algorithm cannot detect the faults as efficient as expected [23-24]. In addition, this approach also requires each sensor node to be aware of its physical location. We have tried to make this algorithm able to identify suspicious nodes even when half neighbors are faulty, we have conserved the same steps of the algorithm and tried to apply it in a centralized way where the sink node or the cluster head is responsible of all decisions, the other nodes have just to transmit their data without making any exchange between each other. The steps of the algorithm are described as follows:.

- Step 1: For each node Si and any node Sj in neighbors of (Si), dij (t) is the difference between the measurements of the neighbors nodes, Cij is the test result, N(Si) is the number of node's neighbors and θ is the threshold.

  $$\textbf{If } \left| dij(t) \right| < \theta \textbf{ set } C_{ij}=0 \textbf{ else } C_{ij}=1$$

- Step 2: **If** $\sum Sj \in N(Si) < N(S_i)-1$, **set** the initial detection status of S_i as possibly normal 'LG' , **otherwise** it is possibly faulty.

- Step 3: $Num(N(S_i)_{(Tj=LG)}$ is the number of neighbor nodes of (S_i) whose initial detection status is *LG*.

  If $\sum Sj \in N(Si)_{(Tj=LG)} < N(S_i)\ )_{(Tj=LG)}-1$, **set** the status of S_i as normal, **otherwise** it is faulty.

B.2 Application of DFD algorithm on True Time Simulator

Our network is defined as follows: Node 4 is the sink node which collects all information from the other nodes and applies the DFD algorithm to decide about every node status. Nodes 1, 2 and 3 are sensors that also route data [fig 2]. They have the same hard-ware and require the same amount of execution time to collect sensing data. It takes 1 time unit, which is 10 milliseconds, to transmit or receive one data packet. All data is forwarded to the sink node. In the centralized mode, a node transmits data according to pre-configured schedule from the cluster head.

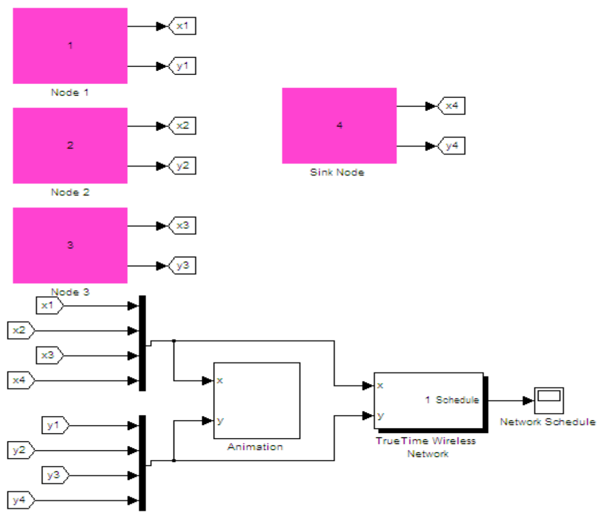


Fig.2: The wireless sensor network design

B.3 Simulation Results

We choose to introduce different measurements for each node to see if the sink node could detect them all faulty. After running the simulation we get:

- The original positions of the four nodes and their respective signal reach [fig3].
- The Ad-hoc routing protocol and the different transmissions between nodes [fig4].
- The status of each one judged by the sink node [fig4].

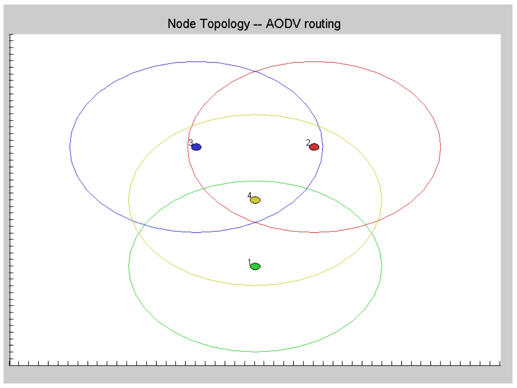
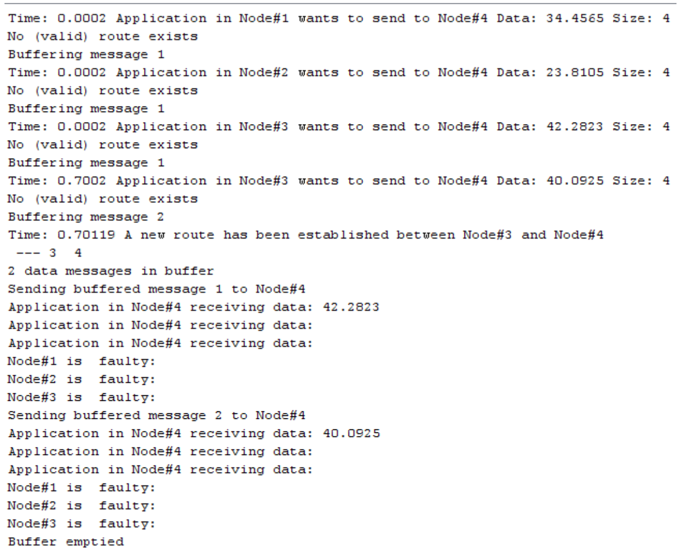
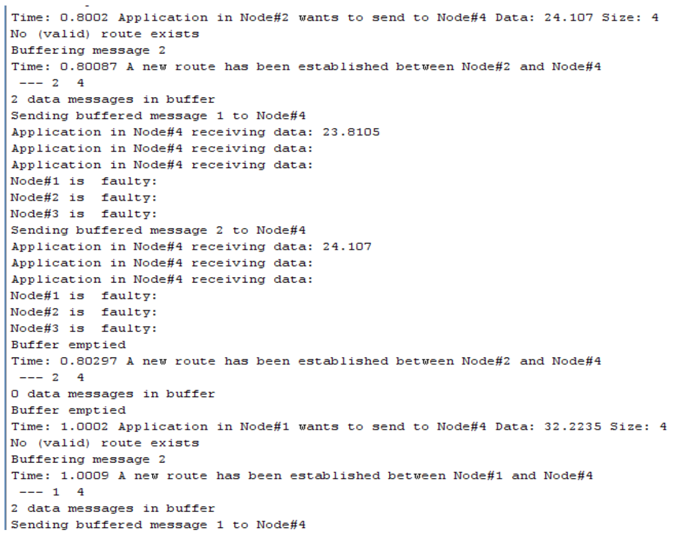


Figure 3. The physical node deployment



```
Time: 0.0002 Application in Node#1 wants to send to Node#4 Data: 34.4565 Size: 4
No (valid) route exists
Buffering message 1
Time: 0.0002 Application in Node#2 wants to send to Node#4 Data: 23.8105 Size: 4
No (valid) route exists
Buffering message 1
Time: 0.0002 Application in Node#3 wants to send to Node#4 Data: 42.2823 Size: 4
No (valid) route exists
Buffering message 1
Time: 0.7002 Application in Node#3 wants to send to Node#4 Data: 40.0925 Size: 4
No (valid) route exists
Buffering message 2
Time: 0.70119 A new route has been established between Node#3 and Node#4
 --- 3  4
2 data messages in buffer
Sending buffered message 1 to Node#4
Application in Node#4 receiving data: 42.2823
Application in Node#4 receiving data:
Application in Node#4 receiving data:
Node#1 is  faulty:
Node#2 is  faulty:
Node#3 is  faulty:
Sending buffered message 2 to Node#4
Application in Node#4 receiving data: 40.0925
Application in Node#4 receiving data:
Application in Node#4 receiving data:
Node#1 is  faulty:
Node#2 is  faulty:
Node#3 is  faulty:
Buffer emptied
```



```
Time: 0.8002 Application in Node#2 wants to send to Node#4 Data: 24.107 Size: 4
No (valid) route exists
Buffering message 2
Time: 0.80087 A new route has been established between Node#2 and Node#4
 --- 2  4
2 data messages in buffer
Sending buffered message 1 to Node#4
Application in Node#4 receiving data: 23.8105
Application in Node#4 receiving data:
Application in Node#4 receiving data:
Node#1 is  faulty:
Node#2 is  faulty:
Node#3 is  faulty:
Sending buffered message 2 to Node#4
Application in Node#4 receiving data: 24.107
Application in Node#4 receiving data:
Application in Node#4 receiving data:
Node#1 is  faulty:
Node#2 is  faulty:
Node#3 is  faulty:
Buffer emptied
Time: 0.80297 A new route has been established between Node#2 and Node#4
 --- 2  4
0 data messages in buffer
Buffer emptied
Time: 1.0002 Application in Node#1 wants to send to Node#4 Data: 32.2235 Size: 4
No (valid) route exists
Buffering message 2
Time: 1.0009 A new route has been established between Node#1 and Node#4
 --- 1  4
2 data messages in buffer
Sending buffered message 1 to Node#4
```

Figure 4. The true time simulation results capture

B.4 Interpretation of Results:

The TRUETIME wireless network simulates communication in an ad-hoc network. This example describes a TRUETIME implementation of one such ad-hoc wireless routing protocol on the DFD algorithm. The network uses three basic types of control messages in order to build and invalidate routes: route request (RREQ), route reply (RREP), and route error (RERR) messages. These control messages contain source and destination sequence numbers, which are used to ensure fresh and loop-free routes. Each node requires a route to the sink node initiates route discovery by broadcasting an RREQ message to the destination node. The sink node receiving an RREQ starts by updating its routing information backwards towards the source. If a route exists with a sequence number greater than or equal to that contained in the RREQ, an RREP message is sent back towards the source as mentioned in the fig-4 at time t=0.0002 and t=0.7002 which announced that there is not a valid route . Otherwise, the node rebroadcasts the RREQ. When an RREP has propagated back to the original source node, the established route may be used to send data.

The simulation example consists of four nodes representing a cluster network, while running we get an animation window which shows the original positions of the four nodes and their respective signal reach (figure 3). In the simulation scenario, the head cluster or the sink node (node 4) receives data periodically from node 1, 2 and 3 while finding the established route (figure 4). After collecting data from the three nodes, the sink node apply the DFD algorithm for each node (the three steps mentioned in B.1), after it decides about the status of every node which is printout in the Matlab command window (figure 4).

We remark that even the number of faulty nodes is more than the half of the total number in the cluster network, the algorithm could detect them all faulty not like in the decentralized approach where it gives us a wrong fault detection when the number of distributed faulty nodes is more than the half of the total number of nodes.

V. CONCLUSION

Wireless sensor network has gradually emerged as a cutting-age technology to develop new wireless applications for pervasive computing in the 21st century. One of challenges to success this vision is fault detection. In this paper, we provided an overview of fault detection in WSNs by comparing existing proposed approaches and we have tried to apply the DFD algorithm on a TRUETIME application in a clustering approach which consumes limited resource of sensor nodes, such as battery energy. In addition, we can see through the simulation results that the centralized fault detection based on clustering approach make the DFD algorithm more efficient than in the decentralized approach; the node doesn't need any more to be expected to constantly police all their neighbors and to consequently end up routing into a new neighbor that has also failed, it has just to send its data to the sink node which collect all the data and apply the algorithm to detect the faulty nodes.

To conclude we can say that by implementing the DFD algorithm on a TRUETIME simulator, we have improved our previous work which had a lack of real application, also we have solved the problem of limited accuracy of detection in the DFD algorithm when the faulty nodes exceeds the half of the total number of nodes. In our future work, we will try to apply our proposed approaches published in [23-24] in a TRUETIME application to validate our work.

REFERENCES

[1] Chen, D., A.K. Mok and S.K. Buruah (1997). On modeling real-time task systems. Lecture Notes in Computer Science - Lectures on Embedded System.

[2] T.Y Wang, L.Y Chang, D.R. Dun and J.Y Wu, Distributed fault-tolerant detection via sensor fault detection in sensor networks, in Proc. of The IEEE lath International Conference on Information Fusion, Quebec, Canada, pp. 1-6, 2007.

[3] Jia, Z. and P. Varaiya (2001). Heuristic methods for delay constrained least cost routing using k-shortest-paths. INFOCOM.

[4] Song, J., A.K. Mok, D. Chen and M. Nixon (2006). Using real-time logic synthesis tool to achieve process control over wireless sensor networks. The 12th IEEE Conference on Embedded and Real-Time Computing Systems and Applications.

[5] M. Yu, H. Mokhtar, et M. Merabti. A Survey on Fault Management in Wireless Sensor Networks. IEEE Wireless Communications (2007) 13-19.

[6] E Y. Singh, S. Saha, U. Chugh, C. Gupta, 2013 Distributed Event Detection in WSN for Forest Fires, Internation Conference of Computer Modelling and Simulation (UKSIM'13).

[7] Karim, L et al., 2009 A Fault Tolerant Dynamic Clustering Protocol of Wireless Sensor Networks, IEEE conference of Global Telecommunications (GLOBECOM).

[8] ZigBee (2007). http://www.zigbee.org.

[9] M. Yu, H. Mokhtar, and M. Merabti, 2007 A Survey of Fault Management in Wireless Sensor Networks, Journal of Network and Systems Management, Volume 15, Issue 2 , pp 171-190.

[10] Jannatul Ferdous et al., 2010 "A Comprehensive Analysis of CBCDACP in Wireless Sensor Networks", Journal of Communications, vol. 5, no. 8 627-636, Aug 2010.

[11] Sang Hyuk Lee et al., 2011 Gradual Cluster Head Election for High Network Connectivity in Large-Scale Sensor Networks, ICACT,vol. 13, pp.168-172.

[12] M. Saihi, B. Boussaid et A. Zouinkhi. Détection des défauts distribués dans les réseaux de capteurs sans fil, 2012, pp.26-29.

[13] J. Shi and J.P. Fonseka, "Hierarchical self-healing rings," IEEE/ACM Trans. Networking, vol. 3, no. 6, pp. 690-697, Dec. 1995. Smart Transducer Interface Standard, IEEE 1451, Sensors Expo, Philadelphia, Oct. 2001.

[14] S.H. Low, F. Paganini, and J.C. Doyle, "Internet congestion control," IEEE Control Systems Mag., pp. 28-43, Feb. 2002.

[15] Linnyer Beatrys Ruiz, Isabela G. Siqueira, Leondardon B. Oliveria, Hao Chi Wong, Jose Marcos S. Nogueira, Antonio A.F. Loureiro, On the Design of a Self-Managed Wireless Sensor Network. IEEE Communications Magazine, 2005: p. 95-102.

16] Jinran Chen, Shubha Kher, Arun Somani. Distributed Fault Detection of Wireless Sensor Networks. in DIWANS'06. 2006. Los Angeles, USA.

[17] Winnie Louis Lee, Amitava Datta, Rachel Cardell-Oliver, WinMS: Wireless Sensor Network-Management System, An Adaptive Policy-Based Management for Wireless Sensor Networks. 2006, UWA, Australia.

[18] NIthya Ramanathan, Kevin Chang, Rahul Kapur, Lewis Girod, Eddie Kohler, Deborah Estrin. Sympathy for the Sensor Network Debugger. In 3rd Embedded networked sensor systems. 2005. San Diego, USA.

[19] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar. SPINS: Security protocols for sensor networks. in ACM MobiCom'01. 2001. Rome, Italy.

[20] S. Tanachaiwiwat, P. Dave, R. Bhindwale, A. Helmy, Secure Locations: routing on trust and isolating compromised sensors in location-aware sensor networks. SenSys '03 Proceedings of the 1st international conference on Embedded networked sensor systems, pp. 324 – 325, ACM New York, NY, USA , 2003

[21] Z.Ying, X.Debao. Mobile Agent-based Policy Management for Wireless Sensor Networks. in WCNM. 2005.

[22] Winnie Louis Lee, Amitava Datta, Rachel Cardell-Oliver, WinMS: Wireless Sensor Network-Management System, An Adaptive Policy-Based Management for Wireless Sensor Networks. 2006, UWA, Australia.

[23] M. Saihi, B. Boussaid, A. Zouinkhi, M. N. Abdelkrim (2012). A New Approach for Decentralized Fault Detection in Wireless Sensor Network. 13th international conference on Sciences and Techniques of Automatic control and computer engineering (STA 2012), December 18-21, 2012, Monastir, Tunisia.

[24] M. Saihi, B. Boussaid, A. Zouinkhi, M. N. Abdelkrim (2013). Decentralized Fault Detection in Wireless Sensor Network based on function error, 10th international multi-Conference on systems, signals and devices (SSD 2013). Mars, 17-19, 2013, Hammamet, Tunisia.