A pragmatic testing approach for wireless sensor networks

Cédric Ramassamy, Hacène Fouchal, Philippe Hunel, Nicolas Vidot

A Pragmatic Testing Approach for Wireless Sensor Networks

Cédric Ramassamy
LAMIA
Université des Antilles et de la Guyane
France
cramassa@gmail.com

Hacène Fouchal
CReSTIC
Université de Reims Champagne-Ardenne
France
Hacene.Fouchal@univ-reims.fr

Philippe Hunel
LAMIA
Université des Antilles et de la Guyane
France
Philippe.Hunel@univ-ag.fr

Nicolas Vidot
LAMIA
Université des Antilles et de la Guyane
France
Nicolas.Vidot@univ-ag.fr

ABSTRACT

Applications over wireless sensor networks are growing quickly. Traditional software development tools are not well adapted to this technology. In particular, adequate testing methodologies are required. Many issues have not been formally addressed as energy-conservation, congestion control, reliability data dissemination, security. In this paper, we present a pragmatic approach to detect some kind of anomalies based on defined scenarios on wireless sensor networks. A scenario is considered as a set of events which should happen on the network in an ordered way. We discuss a formal architecture able to perform these scenarios over the wireless sensor networks and to raise alarms if necessary. This methodology has been implemented on a prototype and has been experimented with various examples. This contribution is a first attempt for a formal testing methodology for wireless sensor networks.

Categories and Subject Descriptors

C.4 [**Performance of systems**]: Fault tolerance; D.2.8 [**Software Engineering**]: Software/Program Verification—*Formal methods, Model checking, Reliability*

General Terms

Reliability

Keywords

Wireless Sensor Networks, Software Testing, Runtime Monitoring, Fault Detection

1. INTRODUCTION

Presently, in software industry, an important validation process is performed using various testing techniques as conformance, robustness, performance and interoperability of a system. Conformance testing consists of 2 steps: the first one deals with the generation of test sequences from a specification and the second one takes care of the execution of the sequences on an implementation which is referred to as an Implementation Under Test (IUT), and checks the reactions of the implementation in order to detect faults.

Many well known formalisms for system description are proposed like LTS[1] and FSM[2] for untimed systems and TIOA[3] and ETIOA[4] for timed systems [2, 20, 21]. Many methodologies have been proposed since many decades for test sequence generation. For example, in [3] the authors propose a model to describe a system and a method to test communicating systems. In [5] the authors propose a testing method for a system which can be described as an Extended Finite State Machine. In [6] the authors propose a generalization of the W-method (from T. S. Chow [10]) for a specification which do not has a characterization set. A survey about testing Real-Time systems can be found in [7, 11, 13] and FSM [16]. Some other works [8, 19, 18, 22, 14] are dedicated to embedded systems and distributed systems.

Wireless sensor networks have been studied in academia and industry in recent years. Their main advantages are the ability to have computation and communication capabilities as well as a reduced cost. WSN can be used for many applications such as habitat monitoring, in-door monitoring, target tracking, and security monitoring, etc. However, embedded applications over WSN are not widely deployed since WSN have still some issues to overcome, for exam-

[1]Labelled Transition System
[2]Finite State Machine
[3]Timed Input Output Automaton
[4]Extended Timed Input Output Automaton

Q2SWinet'10, October 20–21, 2010, Bodrum, Turkey.

ple, energy-conservation, congestion control, reliability data dissemination, security and management of a WSN itself.

In this study, we propose an architecture to test wireless sensor networks. The user may monitor the system by using this methodology. He may also execute test sequences using defined scenarios on WSN. On one hand, the monitoring process checks if the system works properly, and on another hand the scenario ensures *a low level conformance* for the system. A scenario describes a test sequence to be experimented on the implementation. It is composed of an ordered set of input actions with the average waiting time before two successive actions. The formalism suggested in [18, 24] is used to represent our scenarios. This paper is organized as follows. Section 3 presents a description of scenario generation, test architecture and test execution. Section 4 describes a prototype case study to illustrate our approach. Section 5 concludes the study.

2. RELATED WORK

Distributed systems may be viewed as a set of components connected via a communication system, where each component sends and/or receives messages through one or more communication channels. Distributed testing was studied with different test architectures. [17] proposed a distributed tester, with the distribution of global test sequence (GTS) into local test sequences executed over local testers, where the synchronization and the fault detection problems are discussed.

A noticeable work has been done on distributed testing [17]; however, only few of them, [4], [30] dealt with testing component based systems, and others [25], [29], [27] studied the construction of component based systems. [17], handles testing distributed real-time systems, proposing a method for solving the controllability and observability problem by holding some timing constraints. [17] assumed a more realistic clock between testers synchronizing with a reference clock with a given inaccuracy rather than synchronizing with other tester's clock.

In Component Based Systems (CBS), components must interact with different external elements [29]. Those components may run on a single, multiple, distributed or network systems in a sequential or individual concurrent manner, and may communicate with every other component. Such systems require a tight and loose coupling of components of diverse granularity. [25] takes into consideration some major requirements such as, flexibility, time-to-market, quality, cross-department standardization. With all its advantages, CBS still lacks some aspects of real-time systems that cannot be encapsulated in a component such as synchronization, and memory optimization [27]. From the above requirements, it is clear how complex and sensitive it is to build and to have a reliable tested component-based system.

In [28], MoteLab, a Web-based sensor network testbed is presented. MoteLab consists of a set of permanently-deployed sensor network nodes connected to a central server which handles reprogramming and data logging while providing a web interface for creating and scheduling jobs on the testbed. MoteLab accelerates application deployment by streamlining access to a large, fixed network of real sensor network devices; it accelerates debugging and development by automating data logging, allowing the performance of sensor network software to be evaluated offline. Additionally, by providing a web interface MoteLab allows both local and remote users access to the testbed, and its scheduling and quota system ensure fair sharing.

In [26], authors deal with the observation of faults in real deployments. They considered three types of transient faults, caused by faulty sensor readings that appear abnormal. To understand the prevalence of such faults, they explore four qualitatively different classes of fault detection methods: 1) Rule-based methods leverage domain knowledge to develop heuristic rules for detecting and identifying faults; 2) Learning methods about the system behavior; 3) Estimation methods predict to normal sensor behavior by leveraging sensor correlations, flagging anomalous sensor readings as faults; 4) Time-series-analysis-based methods start with an a priori model for sensor readings. A sensor measurement is compared against its predicted value computed using time series forecasting to determine if it is faulty. They concluded that: training data, and then statistically detect and identify classes of faults. These four classes of methods sit at different points on the accuracy/robustness spectrum. Rule-based methods can be highly accurate, but their accuracy depends critically on the choice of parameters. Learning methods can be cumbersome to train, but can accurately detect and classify faults. Estimation methods are accurate, but cannot classify faults. Time-series-analysis-based methods are more effective for detecting short duration faults than long duration ones, and incur more false positives than the other methods. They apply these techniques to four real-world sensor datasets and find that the prevalence of faults as well as their type varies with datasets. All four methods are qualitatively consistent in identifying sensor faults, lending credence to our observations.

In [1] is a tiered wireless sensor network testbed. It consists of 13 clusters, with each cluster consisting of a stargate and several motes attached to it via USB cables. These stargates communicate with a central PC over 802.11b, from where any node on the testbed can be programmed. Thus a testbed consisting of 13 stargates and 104 motes (91 tmoteSky and 13 MicaZ) is currently running in at University of Southern California. The main features of Tutornet are Tiered sensor network testbed: consists of 3 tiers staring with top tier, the testbed sever is on the top, Stargates, and then motes respectively. Wireless multi-hop routing between the testbed server and a stargate or among stargate is performed.

[15] extends the Emulab network testbed software to provide a remotely-accessible mobile wireless and sensor testbed. Robots carry motes and single board computers through a fixed indoor field of sensor-equipped motes, all running the user's selected software. In real-time, interactively or driven by a script, remote users can position the robots, control all the computers and network interfaces, run arbitrary programs, and log data. This mobile testbed provides simple path planning, a vision-based tracking system accurate to 1 cm, live maps, and webcams. Precise positioning and automation allow quick and painless evaluation of location and mobility effects on wireless protocols, location

algorithms, and sensor-driven applications. The system is robust enough that it is deployed for public use.

[12] presents TWIST, a scalable and reconfigurable testbed architecture for indoor deployment of wireless sensor networks. The design of TWIST is based on an analysis of typical and desirable use-cases. It provides basic services like node configuration, network-wide programming, out-of-band extraction of debug data and gathering of application data. TWIST supports experiments with heterogeneous node platforms. it also supports active power supply control of the nodes. This enables easy transition between USB-powered and battery-powered experiments, dynamic selection of topologies as well as controlled injection of node failures into the system. Thirdly, TWIST supports creation of both at and hierarchical sensor networks. The self-configuration capability, the use of hardware with standardized interfaces and open-source software makes the TWIST architecture scalable, affordable, and easily replicable.

In all these studies, the main objective is to check if the data handled by the wireless sensor network are computed without errors. The testing is data oriented. In our case, we intend to focus on the behavior checking. We will test how the network react to stimulus sent by the environment. This kind of testing aims to prove if the expected correct behavior of the whole network will be executed without any fault. Our technique is inspired from protocol testing engineering field that we adapt to the wireless sensor networks.

3. CONTRIBUTION

Our goal is to check if a wireless sensor network works without executing faulty actions. The approach aims to detect anomalies related to defined scenarios on wireless sensor networks. The objective is to provide an appropriate testing methodology for WSN.

3.1 Definitions

In this section, we introduce some notions needed to describe the approach as an observer, a scenario, a node and some other notions and notations used in the paper.

DEFINITION 1. *(Node) A node is a sensor which could be connected to some neighbor nodes.*

In our case the communication is performed through a wireless method.

DEFINITION 2. *(Network) A WSN is a network composed of nodes. It executes a distributed application. Each node execute a part of the application (usually all of them has the same code)*

DEFINITION 3. *(Reaction Time) Reaction time is an upper bound of time amount between : (i) the instant when an event e is received by a node and (ii) the instant when a node has ended to send all its outputs (if any) as a reaction to the reception of e.*

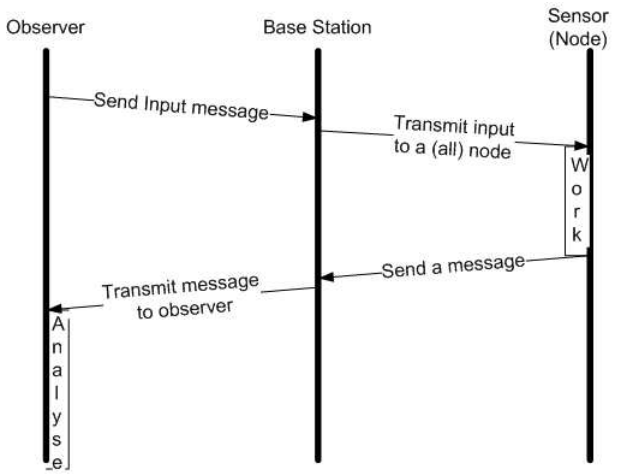

Figure 1: Sequence exchange

This amount of time is used when the application has time constraints like real time applications.

DEFINITION 4. *(Observer) An observer is a specific node which is dedicated to listen, receive and analyze messages responses of request on WSN.*

This node is particular. Its role is to monitor the network in a remote way and will not participate to the application execution.

DEFINITION 5. *(Transfer Time) Transfer time, denoted Δ, between observers and nodes, is defined as the time amount between : (i) the instant when a message M is sent and (ii) the instant when M is received.*

In general this delay expresses network topology complexity. In most of cases, we assume that we have no delay between observers or sensors.

DEFINITION 6. *(WSN application) A WSN application is characterized by a set of events and tasks.*

The application is composed of tasks over all nodes. We need to describe the events observed over the network. These events are the key point of our testing process.

DEFINITION 7. *(Black-box testing) Black-box testing uses external descriptions of the software, including specifications, requirements, and design to derive test cases. These test cases can be functional or non-functional. There is no knowledge about internal structure of the implementation to test.*

This kind of testing is very common in hardware and protocol engineering. For competition reasons, many companies

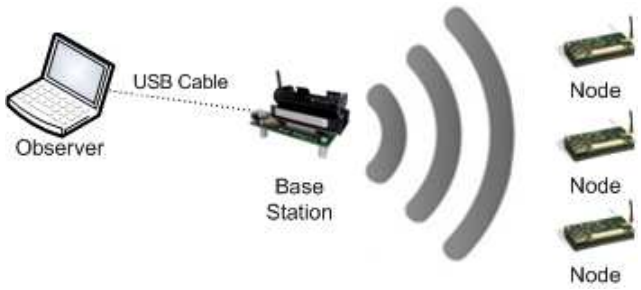

Figure 2: Observer with Base Station

(like electronic device producers) do not wish to show the internal structure of their products. But all companies which need to use these kind of products should perform black-box testing in order to check if some non required behavior is observed on the products they intend to use (like telecom providers).

DEFINITION 8. *(Scenario) A scenario is described by a test sequence. A test sequence is an ordered set of of action/reactions where each event corresponds to the reception of an input and a response is the output generated after receiving an event.*

Scenario generation may be achieved by an user. A scenario must be generic enough to check if the system works properly. Automatic generation is widely required for large systems and will ensure formally fault coverage as described in [18, 24]. The base station is the interface between the observer and the WSN. The observer is connected to the base station through a serial port, a parallel port or a network port.

DEFINITION 9. *(Correct scenario) Let a sequence of input action $t_{ia} = a_1, a_2, ..., a_n$ and output reaction $t_{rm} = b_1, b_2, ..., b_n$. An execution of scenario on a WSN is correct iff : (1)b_i is the reaction to a_i (2) a_i is executed before a_{i+1} and b_i before b_{i+1}; and (3) the timing constraints (reaction time) between the input event and output reaction are respected.*

Due to radio range coverage of a node, we use distributed observers to be sure to cover all nodes in the WSN. In Sect. 3.2, we detail the repartition method of observers.

3.2 The approach

Our testing approach is based on monitoring scenario execution on a WSN. This monitoring is ensured by distributed observers. Observers are controlled by a user (or by an automated process). In order to determine the observers location on the network, we split the network into several sub-networks. This is done according to the radio range coverage of sensors. For each sub-network, we have a local base station which will be used to collect the sub-network data.

3.2.1 Test Architecture

For each sub-network, we set an observer and a base station. The approach is based on the following parts :

- **The base station** is a specific node. It is a relay node able to provide a communication between an observer and all sensors of the related sub-network .
- **The observer** is the tester. Its role is to send input action and receive output reaction through the base station. The observer analyzes output reactions and checks if timed constraints are respected. The aim is to check if a scenario is executed properly over the network.
- **The user** controls the scenario execution. He can check if an anomaly is observed over the network (dead links, energy decrease, congestion control, overload, etc).
- **The verdict** is produced by an observer and concludes if the WSN encounters anomalies.

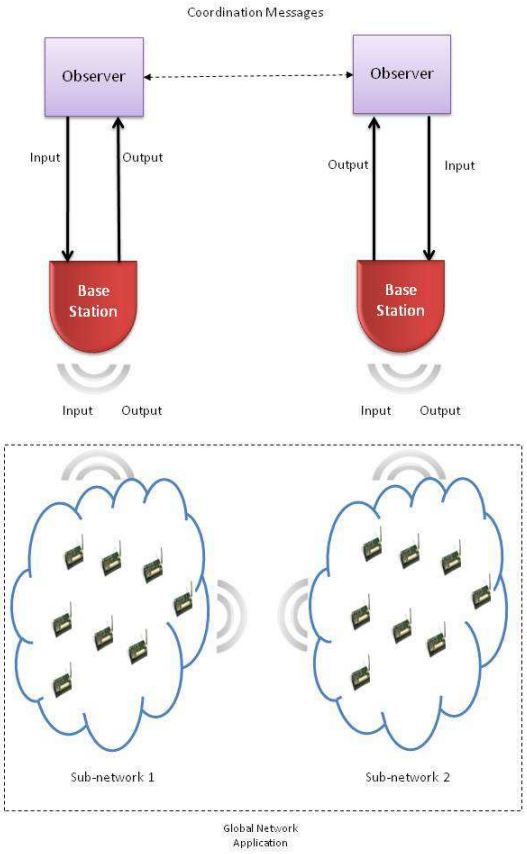

Figure 3: Test Architecture

3.2.2 Scenario generation

In order to achieve the generation of scenarios, we have to describe the application with a formal formalism like LTS (Labelled Transition Systems). Then a generation of most pertinent sequences could be done by a well-known technique as W-method for example. That means we do not

need to generate all possible sequences. When the application is very large, a space explosion can be expected. In practical cases, this generation is oriented by the user.

4. EXPERIMENTATION

We have implemented a prototype tool which implements our approach.

4.1 The prototype

This prototype is composed of three parts :

- ➢ The application.
- ➢ The scenario management,
- ➢ The analysis and monitoring of messages sensors.

4.2 The environment

In our tool, we achieved the testing process on a real environment. The application to test is deployed over a WSN. It is written in NesC language. The framework is composed of two base stations, five sensors and a computer. Sensors communicate using the Zigbee protocol (802.15.4). Computer communicates with base station using USB connection Figure. 2.

4.3 The network

The sensor radio coverage may be represented by groups denoted sub-networks (Figure 4). Sensors are spread using geographically random way, they represent sub-networks communication (Figure 4 is an example).

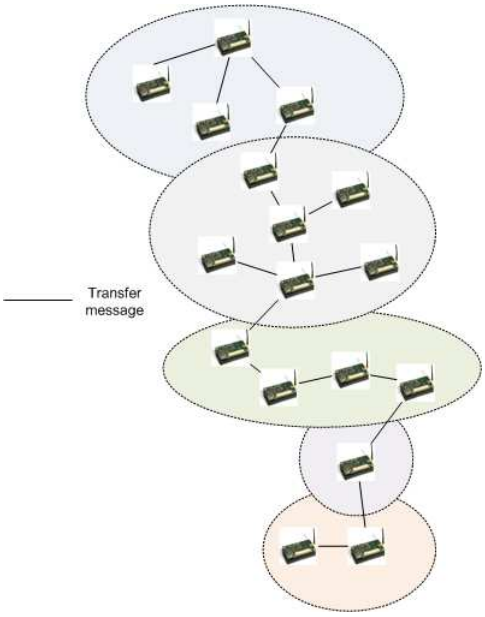

Figure 4: Sub networks

4.4 The application

The application is deployed over the whole WSN. The following three events are possible on each sensor:

- ➢ The timer management event
- ➢ The radio management event (switch on or off),
- ➢ The message sending/receiving management event.

Each sensor has a message setup to 0. On the sub-network, one sensor is dedicated to sense a value from the environment (temperature, pressure,..). This sensor updates its message with the sensed value. Every 20 seconds, it broadcasts its message on the network. When it receives a message (from the base station or from another sensor), it compares its local message to the received one and updates its local message with the received one if this later is greater.

The first aim is to guarantee that messages observed on sub-networks are identical on all nodes.

4.5 The observer

Here we describe the observer application we have implemented over all observers.

Our application is written in JAVA, thus we have to implement a library allowing communication in USB port. Although current libraries are very limited [5], then we use TinyOS library for USB communications between the base station and the computer.

Our application is split in two parts. The first part is dedicated to sending input actions to the network and the second part is dedicated to receive reactions.

4.5.1 Network listening

The network listening is achieved according to radio range coverage of the base station. We catch, during data sending, messages from all sensors and we analyze their contents. The packet structure is detailed as follows:

Src	Dest	AM	Grp	Len	Data
(2)	(2)	(1)	(1)	(1)	(0...29)

Table 1: TinyOS packet format. The byte size of the fields are detailed

We catch messages in transit on the network and analyze if messages from all sensors are the same. Our main aims are:

- ➢ Collecting sensor messages,
- ➢ Analyzing messages and timing constraints.

4.5.2 Scenario Sending

The aim of our scenario is to send every 5 seconds a message on the network via the base station. This message represents a number which will be increased after each step. Message sending is a broadcast to all sensors located in the radio range of the sub-network.

Figure 5: First Representation

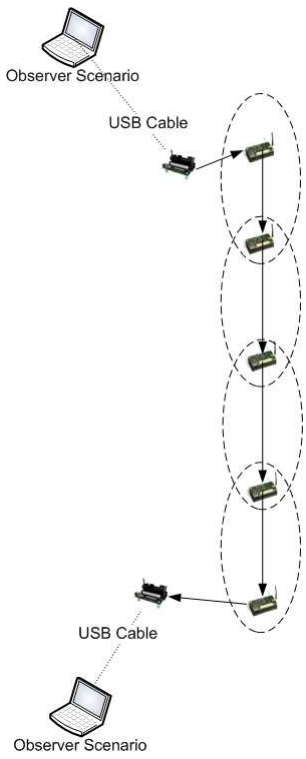

Figure 6: Second Representation

4.6 Results

We tested two representations of the network described on Figure 5 and on Figure 6.

In the first case, all sensors communicate in the radio range of the base station. We notice that dead nodes and bad communications are detected by the observer application in all cases. Nevertheless, we have no guarantee on packet lost. The application can check in the first iteration that the sensor has some troubles. But if in the second iteration sensor message is received, the application can decide if this sensor is in bad location regarding radio range coverage.

In the second case, some sensors are located in the radio range of the sub-network and all others are outside of it. But the connectivity is guaranteed by having some relaying sensors on the sub-network. We have observed the same results than in the previous case. But if a sensor relay is switched off (Figure 7), we have observed communication troubles. That allows us to expect the use of a *router* node for future works.

Up to now, we have not described formally all faults we are able to detect. But next investigations of our work will be dedicated to collect all possible faults and to show formally how are we able to detect these faults by our methodology.

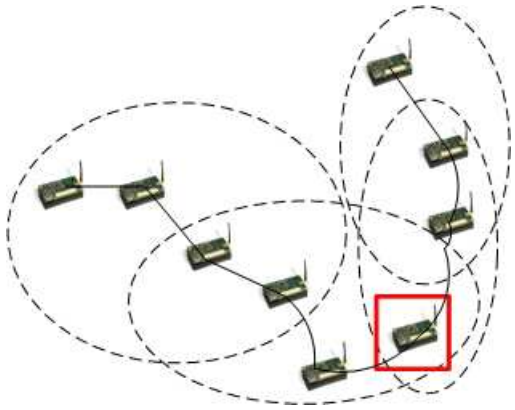

Figure 7: A router node

5. CONCLUSION AND FUTURE WORK

We described a distributed test architecture for Wireless Sensor Networks by using observers. We presented a methodology to check if some faults occur on a wireless sensor network. The main contribution is the use of collaboratives observers over the WSN. The observers checks the correctness of the application by means of monitoring a scenario over the whole network. Thanks to this, we can establish a kind of conformance relation.

Two important issues will be investigated in the next future:

- ➢ Formal description of faults: this issue is very useful and could be used for fault coverage.
- ➢ Robustness of the monitoring process: this issue en-

[5]JUSB, javax.usb

sures that the monitoring is able to be maintained even if some observers have troubles.

R. Cardell-Oliver shows in [9] works about robustness and longevity of network on WSN to rainfall.

Finally, we intend also to handle a real-time monitoring for energy levels, network quality, sensor workloads.

6. REFERENCES

[1] Tutornet: A tiered wireless sensor network testbed. 2005.

[2] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[3] Ismail Berrada, Richard Castanet, and Patrick Félix. Testing communicating systems : a model, a methodology and a tool. In *17th IFIP International Conference on Testing of Communicating Systems; Lecture Notes in Computer Science*, Montreal, Canada, may 2005. Elsevier.

[4] A. Bertolino, F. Corradini, P. Inveradi, and H. Muccini. Deriving test plans from architectural descriptions. In *ACM Proceedings, International Conference on Software Engineering ICSE2000, June 2000*, June 2000.

[5] Johan Blom, Anders Hessel, Bengt Jonsson, and Paul Pettersson. Specifying and generating test cases using observer automata. In *Proc. 4 th International Workshop on Formal Approaches to Testing of Software 2004 (FATES'04), volume 3395 of Lecture Notes in Computer Science*, pages 125–139. Springer–Verlag, 2005.

[6] Adilson Luiz Bonifácio, Arnaldo Vieira Moura, and ao Sim Adenilso da Silva. A generalized model-based test generation method. In *SEFM '08: Proceedings of the 2008 Sixth IEEE International Conference on Software Engineering and Formal Methods*, pages 139–148, Washington, DC, USA, 2008. IEEE Computer Society.

[7] Manfred Broy, Bengt Jonsson, Joost-Pieter Katoen, Martin Leucker, and Alexander Pretschner. *Model-Based Testing of Reactive Systems: Advanced Lectures (Lecture Notes in Computer Science)*. Springer-Verlag New York, Inc, Secaucus, NJ, USA, 2005.

[8] L. Cacciari and O. Rafiq. Controllability and observability in distributed testing. *Information and Software Technology*, 41(11-12):767–780, 9/15 1999.

[9] Rachel Cardell-oliver, Keith Smettem, Mark Kranz, and Kevin Mayer. Field testing a wireless sensor network for reactive environmental monitoring. In *In International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 14–17, 2004.

[10] T. S. Chow. Testing software design modeled by finite-state machines. *IEEE Trans.Softw.Eng.*, 4(3):178–187, 1978.

[11] Camille Constant. Génération automatique de tests pour des modèles avec variables ou récursivité., 2008-11-24 2008. ID: tel-00424546, version 1.

[12] Vlado Handziski, Andreas Köpke, Andreas Willig, and Adam Wolisz. Twist: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In *REALMAN '06: Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, pages 63–70, New York, NY, USA, 2006. ACM.

[13] Anders Hessel and Paul Pettersson. A test case generation algorithm for real-time systems. *Quality Software, International Conference on*, 0:268–273, 2004.

[14] William Hoarau. Injection de fautes dans les systèmes distribués, 21 mars 2008.

[15] David Johnson, Tim Stack, Russ Fish, Daniel Montrallo Flickinger, Leigh Stoller, Robert Ricci, and Jay Lepreau. Mobile emulab: A robotic wireless and sensor network testbed. In *INFOCOM*. IEEE, 2006.

[16] Fujiwara Bochmann Khendek, S. Fujiwara, G. V. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi. Test selection based on finite state models. *IEEE Transactions on Software Engineering*, 17:591–603, 1991.

[17] A. Khoumsi. Testing distributed real-time systems in the presence of inaccurate clock synchronization. *Journal of Information Soft. Technology (IST)*, 45, Dec 2003.

[18] Ahmed Khoumsi. Test execution for distributed real time systems.

[19] Ahmed Khoumsi. Testing distributed real time systems using a distributed test architecture. *Computers and Communications, IEEE Symposium on*, 0:0648, 2001.

[20] David Lee and Mihalis Yannakakis. Principles and methods of testing finite state machines — a survey.

[21] Frank Jin Ye Luo. A diagnostic method for non-deterministic finite state machines, 1996.

[22] Carlos Eduardo Pereira and Luigi Carro. Distributed real-time embedded systems: Recent advances, future trends and their impact on manufacturing plant control. *Annual Reviews in Control*, 31(1):81–92, 2007.

[23] D. Raychaudhuri, M. Ott, and I. Secker. Orbit radio grid tested for evaluation of next-generation wireless network protocols. In *TRIDENTCOM '05: Proceedings of the First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities*, pages 308–309, Washington, DC, USA, 2005. IEEE Computer Society.

[24] Ahmad A. Saifan, Ernesto Posse, and Juergen Dingel. Run-time conformance checking of mobile and distributed systems using executable models. In *PADTAD '09: Proceedings of the 7th Workshop on Parallel and Distributed Systems*, pages 1–11, New York, NY, USA, 2009. ACM.

[25] H. Schmidt. Trustworthy components-compositionality and prediction. *The Journal of Systems and Software*, 65:215–225, 2003.

[26] Abhishek B. Sharma, Leana Golubchik, and Ramesh Govindan. Sensor faults: Detection methods and prevalence in real-world datasets. *ACM Transactions on Sensor Networks (TOSN)*, 6(3):1–39, 2010.

[27] A. Tesanovic, D. Nystrom, J. Hansson, and C. Norstrom. Towards aspectual component-based development of real-time systems. In *Proceeding of the 9th International Conference on Real-Time and Embedded Computing Systems and Applications (RTCSA 2003), February 2003*, February 2003.

[28] Geoffrey Werner-Allen, Patrick Swieskowski, and Matt Welsh. Motelab: a wireless sensor network testbed. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 68, Piscataway, NJ, USA, 2005. IEEE Press.

[29] J. Zalewski. Developing component-based software for real-time systems. In *27th Euromicro Conference 2001: A Net Odyssey (euromicro'01), September 2001*, September 2001.

[30] Peter Zimmerer. Test architectures for testing distributed systems. In *12th International software quality week (QW'99), May 1999*, May 1999.