7-2016

Passively testing routing protocols in wireless sensor networks

Xiaoping CHE

Stephane MAAG

Hwee-Xian TAN

Hwee-Pink TAN
Singapore Management University, hptan@smu.edu.sg

Citation

CHE, Xiaoping; MAAG, Stephane; TAN, Hwee-Xian; and Hwee-Pink TAN. Passively testing routing protocols in wireless sensor networks. (2016). *Proceedings of 2015 IEEE 12th International Conference on Ubiquitous Intelligence and Computing: 2015 IEEE 12th International Conference on Advanced and Trusted Computing, Beijing, China, 2015 August 10-14*.
Available at: https://ink.library.smu.edu.sg/sis_research/3565

Passively Testing Routing Protocols in Wireless Sensor Networks

Xiaoping Che
School of Software Engineering
Beijing Jiaotong University
Beijing, China

Stephane Maag
CNRS UMR 5157 Samovar
IMT/Telecom SudParis
Evry, France

Hwee-Xian Tan
Institute for
Infocomm Research (I2R)
Singapore

Hwee-Pink Tan
School of Information Systems
Singapore Management University
Singapore

Abstract—**Smart systems are today increasingly developed with the number of wireless sensor devices that drastically increases. They are implemented within several contexts through our environment. Thus, sensed data transported in ubiquitous systems are important and the way to carry them must be efficient and reliable. For that purpose, several routing protocols have been proposed to wireless sensor networks (WSN). However, one stage that is often neglected before their deployment, is the conformance testing process, a crucial and challenging step. Active testing techniques commonly used in wired networks are not suitable to WSN and passive approaches are needed. While some works propose to specify the protocol with state models or to analyze them with simulators and emulators, we here propose a logic based approach for formally specifying some functional requirements of a novel WSN routing protocol. We provide an algorithm to evaluate these properties on collected protocol execution traces. Further, we demonstrate the efficiency and suitability of our approach by its application into common WSN functional properties as well as specific ones designed from our own routing protocol. We provide relevant testing verdicts through a real indoor testbed and the implementation of our protocol. We show that our approach may model and passively test common and particular test objectives illustrating its flexibility, genericity and practicability. As far as we know, this is the first work on formal passive testing of routing protocols in wireless sensor networks.**

I. Introduction

The phenomenon whereby data is acquired from several network-connected sensors for the purposes of sense-making and intelligent decision-making in smart systems, is drastically accelerating. It is expected that there will be increasing numbers of large-scale deployments of devices that are instrumented with multi-modal sensors, in the near future (e.g., in cities, transportations, museums, energy systems, etc.). These sensor devices generally have small form factors, and possess limited processing and storage capabilities. In addition, they are typically powered by limited battery supplies and/or energy harvesting sources.

An essential component for the realization of these wireless sensing devices is the data transport service - i.e., the 'routing protocol' - to carry data from the sensor devices to the backend servers via one or more (Internet-enabled) gateways, for further processing and analysis. Due to the energy constraints of the nodes, the limited transmission range, and the few resources of the sensor nodes, providing efficient and reliable routing protocols is an important and challenging problem. For years now, several routing protocols for wireless sensor networks (WSN) have been standardized and developed [1], [2]. While many efforts have been produced for their development, integration and performance evaluation, the conformance testing of their implementations have been somehow neglected.

The conformance testing of routing protocols is crucial, as a data transport service that does not behave in its expected manner, can have detrimental impacts on data collection, and severely impact the overall quality as well as usefulness of the system. Furthermore, in large-scale sensor systems, a single bug fix can be an extremely time-consuming and labor intensive process - especially if the nodes are placed in inaccessible locations across varying spatial locations.

However, though the conformance testing of these protocols is crucial to the reliability of the systems, there are paradoxically very few works. Indeed, the major ones are mostly dedicated to their performance analysis, application and security studied through simulation or emulation relying on non-formal models [3], [4], [5], [6], [7]. We propose in this paper a formal syntax and semantics to express the requirements of WSN routing protocols and to passively test them on real execution traces. These syntax and semantics do not need to be inevitably used on top of any protocol standard. We herein model and test common WSN requirements as well as specific ones designed from a novel non-standardized routing protocol. We show that our approach is helpful since the very first stage of the system development life cycle and is complementary to the normalization phase as well.

There are two main ways for testing the conformance of a protocol, either in an active or passive way. While the active ones require a stimulation of the Implementation Under Test (IUT) and an important testing architecture, the passive ones are based on the observation of input and output events of an IUT at runtime. Basically, passive testing techniques are applied whenever the state of an IUT cannot be controlled by means of test sequences either because access to the interfaces of the system is unavailable or a reset of the IUT is undesired. This is specifically the case with the WSN where the topology is dynamic and the testing interfaces not always available. Morever, in actively testing WSN protocols, we often deal with no direct access, wide fluctuations in wireless channel and a high dependance on prevailing environmental characteristics.

The passive testing approaches are based on the record of the observation during runtime and its comparison with the expected protocol behavior defined by either a formal model or as a set of formally specified properties obtained from the requirements of the protocol [8]. The observation is performed through a set of Points of Observation (PO) on monitored entities composing the System Under Test. The protocol messages observed in execution traces are generally modelled and analyzed through their control and data parts [9]. These approaches are efficient but are not suitable for wireless environments like WSN and its own inherent constraints as above mentioned. Although we already studied and tested wired networks with our formal testing approach [10], we present in this paper an adaptation of this approach to passively test a routing protocol in a wireless sensor based smart systems. Common and specific WSN protocol properties are evaluated on collected execution traces to provide testing verdicts. We successfully assess our approach on an indoor testbed. As far as we know, this is the first work on formal passive testing of routing protocols in WSN.

Our paper's primary contributions are:

- A logic based approach for formally specifying some functional requirements of WSN routing protocol. We provide an algorithm to evaluate these properties on protocol execution traces that are collected at runtime through PO set on the nodes.
- With a real indoor testbed and the implementation of a new routing protocol and the implemented algorithm, we demonstrate the efficiency and suitability of our approach by providing relevant testing verdicts. The modeled requirements do not need to be compulsorily standardized, which helps the development life cycle processes.

The rest of this paper is organized as follows: Section II discusses existing work in the literature. We describe the formal approach of our passive testing framework in Section III and provide details of the sensor networking protocol to be tested in Section IV. Section V contains details of the experimental setup and evaluation results. We conclude our work in Section VI.

II. Related works

Although there exist many studies on the protocols evaluation in WSN, they mostly tackle their performance, application or security issues [3], [4]. Very few works focus on the conformance testing of these protocols. In the following, we present the ones from which we got inspired and compared with.

In [11], the authors study common functional testing approaches to tackle the security issues in wireless sensor networks. Since conformance testing techniques cannot assess the efficiency of the implementation of a security protocol (mainly because most of the security concepts are close to non-functional mechanisms), the authors propose to add randomness tests to conformance tests dedicating to the evaluation of security functions. Like in the previous proposal, [12] recently proposed a remote testing method by stating the design and development of test routers in WSN. For that purpose, a remote testing platform is developed and assessed. The authors tackle the bad environment in which nodes are intensively placed. They study the energy consumption and the dynamic change of the network topology. These inherent constraints often prevent the testers to meet the tested protocol requirements. Still with the same objectives, a formal testing approach is proposed in [13]. A nodes' self similarity approach for testing wireless mobile ad hoc networks (MANET) is presented. Formal specifications and emulations are applied through the DSR protocol to provide relevant verdicts about the test of one of its implementations DSRuu. We also got inspiration from [14] in which real wireless sensor nodes are deployed to test, through virtualized emulated nodes, TCP and some functions of their own faulty routing protocol RMRP (Rime mesh routing protocol). A state model is defined, test suites generated and executed through their architecture based on an interesting generated dynamic topology. In recent papers [15], [16], the authors provided suitable open conformance test system for standard-based WSNs, dedicated to the protocol evolution and the various hardware interfaces of sensor nodes. Though the approaches are interesting, they do not provide a way of modeling non-standardized requirements as we tackle in this work.

Though interesting results have been obtained by these works, the tests are applied through active testing architectures. The tests are intrusive and need to generate important test suites. One of our challenges is at the contrary to avoid any interaction with the implementation. By passively testing the implementation protocols, we also do not need any testing scripts and the related specification stages. Moreover, we here work on a real testbed avoiding the simulation or emulation drawbacks often met in these kind of studies.

Recently, interesting works on passive diagnostic in WSN have been presented. First we may cite [17] in which the authors propose a probabilistic diagnosis approach for inferring the root causes model of abnormal phenomena in wireless sensor networks through the passive observation of eventual symptoms. A light-weight packet marking scheme is defined to collect relevant hints and the probabilistic inference model located at the sinks targets the expected hints. Second, [18] proposes a passive observation and mining of relevant collected information to diagnose WSN performance failures. The authors technique deduces the root causes of failures by focusing on the relationships between the sensing data, the failures in the networks and a failure knowledge library. Finally, a combination of active and passive testing approach is presented in [19] for fault localisation in WSN. In this work, the scope is different from ours. The faulty components are searched by optimal end-to-end test suites whereas we test the conformance of the protocols without trying to point at the faulty entity since the implementation of the tested protocol is our objective. However, the proposed passive observation and the data aggregation procedure are of high interests in the definition of our formal approach.

We got inspired of these above mentioned approaches even if they are applied to the diagnosis or fault localization of WSN. Indeed, their symptoms or failure signatures can be compared to our formal properties although in our framework a logic-based approach is used. Moreover, some correlation processes are commonly applied in passive testing techniques. Nevertheless, we do not need any inferred models and do not impact the nodes reliability by any active processes.

III. Formal passive testing approach

A. Preliminaries

We here define the syntax allowing to describe some functional properties of network protocols.

Definition 1. A *message* of a protocol P is any element $m \in M_p$.

For each $m \in M_p$, we add a real number $t_m \in \mathbb{R}^+$ which represents the time when the message m is received or sent by the monitored entity. The data domains are defined as *atomic* or *compound*. Once given a network protocol P, a *compound* domain M_p can be defined by the set of labels and data domains derived from the message format defined in the protocol specification/requirements.

Definition 2. A *term* is defined in Backus-Naur Form (BNF) as $term ::= c \mid x \mid x.l.l...l$ where c is a constant in some domain, x is a variable, l represents a label, and $x.l.l...l$ is called a *selector variable*.

Definition 3. An *atom* is defined as the relations between *terms*, $A ::= p(term, ..., term) \mid term = term \mid term \neq term \mid term < term$.

Definition 4. A *clause* is an expression of the form $A_0 \leftarrow A_1 \wedge ... \wedge A_n$, where $A_1, ..., A_n$ are *atoms*. The relations between *atoms* are stated by the definition of *clauses*.

Definition 5. A *formula* is defined by the BNF: $\phi ::= A_1 \wedge ... \wedge A_n \mid \phi \rightarrow \phi \mid \forall_x \phi \mid \forall_{y>x} \phi \mid \forall_{y<x} \phi \mid \exists_x \phi \mid \exists_{y>x} \phi \mid \exists_{y<x} \phi$, where $\exists$ and $\forall$ represent for "it exists" and "for all" respectively.

According to these definition, we define the syntax based on the basic unit $term$. In the following sub section, we introduce the semantics and algorithm used for our testing approach.

B. Semantics and Algorithm

The semantics used in our work is related to the traditional Apt-Van Emdem-Kowalsky semantics for logic programs [20], from which an extended version has been provided in order to deal with messages and trace temporal quantifiers.

Definition 6. A *substitution* θ is a finite set of bindings $\theta = \{x_1/term_1, ..., x_k/term_k\}$ where each $term_i$ is a *term* and x_i is a variable such that $x_i \neq term_i$ and $x_i \neq x_j$ if $i \neq j$.

Given a formula ϕ defined by using a set of clauses K as we mentioned above, a satisfaction result '$\top$' ('*Pass*'), '$\bot$' ('*Fail*') or '?' ('*Inconclusive*') will be given to a particular trace ρ. Using substitution θ, we recursively evaluate the formula ϕ (i.e. protocol property) on the real protocol execution trace ρ, coupled with a modification of SLD (Selective Linear Definite-clause) resolution algorithm [21] for evaluation of Horn clauses.

The evaluation process is described as follows:

$$eval(A_1 \wedge ... \wedge A_k, \theta, \rho) = \begin{cases} \top & \text{if } A_1 \wedge ... \wedge A_k\theta \\ & \text{has a result} \\ \bot & \text{otherwise} \end{cases}$$

For every possible value x in the trace, the following formula is used:

$$eval(\forall_x \phi, \theta, \rho) = \{eval(\phi, \theta \cup x/m, \rho) | \forall_m \in \rho\}$$

If the formula $\forall_x \phi$ is included in another formula, the result of evaluation is provided by:

$$eval(\forall_x \phi, \theta, \rho) = \begin{cases} & \text{if } \exists_m \in \rho \text{ where} \\ \bot & eval(\phi, \alpha, \rho) = \top \text{ with} \\ & \alpha = \theta \cup x/m \\ ? & \text{otherwise} \end{cases}$$

The result "$\bot$" represents for any violation has been found. The evaluation of $\exists_x$ is quite similar to the $\forall_x$, but it is looking for a "$\top$" result.

$$eval(\exists_x \phi, \theta, \rho) = \begin{cases} & \text{if } \exists_m \in \rho \text{ where} \\ \top & eval(\phi, \alpha, \rho) = \top \text{ with} \\ & \alpha = \theta \cup x/m \\ ? & \text{otherwise} \end{cases}$$

The evaluation processes for $\forall_{y>x}$, $\exists_{y>x}$ and $\phi \rightarrow \varphi$ are the same as described before, we will not repeat them here. The interesting readers can have a look at our previous work [10], [22].

We also provide the evaluation algorithm in the following. The algorithm starts by checking the existence of a trace

Algorithm 1: Algorithm for eval(ϕ, θ, ρ)

```
Input: Formalized requirement $\phi$, Substitution $\theta$ with initial
       bindings, and finite trace $\rho$
Output: Pass if the formula has a solution, Fail if exist a violation of
        the requirements, '?' if no definite response can be provided
if $\rho$ is not empty, $\phi$ is not empty then
    $verdict \leftarrow \top$;
    for $(m_0 \wedge ... \wedge m_n) \in \rho$ do
        $\theta^{'} \leftarrow \theta m_i$;
        for $(A_0 \wedge ... \wedge A_n) \in \phi$ where $verdict \neq \perp$ do
            if $\theta^{'} A_0 \wedge ... \wedge \theta^{'} A_n = \top$ then
                $verdict \leftarrow \top$, return $verdict$;
            end
            else
                $verdict \leftarrow \perp$, return $verdict$;
            end
        end
        if $verdict = \top$ then
            next $m_i$, $final \leftarrow pass$, return $final$;
        end
        else
            next $m_i$, check_end_of_file($m_i$);
            if $check\_end\_of\_file(m_i) = \top$ then
                $final \leftarrow inconclusive$, return $final$;
            end
            else
                $final \leftarrow fail$, return $final$;
            end
        end
    end
end
```

ρ and a requirement ϕ. Then if ϕ contains sub formulas, they will be sequentially tested by using recursive calls. For testing a formula ϕ on a finite trace ρ, the algorithm will firstly assigns the values to substitution θ from each message m in the trace. Then the obtained $\theta^{'}$ will be used to compare with each atom in the formula ϕ. If all the atoms in ϕ are satisfied, a truth value '$\top$' will be assigned and the algorithm will step to test the next message m. Otherwise, any violation of the atoms will result to a truth value '$\perp$' and the algorithm will immediately terminate the comparing process and step to test the next message m. The truth values '$\top$' and '$\perp$' will be eventually transformed to the verdict '*Pass*', '*Fail*' or '*Inconclusive*' as the semantics defined in Section 3. Finally, a final report will be provided when all the sub formulas of ϕ are tested through the trace ρ.

The complexity of the algorithm is decided by the number of quantifiers used in the formula being tested. In the worst-case, the time complexity of our algorithm with k quantifiers is $\mathcal{O}(n^k)$ to analyze the trace, where n represents the number of messages in the trace. In the following section, we present a newly designed routing protocol to be tested. It has been specified, developed and integrated in a real indoor testbed.

IV. Protocol Description

We have designed our proprietary end-to-end networking stack for large-scale WSN deployments, with modular components that are configurable, extensible as well as plug-and-playable, depending on sensing application requirements and physical environment of the sensing region of interest. It is designed for scalability through the use of lightweight protocols that minimize both communication and storage overheads. Existing networking protocols are often designed in silo, and thus unable to support our multi-faceted sensor network testbed requirements. In the following, we briefly describe its basics, the route establishment, the data transmission and packet format.

A. Basics

A beacon is periodically broadcast by each node (sensor and gateway) throughout its network lifetime, and is used for:

1) Neighbour discovery
2) Estimation of link quality between neighbouring nodes
3) Update of gradients throughout the network

It contains the:

1) Source node identifier
2) Monotonically increasing sequence number for detection of expired links and prevention of routing loops
3) Gradient towards the gateway

During network initialization, each sensor node has a gradient value of infinity. The gateway then updates its gradient to be zero and sends this information together with its current sequence number, in its next beacon. Upon receiving a non-infinite value of the gradient in a beacon from a neighbouring node v_j, the sensor node v_i updates its gradient if the following conditions are satisfied: (i) estimated link quality between v_i and v_j is above a pre-defined threshold T; and (ii) gradient of v_j is smaller than the gradient of v_i.

When a sensor node v_i has a data packet to forward to the gateway, the packet is broadcast into the wireless medium together with the gradient of v_i. A neighbouring node v_j that receives the data packet from v_i will send an acknowledgement and help to forward the packet only if the gradient of v_j is smaller than that of v_i. The process is repeated along each hop until the gateway receives the data packet.

B. Route Establishment

At periodic intervals of 30 seconds, each node broadcasts a BEACON message to its one-hop neighbours for the purpose of neighbour discovery, topology maintenance, time synchronization and route discovery to the gateway. Each sensor node is associated with a gradient (metric) towards the gateway node. The metric of an arbitrary node v_i is denoted as m_i. Generally, packets flow from nodes with higher gradients to nodes with lower gradients. In our testbed, the gateway v_g is initialized with a metric value of $m_g = 0$; all other nodes in the network are initialized with a default invalid metric value of -1. This metric information is piggybacked in the beacons that are transmitted by each node.

C. *Opportunistic Data Transmission*

Data transmission takes place in an opportunistic fashion within the network. A node v_j with data to send to the gateway will include its current metric m_j in the DATA packet, which is broadcast to the local one-hop neighbourhood. An arbitrary neighbour v_i that receives the data packet will send an acknowledgement packet ACK to v_j, if it has a better metric to the gateway, i.e. $m_i < m_j$. As wireless links may fluctuate, the transmitting node v_j can retransmit data packets up to a maximum of 5 times before the packet is discarded from its transmit buffer. The process is repeated along each hop until the gateway receives the data packet. Due to the opportunistic nature of the routing protocol, it is possible for duplicate copies of the same packet to arrive at the gateway by using sequence numbers. The duplication of packets is resolved at the back-end through the use of sequence numbers.

D. *Protocol Format*

There are three possible types of packets that are used in the protocol:

- BEACON packets that are broadcast to neighboring nodes at periodic intervals of 30 seconds.
- DATA packets that are generated by the sensor nodes at periodic intervals of 120 seconds.
- ACK packets that are generated by the gateway or intermediate forwarding node (i.e. node that receives the data from another node with a higher metric).

The corresponding format of each packet is as follows:

- BEACON: [sequenceNum], [unixTime], [gatewayAddress], [gatewaySequenceNum], [metricToGateway]
- DATA: [portNumber], [dataLength], [metricToGateway], [dataSequenceNumber], [routeIncluded]
- ACK: [dataSource], [dataSequenceNumber]

Example 1. Transmitted message (BEACON): 705, 65535, 0, 3, 9, 1404388288, 700, 1404388247, 30

In Example 1, we illustrate a beacon packet of sequence number 9, that is broadcast by node 705 to its neighboring nodes at time 1404388288. The beacon contains the information that the metric from node 705 to the gateway 700 is 30, and that the last known gateway sequence number from 700 is 1404388247. The gateway sequence number is used to expire stale routes and prevent routing loops in the network. We present in the following the testing results we obtained while testing this new protocol from four properties.

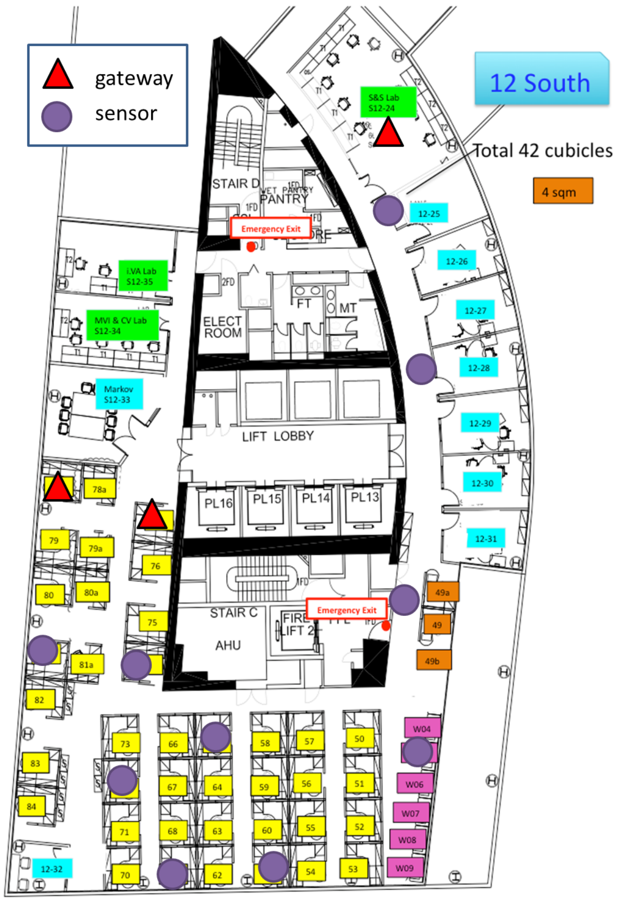

Figure 1. Experiments environment

V. Experiments

A. *Testbed description*

We evaluate our protocol in an indoor wireless sensor network testbed that is deployed on the twelfth floor of an office building. The testbed comprises a total of ten Arduino-based sensor nodes and multiple Linux-based gateways. Each sensor node is equipped with appropriate sensor hardware that enables it to collect data of various physical environmental modalities such as temperature, noise and humidity. The sensor data from each node is periodically transmitted to the gateways through multiple hops, via an IEEE 802.15.4 compliant radio interface.

Figure 1 illustrates the location of the nodes (101 to 110 for the sensor nodes, and 500 and 501 for the gateways) in the testbed, which forms a connected multi-hop topology. Due to the lack of energy harvesting sources in the building, each node is connected to a constant power supply source.

B. *Traces description*

As illustrated in Figure 1, ten sensor nodes and multiple gateways are used in our experiments. Traces are collected from each of these nodes in the following way: as the nodes have limited on-board storage, and solely for the testing purpose of packet trace collection, a wired USB connection between each node and a PC is provided. Whenever a node transmits or receives a message, the message content is recorded in the PC, alongside with the timestamp of the message (for both transmission and reception) and the RSSI value of the message (for reception only).

Packet traces have been collected from each of the nodes in the network, over a period of two days, and with differing test scenarios. During certain periods in time, one of the gateways may be deliberately switched off - to simulate the effect of an actual gateway that has depleted its energy in the outdoor deployment. The routing protocol is expected to recover in such scenarios by re-routing data packets to the only other existing gateway in the network.

C. Test Results

The tests are performed in a real machine with 8GB RAM and one processor Intel i7 @3.4GHz. They are also performed through a prototype implementation of the formal approach above mentioned, it is developed in C code by using the algorithm introduced in the previous section. Firstly, we test a property commonly used for all WSN routing protocols–No routing loops.

1) Property1: A packet should not be routed to the same node more than once before reaching the intended destination. By using the syntax and semantics we mentioned before, this property can be simply formalized as:

$$\forall_{y>x}(receive(y) != message(x))$$

where $receive(y)$ represents for any message received after message x.

Traces	Total Messages	Pass	Fail	Inconclusive
101	67409	67409	0	0
102	95206	95206	0	0
103	81709	81709	0	0
104	84590	84590	0	0
105	69274	69274	0	0
106	70113	70113	0	0
107	79807	79807	0	0
108	78483	78483	0	0
109	87196	87196	0	0
110	62235	62235	0	0
Gateway 500	60805	60805	0	0
Gateway 501	52444	52444	0	0

Table I
TEST RESULTS FOR PROPERTY 1

The testing results are shown in Table I, unsurprisingly, no Fail verdict has been found. All the messages in the traces passed this property. Since this property is just a simple one for all the WSN routing protocol, for verifying the efficiency of our approach, we step to test more complex properties.

2) Property2: Every DATA packet will be retransmitted until one of the condition has been satisfied: (i) Up to a maximum of 5 times or (ii) An ACK packet is received. This property can be differentiated to two situations, and by using the syntax and semantics we mentioned before, they are formalized as:

(i) Up to a maximum of 5 times

$$\forall_x(repeat(x, \mathbf{DATA}) \to \exists_{u<x}(repeat(u, \mathbf{DATA})) \to \exists_{v<x}(repeat(v, \mathbf{DATA})) \to \exists_{w<x}(repeat(w, \mathbf{DATA})))$$

where $repeat(x, \mathbf{DATA})$ is defined by:

$$(request(x) \wedge x.type = \mathbf{DATA} \to \exists_{y<x}(request(y) \wedge y.type = \mathbf{DATA} \wedge x.dataSequenceNum = y.dataSequenceNum))$$

(ii) An ACK packet is received

$$\forall_x(repeat(x, \mathbf{DATA}) \to \exists_{z<x}(responds(z, x) \wedge z.type = \mathbf{ACK}))$$

Compared to the property 1, this property written in the new protocol requirements is far more complex. We test this property through the traces separately collected in one day. The results are shown in Table II. The tables include the results for traces collected from different nodes (101-110) and gateways (500, 501).

Traces	Total Messages	Pass	Fail	Inconclusive
101	67409	22446	0	0
102	95206	29552	0	0
103	81709	32243	0	1
104	84590	29128	0	0
105	69274	23981	0	0
106	70113	23983	0	0
107	79807	26717	0	0
108	78483	29330	0	1
109	87196	26117	0	0
110	62235	19837	0	0
Gateway 500	60805	17639	0	1
Gateway 501	52444	14659	0	0

Table II
TEST RESULTS FOR PROPERTY 2

For better illustrating the results, the proportion of verdicts are also shown in Figure 2. As expected, most of the traces return 'Pass' verdicts and no 'Fail' are detected. However, several 'inconclusive' verdicts can be observed from 500, 103 and 108 in Table II.

After the analysis of these verdicts through the collected traces, we find out that they are caused by uncertain existence of 'ACK' packets. Indeed, according to the formalized formula, we search an 'ACK' before the received 'DATA' packet. Nevertheless, when the 'DATA' packet is received at the beginning of a trace, we cannot conclude whether there is an 'ACK' or not. As a result, our algorithm produces 'Inconclusive' verdicts.

Although few 'Inconclusive' have been observed, we may notice that no 'Fail' are raised during the testing process.

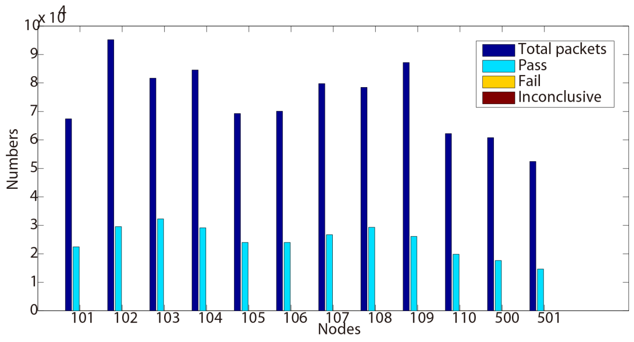

Figure 2. Test results for property 2

This shows that this functional tested property conforms to the requirements through these nodes during one day.

Another discussion can also be opened about these 'Inconclusive'. When testing a protocol, these kinds of verdicts could be compared to interferences. The human testers have then to analyze in depth if they are not false positives/negatives or errors arised during the testing process. In our case here, these verdicts are detected on different nodes in one day. The reason is that our sniffer is manually triggered forgetting the beginning of the executions.

Therefore, in order to reduce these types of verdicts (in some experiments, depending on the protocol and the tested properties, their number can be very important), works could be performed to rewrite the formula to avoid looking in the past of the trace. Two consequences could however occur: (i) our issue is moved to the end of the trace and (ii) the analysis complexity can be increased (or eventually decreased as well). These aspects also make part of our future works.

3) Property3: An ACK packet from a node v_i will be sent to another node v_j only if both conditions are satisfied: (i) v_i receives a DATA packet from v_j and (ii) v_i (currently) has a smaller metric than v_j. This property can be formalized as:

$$\forall_x(response(x) \wedge x.type = \textbf{ACK} \rightarrow$$
$$\exists_{y<x}(request(y,x) \wedge y.type = \textbf{DATA} \wedge i.metric < j.metric))$$

We still test this property through the traces collected and used for previous properties. The results are shown in Table III and Figure 3.

Different from no 'Fail' verdicts in property 2, we can be observe large number of fail verdicts in node 101, 103, 107 and 110. When we analyze these fails verdict, they are all caused by the violation of 'v_i has a smaller metric than v_j'. These 'ACK' packets are transmitted even the v_i has a greater metric than v_j.

This draws our attention on the implementation of these nodes. We went through the codes for node 101, 103, 107 and 110, found out that there exists a configuration error in these nodes, which leads to this phenomenon.

Traces	Total Messages	Pass	Fail	Inconclusive
101	67409	11105	11341	0
102	95206	29552	0	0
103	81709	16171	16072	0
104	84590	29128	0	1
105	69274	23981	0	0
106	70113	23983	0	2
107	79807	13252	13365	0
108	78483	29330	0	1
109	87196	26117	0	0
110	62235	10016	9821	1
Gateway 500	60805	17639	0	0
Gateway 501	52444	14659	0	0

Table III
TEST RESULTS FOR PROPERTY 3

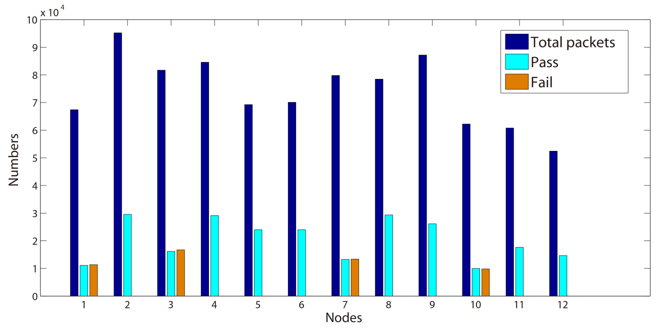

Figure 3. Test results for property 2

After we fixed this implementation error, there is no such fail verdict report again in the second day testing results. These testing results sufficiently prove our approach can detect errors, and can help the developers to find and fix existing bugs in the implementation.

VI. Conclusion

This paper presents a formal approach for formally testing functional requirements of a novel routing protocol used in wireless sensor networks. The routing protocol is designed by considering the inherent constraints of our smart system requirements and to cope with the drawbacks and lacks of current available commercial protocols for our outdoor urban large-scale network. The main objective was therefore to test this new developed WSN routing protocol before its deployment, in a formal way, through a real testbed and without being intrusive. For that purpose, we defined a logic based syntax and semantics to model the functional properties of the protocol. We also designed an efficient algorithm to evaluate these properties on real extracted execution traces.

Our approach has been successfully evaluated by experiments on the WSN testbed. We have shown that our formal approach has several advantages, among others: (i) the syntax and semantics are simple enough to allow any engineers (not only testers) to design their own functional properties to be tested, (ii) we may test several nodes at the same moment in a very short time and (iii) we do not need neither standardized properties nor complete formal models to test a protocol, this is highly convenient while testing a new protocol during its development and deployment periods.

For future work, we will study the rewriting of some specific functional properties and its impacts on the testing process. We will apply and assess our approach on our large-scale outdoor deployed WSN focusing on the reliability of our algorithm, our protocol and specific behaviors between gateways.

Acknowledgment

This work was supported by the French ICT Asian project URSA - Grant n377 and Chinese Fundamental Research Funds for the Central Universities, BJTU Talent Fond (2015RC046). The authors also would like to thank Prof. Chao CHEN (Chongqing University) for his useful advices.

References

[1] Mallanagouda Patil and Rajashekhar C. Biradar. A survey on routing protocols in wireless sensor networks. In *18th IEEE International Conference on Networks*, pages 86–91, 2012.

[2] Sheng Yu, Baoxian Zhang, Cheng Li, and Hussein T. Mouftah. Routing protocols for wireless sensor networks with mobile sinks: a survey. *IEEE Communications Magazine*, 52(7):150–157, 2014.

[3] M. Woehrle. *Testing of wireless sensor networks*. PhD thesis, 2010.

[4] Arunanshu Mahapatro and Pabitra Mohan Khilar. Fault diagnosis in wireless sensor networks: A survey. *IEEE Communications Surveys and Tutorials*, 15(4):2000–2026, 2013.

[5] Miao Xie, Jiankun Hu, and Song Guo. Segment-based anomaly detection with approximated sample covariance matrix in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 26(2):574–583, Feb 2015.

[6] X. Chang, J. Huang, S. Liu, G. Xing, H. Zhang, J. Wang, L. Huang, and Y. Zhuang. Accuracy-aware interference modeling and measurement in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 2(5):72–85, Feb 2015.

[7] W. Si, M. Hashemi, L. Xin, D. Starobinski, and A. Trachtenberg. Teacp: a toolkit for evaluation and analysis of collection protocols in wireless sensor networks. *IEEE Transactions on Network and Service Management*, 4(3):1–15, Feb 2015.

[8] David Lee et al. Network protocol system monitoring: a formal approach with passive testing. *IEEE/ACM Trans. Netw.*, 14(2):424–437, 2006.

[9] Robert M. Hierons et al. Using formal specifications to support testing. *ACM Comput. Surv.*, 41(2), 2009.

[10] Felipe Lalanne and Stephane Maag. A formal data-centric approach for passive testing of communication protocols. *IEEE / ACM Transactions on Networking*, 21(3):788–801, 2013.

[11] Shirui Ji, Qingqi Pei, Yong Zeng, Chao Yang, and Shu po Bu. An automated black-box testing approach for WSN security protocols. In *Seventh International Conference on Computational Intelligence and Security (CIS)*, pages 693–697, 2011.

[12] Haofei Xie, Liu Wei, Jinyan Zhou, and Xiao Hua. Research of conformance testing of low-rate wireless sensor networks based on remote test method. In *Fifth International Conference on Computational and Information Sciences (ICCIS)*, pages 1396–1400, June 2013.

[13] Cyril Grepet, Stephane Maag, and Ana R. Cavalli. A formal validation methodology for manet routing protocols based on nodes' self similarity. *Computer Communications*, 31(4):827–841, 2008.

[14] Junjie Xiong, E.C. Ngai, Yangfan Zhou, and M.R. Lyu. Realproct: Reliable protocol conformance testing with real nodes for wireless sensor networks. In *IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 572–581, Nov 2011.

[15] Zhonghua Zhao, Wei Huangfu, Limin Sun, Zhiqiang Shi, and Wei Gan. An open conformance test system towards the standardization of wireless sensor networks. *IJDSN*, 2012, 2012.

[16] E.Y. Song, K.B. Lee, and F. Proctor. Testing system for ieee 1451.5-802.11 standard-based wireless sensors. In *Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, 2014 IEEE International*, pages 862–867, May 2014.

[17] Yunhao Liu, Kebin Liu, and Mo Li. Passive diagnosis for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 18(4):1132–1144, 2010.

[18] Jiangwu Nie, Huadong Ma, and Lufeng Mo. Passive diagnosis for wsns using data traces. In *8th IEEE International Conference on Distributed Computing in Sensor Systems*, pages 273–280, 2012.

[19] Bing Wang, Wei Wei, Hieu Dinh, Wei Zeng, and Krishna R. Pattipati. Fault localization using passive end-to-end measurements and sequential testing for wireless sensor networks. *IEEE Trans. Mob. Comput.*, 11(3):439–452, 2012.

[20] MH Van Emden and RA Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(44), 1976.

[21] Krzysztof R. Apt and Maarten H. van Emden. Contributions to the theory of logic programming. *J. ACM*, 29(3):841–862, 1982.

[22] Xiaoping Che and Stephane Maag. Passive performance testing of network protocols. *Computer Communications*, 51:36–47, 2014.