A Passive Conformance Testing Approach for a Manet Routing Protocol

Ana Cavalli
TELECOM & Management SudParis
CNRS Samovar
9, rue Charles Fourier
F-91011 Evry Cedex 11, France
Ana.Cavalli@it-sudparis.eu

Stephane Maag
TELECOM & Management SudParis
CNRS Samovar
9, rue Charles Fourier
F-91011 Evry Cedex 11, France
Stephane.Maag@it-sudparis.eu

Edgardo Montes de Oca
Montimage
39, rue Bobillot
F-75013 Paris, France
Edgar.Montesdeoca@montimage.com

ABSTRACT

In this paper we propose a passive conformance testing technique applied to a Mobile ad hoc network (MANET) routing protocol, OLSR, that is characterized by a dynamically changing topology and lack of centralized management. This makes it necessary to investigate new ways to test complex scenarios and configurations. The work here proposes a formal passive testing method to test the conformance and reliability of the protocol. The method developed has been performed on a real case study showing that the approach can be successful applied and that it allows reducing inconclusive verdicts often observed using other methods.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing protocols*; D.2.5 [**Software Engineering**]: Testing and Debugging—*Testing Tools*; I.6 [**Simulation and Modeling**]: Model Validation and Analysis

General Terms

Design, Reliability, Verification

Keywords

MANET, Routing Protocols, Conformance Testing

1. INTRODUCTION

A wireless mobile ad hoc network (MANET) is a self-organizing network that can rapidly be deployed since neither a centralized control nor a predefined infrastructure is necessary. The network is composed of mobile nodes that communicate with each other, participating in the establishment of reliable operations. Node movement may lead to a volatile network topology with continuous modification of the node interconnections. Since the network is infrastructureless, the nodes must interact using their radio range with open transmission medium and, in order to establish end-to-end communications, some of the nodes will behave as routers. Due to these aspects and the limited resources of the nodes, efficient and reliable routing protocols are required leading to the crucial and challenging issue of the quality of the communication system.
Several routing protocols have been proposed and many works have been carried out in order to design and test them. Most of this work relies on protocol descriptions using simulators in order to have an idea of the implemented protocol behavior. However, the test coverage is rather low and the verdicts are only restricted to the simulated context. Nevertheless, some works [2] have shown, for instance, that the AODV implementation in NS-2 was false regarding some properties such as the initial value for the hops number in a RREP. All of this illustrates the importance of formal methods to test routing protocols.

In this paper, we propose, therefore, a conformance passive testing technique applied to a MANET routing protocol. The methodology is performed on a real case study through a real implementation of the Optimized Link State Routing protocol (OLSR). The motivation of our work is based on the observation that using active testing methods, the verdicts of an executed test scenario were sometimes (not to say often) inconclusive[9]. This is due to the mobility of the nodes and the inherent constraints of wireless ad hoc networks, making active methods not always adapted. On the other hand, the non-intrusiveness of passive testing makes it possible to take into account topological changes in the network and to test the protocol under operation in complex changing conditions. We herein present the contribution of a formal passive testing approach that increases the testing coverage, illustrating the complementarity of these methods.

The paper is organized as follows. Section 2 presents related works on passive testing techniques based on formal methods. Section 3 gives an outline of the OLSR protocol. In section 4, the passive testing approach based on invariants is presented. Section 5 presents the testbed, the experiments that have been performed and the results. Finally we conclude and present future works in Section 6.

2. RELATED WORKS

There are few works based on formal passive testing ap-

SAC'09 March 8-12, 2009, Honolulu, Hawaii, U.S.A.

proaches and, in particular, for MANET routing protocols. This is mainly due to the specific constraints of MANET, e.g. the integration of testers in a mobile, wireless and infrastructureless environment. Nevertheless, some interesting works dealing with a passive testing approach can be mentioned. Some of them apply these techniques to wired protocols as in [6] where two algorithms based on Event-driven Extended Finite State Machines are proposed. They are designed to test the protocol data part. The expected properties are specified in a symbolic logic expression in order to be able to define in detail the set of valid variable values. The method is applied to the wired network routing protocol, OSPF, but it is not really suitable for the MANET.

In [7], the authors propose a formal methodology to specify and analyze a MANET routing protocol. It is based on the Relay Node Set (RNS) concept. A RNS is a set of nodes that allow reaching all nodes in the network. According to the studied protocol, the set is built differently: the reactive protocols build the set during the route discovery while the proactive ones build it in a regular manner. The framework illustrated in this paper has as main goal the analysis of the implementation under test using non-functional metrics. The approach we propose has as main goal to check functional properties of the protocol.

Higashino et al. [11] propose a testing architecture for DSR in order to check functional and non-functional properties. A passive technique is applied through a network simulator MobiREAL[1]. The impact of the chosen mobility models and the underlying wireless transmission layers are measured. The use of a passive testing technique is interesting but it has only been applied on a simulator.

[5] is another passive approach to formalize testing of a MANET routing protocol by applying game theory models and concepts. A strong hypothesis commonly applied in game theory consists in the complete knowledge by the "players" of the "game". It means that each node is supposed to have a complete knowledge of the network topology as well as of the nodes or links states. Furthermore, it forbids the non-determinism, making it impossible to model the dynamicity of nodes. These assumptions may not hold in a real case study.

In this paper, we contribute to the state of the art by proposing a new methodology based on passive testing techniques which describes relevant properties of OLSR and check them both on the execution traces of the protocol implementation and on the formal specification.

3. OLSR AND ITS FORMAL MODEL

3.1 Optimized Link State Routing protocol

OLSR is a proactive, link state routing protocol (LSR) which uses periodic message exchanges to update topological information in each node of a network [3]. OLSR distinguishes itself from classical LSR protocols by introducing a simple optimized flooding strategy through the use of Multi Point Relays (MPR). Each node must select MPRs among its neighbors in order to reach all 2-hops neighbors through its MPR set. The MPR nodes themselves maintain a list of MPR selectors, describing the set of nodes which have selected it as a MPR. The MPRs, then, have the responsibility to forward all messages from their MPR selectors and not relay messages from any other node.

[1]http://www.mobireal.net.

3.2 The OLSR formal model

In this work, a formal model of the protocol is provided to check the correctness of the manually derived properties from the IETF RFC [3]. It is specified using the Extended Finite State Machine (EFSM) formalism that is commonly used to specify and test communication protocols.
For our purpose, according to our OLSR implementation and real testbed, only the main functionalities of the protocol are specified. This means that we consider each node as having only one interface and with only one link to a same node. Since OLSR is a link state routing protocol, the behavior of every node is specified according to its state and its connectivity with its neighbors (**I**dle, **A**symmetric, **S**ymmetric and **M**PR). OLSR defines different types of links, asymmetric or bidirectional and also different types of nodes, normal nodes or multipoint relays that implies that they have a different role in the establishment of connections. Our EFSM only represents the interactions between two nodes. This implies that, in our network, for each link between two nodes there is a unique EFSM. For further details about the formal specification, you can find a complete description and its implementation in [10].

4. AN INVARIANT-BASED PASSIVE TESTING APPROACH

4.1 Passive versus Active Conformance Testing

Conformance testing usually relies on the comparison between the behavior of an implementation and a formal specification, that should be the same. Two main approaches are commonly applied: passive and active testing.

Active testing is based on the execution of specific test sequences against the implementation under test. These test sequences are generated from the formal model. The test may be generated automatically or semi-automatically from formal models that represent test criteria, hypothesis, and test goals. These sequences (with an executable format) are performed by establishing points of control and observation (execution interfaces) defined by the testers.

Passive testing consists in observing the exchange of messages (input and output events) of an implementation under test during run-time. The term passive means that the tests do not disturb the natural operation of the protocol. The record of the observed events is called a trace. This trace will be compared to properties (also called invariants) derived from the the standard (e.g. RFC) or proposed by the protocol experts. The passive testing techniques are applied particularly because they are non-intrusive whereas the active testing techniques require the set-up of important testing architectures where the testers need to be able to control the implementation at some specific points. This is sometimes not feasible, especially when there is not direct access to the implementation under test. For MANET protocols, with an infrastructureless environment and where the mobile topology brings constraints that do not exist in wired networks, active testing techniques are not always well adapted to check the given properties. Moreover, while active testing may cover a wide set of automatically generated properties, it may also provide many inconclusive verdicts [9]. For these

reasons we focus here on a passive testing approach where no inconclusive verdicts are obtained. This is possible because we define obligation invariants (defined in the next section), meaning that the verification is performed for events that have already occurred in the past and, thus, we can be sure that they are in the traces if the capture of the events includes the initialization of the protocol at the start.

4.2 The concept of Invariant

The passive testing approach used in this work is based on invariant analysis, where the invariants are properties the Implementation Under Test (IUT) (in this work the OLSR implementation) is expected to satisfy.
An invariant (or *obligation* invariant) is defined as follows:
Let $M = (S, I, O, s_{in}, Tr)$ be a FSM where S is a finite set of states, I a set of input actions, O a set of output actions, s_{in} an initial state and Tr a set of transitions. Each transition $t \in Tr$ is a tuple $t = (s, s', i, o)$. Intuitively, a transition $t = (s1, s2, i, o)$ indicates that if the machine is in state $s1$ and receives the input i then the machine emits output o and moves to the state $s2$.
Intuitively, a trace such as $i1/o1, ..., i_{n1}/o_{n1}, i_n/o_n$ is an obligation invariant for M if each time i_n/o_n is observed, then the trace $i1/o1, ..., i_{n1}/o_{n1}$ happens before. An invariant that expresses a property, such as "if y happens then we must have that x has happened before", is an obligation invariant. They may be used to express properties where the occurrence of an event must be necessarily preceded by a sequence of events. In addition to sequences of input and output symbols, the wild-card characters '?' and '*' are allowed, where '?' represents any single symbol and '*' represents any sequence of symbols.

Invariants can be checked both on the specification and on the IUT, allowing to determine whether the property is correct according to the formal model and the IUT behavior is as defined by the RFC. The algorithms, as well their complexity analysis, are detailed in [1] for the specification/invariants checking and in [4] for the implementation/invariants checking.

4.3 The Conformance Passive Testing approach

Conformance testing aims to test the correctness of an implementation through a set of invariants (or properties) and traces (extracted from the running implementation). The conformance passive testing process follows these five steps:
- Step 1: Properties formulation. The relevant protocol properties to be tested are provided by the standards or by protocol experts. These properties express the main requirements related to wireless communication between nodes.
- Step 2: Properties as invariants. Properties have to be formulated by means of obligation invariants that expresses local properties; i.e. related to a local entity.
- Step 3: Extraction of execution traces from local observers by means of a network sniffer installed on one of the nodes. The captured traces are in XML format.
- Step 4: Transformation of traces in an adapted format. The traces are transformed by the application of filtering rules defined by a XSL sheet. These rules hold the network information needed to perform the checking of the properties described by the invariants.
- Step 5: Verification of the invariants on the traces. The verification of the expected properties is performed on the traces and a verdict is emitted (Pass, Fail or Inconclusive).

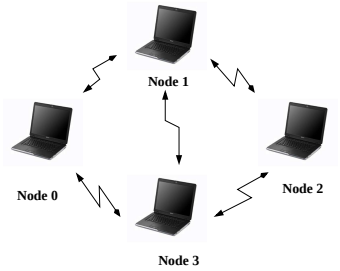

Figure 1: The wireless testbed

This step is based on the application of pattern matching algorithms we have developed (presented in [4]).

5. EXPERIMENTATION AND RESULTS

5.1 The Real Wireless Testbed

Our testbed is composed of four laptops as illustrated in the Figure 1. Let us note that we could increase the number of nodes using an emulation technique embedded in one of the nodes (see [9]). This technique is nevertheless useless regarding our current test purposes and four nodes is the minimal number of nodes necessary to check the properties we are interested in.

Three laptops embed an Intel Pro Wireless 802.11 a/b/g Wifi card (two of them with Fedora Core 8 and one with a Suse Linux 9.3). The fourth one embeds a WPN111 Wireless USB adapter on a Fedora Core 8. All the nodes were configured in ad hoc mode and they run an OLSR implementation version olsrd-0.5.6-rc1[2]. The protocol is configured to run in the wireless interfaces of the nodes and with no link quality measurements in order to comply with the RFC.
At the beginning, all the nodes are close to each other as shown in the figure. Then, Node 2 moved away from the others until it lost the communication. After, it joined the network again.
The trace is captured using Wireshark Network Analyzer [3] in Node 0 at the beginning of all packet transmissions in order to ensure that the start state is captured in the trace (mainly for the obligation invariant).
The provided trace contains all the packet information extracted from the network including information from the different protocol layers (ethernet, ip, udp and olsr). The OLSR layer provides the information that we need (source, destination, message type, link type and list of neighbors). The trace is in XML format allowing the use of open source tools (i.e. Michael Kay's SAXON XSLT processor) that use XSL style sheets to filter and format the information based on the data provided by the user (address of the node that is being examined to be able to define which packets are input and which are output.

5.2 The Invariants

The invariants presented in this section describe the process of connectivity by ensuring that a node $n0$ will respect the procedure followed by the protocol to establish the links with its neighbors. This procedure also ensures that a node $n0$ cannot be fooled by another node $n1$ by announcing a

[2] `http://www.olsr.org`
[3] `http://www.wireshark.org/`

non-existent connection. The process of connectivity follows an implicit logical order in the exchange of link sensing messages. Thus, an asymmetrical link announcement can only reply to a first empty *HELLO* message broadcasted from a node announcing itself. In the same way, a symmetrical link advertisement must follow the reception of an asymmetrical one from a neighbor node. A node can claim to be a *MPR* only if it has been selected as such by at least one 1-hop neighbor. This implicitly implies that a symmetrical link already exists between them. It is important to notice that for every invariant, the number of messages to be checked is always bounded. This number directly depends on the number of neighbors for each node, which is supposed to be fixed (at a particular moment). So the number of address embedded within each message is limited. Moreover, the broadcast of control packets are timed by constants. Admittedly, *HELLO* messages can be emitted as long as a link change is detected, but nodes need to wait for a timeout (2 seconds by default) to announce the change anyway. Thus, even though the mobility of nodes is normally very high, the number of messages exchanged during the life of a link is finite. The invariants presented have the following form (following the definitions given in Section 4.2): "$?/output1, *, input/output2$" where $*$ represents any sequence of symbols, ? represents any message and $?/output1$ and $input2/output2$ represent that after input ?, $output1$ is produced and that after $input2$, $output2$ is produced.

Invariant 1

$? \ / \ MsgType = Hello \ \wedge \ NeighAddr = none \ \wedge \ Source = n0 \ \wedge \ Dest = any, \ *, \ MsgType = Hello \ \wedge \ Source = n1 \ \wedge \ Dest = any \ \wedge \ LinkType = Asym \ / \ ?$

This invariant illustrates the case where a node starts the neighbor detection mechanism by sending *HELLO* messages. In this case, the link established is asymmetrical because a neighbor answers, announcing this link. It is expressed as an obligation invariant, which means that if a packet is received by node $n0$ announcing an asymmetrical link between nodes $n0$ and $n1$, then it is mandatory that $n0$ should have already sent an empty *HELLO* message to $n1$.

Invariant 2

$MsgType = Hello \ \wedge \ LinkType = Asym \ \wedge \ Source = n1 \ \wedge \ NeighAddr = n0(AsymList) / \ ?, \ *, \ MsgType = Hello \wedge LinkType = Sym \ \wedge \ Source = n1 \ \wedge \ NeighAddr = n0(AsymList) \ / \ ?$

This invariant illustrates the case where a node establishes a symmetrical link with another node. This is the case when it previously received an asymmetrical link from this other node. It is expressed as an obligation invariant, which means that if a packet is detected announcing a symmetrical link between nodes $n0$ and $n1$, then it is mandatory that $n1$ should have already established an asymmetrical link with $n0$.

Invariant 3

$MsgType = Hello \ \wedge \ LinkType = Sym \ \wedge \ Source = n1 \ \wedge \ NeighAddr = n0(AsymList) \ / \ ?, \ *, \ MsgType = Hello \wedge NeighAddr = n0 \ \wedge \ Source = n1 \ \wedge \ Dest = any \ \wedge \ LinkType = MPR \ / \ ?$

This invariant illustrates the case where the current node receives a message from a node with a symmetric link, announcing that it has been selected as MPR by this node. It is expressed as an obligation invariant, which means that if a packet is received by $n0$ from $n1$ announcing a MPR link, then it is mandatory that $n1$ should have already established a symmetrical link with $n0$.

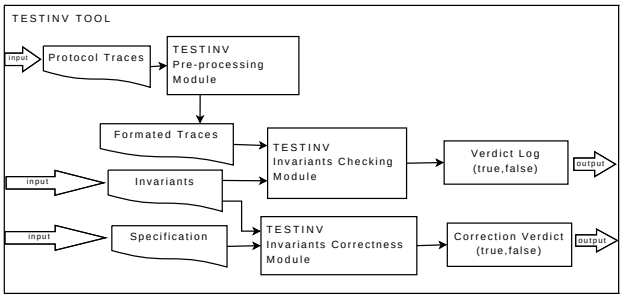

Figure 2: The TestInv tool

Invariant 4

$MsgType = Hello \ \wedge \ NeighAddr = n0 \ \wedge \ Source = n1 \ \wedge \ Dest = any \ \wedge \ LinkType = MPR \ / \ ?, \ *, \ ? \ / \ MsgType = TC \ \wedge \ Source = n0 \ \wedge \ Dest = any$

This invariant illustrates the following property: in order to update their routing tables, nodes must be kept regularly informed of changes in the topology. This is performed using Topology Control (TC) messages. TC messages are emitted periodically by each MPR to all nodes in the network to declare its MPR selector set. The result is that all nodes receive a partial topology graph made up of all reachable nodes and the set of links between a node and its MPR selectors. From this information, it is possible to compute optimal routes from a node to any destination. The property we describe here expresses that if a node has send a TC message to all nodes, then it has been selected as a MPR by at least one of its neighbors. This is expressed as an obligation invariant, which means that if node $n0$ sends a TC message to the all nodes then it must necessarily have been previously selected as MPR by at least one of its neighbors ($n1$ in our example).

5.3 The TestInv tool

The main task of the TestInv prototype is to automate the process of checking the correctness of invariants on real system traces. This prototype code has been completely written in JAVA (J2SE 1.4.0 API specification) and the graphical interfaces have been developed using the Awt and Swing Java packages. The Regex2 Java package has been used to express invariants as regular expressions. This package proposes classes to match character sequences against patterns described by regular expressions. A high level description of the tool is given in Figure 5.3.

In order to start the passive testing process, we first have to obtain real traces from a running implementation (as mentioned in Section 4.3). The Pre-processing module processes the collected trace. The input file or data stream is transformed to a suitable format and is filtered in order to obtain information concerning input and output primitives names as well as relevant data (e.g. source address, destination address, etc.). The Invariant Correctness Module checks the correctness of the invariants on the given specification of the studied system, represented as a FSM (or Extended FSM). Finally, the Invariant Checking module determines if the captured traces satisfy the given list of invariants.

TestInv also enables expressing properties that concerns other systems where properties are not only based on inputs and outputs, but also on actions, time stamps, predicates and references to different parts of the code.

5.4 Experimental results

The testbed described in section 5.1 has been used to capture the packets following the scenario also described. These traces where processed by the filtering tool and the invariant checking module. Previously the necessary input data was written in input files for the TestInv tool. These files must contain the invariants, the key words used by the protocol and the filtering information in XSL. This data and the syntax used is difficult to specify by a normal user, but happily only protocol experts need to do this. Even better, the experts only need to specify this information once for each type of protocol that wants to be observed. In this way users only need to give certain specific information and select the invariants they wish to analyze. An editing tool to help experts edit XSL and invariants is being implemented and another will help the user introduce the information needed concerning the configuration of the IUT. Currently this is done using a text editor.

The verdicts obtained where all PASS. To determine if the TestInv tool effectively detected errors we introduced false packets into the XML trace. This showed that errors could be detected, such as a node n1 that tries to fool a node n0 by announcing a non-existent connection, a node trying to convince another that it has already established a symmetrical link without following the correct sequence and a node declaring itself as a MPR node.

The results of the experimentation showed that the techniques used worked correctly on traces captured and where able to supply the verdicts on 8 Mo traces in less than one second. A challenge that remains is to use these techniques to continuously monitor the network to detect problems. For this it is necessary to improve the real-time performance of the capture and analysis of the packets. This is possible because the information needed (at least for the OLSR protocol) is limited. A packet sniffing tool needs to be developed that captures only this information, formats it and injects it into the TestInv tool, without the need to use the previously mentioned filtering tool. Nevertheless, high-speed networks will pose performance problems that need to be studied in more detail for certain types of protocols. The feasibility of real-time monitoring in the case of OLSR becomes possible due to the fact that the dynamic reconfiguration mechanism of this protocol relies on imposed timeouts (by default equal to 2 seconds).

6. CONCLUSION AND FUTURE WORKS

This paper falls in the continuity of [8] and introduces a novel way to detect flaws in MANET protocols. This was motivated by the fact that in active testing, it is not always possible to detect some types of behavior. In this paper, we propose an extensible and more flexible approach where properties can always be added depending on specific needs. This makes it possible to improve the model by adding new properties required by the standards or by the users.

As future work, we plan to provide mechanisms that allow checking global properties on nodes composing an ad hoc network. To test these properties in a distributed network, it is necessary to define time synchronization mechanisms to perform trace correlation. In this way invariants, expressing global properties, can be checked on the correlated traces. Global properties are crucial to enable the definition of network interoperability and security properties. The second aspect of the future work consists in extending our model in order to manage different types of protocols. We argue that the same approach can be generalized to all proactive protocols (e.g. TBRPF, DSDV) since many routing process aspects are similar. We also need to study how to manage reactive protocols since the problematic is different. The idea is also to extend this work to check security properties as well as conformance properties, making it possible to detect inconsistencies between the implementation of security mechanisms and their specification.

7. ADDITIONAL AUTHORS

Willy Jimenez, *TELECOM & Management SudParis*, 9, rue Charles Fourier, email: Willy.Jimenez@it-sudparis.eu

8. REFERENCES

[1] B.Alcalde, A.Cavalli, D.Chen, D.Khuu, and D.Lee. Network protocol system passive testing for fault management: A backward checking approach. In *FORTE*, pages 150–166, 2004.

[2] K. Bhargavan, C. Gunter, I. Lee, O. Sokolsky, M. Kim, D. Obradovic, and M. Viswanathan. Verisim: Formal analysis of network simulations. *IEEE Trans. Softw. Eng.*, 28(2):129–145, 2002.

[3] T. Clausen and P. Jacquet. *IETF RFC 3626: Optimized Link State Routing Protocol (OLSR).* http://www.ietf.org/rfc/rfc3626.txt, 2003.

[4] E.Bayse, A.Cavalli, M.Nunez, and F.Zaïdi. A passive testing approach based on invariants: application to the wap. *Comput. Netw.*, 48(2):247–266, 2005.

[5] I.Zakkuidin, T.Hawkins, and N.Moffat. Towards a game theoretic understanding of ad hoc routing. *Electr. Notes in Theor. Comp. Sc., 2005*, 119, 2005.

[6] D. Lee, D. Chen, R. Hao, R. Miller, J. Wu, and X. Yin. A formal approach for passive testing of protocol data portions. In *Proc. of ICNP*, pages 122–131, Washington, DC, USA, 2002. IEEE Computer Society.

[7] T. Lin, S. Midkiff, and J. Park. A framework for wireless ad hoc routing protocols. In *Proc. of IEEE Wireless Comm. and Networking Conf.*, 2003.

[8] S. Maag and C. Grepet. Interoperability testing of a manet routing protocol using a node self-similarity approach. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1908–1912, New York, NY, USA, 2008. ACM.

[9] S. Maag, C. Grepet, and A. Cavalli. A formal validation methodology for manet routing protocols based on nodes' self similarity. *Comput. Commun.*, 31(4):827–841, 2008.

[10] J. Orset, B. Alcalde, and A. Cavalli. An EFSM-based intrusion detection system for ad hoc networks. In *Proceedings of Automated Technology for Verification and Analysis (ATVA)*, Taipei, Taiwan, 2005.

[11] T.Higashino and H.Yamagushi. A testing architecture for designing high-reliable manet protocols. In *The 25th IFIP Formal Techniques for Networked and Distributed Systems - FORTE*, 2005.