Analysis of the OLSR Protocol by Using Formal Passive Testing

César Andrés*, Stéphane Maag†, Ana Cavalli†, Mercedes G. Merayo*, Manuel Núñez*
* Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid, Spain
e-mail: c.andres@fdi.ucm.es, mgmerayo@fdi.ucm.es, mn@sip.ucm.es
† TELECOM & Management SudParis
CNRS UMR Samovar, Evry, France
e-mail: stephane.maag@it-sudparis.eu, ana.cavalli@it-sudparis.eu

***Abstract*—In this paper we apply a passive testing methodology to the analysis of a non-trivial system. In our framework, so-called invariants provide us with a formal representation of the requirements of the system. In order to precisely express new properties in multi-node environments, in this paper we introduce a new kind of invariants. We apply the resulting framework to perform a complete study of a MANET routing protocol: The Optimized Link State Routing protocol.**

***Keywords* - Protocol Testing; Formal Methods; Timed Systems; Passive Testing; MANET routing protocols;**

I. Introduction

The complexity of current software systems requires the application of sound Software Engineering techniques. Among them, *testing* [1], [2] is the most widely used in industrial environments. Traditionally, formal methods and testing have been seen as rivals. Thus, there was very little interaction between the two communities. In recent years, however, testing and formal methods are learning from each other and are currently seen as complementary fields [3], [4], [5], [6].

Usually, testing is based on the ability of a tester that stimulates the implementation under test (IUT) and checks the correction of the answers provided by the IUT. However, sometimes this activity becomes difficult and even impossible to perform. For example, this is the case if the tester is not provided with an interface to interact with the IUT or if the implementation is built from components that are running and cannot be shutdown or interrupted for a long period of time. In these situations, there is a particular interest in using other types of validation techniques such as *passive testing*. Passive testing consists in analyzing the traces recorded from the IUT and trying to find a *fault* by comparing these traces with either the complete specification or with some specific requirements (see, for example, [7], [8], [9], [10], [11]). A new methodology to perform passive testing was presented in [12], [13], [14]. The main novelty is that a set of *invariants* is used to represent the most relevant properties of a system. An invariant can be seen as a restriction over the traces allowed to the IUT.

Even though there are several proposal to perform active testing of timed systems (see, for example, [15], [16], [17], [18], [19], [20], [21], [22], [23], [24]), work on formal passive testing of timed systems did not attract much attention until recently. Taking as initial step [14], the approach presented in [25], [26] included the possibility of adding time constraints as properties that traces extracted from the IUT must hold. Even though this framework was very appropriate to analyze a large variety of timed systems, we observed that our invariants become large and complicated when there is a big number of nodes in the environment. In this paper we present a new temporal framework that takes as starting point our previous work. We consider a new type of invariants that are useful to analyze properties that appear in the type of systems that we are interested in. The new class of invariants, called *continuous timed invariants*, allows us to include a "message filter" behavior to abstract those packages that are not used during the testing phase. After performing different experiments on systems running over networks with a big number of nodes, we consider that the addition of continuous timed invariants to our former framework eases the task of defining relevant properties.

In addition to the theoretical framework, we have extended the tool PASTE (PASsive TEsting) to include the new class of invariants. In this paper we show how the off-line monitoring module of the tool is used to study the applicability of our formal techniques to the analysis of the OLSR MANET routing protocol (Optimized Link State Routing protocol). We will present a set of relevant invariants, denoting the requirements to be checked, and we will report the results of the experiments and discuss them.

It is worth to point out that our passive testing approach is somehow related to *runtime verification* [27] since they share similar objectives and procedures. Runtime verification techniques are used to dynamically verify the behavior observed in the target system with respect to some requirements. These requirements are often formally specified and verification techniques are used both to check that the observed behavior matches the specified properties and to explicitly recognize undesirable behaviors in the target system. The main difference between runtime verification and (formal) passive testing is given by the theoretical tools underlying the application of the techniques. In the case of runtime verification, casual relations among actions and temporal constraints on their performance are defined by using a certain temporal logic. Such languages

are, in general, more expressive than our notions of invariant. The main problem with such expressive languages is that it is far from easy for a (passive or active) tester, used to define relations between applied inputs and observed outputs, to write properties to be checked against the IUT as a temporal logic formula. On the contrary, the syntax of invariants is very similar to the usual sequences of inputs and outputs used in model-based testing. Also related to this issue, if we have an invariant expressing an interesting property of the system and we are granted with access to the system, so that we can become an active testing, it is straightforward to transform the invariant into test cases; this transformation is not so simple if we are dealing with temporal logic formulae.

The rest of the paper is structured as follows. In Section II we present our formal framework. In Section III, containing the bulk of the paper, we present the Optimized Link State Routing protocol (OLSR), we give a set of representative timed invariants, and we perform some experiments on real traces. We conclude with Section IV where we present the conclusions and some lines of future work.

II. Definition of Timed Invariants

First we introduce some auxiliary notation. In this paper we are only interested in the *messages* that are recorded from the environment. These messages are generated for different purposes (mainly control and exchange). We will focus on the different *tags* and *values* that can be found inside them. For example, in a wireless network, a package usually contains some neighbors address, link status and types, which raise several interesting constraints.

Definition 1: A *tag* is any string of characters. The set of all tags is denoted by TAG. A *message* is a tuple $m = (i, f, N, \mathcal{N}, \Im)$ where $i \in \texttt{TAG}$ denotes the kind of message. We assume that this string begins with either ? or !. $f \in \mathbf{N}$ denotes the source of the message, $N \subseteq \mathbf{N}$ is the set of neighbors of this message, $\mathcal{N} \subseteq \texttt{TAG}$ is the set of possible links that this message can hold, we force that no element of $\mathcal{N}$ starts with either ? or !. Finally, $\Im : \mathcal{N} \mapsto \wp(\mathbf{N})$ is a function that associates each link belonging to $\mathcal{N}$ to the set of neighbors that share this link.

The set of all messages is denoted by $\mathcal{M}$. We will define the function $\imath : \mathcal{M} \mapsto \texttt{TAG}$ such that for each $m = (i, f, N, \mathcal{N}, \Im)$ we have $\imath(m) = i$.

Given two messages $m_1 = (i_1, f_1, N_1, \mathcal{N}_1, \Im_1)$ and $m_2 = (i_2, f_2, N_2, \mathcal{N}_2, \Im_2)$, we say that m_1 matches m_2, denoted by $\mathbf{m}(m_1,m_2)$, iff $i_1 = i_2$, $f_1 = f_2$, $N_1 \subseteq N_2$ and for all $t \in \mathcal{N}_1$, there exists $t' \in \mathcal{N}_2$ such that $\Im_1(t) \subseteq \Im_2(t')$.

We will denote by θ the *wild-message*. This message has the following two properties: $\theta \notin \mathcal{M}$ and for all $m \in \mathcal{M}$ we have that $\mathbf{m}(\theta,m)$=true. □

We say that a message starting with ? is an *input message* while messages starting with ! are called *output messages*. Along this paper, we will "spread" any tuple-message to a char string in order to make it easier to read. For example, if we have the message $(\texttt{!hello}, \mathbf{n_0}, \varnothing, \{\texttt{AsymLink}\}, \Im)$, where $\Im(\texttt{AsymLink})=\{\mathbf{n_1}\}$ we will write the following char string: !Hello_FROM_$\mathbf{n_0}$_AsymLink_$\mathbf{n_1}$. Let us remark that a message has associated several spread strings. So, if there could be any confusion then we would write the complete tuple.

Passive testing makes use of the exchanged messages in order to detect faults. *Logs* extracted from IUTs do not only represent the value of the messages, but also include time information that denotes the amount of time elapsed between consecutive messages.

Definition 2: We say that a *log* of a system is a finite sequence belonging to $(\mathcal{M} \cdot \mathbf{R}_+)^\star \cdot \mathcal{M}$. The set of logs is denoted by $\mathcal{L}$. Let σ and σ' be two logs. We write $\sigma < \sigma'$ if σ is contained in σ'.

The function $\texttt{TT} : \mathcal{L} \mapsto \mathbf{R}_+$ is defined, for each $\sigma = \langle m_1, t_1, \ldots, t_{n-1}, m_n \rangle \in \mathcal{L}$, as $\texttt{TT}(\sigma) = \sum_{i=1}^{n-1} t_i$. □

Before introducing timed invariants, we give notation regarding the definition of time intervals. Since these intervals will represent time, we consider that the lower bound cannot be negative.

Definition 3: We say that $\hat{p} = [p_1, p_2]$ is a *time interval* if $p_1 \in \mathbf{R}_+$, $p_2 \in \mathbf{R}_+ \cup \{\infty\}$, and $p_1 \leq p_2$. We assume that for all $t \in \mathbf{R}_+$ we have $t < \infty$ and $t + \infty = \infty$. We consider that $\mathcal{IR}$ denotes the set of time intervals. Let us note that in the case of $[t, \infty)$ we are abusing the notation since this interval represents, in fact, the interval $[t, \infty]$.

We consider the following functions:

- $+ : \mathcal{IR} \times \mathcal{IR} \to \mathcal{IR}$. If $\hat{p} = [p_1, p_2]$ and $\hat{q} = [q_1, q_2]$ are time intervals then $\hat{p} + \hat{q} = [p_1 + q_1, p_2 + q_2]$.
- $\odot : \mathcal{IR} \times \mathbf{R}_+ \to \{true, false\}$. If $\hat{p} = [p_1, p_2]$ is a time interval and $t \in \mathbf{R}_+$ then $\hat{p} \odot t = (t \leq p_2)$.

We say that the sequence ψ is a *timed invariant* if ψ is defined according to the following EBNF:

$$\psi ::= m/\hat{p}, \psi \mid \star/\hat{p}, \psi' \mid m' \rhd_{\hat{p}} A, \hat{q}$$
$$\psi' ::= m'/\hat{p}, \psi \mid m' \rhd_{\hat{p}} A, \hat{q}$$

In this expression we consider $\hat{p}, \hat{q} \in \mathcal{IR}$, $m' \in \mathcal{M}$, $m \in \mathcal{M} \cup \{\theta\}$, and $A \subseteq \mathcal{M}$. □

Timed invariants are used to represent a set of functional and non-functional requirements which we will use to check the correctness of the logs. Intuitively, the previous EBNF expresses that an invariant is a sequence of symbols where each component, excluding the last one, is either:

- A pair $m/\hat{p}$, with m being a message in $\mathcal{M}$ or the wild-message θ, and $\hat{p}$ being a time interval,
- or an expression $\star/\hat{p}$.

There are two restrictions to this $\star$-rule. First, an invariant cannot contain two consecutive $\star$'s as $\star/\hat{p_1}$ and $\star/\hat{p_2}$ since this is the same as having $\star/(\hat{p_1} + \hat{p_2})$. The second restriction is that an invariant cannot present a pair $\star/\hat{p}$ followed by θ, that is, the message of the next component must be a *real* message belonging to $\mathcal{M}$. In fact, given an invariant such as $\psi = \ldots, \star/\hat{p_1}, m'/\hat{p_2}, \ldots$ we have $\star/\hat{p_1}$ represents a sequence of pairs message/time $\langle m_1/t_1, \ldots, m_n/t_n \rangle \in (\mathcal{M} \times \mathbf{R}_+)^n$, with $\sum_{j=1}^{n} t_j \in \hat{p_1}$, and for all $1 \leq i \leq n$ we have $\imath(m') \neq \imath(m_i)$.

Finally, the last component of an invariant corresponding to the expression $m' \rhd_{\hat{p}} A, \hat{q}$, that is, a message associated with a time interval followed by a set of messages and another time interval. This last interval is used to control the sum of time values associated to all the actions performed during matching the invariant. For a more precise and formal explanation of timed invariants, the interested reader is referred to [25].

Next, we will provide the notion of *correctness* of a log with respect to a time invariant. In order to define it, we have first to introduce the notion of *matching* a log with respect to an invariant.

Definition 4: Let $\sigma = \langle m_1, t_1, \ldots, m_r \rangle$ be a log and $\psi = \xi_1/\hat{p}_1, \ldots, \xi_n/\hat{p}_n, m_f \rhd_{\hat{q}_f} A, \hat{p}_f$ be a timed invariant. We say that σ *matches* ψ if after applying Algorithm 1 we have that $n = j$ (the loop has visited the first n positions of the invariant) and $x < r$ (the trace contains at least one element so that its *head* can be compared with the last part of the invariant: $m_f \rhd_{\hat{q}_f} A, \hat{p}_f$). □

For example, an invariant such as

$$m/\hat{p}, \star/\hat{p_\star}, m_f \rhd_{\hat{p'}} A, \hat{q}$$

indicates that **if** we match a message of the trace with the message m, and it is followed by a time value belonging to the interval $\hat{p}$, and **if** we have any sequence of pairs message/time not matching the message m_f and the sum of time values belongs to $\hat{p_\star}$, and **if** this sequence is followed by a message m'' matching m_f **then** we should have a lapse of time belonging to the interval $\hat{p'}$ **and** we will observe a message m_a that matches an existing message belonging to A. Finally, the performance of the complete sequence, from m to m_a, must belong to the interval $\hat{q}$.

Definition 5: Let $\psi = \xi_1/\hat{p}_1, \ldots, \xi_n/\hat{p}_n, m_f \rhd_{\hat{p}_f} A, \hat{q}_f$ be a timed invariant and σ be a log. We say that σ is *correct* with respect to ψ, iff either σ does not match ψ (see Definition 4) or σ matches ψ and $\mathbf{m}(m_f, m_x)$ implies

$$\exists m \in A: \quad \mathbf{m}(m, m_{x+1}) \wedge t_x \in \hat{p}_f \wedge \\ \texttt{TT}(m_{x'}, t_{x'}, \ldots, t_x, m_{x+1}) \in \hat{q}_f$$

where the variables x and x' contain the values returned by Algorithm 1. □

In this paper we present another kind of invariants called *continuous timed invariants*. This new family allows us to express properties in networking environments where the number of nodes is greater than two. The idea is to provide an easy representation that allows to *filter* the messages that we do not need to check, making them non-observable ones. As in the previous definition, first we present the grammar to build the new invariants and next we present the formal correctness of logs with respect to them.

Definition 6: We say that the sequence ψ is a *continuous timed invariant* if ψ is defined according to the following EBNF:

$$\psi ::= m/\hat{p}, \psi \mid \star/\hat{p}, \psi' \mid m' \rhd\rhd_{\hat{p}} A, \hat{q}$$
$$\psi' ::= m'/\hat{p}, \psi \mid m' \rhd\rhd_{\hat{p}} A, \hat{q},$$

In this expression we consider $\hat{p}, \hat{q} \in \mathcal{IR}$, $m' \in \mathcal{M}$, $m \in \mathcal{M} \cup \{\theta\}$, and $A \subseteq \mathcal{M}$. □

in : $\psi = \xi_1/\hat{p}_1, \ldots, \xi_n/\hat{p}_n, \alpha_f \texttt{op}_{\hat{q}_f} A, \hat{p}_f$
// either $\xi_k = m \in \mathcal{M}$ *or* $\xi_k = \theta$, *or* $\xi_k = \star$,
// and $\texttt{op} \in \{\rhd, \rhd\rhd\}$
$\sigma = \langle m_1, t_1, \ldots, t_{r-1}, m_r \rangle$
out: $x, x' \in \mathbf{N}$
$x = 1$;
// is the current position of the trace
$x' = 1$;
// is the base-position of the trace
$j = 1$;
// is the current position of the invariant to match
while $j \leq n \wedge (x' + n + 1) < r \wedge x < r$ **do**
  if $j == 1$ **then**
    $x' = x' + 1$
  end
  $(m_e, t_e) = \sigma[x]$*//x-th* element of σ;
  switch ξ_j **do**
    case m
      if $\mathbf{m}(m_j, m_e) \wedge t_e \in \hat{p}_j$ **then**
        $x = x + 1$;
        $j = j + 1$;
      else
        $j = 1$;
        $x = x'$;
      end
    end
    case θ
      if $t_e \in \hat{p}_j$ **then**
        $x = x + 1$;
        $j = j + 1$;
      else
        $j = 1$;
        $x = x'$;
      end
    end
    case $\star$
      $x'' = x$; $t'' = 0$;
      $m' = \xi_{j+1}$;
      while $\neg\mathbf{m}(m', m_e) \wedge \hat{p}_j \odot t'' \wedge x'' < r$ **do**
        $x'' = x'' + 1$;
        $t'' = t'' + t_e$;
        $(m_e, t_e) = \sigma[x'']$;
      end
      if $t'' \in \hat{p}_j$ **then**
        $x = x''$;
        $j = j + 1$;
      else
        $j = 1$;
        $x = x'$;
      end
    end
  end
end

Algorithm 1: Matching σ and ψ.

Next we describe the differences between the two classes of invariants. Let us consider $\psi_1 = m_f \rhd_{\hat{p}} A, \hat{q}$ and $\psi_2 = m_f \rhd\rhd_{\hat{p}} A, \hat{q}$. The meaning of ψ_1 is that **if** we see in the trace a message that matches m_f **then** it will be followed by an amount of time in $\hat{p}$ and with a message matching one of the elements in A. The meaning of ψ_2 is that **if** we observe the message m_f **then** *we will observe any sequence of pairs message-time* before having a message that matches a message in A. The amount of time that this allowed sequence can spend has to belong to $\hat{p}$. Regarding this new notion, let us comment that although sometimes it might seem that a timed invariant and a continuous timed invariant reflect the same property, there always exist some differences.

Example 1: Let us consider the following two timed invariants

$$\psi_1 = m' \rhd_{[0,0]} \{m_1, m_2\}, [0, 2]$$
$$\psi_2 = m' \rhd\rhd_{[0,0]} \{m_1, m_2\}, [0, 2]$$

and let $\sigma = \langle \alpha, 4, m', 0, m'_3, 0, m_2 \rangle$ be a log. We have that σ is correct with respect to ψ_2 but it is not with respect to ψ_1. Let us remark that continuous timed invariants always *accept* more logs than its corresponding timed invariant where we replace the symbol $\rhd\rhd$ by the symbol $\rhd$. □

Next, we will define the formal correctness criterion for continuous timed invariants. Similar to the previous formal correctness definition, we will make use of Definition 4.

Definition 7: Let $\psi = \xi_1/\hat{p}_1, \ldots, \xi_n/\hat{p}_n, m_f \rhd\rhd_{\hat{p}_f} A, \hat{q}_f$ be a continuous timed invariant and σ be a log. We say that σ is *correct* with respect to ψ, iff either σ does not match ψ (see Definition 4 since the notion of matching is exactly the same if we replace $\rhd$ by $\rhd\rhd$) or σ matches ψ and $\mathbf{m}(m_f, m_e,)$ implies

$$\begin{array}{c} \exists \sigma' = \langle m_e, t_e, \ldots, m_n, t_n, m_{n+1} \rangle : \sigma' < \sigma \wedge \\ \forall m' \in \{m_e, \ldots, m_n\}, \nexists m'' \in A : \\ (\imath(m') \neq \imath(m'') \wedge \mathbf{m}(m'', m_{n+1})) \wedge \\ \texttt{TT}(m_e, t_e, \ldots, t_n, m_{n+1}) \in \hat{p}_f \wedge \\ \texttt{TT}(m_{x'}, t_{x'}, \ldots, t_n, m_{n+1}) \in \hat{q}_f \end{array}$$

where $(m_e, t_e) = \sigma[x]$, having the variables x and x' the values returned by Algorithm 1. □

III. A Real Case Study: The OLSR Protocol

In this section we will apply our timed passive testing methodology to a non-trivial case study. First, we will present the OLSR protocol. We will focus on some stages of it and we will propose a set of timed invariants to check them. At the end of this section, we will present our tool PASTE, which implements the formal timed framework presented in the previous section and we will discuss the results of applying the set of proposed invariants to traces extracted from systems running this protocol.

A. The Protocol

The OLSR protocol, standardized by the IETF, is a link-state proactive protocol dedicated to the routing in the MANET [28]. OLSR manages to diffuse routing information through an efficient flooding technique. The key innovation of this protocol is the concept of Multi-Point Relays (MPRs). A node's multi-point relay is a subset of its neighbors whose combined radio range covers all nodes two hops away. In order for a node to determine its minimum multipoint relay set based on its two-hop topology, periodic broadcasts are required. Similar to conventional link-state protocols, the link information updates are propagated throughout the network. However in OLSR, when a node has to forward a link update it only forwards it to its MPR set of nodes. Finally, the distribution of topological information is realized with the use of periodic topology control messages and as a result, each node has a partial graph of the topology of the network that is used to calculate the optimal routes. OLSR is mostly preferred when the ad hoc network consists of a large number of nodes and has a high density. One of the main advantages of the OLSR protocol is that it does not make any assumption concerning the underlying link layer, allowing it to be used in a variety of configurations.

Three main functionalities are studied in this paper: The link sensing, the neighbor detection and the dissemination of link state information. The two first are obtained by broadcasting and analyzing Hello() messages in order to discover 2-hop neighbor information and to perform a distributed selection of MPR sets. The third one is taken in charge by the Topology Control (TC) messages sent in particular by the MPR nodes.

B. The Real Wireless Testbed

Our testbed is composed of four laptops as illustrated in Figure 1. It must be noted that we could increase the number of nodes by using an emulation technique embedded in one of the laptops (see [29] for more details). This technique is nevertheless useless regarding our current test purposes since four nodes is the 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 run an OLSR implementation version olsrd-0.5.6-rc1. The protocol is configured to run in the wireless interfaces of the nodes and with the link quality measurements option deactivated in order to comply with the RFC.

The trace is captured using the Wireshark Network Analyzer http://www.wireshark.org/ installed in Node 0 and launched at the beginning of all packet transmissions to ensure that the start state is captured in the trace.

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 Wireshark process sniffing the OLSR protocol was run during 9 minutes and 29 secondes allowing to provide exactly 1350 actions in that trace. The trace has been transformed in plain text format by that same tool allowing the use of PASTE (described below).

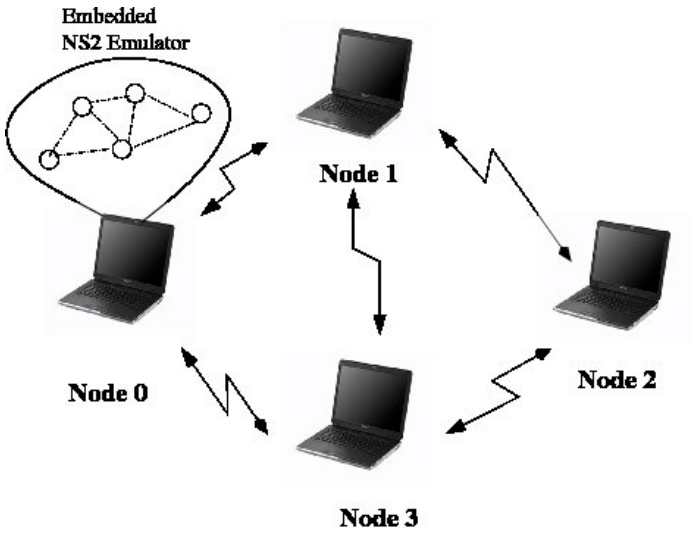

Figure 1. The wireless testbed.

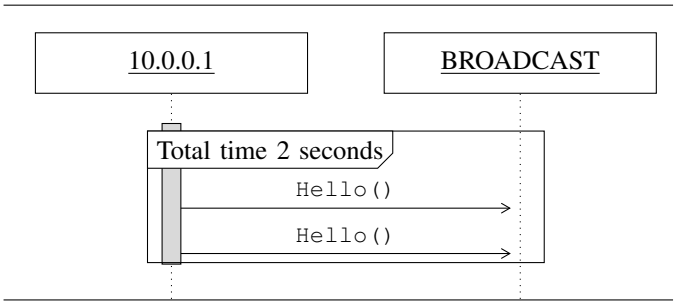

Figure 2. OLSR diagram for φ_1.

C. The Invariants

The invariants presented in this section describe the process of connectivity by ensuring that a node $\mathbf{n_0}$ will respect the procedure followed by the protocol to establish the links with its neighbors. This procedure also ensures that a node $\mathbf{n_0}$ cannot be fooled by another node $\mathbf{n_1}$ by announcing a nonexistent connection.

Let us remark that it is important 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 addresses embedded within each message is limited. Moreover, the broadcast of control packets are temporized by constants. `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. Thus, even though the mobility of nodes is normally very high, the number of messages exchanged during the life of a link is finite.

Next we present a set of relevant invariants for the `OLSR` protocol. Besides, in order to match the sniffed trace, we have that the variables $\mathbf{n_0}$ and $\mathbf{n_1}$ correspond to the nodes whose `IP` address are respectively 10.0.0.1 and 10.0.0.10. Let us comment than in the formal definition of messages, the source of the tuple is a natural number instead of `IP` numbers, but there do not exists any problem to transform `IPs` into naturals. We briefly comment the main behavior that each invariant represents.

$$\varphi_1 = !\texttt{Hello\_FROM\_}\mathbf{n_0} \rhd_{[2,2]} !\texttt{Hello\_FROM\_}\mathbf{n_0}, [2,2]$$

The invariant φ_1 represents that always **if** an isolated `OLSR` node has to emit the `Hello()` message, **then** it has to repeat the same message with a frequency of two seconds. This idea is graphically expressed in Figure 2.

$$\begin{aligned}\varphi_2 = {} & !\texttt{Hello\_FROM\_}\mathbf{n_0}\texttt{\_SymLink\_}\mathbf{n_1}/[0,2], \star/[0,2], \\ & ?\texttt{Hello\_FROM\_}\mathbf{n_1}\texttt{\_SymLink\_}\mathbf{n_0} \rhd\rhd_{[0,4]} \\ & \{!\texttt{Hello\_FROM\_}\mathbf{n_0}\texttt{\_MPRLink\_}\mathbf{n_1}, \\ & ?\texttt{Hello\_FROM\_}\mathbf{n_1}\texttt{\_MPRLink\_}\mathbf{n_0}\}, [0,\infty]\end{aligned}$$

The invariant φ_2 expresses that **if** a node $\mathbf{n_0}$ has associated a symmetric link with the node $\mathbf{n_1}$ and **if** we observe that $\mathbf{n_1}$ has declared a symmetric link with the node $\mathbf{n_0}$, **then** it is expected to observe **either** an MPR link connection from $\mathbf{n_1}$ adding in its neighbor list $\mathbf{n_0}$ **or** viceversa. In Figure 3, up, we have a graphical representation of both diagrams.

$$\begin{aligned}\varphi_3 = {} & ?\texttt{Hello\_FROM\_}\mathbf{n_1}/[0,2], \star/[0,2], \\ & \texttt{Hello\_FROM\_}\mathbf{n_0}\texttt{\_AsymLink\_}\mathbf{n_1} \rhd\rhd_{[0,2]} \\ & \{?\texttt{Hello\_FROM\_}\mathbf{n_1}\texttt{\_SymLink\_}\mathbf{n_0}\}, [0,4]\end{aligned}$$

The invariant φ_3 represents that **if** we observe in the trace a `Hello()` message from $\mathbf{n_1}$ and **if** $\mathbf{n_0}$ sends a `Hello()` mentioning an asymmetric link with $\mathbf{n_1}$, **then** $\mathbf{n_0}$ will be declared into the neighbor list of the symmetric link in $\mathbf{n_1}$ and the elapsed time associated with all these messages, that

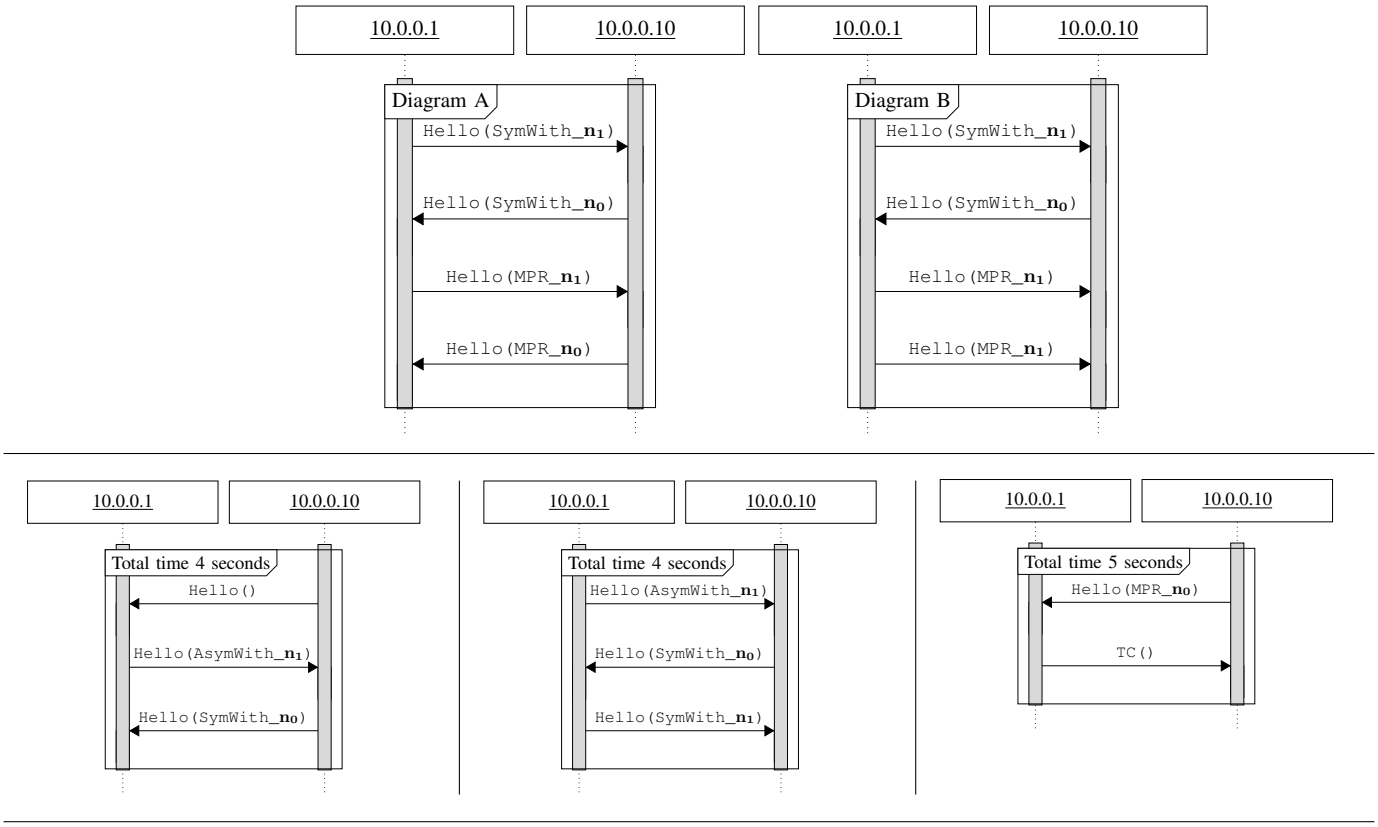

Figure 3. OLSR diagram for $\varphi_2, \varphi_3, \varphi_4$ and φ_5.

is, from the first Hello() from $\mathbf{n_1}$ until receiving the confirmation Hello() from $\mathbf{n_1}$, must belong to [0,4]. In Figure 3, below left corner, we have the graphical representation of this invariant.

$$\varphi_4 = !\texttt{Hello\_FROM\_}\mathbf{n_0}\texttt{\_AsymLink\_}\mathbf{n_1}/[0,2], \star/[0,2], ?\texttt{Hello\_FROM\_}\mathbf{n_1}\texttt{\_SymLink\_}\mathbf{n_0} \rhd\rhd_{[0,2]} \{!\texttt{Hello\_FROM\_}\mathbf{n_0}\texttt{\_SymLink\_}\mathbf{n_1}\}, [0,4]$$

The invariant φ_4 concerns symmetric and asymmetric links. This invariant represents that **if** the node $\mathbf{n_0}$ has associated an asymmetric link with $\mathbf{n_1}$ and **if** there exists a petition from $\mathbf{n_1}$ to make this link as symmetric, **then** $\mathbf{n_0}$ will answer this request by adding $\mathbf{n_1}$ in its list of neighbors of symmetric link. Figure 3, below center, shows the graphical representation of φ_4.

$$\varphi_5 = ?\texttt{Hello\_FROM\_}\mathbf{n_1}\texttt{\_MPRLink\_}\mathbf{n_0} \rhd\rhd_{[0,5]} \{!\texttt{TC\_FROM\_}\mathbf{n_0}\}, [0,5]$$

Concerning the typology control messages, the invariant φ_5 reflects that **if** a node $\mathbf{n_1}$ has into its neighbor MPR link list a node $\mathbf{n_0}$ and $\mathbf{n_1}$ broadcasts this information inside of a Hello() message, **then** we must observe a TC message from $\mathbf{n_0}$ in a time less than or equal to 5 seconds. In Figure 3, below right, we give the graphical representation of φ_5.

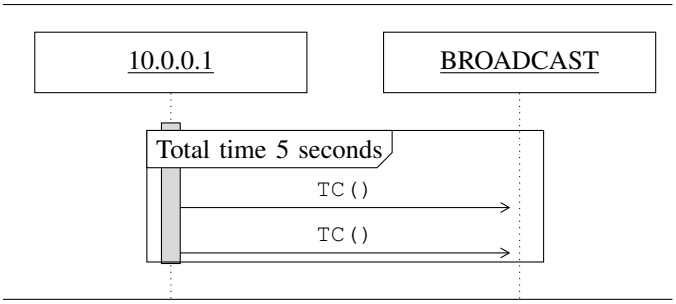

Figure 4. OLSR diagram for φ_6.

$$\varphi_6 = !\texttt{TC\_FROM\_}\mathbf{n_0} \rhd\rhd_{[0,5]} \{!\texttt{TC\_FROM\_}\mathbf{n_0}\}, [0,5]$$

This last invariant represents that always **if** a node $\mathbf{n_0}$ (according with φ_5 is in the neighbor MPR list of another node) starts sending a TC message, **then** it will send this information with an associated frequency less than or equal to 5 seconds. Figure 4 shows the graphical representation of φ_6.

D. Experimental Results using PASTE

We have extended our PASTE tool (some experiments with this tool are reported in [30]) with the new kind of invariants. The original core was implemented in JAVA

and it was an isolated project. Later, it was decided to include our academic tool as a module of a more important monitoring tool, developed by the SME Peopleware, called OSMIUS (see http://www.osmius.net). This is a monitoring tool released under the GPLv2 license and available in the open source repository Sourceforge. Osmius is designed to be extensible in such a way that it can be used to monitor any device or software connected to a network. Therefore, the new version includes the complete migration of the code from JAVA to C++ (more information in http://kimba.mat.ucm.es/paste/).

Next we report the results, using PASTE, over the recorded logs containing 1350 tokens. The results are presented in Figure 5. In these tables, the first column lists the set of invariants. The other columns represent different freedom degrees over the timed constrains (for example, at φ_1, which has associated the time interval [2,2], the column ± 0.4 represents the results obtained by changing the time constrains to $[1.6, 2.4]$). Each cell contains two values. The first one corresponds to the number of time values belonging to the considered intervals while the second value corresponds to the number of values that do not belong to the interval and therefore represent a (performance) fault.

As we had already mentioned, the invariant φ_1 is *only* useful when there are two nodes interacting between themselves. In the studied trace there are four nodes. So, we have to adapt φ_1 with:

$$\varphi_{1a} = !\texttt{Hello\_FROM\_}\mathbf{n_0} \rhd\!\rhd_{[2,2]} !\texttt{Hello\_FROM\_}\mathbf{n_0}, [2,2]$$

Concerning the results for φ_1 and φ_{1a}, we did not observe a message that arrived "just in time", that is, in 2 seconds. When we allow some freedom degree, the number of values belonging to the interval increases. Let us also note that the invariant φ_1 returns 262 *faults*. However, these values are false positive because there are more than two nodes in the network; when we check its modification, φ_{1a}, we observe that there are not so many faults. Let us remark that the number of faults decreases as the time interval increases. This makes us suspect that we did not find functional errors, as expected, but some deviations in performance. In fact, if we check the time values when we increase the time interval in ten seconds for φ_{1a}, then all the values belong to the interval and we can claim that we did not find functional errors. In Figure 5 (top), we also report the results for φ_6. We group them together because they are studied by using the same scale of freedom degree.

In Figure 5, middle, we give the results for φ_2. The scale of the freedom degree has changed. In fact, they are several times the ones applied to φ_1, φ_{1a}, and φ_6. The reason for this is that this invariant should only denote functional properties. Therefore, we should consider that all intervals are set to $[0, \infty]$ (last column).

In Figure 5, bottom, we present the results for the invariants φ_3, φ_4, and φ_5. These are the invariants that are checked less times. Almost all of them are checked correctly with their initial time constrains. We only needed to add a small freedom degree to φ_5 in order to obtain the full set of correct cases.

	± 0	± 0.2	± 0.4	± 1	± 10
φ_1	$\mathbf{0}/266$	$\mathbf{4}/262$	$\mathbf{4}/262$	$\mathbf{4}/262$	$\mathbf{4}/262$
φ_{1a}	$\mathbf{0}/266$	$\mathbf{204}/62$	$\mathbf{217}/49$	$\mathbf{245}/25$	$\mathbf{266}/0$
φ_6	$\mathbf{38}/68$	$\mathbf{56}/50$	$\mathbf{71}/35$	$\mathbf{90}/16$	$\mathbf{106}/0$

	± 0	± 3	± 10	± 40	$\pm \infty$
φ_2	$o\mathbf{4}/160$	$\mathbf{7}/157$	$\mathbf{11}/153$	$\mathbf{38}/126$	$\mathbf{164}/0$

	± 0	± 2
φ_3	$\mathbf{4}/0$	$\mathbf{4}/0$
φ_4	$\mathbf{4}/0$	$\mathbf{4}/0$
φ_5	$\mathbf{6}/1$	$\mathbf{7}/0$

Figure 5. Results after the application of invariants.

Finally, we discuss the most relevant features of this analysis. First, let us remark that the full set of proposed invariants are matched at least one time over a normal trace. The invariants that are checked more times are φ_1 and φ_{1a}. Another conclusion is that when we are working with environments containing more than two nodes, continuous timed invariants are always better than timed ones to detect faults. The set of invariants could be also classified with respect to the number of times checked on the traces. The subset composed by invariants φ_4, φ_5 and φ_6 can be considered as *security* invariants. The reason is that in almost 100% checked times, these invariants hold. The rest of invariants can be seen as *performance* invariants. They check the real behavior (e.g. delays, white noise of the net, etc) that the communication is having. We can use them to detect communication bottlenecks.

IV. Conclusions and Future work

In this paper we have presented a case study of formal passive testing in a network protocol. We have reviewed our previous methodology and introduced a new notion of timed invariant. We have also presented a complete study of a MANET routing protocol: The Optimized Link State Routing protocol. We have defined a set of representative invariants. We have performed passive testing against real logs of this protocol and we have reported and commented the results. We hope that this study shows that any tester should be able to define and check interesting properties about the typology of the network, its global performance, security properties, etc.

As future work we plan to consider the study of other systems and to extend the types of invariants by defining appropriate timed extensions of obligation invariants [14].

Acknowledgements

This research was partially supported by the FP7 SHIELDS project, the WEST project (TIN2006-15578-C02-01), and the UCM-BSCH programme to fund research groups (GR58/08 - group number 910606). The work was carried out while

the first, fourth and fifth authors were visiting TELECOM & Management SudParis.

REFERENCES

[1] G. Myers, *The Art of Software Testing*, 2nd ed. John Wiley and Sons, 2004.

[2] P. Ammann and J. Offutt, *Introduction to Software Testing*. Cambridge University Press, 2008.

[3] M. Utting and B. Legeard, *Practical Model-Based Testing: A Tools Approach*. Morgan-Kaufmann, 2007.

[4] R. Hierons, J. Bowen, and M. Harman, Eds., *Formal Methods and Testing, LNCS 4949*. Springer, 2008.

[5] R. Hierons, K. Bogdanov, J. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Luettgen, A. Simons, S. Vilkomir, M. Woodward, and H. Zedan, "Using formal methods to support testing," *ACM Computing Surveys*, vol. 41, no. 2, 2009.

[6] I. Rodríguez, "A general testability theory," in *20th Int. Conf. on Concurrency Theory, CONCUR'09, LNCS 5710*. Springer, 2009, pp. 572–586.

[7] D. Lee, A. Netravali, K. Sabnani, B. Sugla, and A. John, "Passive testing and applications to network management," in *5th IEEE Int. Conf. on Network Protocols, ICNP'97*. IEEE Computer Society Press, 1997, pp. 113–122.

[8] M. Tabourier and A. Cavalli, "Passive testing and application to the GSM-MAP protocol," *Information and Software Technology*, vol. 41, no. 11-12, pp. 813–821, 1999.

[9] R. E. Miller, D. Chen, D. Lee, and R. Hao, "Coping with nondeterminism in network protocol testing," in *17th Int. Conf. on Testing of Communicating Systems, TestCom'05, LNCS 3502*. Springer, 2005, pp. 129–145.

[10] A. Benharref, R. Dssouli, M. Serhani, A. En-Nouaary, and R. Glitho, "New approach for EFSM-based passive testing of web services," in *Joint 19th IFIP TC6/WG6.1 Int. Conf. on Testing of Software and Communicating Systems, TestCom'07, and 7th Int. Workshop on Formal Approaches to Software Testing, FATES'07, LNCS 4581*. Springer, 2007, pp. 13–27.

[11] H. Ural and Z. Xu, "An EFSM-based passive fault detection approach," in *Joint 19th IFIP TC6/WG6.1 Int. Conf. on Testing of Software and Communicating Systems, TestCom'07, and 7th Int. Workshop on Formal Approaches to Software Testing, FATES'07, LNCS 4581*. Springer, 2007, pp. 335–350.

[12] J. Arnedo, A. Cavalli, and M. Núñez, "Fast testing of critical properties through passive testing," in *15th Int. Conf. on Testing Communicating Systems, TestCom'03, LNCS 2644*. Springer, 2003, pp. 295–310.

[13] A. Cavalli, C. Gervy, and S. Prokopenko, "New approaches for passive testing using an extended finite state machine specification," *Information and Software Technology*, vol. 45, no. 12, pp. 837–852, 2003.

[14] E. Bayse, A. Cavalli, M. Núñez, and F. Zaïdi, "A passive testing approach based on invariants: Application to the WAP," *Computer Networks*, vol. 48, no. 2, pp. 247–266, 2005.

[15] D. Mandrioli, S. Morasca, and A. Morzenti, "Generating test cases for real time systems from logic specifications," *ACM Transactions on Computer Systems*, vol. 13, no. 4, pp. 356–398, 1995.

[16] D. Clarke and I. Lee, "Automatic generation of tests for timing constraints from requirements," in *3rd Workshop on Object-Oriented Real-Time Dependable Systems, WORDS'97*. IEEE Computer Society Press, 1997, pp. 199–206.

[17] T. Higashino, A. Nakata, K. Taniguchi, and A. Cavalli, "Generating test cases for a timed I/O automaton model," in *12th Int. Workshop on Testing of Communicating Systems, IWTCS'99*. Kluwer Academic Publishers, 1999, pp. 197–214.

[18] J. Springintveld, F. Vaandrager, and P. D'Argenio, "Testing timed automata," *Theoretical Computer Science*, vol. 254, no. 1-2, pp. 225–257, 2001, previously appeared as Technical Report CTIT-97-17, University of Twente, 1997.

[19] A. En-Nouaary, R. Dssouli, and F. Khendek, "Timed Wp-method: Testing real time systems," *IEEE Transactions on Software Engineering*, vol. 28, no. 11, pp. 1024–1039, 2002.

[20] M. Merayo, M. Núñez, and I. Rodríguez, "Formal testing from timed finite state machines," *Computer Networks*, vol. 52, no. 2, pp. 432–460, 2008.

[21] ——, "Extending EFSMs to specify and test timed systems with action durations and timeouts," *IEEE Transactions on Computers*, vol. 57, no. 6, pp. 835–848, 2008.

[22] M. Uyar, S. Batth, Y. Wang, and M. Fecko, "Algorithms for modeling a class of single timing faults in communication protocols," *IEEE Transactions on Computers*, vol. 57, no. 2, pp. 274–288, 2008.

[23] Y. Wang, M. Uyar, S. Batth, and M. Fecko, "Fault masking by multiple timing faults in timed EFSM models," *Computer Networks*, vol. 53, no. 5, pp. 596–612, 2009.

[24] R. Hierons, M. Merayo, and M. Núñez, "Testing from a stochastic timed system with a fault model," *Journal of Logic and Algebraic Programming*, vol. 78, no. 2, pp. 98–115, 2009.

[25] C. Andrés, M. Merayo, and M. Núñez, "Passive testing of timed systems," in *6th Int. Symposium on Automated Technology for Verification and Analysis, ATVA'08, LNCS 5311*. Springer, 2008, pp. 418–427.

[26] ——, "Formal correctness of a passive testing approach for timed systems," in *5th Workshop on Advances in Model Based Testing, A-MOST'09*. IEEE Computer Society Press, 2009, pp. 67–76.

[27] M. Leucker and C. Schallhart, "A brief account of runtime verification," *Journal of Logic and Algebraic Programming*, vol. 78, no. 5, pp. 293–303, 2009.

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

[29] S. Maag, C. Grepet, and A. Cavalli, "A formal validation methodology for MANET routing protocols based on nodes' self similarity," *Computer Communications*, vol. 31, no. 4, pp. 827–841, 2008.

[30] C. Andrés, M. Merayo, and M. Núñez, "Passive testing of stochastic timed systems," in *2nd Int. Conf. on Software Testing, Verification, and Validation, ICST'09*. IEEE Computer Society Press, 2009, pp. 71–80.