A Formal Validation Methodology for MANET Routing Protocols based on Nodes' Self Similarity

Stephane Maag [a,*] Cyril Grepet [b] Ana Cavalli [a]

[a] *National Institute of Telecommunications*
CNRS SAMOVAR
9, rue Charles Fourier
F-91011 EVRY Cedex

[b] *CEA/Saclay,*
SOL, Bat.451,
91191 Gif-sur-Yvette Cedex

Abstract

Conformance testing for ad hoc routing protocols is crucial to the reliability of wireless mobile ad hoc networks. Most of the works related to the validation of such protocols are based on simulation and emulation analysis of simulated/emulated implementations without taking into account formal specifications. However, for many reasons, it has often been shown that the results obtained from simulator studies are far from the real case studies. In this paper we propose a new conformance testing methodology dedicated to the validation of MANET routing protocols, especially Dynamic Source Routing. After generating test sequences from the formal model, the main issue is to execute them. There exist many techniques in wired systems, but several MANET inherent constraints lead to cope with the diverse mobility patterns and the topological changes. Therefore, a nodes' self similarity approach is introduced as well a specific testing architecture dealing with unexpected and unpredictable topology and messages. Interesting results have been provided as illustrated through experiments.

Key words: MANET, Conformance testing, Formal model, Self similarity, DSR

* Fax:+33 1 60 76 47 11, Tel: +33 1 60 76 44 70
Email addresses: `Stephane.Maag@int-edu.eu` (Stephane Maag), `Cyril.Grepet@cea.fr` (Cyril Grepet), `Ana.Cavalli@int-edu.eu` (Ana Cavalli).

1 Introduction

In wireless mobile ad hoc networks (MANETs) there are no predefined infrastructures, no administrative node and each node participates in the provision of reliable operations in the network. The nodes may move continuously leading to a volatile network topology with interconnections between nodes that are often modified. As a consequence of this infrastructureless environment, each node communicates using their radio range with open transmission medium and some of them behave as routers to establish multi-hop connections. Due to these aspects and the limited resources of the mobile nodes, efficient routing in ad hoc networks is an important and challenging problem for the quality of the communication systems.

Conformance testing for ad hoc routing protocols is crucial to the reliability of those networks. But paradoxically, few works currently exist on the formal specifications to analyze and test routing protocols. The major techniques used by the ad hoc network experts to design and ensure the quality of their routing protocols essentially rely on descriptions for simulations and/or emulations. The majority of these works rely on non-formal models provided as input to simulators such as NS-2 [1], OpNet [2], SSF [3] or Glomosim [4]. [5] tries to illustrate some comparisons between NS-2, OpNet and Glomosim. But it is not really possible to tell that one is better than another. This study shows that by implementing the same flooding algorithm, same topology and the same traffic model in the simulators, we obtain different results denoting that some of the protocols seem better than others in some conditions and vice-versa. Nevertheless, even if some functional or non-functional aspects of the protocols may be observed, it has been proved that these observations may be erroneous. Indeed, results providing by the simulation testing is sometimes far from the one obtained in a real case study. Halfway between real life testing and simulation testing, we find the emulation testing [6]. This kind of architecture enables for instance to use one or several real protocol stacks linked to a simulator in order to generate a real traffic and provide an environment to the "virtual network" and then observe the outputs provided by the real nodes. However, even if this architecture and the results come closer to the reality, the simulation side is still important. By the way, it may be noted that two identical simulation scenarios used in two different simulators will not give the same results.

Finally, as it is often noticed, the simulation and emulation techniques do not replace a real case study [7]. Normal or ideal behaviors obtained by simulation may be proved erroneous in the real case. This is why formal description techniques are required to test this kind of protocols.
Formal description techniques and their testing tools are rarely applied in such kind of networks. The main reasons are the difficulty to take into account the

inherent MANET protocol characteristics and the mobility of nodes in the test sequences generation and their execution. Therefore, our work focuses on a new testing technique based on new algorithms to check the conformance of these ad hoc routing protocols. We present in this paper a formal specification of the Dynamic Source Routing (DSR) protocol from which we may generate some test sequences. Nevertheless, the execution of these scenarios is currently an issue. Indeed there is often a gap between the dynamic topology designed in a specification and the one of a real case study. Therefore, we illustrate in this paper the concept of node self-similarity in order to generate test sequences and execute them on a real wireless ad hoc routing protocol taking into account the network topologies. A methodology and an algorithm to maintain the testing process through the testers will also be introduced.

The paper is organized as it follows. In the Section 2, the related works on testing MANET routing protocols using formal methods are mentioned. Then, in Section 3, the conformance testing process is depicted taking into account the specific constraints of these kind of networks. In Section 4, our DSR specification is illustrated. Therefore, in Section 5, the notion of nodes' self similarity will be given and definitions allowing to combine nodes will be enunciated. The possibility to reduce the specification from which test sequences will be generated and executed on a real implementation is presented in the Section 6. The Section 7 illustrates an emulation example from which the necessity to maintain the testers when receiving unexpected messages is raised. Finally we will conclude.

2 Related Works

As above mentioned, very few works have been performed on formal specification to test MANET routing protocols. We may however cite the following results.

Most of the time, the interest of simulation is to observe and analyze the protocol performances and to verify expected properties (e.g. infinite loops free). If the simulation code is erroneous, many troubles may occur such as a decreased performance, wrong parameterization and finally results that are false. Verisim [7] is a model combining NS-2 and the trace verification component provided by the *Monitoring and Checking* system namely MAC [8]. The goal is to generate a NS-2 trace T and to verify if the expected properties are included in the implementation I according to a scenario S. In [7], the authors have shown that the AODV implementation in NS-2 was false regarding some properties as for instance the initial value for the hops number in a RREP. This first work was very interesting, disturbing and raised several issues regarding the efficiency/reliability of the simulation/emulation.

While Verisim performs a validation through invariants in a simulator, [9] proposes 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 allowing to reach 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. However, despite this formal description, RNS is victim of its own objectives keeping it from being usable to the conformance testing. Indeed, this framework has as main goal to analyze the protocols that are specified by metrics through non-functional aspects which are usually applied for a performance analysis (e.g. overhead). It does not allow to study functional properties neither the nodes' interactions. In addition, only a RNS specialist seems able to correctly specify the protocol behavior regarding its standard.

From a need of specification, it came another work. In order to study the interest and the performances of their new routing protocols (LTLS, Logical Topology based Location Service), the authors [10] developed a formal model namely Distribued Abstract State Machines (DASM). While this model allows to verify the behavior of a node in a functional way, unfortunately the model is non-executable and does not allow to observe the nodes' interoperability.

[11] is another approach to formalize the routing in a MANET by applying game theory concepts. The game theory is based on the "income" calculus. In our case, the income means for instance the convergence when the topology is modified or the induced overhead. Despite an innovative approach, many issues are still present. First, the game theory consists on the complete knowledge by the "players" of the "game". Thus, in the MANET context, each node is supposed to have a complete knowledge of the network topology as well the nodes or links states. Furthermore, it forbids the non-determinism perventing therefore to model the nodes' dynamicity or other MANET inherent constraints. These two aspects decrease the interest of this method.

In our work, we propose a new testing approach relying on well known formal methods, especially a nodes self similarity concept by considering the eventual topology modifications. Our main goal is to facilitate the execution of generated test sequences and to reduce the inconclusive verdicts when checking the conformity of an implementation in relation with its specification.

3 Testing the Conformance of an Ad Hoc Routing Protocol

Testing techniques can be divided in two categories : active testing which relies on stimulation and observation of an implementation, and passive testing which only observes the system without interactions [12], [13], [14]. Our research focuses on active testing of ad hoc routing protocols.

3.1 Conformance testing

Conformance testing usually relies on the comparison between the behavior of an implementation and the formal specification of a given protocol i.e. a conformed implementation has to behave as its specification.
The conformance testing procedure follows these steps :
- Step 1. Define a *testing architecture* with respect to the characteristics of the system under test and its possible implementations. This step could impact on each following step and has to be defined according to the context.
- Step 2. Make some *assumptions* that are sometimes required to enable the test.
- Step 3. Construct a precise *formal specification* of the system to be tested. This specification takes into account the system functionalities as well as the data specific to the test environment (test architecture, test interface, etc.).
- Step 4. *Select* the appropriate tests. This step corresponds to the definition of the test purposes. A test purpose can be a specific property of the system such as tasks or assignments with regard to values of variables, or the behavior of a specific component of the system taking into account the current values of the variables.
- Step 5. *Generate* the test sequences. The test purposes are used as a guide by an algorithm based on simulation to produce the test sequences. As a result, our algorithm computes a test sequence that can be applied to the implementation under test to verify the test purpose. A sequence is a suite of interactions (between the system and the environment) that includes the interactions that represent a test purpose.
- Step 6. *Format* the test sequences i.e. to produce test sequences in some accepted formalism as *Message Sequence Charts* (MSC), a formalism widely used in industry to describe message exchanges, or in *Testing and Test Control Notation* (TTCN), the ITU-TS standard language used for test specification.

Problematic The main goal tackled in this paper is to provide a reliable method to test a routing protocol in a network in which we do not control neither the number of nodes nor the mobility scenario. Therefore, three main and relevant problems may be defined.

- *Mobility representation* : The mobility in ad hoc networks implies that a specification has to represent more than one node communicating with each other. Thus the specification has to allow the creation or the suppression of a link between a pair of nodes in order to represent their mobility.
- *Test sequences generation* : Another objective is to try to maximize the automation of the test sequences generation from classical tools avoiding the well-known state space explosion problem.
- *Test execution* : The mobility of nodes and the hazard of radio communications can lead to many inconclusive verdicts or even to prevent the test. Thus we may not be able to give a verdict about the conformity of the implementation. Moreover, the testing architecture to be used is also subject to the same problems. This problematic has to be study in order to provide a reliable verdict to the implementation conformity.

In the remaining of the paper we present our solutions to these problems with respect to the six steps aforementioned.

3.2 Testing Architectures

In order to verify the conformance of an implementation, active testing requires a set of test sequences. A test sequence is a succession of inputs provoking outputs obtaining from the formal model. They are used to check the expected or critical behaviors of the requirements. Once we obtain these sequences, they are injected into the implementation under test (IUT) and the outputs are collected and compared with the ones expected by the model to finally provide a verdict.
In order to interact with the IUT, a testing architecture is needed and is composed by the following elements. First, in order to observe packets, we need some Points of Observation (PO), whereas to observe and control these packets (if the IUT allows it (white/grey/black box testing)) we need Points of Control and Observation (PCO). These points are connected with the upper and lower testers (UT and LT) which are controlled by the test coordination procedure (TCP). The PO/PCO connected to UT aim to control the packets between the IUT and the upper layer (IP for example) whereas those connected to LT aim to control communications between the IUT and the lower layer (the link layer for instance). Each time one of the testers observes a packet, the TCP checks if it is the one expected regarding the specification.

The International Organization for Standardization proposes in one of its standard [15] different conformance testing architectures. The main differences between the presented architectures rely on the position, communication and synchronization of the components: lower and upper testers as well the test coordination procedure (TCP).

In a wired environment, the choice of the testing architecture is not a difficulty especially because of the fixed node, the reliable communications, as well the realist assumption that all received packets will be processed.
In the MANETs, these problems have another dimension. The communication are not so reliable, subject to distortion, multiple retransmissions, routes modifications, delays, collisions, etc. These aspects raise several issues about the dynamicity/reactivity to the observed events of the testing architecture depending on the radio interfaces.
Thus, in a distributed architecture, the lower tester is distant of the IUT contrary to the upper tester, and the TCP synchronizes the two testers actions. They use the radio interfaces that may provoke packet loss and disruption providing finally erroneous verdicts. Indeed, when a test sequence is applied, at each time the TCP indicates to the testers an action to be processed and receives the collected information or observations.
This last aspect is another issue. Indeed, all remote testers are subject to delays or to the radio medium occupation when some observations are required. The results could therefore be wrong or inconclusive.
Based on these observations, we may suppose that a testing architecture with components that are outside the IUT site will give incorrect verdicts for conformance testing.

Therefore, the architecture allowing to be as close as possible to the implementation under test by assuring a reliable synchronization between the testers is the local architecture (see Fig.1).

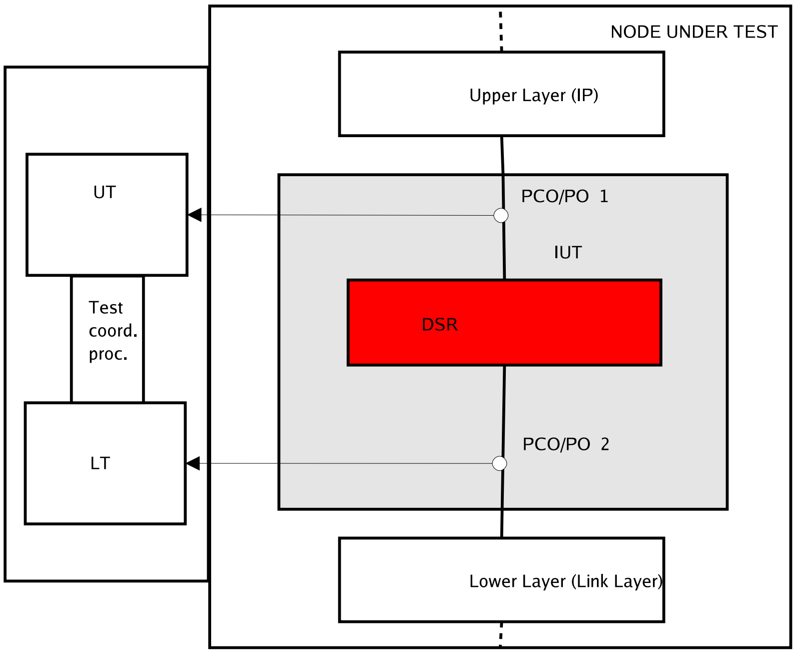

Figure 1. Local test architecture applied on IUT.

Following the way how the implementation is developed and integrated into the protocol stack, two alternatives may stand. First, the IUT can be a black box (that means the code is unavailable, we may not access the internal behavior of the implementation) with an access to its lower and upper interfaces, in which case PCO will be placed on each interfaces as illustrated by the Figure 1.

Another possibility is that the IUT is not directly reachable through its own interfaces. Then embedded testing techniques will be used [16]. The resulting architecture for the DSR protocol will be based on a PCO to layers 1 or 2 for the LT and on an upper application allowing to transmit and collect some packets.

3.3 Testing Assumptions

In order to execute test sequences in a real network and to provide reliable verdicts, some testing assumptions are required. First, we make the classical assumption that the implementation could be tested, i.e. we can install some Points of Control (PCO) or Points of Observation (PO) (see section 3.2) into the IUT. The specific constraints of ad hoc networks imply that we have to make some assumptions to take into account the mobility of the nodes. Six hypothesis are defined :

(1) *Existing destination nodes* : Each destination node (D) of the packets used by our test sequences exists and is or will be connected by the network to the IUT.
(2) *Connectivity of node D* : We assume that in a reasonable time, one or more paths will enable to execute the test sequences between IUT and D.
(3) *Stability of routes* : The routes which allow IUT and D to communicate with each other will remain stable during the execution of the test sequences. This assumption is necessary to realize conformance testing and relies on the fact that an ad hoc network is created in order to allow communication for a community. If the communications are reliable enough for this purpose, we can suppose that the routes will allow to execute the test sequences.
(4) *Replay* : Despite the connectivity and stability assumptions, the test could sometimes fail or be inconclusive due to the radio or topological hazards. Thus, to avoid wrong decisions, we have to replay test sequences before giving the verdict.
(5) *Fail* : If the test is too many times "inconclusive" then we may consider that the test has failed. The testing replay number has to be decided according to the implementation and the test context.
(6) *Implementation choices*: We assume that the implementation under test

has the same options as the context. Capability testing techniques can be applied to check that aspect [17]. This assumption is necessary in conformance testing as well as in interoperability testing to prevent wrong decisions.

These assumptions allow to ensure a reliable testing environment by reducing wrong final verdicts.

3.4 Formal Specification Language

Following the conformance testing phases introduced in Section 3.1, the third step is to formally describe the system to be tested. It is necessary to choose a formal model to reach this objective. We select the Extended Finite State Machines (EFSMs) [17] that are well adapted to describe communication protocols.

Definition 1 *An EFSM M is defined as :* $M = (I, O, S, \overrightarrow{x}, T)$ *with* I*,* O*,* S*,* $\overrightarrow{x}$ *and* T *respectively a set of input symbols, a set of output symbols, a set of states, a vector of variables and a set of transitions. Each transition* $t \in T$ *is a 6-tuple defined as :* $t = (s_t, q_t, i_t, o_t, P_t, A_t)$ *where*

- s_t *is the current state,*
- q_t *is the next state,*
- i_t *is an input symbol,*
- o_t *is an output symbol,*
- $P_t(\overrightarrow{x})$ *a predicate on the values of the variables,*
- $A_t(\overrightarrow{x})$ *an action on the variables.*

The language selected to provide the specification is the Specification and Description Language (SDL) standardized by ITU-T [18]. This is a widely used language to specify communicating systems and protocols, based on the semantic model of EFSM. Its goal is to specify the behavior of a system from the representation of its functional aspects. It allows to describe the architecture of the system i.e. the connection and organization of the elements (blocks, processes, etc.) with the environment and between them. The behaviors of the entities in terms of their interactions with the environment and among themselves may also be designed. These interactions are described by tasks, transitions between states, and are based on the EFSMs.

4 DSR - Dynamic Source Routing

4.1 The Dynamic Source Routing Protocol

Dynamic Source Routing (DSR) is a reactive protocol that discovers and maintains routes between nodes on demand [19]. It relies on two main mechanisms, *Route Discovery* and *Route Maintenance*. In order to discover a route between two nodes, DSR floods the network with a *Route Request* packet. This packet is forwarded only once by each node after concatenating its own address to the path. When the targeted node receives the *Route Request*, it piggybacks a *Route Reply* to the sender and a route is established. Each time a packet follows an established route, each node has to ensure that the link is reliable between itself and the next node. DSR provides three successive steps to perform this maintenance: link layer acknowledgment, passive acknowledgment, and network layer acknowledgment. If a route is broken, the node which detects the failure sends by piggybacking a *Route Error* packet to the original sender.

4.2 DSR Formal Model

Most of the standards or drafts, especially for routing protocols, are heavy and propose diverse options. In the experimental RFC of DSR, several kind of implementations or functionalities may be developed by the notations MAY, SHOULD, COULD or MUST. Therefore, a complete formal specification being a very important task, choices have been taken as it follows.
Based on the available experimental RFC, it exists multiple implementation and specification possibilities. We have decided to focus on the main characteristics of DSR that are useful for our test purposes (our specification is detailed in [20,21]). The specification developed in SDL models the *Route Discovery* mechanism as well the *CachedRouteReply*, the *RouteMaintenance* with its three acknowledgement procedures, the *Flow State Extension* specified as an option triggered or not when a node is instantiated, and the main structures required by DSR such as the *RouteCache, the SendBuffer* or the *MaintenanceBuffer*. Besides, we do not specify how to support multiple interfaces or security concepts (which is a perspective research).
These mechanisms specify the nodes' behavior that is composed by three processes. A DSR node is represented as illustrated by the Figure 2.

The process *Node* behaves very easily: it represents the interface between the environment and the IP process for the received packet. The process *IP* only generates the IP datagrams that are transmitted to the lower layers after re-

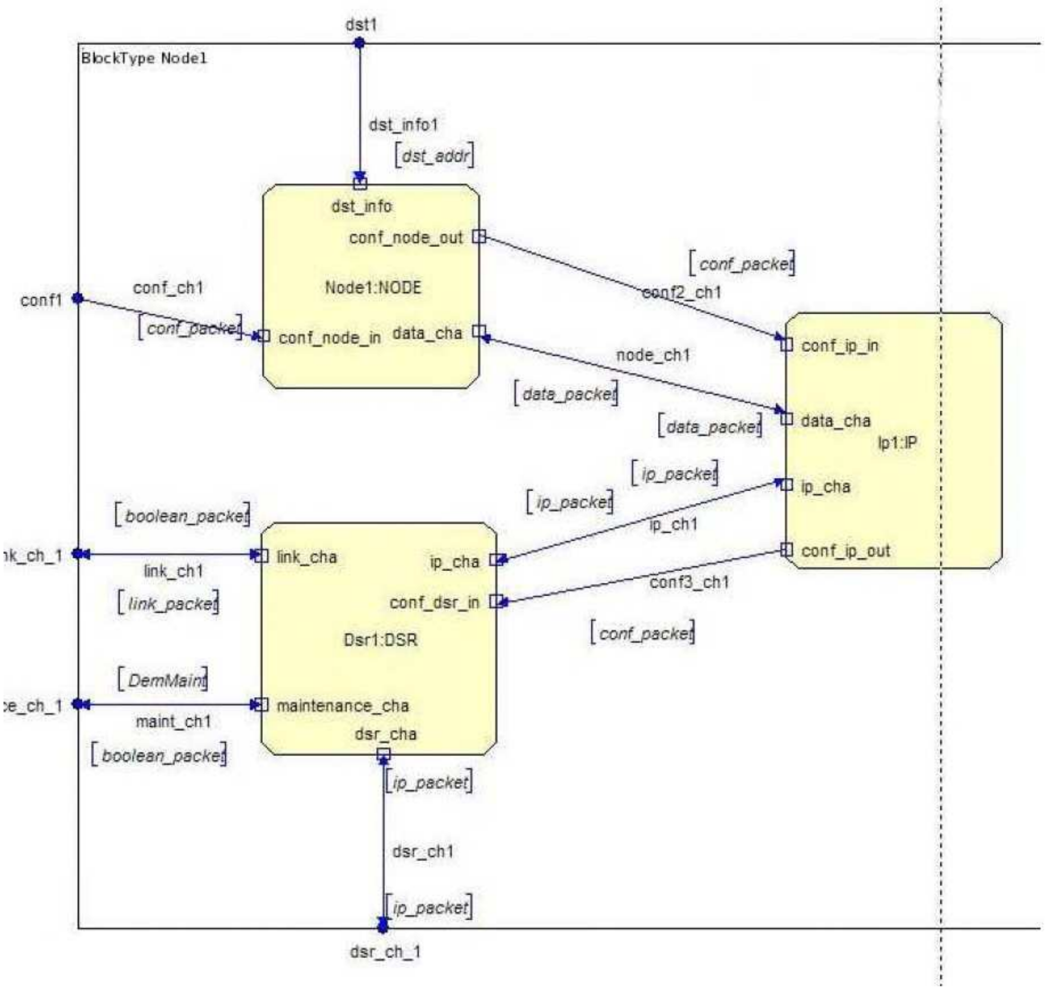

Figure 2. A DSR node specified in SDL.

ceiving a packet from the process Node. The process *DSR* specifies the whole behavior of DSR. A node may be instantiated as much as necessary. Nevertheless, the number of generated nodes determines the test purposes and the results obtained (more nodes there are, more difficult is the test execution). In order to represent the different links between nodes, we use a special block, called *Transmission* that contains a matrix where we define the connectivity in the networks. The matrix may be easily updated by sending information to create or remove a link. It means that we are able to modify dynamically the topology of the network in the purpose of representing the node mobility by changing their neighborhood. The definition of such a matrix is given as following.

Definition 2 *An adjacency matrix.*
Let M be a matrix $N \times N$ where N is the number of instantiated nodes. $\forall (i, j) \in N^2$,

$$M = \begin{cases} M_{ij} = 1 & \textit{if the node } i \textit{ may transmit to the node } j. \\ M_{ij} = 0 & \textit{if } i \textit{ is unable to directly transmit to } j. \end{cases}$$

We notice that if the links are supposed bidirectional, the matrix is symetric. This is an assumption used in our testing experiments.

4.3 Test Sequences Generation

From the above mentioned model, our main goal is to generate test sequences from the specification to validate some functionalities of a real implementation. These properties are formulated as test objectives. In order to produce these sequences, we apply a testing tool developed in our research lab [22]. Its inputs are the formal model and the test purpose (objectives) written in a special SDL syntax and the testing process is entirely automatized by providing the executable test sequences.
Some papers have already been published notably on the conformance and interoperability testing of wireless mobile routing protocols (DSR and OLSR[23]) in which we have produced test sequences taking into account the inherent constraints of MANET routing protocols [24,25,26] and specific scenarios in which the nodes, mobility, packets and topology were controlled.

However, even if the model is formal and that the test sequences may be numerous, to control the above mentioned aspects is an important constraints. Indeed, the ideal would be to perform our conformance testing in an unknown DSR network in terms of topology, number of nodes and their mobility. SDL allows to describe network topologies by using node instances, although it is impossible to guarantee that the topology of the real network will match the one of the specification. In that way, the work presented in this paper mainly aims at reducing the number of instantiated nodes required to generate the test sequences that will be enforceable in a non-controlled MANET. Besides, the minimization of the specification helps to avoid the state space explosion problem that may occur when generating the test sequences. This is the subject of our next sections.

5 Nodes Self Similarity

In opposition to wired and fixed networks, when a test sequence has been generated and has to be executed on a MANET routing protocol, if the network is not controlled (as mentioned before) it becomes very difficult to know how to execute it and how to interpret the verdict provided by the testers.

We therefore need to provide test sequences enforceable on any real network topology. Nevertheless, due to the volatility of these kind of networks, their topology may be quite different that the one represented by the specification. In order to answer to those issues, we present in the following the nodes' self similarity concept.

5.1 Self Similarity Notions

The self similarity notion is presented and applied in [27] especially to lead a real wired network topology to another one that is smaller and equivalent regarding the testing process. Indeed, two nodes in a network being self-similar may be composed to become a single node. This technique allows, under some conditions, to map some topologies from the same equivalence class to a simpler topology from the same class.
In other words, the self similarity means that if a packet is received by a node A and sent to B and forwarded to another one, then A and B has the same behavior according to this packet and it is possible to compose A and B to A' that represents the two nodes with the encapsulated internal communications. Formally, we define the combination of two nodes and then the self similarity notion.

Definition 3 *Nodes combination*
Let $\{N_i\}_{i\in E}$ *where* $E \in [1..n]$ *and* $n \in \mathbb{N}$ *be a collection of model that can be described as EFSMs. We note* $N_1 \circ ... \circ N_n$ *the combination of all* N_i *defined as :*
$O(N) = \bigcup_{i\in E} O(N_i)$
$I(N) = \bigcup_{i\in E} I(N_i) - \bigcup_{i\in E} O(N_i)$
$S(N) = \Pi_{i\in E} S(N_i)$
$\overrightarrow{x}(N) = \Pi_{i\in E} \overrightarrow{x}(N_i)$
$T(N) = (s, s', e, o, P(\overrightarrow{x}), A(\overrightarrow{x}))\ if\ (s_i, s'_i, e, o, P_i(\overrightarrow{x}), A_i(\overrightarrow{x})) \in T(N_i)$
where $P_i(\overrightarrow{x}) \equiv P_i(\overrightarrow{x_i}),\ A_i(\overrightarrow{x}) \equiv A_i(\overrightarrow{x_i}),\ e \in I(N_i)$ *and* $o \in O(N_i)$
Let $\Phi \subset O(N)$*, we define* $ActHide_\Phi(N)$ *as the obtained EFSM from* N *where each action of* Φ *becomes an internal one. This application transforms the communications between the different components of* N *into non-observable actions.*

Thus we may define the self similarity of two nodes as:

Definition 4 *Nodes self similarity*
Let two possible actions for a node be $send(Message, n, m)$ *and* $receive(Message, n', m')$ *where* n *(respectively* m'*) the observed node,* m *(respectively* n'*) the destination of the packet (respectively sender), and* $Message$ *the whole possible contents of a packet. Let* N *be a node specification. We note* $Tr(N)$ *the set*

of observable traces, a trace being an input/output sequence. Beside, $Tr(N)$ is a finite set , indeed the variable domains of the EFSM are discrete and finite (as most of the communication protocols).

Some $N_{i \in I}$ are self-similar if :
$Tr(ActHide_{\Phi}(N_1 \circ N_2)) \subseteq Tr(N)$,
where $\Phi = \{send(Message, N_1, N_2), send(Message, N_2, N_1),$
$receive(Message, N_1, N_2), receive(Message, N_2, N_1)\}$

The self-similarity approach is easily applied in a wired network but due to mobility and the unreliable communications in a MANET, it can not be performed directly. The combination of mobile nodes is impossible, indeed the trace property could not exist if for example two consecutive nodes on a route can not communicate with each other anymore.
Therefore, we use the self-similarity with three restrictions :

(1) The self-similarity is applied from the point of view of a single node which is the IUT.
(2) The self-similarity is applied each time a packet of the test sequences is received or sent in order to simplify the possible topologies known by the IUT.
(3) The self-similarity is applied only for a specific communication on a defined route (not considering all communications in the network) between the IUT and another node.

5.2 Nodes self similarity through MANETs

As the links in those kind of networks are unreliable and unpredictable, it is needed to consider a communication failure. Therefore, a path from a source S to a destination D is divided in two parts. The first one is composed by the nodes following the source S that succeeded the packet forward, and the second one that contains D and all nodes that did not receive the packet.
In the case of conformance testing where the network is perceived only from the IUT point of view (which is often S), three kind of nodes are noted: the source, the destination and the other path nodes N_i. We also assume that except S and D, all other N_i have the same functional behavior.
A route is defined as a succession of S, N_i where $i \in [1..n]$ and D (Figure 3).

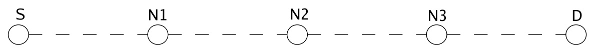

Figure 3. A simple route between S and D.

We consider the nodes in the route from the point of view of S which is the IUT. Two possible cases arise during a communication between nodes on a particular route: either the communication between two successive nodes N_i

and N_{i+1} succeeds, or it fails. We consider a communication as a success if a packet received by N_i is forwarded to N_{i+1} and forwarded after to N_{i+2} without provoking a *RteError* regardless of the meaning used for the acknowledgment.

The process of nodes self-similarity may be illustrated as follows:

- *Transmission success* : If a transmission between N_i and N_{i+1} succeeds, we combine these two nodes in a new node $N_\circ$. The communications between N_i and N_{i+1} are considered as $N_\circ$ internal actions. If the communication between $N_\circ$ and N_{i+2} succeeds, we iterate the process and so on. Thus, in case that the packet from S reaches D without causing a *RteError*, we may combine all the intermediate nodes as illustrated in Figure 4.

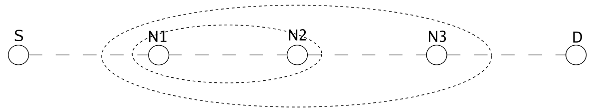

Figure 4. Combination by self-similarity when all communications succeed.

- *Transmission failure* : If a communication fails between N_i and N_{i+1}, it means that all the previous communications have succeeded. So the nodes between N_1 and N_i are combined. Finally, all the nodes after N_{i+1}, including D have the same behavior for an observer placed on the IUT. We therefore combine all the nodes from N_{i+1} to D into a new node D (Figure 5).

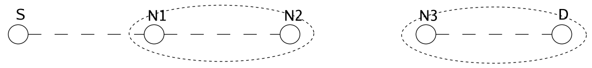

Figure 5. Combination by self-similarity when a communication fails.

Besides, we may note two exceptions : if the failure occurs between S and N_1 or S and D when a direct link exists. However, from the point of view of S and with respect to our test sequences, the length of the main route does not matter. Indeed, a direct connection or a multi-hop route (except for a route sorting in the RouteCache) are observationaly equivalent to S. Therefore, we formalize the definition of $N_\circ$ in order to handle the exceptions.

$$N_\circ = \begin{cases} N_\varnothing & \text{where } N_\varnothing \text{ is the neutral element.} \\ N_x \circ N_{x+1} & \text{where } N_x \in \{N_\varnothing\} \cup \bigcup_{i=1}^{n} \{N_i\}. \end{cases}$$

By making connection with the MANETs components, $N_\varnothing$ is an abstract node representing a node that only forwards the packets without modifying anything in the packet.

The two exceptions are handled as it follows:

· If a direct link between S and D is broken, we introduce $N_\circ = N_\varnothing$ as a node on the route.

· if the broken link takes place between S and N_1, we have combined all the N_i and D in a new node D. Thus this leads to a similar situation than in the first case, we also introduce $N_\circ = N_\varnothing$ between S and the combined node D.

With these definitions, the IUT behavior is not impacted by the path length when the test sequences are executed (excepted for the test sequence selection). However, even if the self-similarity may reduce the specification from which test sequences are generated, their executions on the implementation and the relationship with this specification have to be defined. This is the subject of the following section.

6 Topological Equivalence for the Conformance Testing of MANET routing protocols

6.1 Definitions

We present in this section the needed definitions in order to illustrate that in the conformance testing framework and the test sequences execution phase, it is possible to map a real MANET topology to the one described in a specification.

Definition 5 *Routes' sets*

Let Spec be our routing protocol specification.
Let Imp be our implementation.
Let S and D the source and destination nodes:

- *let S_{spec} and D_{spec} be their representation in the specification.*
- *let S_{imp} and D_{imp} be their representation in the implementation.*

Let $(p_n(x)|\{x, n\} \in \mathbb{N})$, the n^{th} unique path chosen by S_{imp} (according to the DSR procedures) to reach D_{imp} that is composed by x nodes.
Let $P0_{spec}$ and $P1_{spec}$ (respectively $P0_{imp}$ and $P1_{imp}$) two ordered sets composed by nodes connecting S_{spec} to D_{spec} (respectively S_{imp} to D_{imp}).

For example, in the classical case of shortest path decision, the n are affected by ascending length order from 0, functions of hops number needed by S to reach D. If this number is equal, the first discovered route (that is the first RouteReply received) is affected by the smaller n.
More precisely, P_0 and P_1 are defined by:

- *In Spec*

- $\{(S_{spec}, N0, D_{spec})\} = P0_{spec}$ *(also written* $\{(S_{spec}, N0), (N0, D_{spec})\} = P0_{spec}$*)*
- $\{(S_{spec}, N1, D_{spec})\} = P1_{spec}$ *(also written* $\{(S_{spec}, N1), (N1, D_{spec})\} = P0_{spec}$*)*

- *In Imp*
  - $p_n(x) \in P(n \; mod \; 2)_{imp}$ *i.e. all the route with 0 or an even subscript will be in the set* $P0_{imp}$ *and those with an odd subscript will be in the set* $P1_{imp}$*. Thus, if there is a RouteError we preserve the route alternation between the set.*

We also note $P0_{imp}(n)$ and $P1_{imp}(n)$ respectively the n^{th} elements of $P0_{imp}$ and $P1_{imp}$.
Besides, we define P_{imp} (respectively P_{spec}) as $P_{imp} = P0_{imp} \cup P1_{imp}$ (resp. $P_{spec} = P0_{spec} \cup P1_{spec}$) the set of paths known by S to reach D in Imp (resp. in $Spec$).

Corollary 1 *From the definitions of* $P0_{imp}$*,* $P1_{imp}$ *(resp. of* $P0_{spec}$ *and* $P1_{spec}$*) and* $p_n(x)$*, we note that* $P0_{imp} \cap P1_{imp} = \emptyset$ *(resp.* $P0_{spec} \cap P1_{spec} = \emptyset$*).*

The definitions 4 and 5 will allow to define the specification topology that will be applied on a large topological classes of a real MANET.

6.2 The simplified specification

The first behavioral phase of all DSR nodes willing to communicate is a $RReq$ in order to discover the paths. This phase requires at least one path from S to D. Therefore, we need a specification illustrating at least one path. Once the communication is set, a test sequence may be generated from this path unless it implies a $RErr$. This break requires another path especially to test the DSR node adjustment.
We may therefore define two routes $P0_{spec}$ and $P1_{spec}$ representing these two paths. Since in this paper S is the implementation under test and that only the Inputs/Outputs (I/O) concerning it will be analyzed through our test sequence, $P0_{spec}$ and $P1_{spec}$ are defined as in the Definition 5. Due to nodes self-similarity that allows to reduce a route between two nodes in a two-hop one, we can keep only the smallest number of nodes required to generate a test sequence according to specific test objectives. To test functional properties of DSR, no test objectives requiring more than two routes in the network were found out. Then, our specification can be reduced into four nodes, S, N_0, N_1 and D which compose two routes $[S, N_0, D]$ and $[S, N_1, D]$ as represented in Figure 6 and enunciated by the following theorem.

Theorem 1 *Let us assume a network composed by DSR nodes in which a source S is willing to communicate with a destination D, that a path between the two nodes exists and that each intermediate nodes behaves as routers (no attacks). Let us assume the hypothesis of Section 3.3. Let us suppose that we*

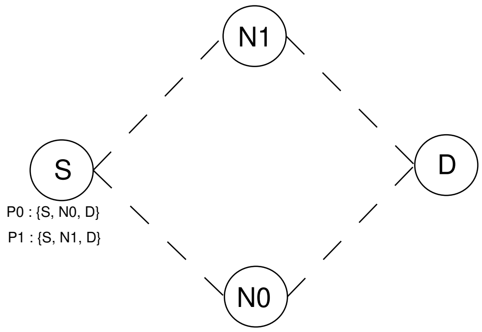

Figure 6. Specification topology.

have a specification of this network with as much instantiated nodes as we wish and that the node S is our DSR IUT.
Then the reduced specification of the network as above illustrated is equivalent to the real network according to the above mentioned testing process. Formally:

Let q(x) be a route in the real network from S to D composed of x hops.
Let $S_{spec}|q$ *be the behavior of* S_{spec} *when S and D communicate with each other through the route q.*
Let $q1 \doteq q2 \Leftrightarrow S_{spec}|q1 = S_{spec}|q2$.

Then $\forall n > 2, q(n) \doteq q(2)$.

That means, with S as the IUT, a route of 1 or n hops has the same effect for conformance testing. Therefore, the specification describes in the Figure 6 may be applied.

□

A detailed proof is given in details in [21].

All along the test execution, the TCP will preserve a relation between $P0_{spec}$ and $P0_{imp}$, and also between $P1_{spec}$ and $P1_{imp}$. For instance, if the routing metric is "minimal hop count" (assumed in the paper), TCP will affect in $P0_{imp}$ the shortest path as $p_0(x)$, in $P1_{imp}$ the second as $p_1(y)$, in $P0_{imp}$ the third as $p_2(z)$ and so on with $x \leqslant y \leqslant z \leqslant ...etc$. Both sets save the theoretical *RouteCache* in the TCP. Here "theoretical" is used because the *RouteCache* could eventually be filled in an other way during the *RouteDiscovery* mechanism. This problem will not be discussed in this paper because we focus here our study on functional properties.
With respect to *Spec*, $P0_{imp}$ and $P1_{imp}$ match possible routes described in the specification. For instance, if a test sequence implies that $P0_{spec}$ disappears : the TCP will detect the RouteError packet as an input, will erase the first element of $P0_{imp}$ i.e. $p_0(x)$ and will select $p_1(y) \in P1_{imp}$ as the new route that IUT must use.
We have to underline the fact that a node N could belong to several routes (a

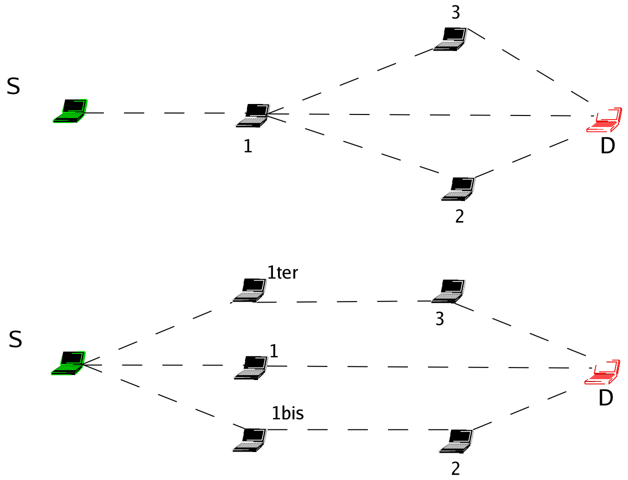

Figure 7. Self-Similarity of a node involved in different routes.

GratuitousRouteReply sent to S by the node involved into different routes). In this case, with respect to our sets, we have to duplicate this node nbr times where nbr is the number of routes containing N in the network representation of the TCP. Thus we create nbr routes in order to apply nodes self-similarity to each one and to split up these routes into our two sets $P0_{imp}$ and $P1_{imp}$ (Figure 7).

We illustrate in the following a short example of a test sequence execution applying our methodologies.

6.3 An example of a test sequence execution

Once the RouteDiscovery has been performed for S, the IUT has a representation of the topology. From the point of view of S, the selected route can be simplified by self-similarity for each subsequent I/O as explained in section 5. In order to insert the test sequence in the IUT, we use a Upper Tester (UP) and a Lower Tester (LT) linked to the TCP. They are respectively connected to the upper and lower interface of the protocol layer.
In our example, a test sequence which contains two objectives is used :
- After having sent a RReq packet, the IUT waits for at least one RteRep packet, before sending a SrcRte packet by the shortest route to the destination in its cache.
- If a RteError is received, the IUT will use the next shortest route to the destination in its cache to send a SrcRte containing the original message.

To generate the test sequences, after the RouteDiscovery mechanism, we remove the link between $N0$ and $N1$ using the adjacency matrix of the *Transmission* block. The test scenario generation is detailed in our previous works [20]. By lack of space, we depict the related simplified MSC in the Figure 8.

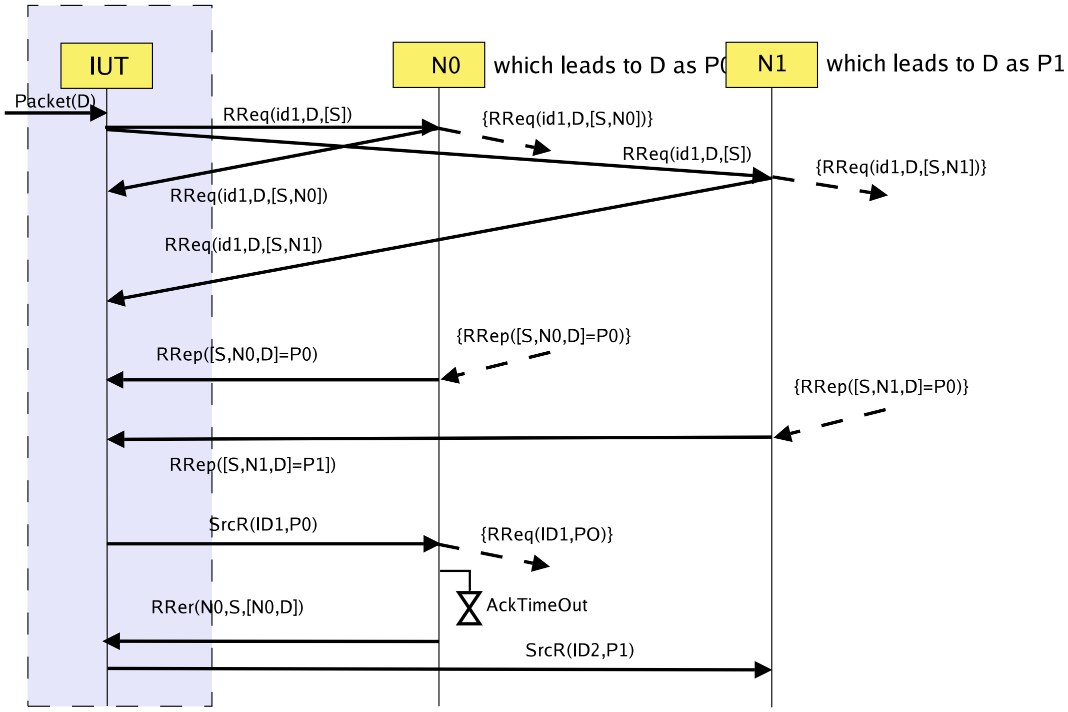

Figure 8. Simplified MSC of a test scenario.

We detail hereafter the different steps of the execution in the network shown in Figure 9.

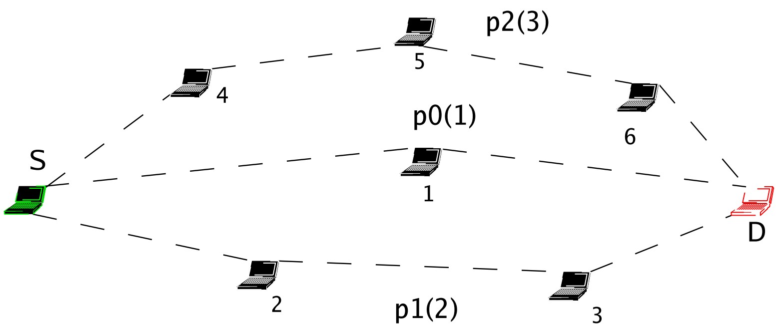

Figure 9. A network example.

For convenience, we use *pass*, *fail* and *inconclusive* to interpret each step of the decision in the TCP and $PASS$, $FAIL$ and $INCONCLUSIVE$ to define the final verdict decided by the oracle i.e. TCP.

1- UT injects a $Packet$ in the IUT to the destination node D - *pass*
2- LT observes an output $RReq(S, D)$ - *pass*
3- LT observes an input $RReq(S, D)$ identical as the one sent where N0 has added its own address - *pass*
4- LT observes an input $RReq(S, D)$ identical as the one sent where N1 has added its own address - *pass*
5- LT observes an input $RRep(S, 1, D)$ and TCP stores this route - *pass*
6- LT observes an input $RRep(S, 2, 3, D)$ and TCP stores this route - *pass*
7- LT observes an input $RRep(S, 3, 4, 5, D)$ and TCP stores this route - *inconclusive* this packet is not expected but do not invalidate the conformance. TCP sorts the different routes by generating $p_0(1) = (S, 1, D)$, $p_1(2) = (S, 2, 3, D)$ and $p_2(3) = (S, 4, 5, 6, D)$ which are put in the sets $P0_{imp}$ et $P1_{imp}$ as defined by the "shortest path" strategy and the Section 6.2. A route pointer defines which route the TCP expects the IUT to use. This pointer indicates the first element of $P0_{imp}$, $p_0(1) = (S, 1, D)$. The sets $P0_{imp}$ and $P1_{imp}$ in Imp are equivalent to $P0_{spec}$ and $P1_{spec}$ in the specification for the test scenario.
8- LT observes an output $SrcR(p_0(1))$ and $p_0(1) \in P0_{imp}$ - *pass* because the route chosen in Imp is equivalent to the one chosen in $Spec$
From this step we can use Node self-similarity to represent the route from the point of view of the IUT.
We assume the link between node 1 and node D to be broken.
9- LT observes an input $RErr(1, D)$, TCP checks $P0$ et $P1$ and removes $p_0(1)$ from $P0$. The pointer moves on $P1(1) = p_1(2) = (S, 2, 3, D)$. - *pass*
10- LT observes an output $SrcR(p_1(2))$ and $p_1(2) \in P1$ - *pass* because the route chosen in Imp is equivalent to the one chosen in $Spec$.
11- TCP, as the oracle, gives the final verdict $PASS$ because each expected I/O has been observed.

7 Experimental Emulation

7.1 Emulation choice

The two main emulation techniques currently applied are the *trace* and *direct* emulation presented in [28]. The first concept allows a realism and a refinement of the mobility and the network simulation as well the communication management with the upper layers. Moreover, it is particularly adapted to the performance and robustness testing of a protocol. In our case, our main objective is to test an implementation of DSR and its correctness in respect to its requirements and not really regarding its performances or non-functional as-

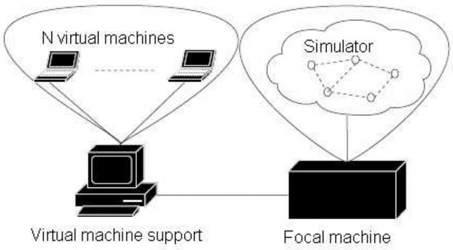

Figure 10. A direct emulation configuration.

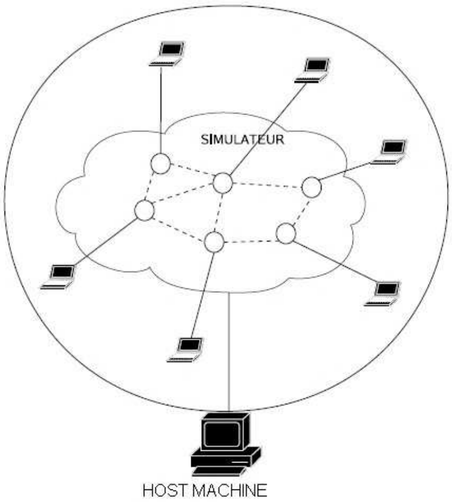

Figure 11. An other direct emulation configuration.

pects reliability. That is why we decided to apply a direct emulation technique. This is based on the following entities:

- a simulator: it manages the nodes' mobility into the network and the communications with each other. A great majority of simulators currently integrates an emulator process.
- a focal machine: this machine hosts the network simulation whose the different components may be simulated or emulated (according to the options available in the simulators).
- virtual machines: virtual nodes (simulated ones) communicating with the focal machines. Let us note that it is possible to connect real computers to the focal machine for load sharing and virtual nodes increasing.

The Figure 10 presents an emulation architecture separating the focal machine and the one managing the virtual nodes whereas Figure 11 illustrates only one host.

For a conformance testing activity on the DSR protocol aiming at checking the functional properties of the protocol, the testing purposes may show that an important number of nodes is not mandatory. Therefore, a discrete events simulator such as NS-2 is sufficient and we will thus use NSe (NS-2 patched as emulator) [29]. Four virtual machines are executed on the focal machine that also runs the simulation side.

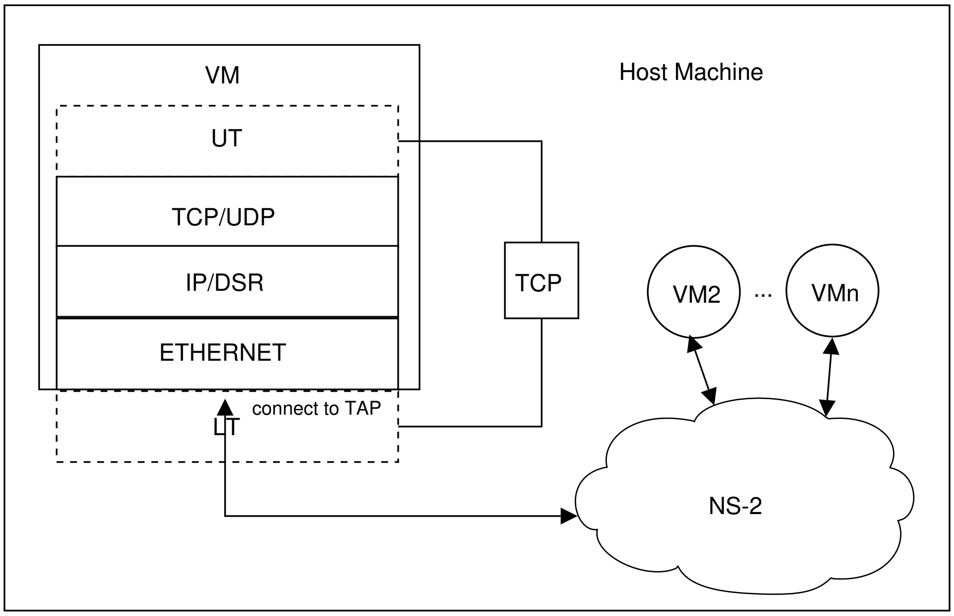

Figure 12. Direct emulation and testing architecture

7.2 DSR-UU, virtual machines and devices

Our approach is applied on experimentation through the DSR-UU implementation [30]. The test sequences are provided by one of our tools TESTGEN-SDL [16] and some test purposes. Direct emulation is used. It allows to use a real implementation of a protocol stack with a simulator to represent the mobility and to manage the communications.
The direct emulation is performed on a focal machine with the following characteristics :

- Pentium©M 1,6 GHz
- 512 Mo Ram
- Fedora-2.6.15 kernel with skas patch
- TUN/TAP interfaces activated

We use *User Mode Linux* [31] to create virtual machines with existing prepared kernel and file system [32]. DSR-UU was added in the kernel.
NS-2 patched for emulation was performed to manage mobility and wireless communication between the virtual machines. We may note that a maximum of six virtual nodes is possible on a same focal machines. If we want a larger collection of nodes, it is necessary to distribute the virtual machines on more than one focal computer. The proposed emulation and testing architecture are depicted in Figure 12.

7.3 Experiment

When test sequences are obtained from a formal specification, the emulation process allows to execute many of these sequences. Moreover, it enables the IUT to communicate with other implementations (that are supposed correct). The NS-2 simulator allows to manage the nodes mobility defined either randomly or performing a Tcl script. In our case, predefined moves are illustrated in such a Tcl file . The test objectives aim at testing the *RouteDiscovery*, the *RouteReply* and the *RouteError*. Interesting results have been obtained and no FAIL verdicts have been provided (see [21] for details).

Nevertheless, by using random topology and mobility patterns, we may notice many INCONCLUSIVE verdicts. This is mainly due to unpredictable topological changes. In other words, if the IUT is considered as an EFSM, it triggers a transition that is different that the one planed by the specification due to a non-anticipated topology. Therefore, as the mobility could have negative impacts on the testing verdicts, new mechanisms have been added in our testing process in order to reduce the mobility's edge effects. This is the subject of the following section.

8 Inconclusive Verdicts Reduction

Despite the test assumptions (section 3.3) and the nodes self similarity (section 5), it is necessary to maintain a relationship between the specified topology and the network as it is known by the IUT. Indeed, during the test process, the IUT may have several routes for a same destination, some of them may be merged at one moment if a *RouteShortening* is operated, or may disappear suddenly.

8.1 Route relationship maintenance

From the aspect above mentioned, it is necessary to know at each moment, which route is applied by the implementation, which one would have to be chosen theoretically and the correspondence of this route with a path in the specification.
We use the Definition 5 which mentions that the routes length defines their allocation in the *RouteCache*.
By definition, the n routes known by the IUT are labeled from 0 to $n-1$ and initially the even routes are placed in $P0_{spec}$ and the odd ones in $P1_{spec}$. These two sets are implemented in the testing coordination procedure and allows to

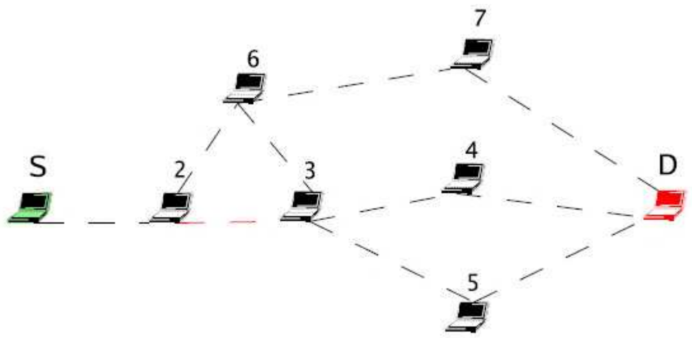

Figure 13. Example of a link common to two routes.

provide a theoretical *RouteCache*. "Theoretical" means that it is filled as it should be according to the sent and received messages.
Within the specification, there are therefore two paths that are alternated when a route is changed due to a *RouteError* or when the link is broken at the first hop. This alternation has to be taken into account during the testing process with a maintenance of $P0_{spec}$, $P1_{spec}$, $P0_{imp}$ and $P1_{imp}$ as it is depicted in the following.

8.2 Exceptions

It exists some situations in which we may not apply directly our methodology relying on the test assumptions, a reduce specification and a local test architectures. These situations were met during the experiments and provoked erroneous or inconclusive verdicts. We then propose an algorithm managing these exceptions.

8.2.1 RouteError on a common link

A problem is met when a link concerning some $p_n(x)$ in $P0$ and $P1$ disappears. Typically, this may be observed when the DSR *GratuitousRouteReply* procedure is performed. Taking the Figure 13 for instance, if the link [2,3] disappears, the node 2 will emit a *RouteError* that leads to the suppression of all routes known by S that contains the link [2,3].

After the route discovery phase, if we assume: $P0_{imp} = \{p_0(4) = (S, 2, 3, 5, D)$, $p_2(4) = (S, 2, 6, 7, D), p_4(5) = (S, 2, 6, 3, 4, D)\}$ and $P1_{imp} = \{p_1(4) = (S, 2, 3, 4, D), p_3(5) = (S, 2, 6, 3, 5, D)\}$ where S is the IUT, thus the *RouteError* will lead to the modification of $P0_{imp}$ and $P1_{imp}$ and then the IUT could use two consecutive routes belonging to the same set. In this case, these sets have to be modified to keep the coherence of sets alternation.

8.2.2 *Unpredictable RouteError*

Another problem is met during the test sequence execution when *RouteError* packets are observed while no such packets are planed to be received. Other messages may obviously be received such as *RouteRequest* or *SourceRoute* coming from other nodes. As these latter do not modify the routing information between the IUT S and D, these packets are simply ignored and discarded by the TCP. An unexpected *RouteError* is not the fact of a incorrect implementation but rather of a real unexpected topological changes sustained by the executed test sequence. Therefore, we need to maintain $P0_{imp}$ and $P1_{imp}$ as in the previous case. By the way, we need to note that this unexpected packet may be received in two well-defined situations:

- Following a *SourceRoute* emitted by S,
- *RouteError* for another node recipient.

8.2.3 *A global algorithm*

In order to answer to the previously mentioned problems, we present below an algorithm depending on whether:

- the *RouteError* is expected or not,
- the *RouteError* is sent to the IUT or not,
- the *RouteError* means that a link of the current applied route is broken.

Algorithm to manage the TCP according to RouteError messages

We note $P_{imp} = P0_{imp} \cup P1_{imp}$ (respectively for spec) an ordered set of 2-uple sets. Let P be a pointer in the TCP pointing on the set containing the current applied route used by the test sequence and $\overline{P}$ its complementary. Let p be a pointer in the TCP pointing on the first element of P.
Let $bl = (i, j)$ be a broken link between the nodes i and j identified by the received RouteError.
We node RE the couple (a,d) where the variable a is 1 if the RouteError is expected, 0 otherwise, and the variable d is 1 if the IUT is the destination node of the RouteError, and 0 otherwise.

1. **If** *$bl \in p$ and $RE = (1, d)$* **then**
(a) $\forall p_n(x) \in P_{imp}$: if $bl \in p_n(x)$ then $P_{imp} = P_{imp} \setminus \{p_n(x)\}$.
(b) Label each route by $n \in [1..m]$ if $P = P0_{imp}$ and $n \in [0..m-1]$ otherwise, where m is the number of remaining routes as defined by Definition 5.
(c) Rebuild the sets $P0_{imp}$ and $P1_{imp}$ as defined in Definition 5.
(d) Assign $P := \overline{P}$.
(e) Assign p to the first element of P.
(f) Resume the testing process if possible. (*)

2. **If** *$bl \in p$ and $RE = (0, d)$* **then**
(a) $\forall p_n(x) \in P_{imp}$: if $bl \in p_n(x)$ then $P_{imp} = P_{imp} \setminus \{p_n(x)\}$.
(b) Label each route by $n \in [0..m-1]$ if $P = P0_{imp}$ and $n \in [1..m]$ otherwise, where m is the number of remaining routes.
(c) Rebuild the sets $P0_{imp}$ and $P1_{imp}$.
(d) Assign p to the first element of P.
(e) Resume the testing process if possible. (*)

3. **If** *$bl \notin p$ and $RE = (a, d)$* **then**
(a) $\forall p_n(x) \in P_{imp}$: if $bl \in p_n(x)$ then $P_{imp} = P_{imp} \setminus \{p_n(x)\}$.
(b) Label each route by $n \in [0..m-1]$ if $P = P0_{imp}$ and $n \in [1..m]$ otherwise, where m is the number of remaining routes.
(c) Rebuild the sets $P0_{imp}$ and $P1_{imp}$.
(d) Resume the testing process.

(*) i.e. the emission of the previous SourceRoute. If the sequence still requires a route to transmit a message but $P_{imp} = \emptyset$, an inconclusive verdict could be obtained anyway.

By this algorithm, we are able to manage efficiently the sets contained in the TCP in order to keep them similar to the ones provided by the specification in reducing the inconclusive verdicts. Nevertheless, it is still not possible to evaluate the improvement rate, it indeed highly depends on the mobility patterns and is therefore currently studied.

9 Conclusion

We have presented in this paper a new methodology for the conformance testing of a MANET routing protocol, namely Dynamic Source Routing (DSR). An SDL formal model of DSR based on the EFSM semantic was specified. This specification is generic and allows the tester to generate as much nodes as necessary, nodes that communicate with each other using an adjacency matrix managing the nodes' connectivity. Besides, realists and adapted testing assumptions were stated.

The network diversity and all the possible nodes' connectivity configurations imply the necessity to map a large network topological class to a more simple one, basis of our specification. We propose in that way the concept of nodes self similarity that allows to map a route viewed by the IUT for a given message in a real network with a path of the specification containing only three nodes avoiding therefore an eventual state space explosion when generating the test sequences. From the self similarity of nodes and precise test objectives, we have succeeded to minimize the needed nodes in the specification.

Some experiments performing on a network with well-defined topology and mobility patterns were obtained. Test sequences were generated from the formal model and an emulation architecture was designed to test the conformance of a real DSR implementation (DSR-UU).

However, even if the defined test hypothesis and the testing architecture allow the test sequence execution, there are still many inconclusive verdicts due to the unreliability of radio communications but especially because of the huge number of mobility patterns and topology. Indeed, in many cases, the Implementation Under Test may receive unexpected and unpredictable messages that lead to inconclusive verdicts about the conformance of the DSR implementation. In order to reduce this number, we have introduced into the test coordination procedure some route sets similar to the ones from the specification as well an algorithm enabling their management when the topology is modified unexpectedly. Currently, one of our perspective researches is to measure the number of inconclusive verdicts prevented by our method in order to apply much more complex test sequences/objectives.

This testing process is very efficient for the DSR protocol and many interesting results were provided. And even if this could be performed on other protocols such as DYMO, it seems quite different for other proactive protocols such as OLSR. Indeed, our researches on that aspects underline that a totally different testing architecture has to be defined. It seems that a distributed architecture should be applied in order to match with the concepts of proactive routing protocols. This makes part of our perspective works.

Biography

Dr Stephane Maag is an Assistant Professor recruited in 2002 at the INT and more precisely into the Software Networks department and the VPS (Validation of Protocol and Services) research team. He is also involved in the CNRS research group Samovar (UMR 5157) since 2003. His current research activities are the testing of protocol and services from formal or semi-formal models in the MANET area. The application domains are especially the routing protocols (DSR, OLSR, etc.) testing phases, from the specification, the verification, validation and test sequences execution regarding the conformance and interoperability of the protocols. He is also currently involved in several European and National projects (such as the FP6 Marie-Curie TAROT (in which INT is the leader.), the @LIS E-LANE project, etc.).

Dr. Cyril Grepet obtained his PhD degree in 2007 from the National Institute of Telecommunications. His main research interests and activities are conformance and interoperability testing in wireless ad hoc networks. He is currently a Post-doctorant at the French Atomic Energy Commission involved in the development and improvement of testing tools.

Pr. Ana Cavalli received the Doctorat d'Etat es Mathematics Sciences and Informatics in 1984 from the University of Paris VII, France. In 1981, she joined the Laboratoire d'Informatique Theorique et Programmation of the C.N.R.S., Paris, where she worked on automatic proof methods for temporal logics and their applications to the specification and verification of protocols. From 1985 to 1990, she was a staff research member of the Department Langages et Systemes de Commutation, at the CNET(Centre National d'Etudes des Telecommunications), where she worked on software engineering and formal description techniques. Since 1990, she has joined as Professor the National Institute of Telecommunications. Currently, A.R. Cavalli is responsible at INT of the research group "Validation of protocols and services". Her research areas are Formal methods for Web services, MANETs, protocol security policies verification and testing.

References

[1] NS, The network simulator, http://www.isi.edu/nsnam/ns (2004).

[2] OPNet, The opnet modeler, http://www.opnet.com/products/modeler/home.html (2005).

[3] J. Cowie, D. Nicol, A. Ogielski, Modeline the global internet, Computing in Science and Engineering (1999) 30–36Http://dimacs.rutgers.edu/Projects/Simulations/darpa/Papers/internet-cise99.pdf.

[4] L.Bajaj, M.Takai, R.Ahuja, K.Tang, R.Bagrodia, M.Gerla, Glomosim: A scalable network simulation environement, Tech. Rep. 990027, UCLA Computer Science Department (May 1999).

[5] D.Cavin, Y.Sasson, A.Shiper, On the accuracy of manet simulator, in: Workshop on Principle of Mobeil Computing, 2002.

[6] Q.Ke, D.Maltz, D.B.Johnson, Emulation of multi-hop wireless ad hoc networks, in: Mobile and Multimedia Communications, 2000.

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

[8] M. Kim, M. Viswanathan, H. Ben-Abdallah, S. Kannan, I. Lee, O. Sokolsky, Formally specified monitoring of temporal properties, in: In Euromicro Conference on Real-Time Systems, 1999.

[9] T.Lin, S.F.Midkiff, J.S.Park, A framework for wireless ad hoc routing protocols, Wireless Communications and Networking 2 (2003) 1162–1167.

[10] U.Glasser, Q-P.Gu, Formal description and analysis of a distributed location service for mobile ad hoc networks, Theoretical Computer Science, 2003.

[11] I.Zakkuidin, T.Hawkins, N.Moffat, Towards a game theoretic understaing of ad hoc routing, Electronic Notes in Theoretical Computer Science, 2005 119.

[12] D. Lee, D. Chen, R. Hao, R. Miller, J. Wu, X. Yin, A formal approach for passive testing of protocol data portions, in: Proceedings of the IEEE International Conference on Network Protocols, ICNP'02, 2002.

[13] B. Alcalde, A. Cavalli, D. Chen, D. Khuu, D. Lee, Network protocol system passive testing for fault management - a backward checking approach, in: Lecture Notes on Computer Science, Vol. 3235, Springer, 2004, pp. 150–166.

[14] J. Arnedo, A. Cavalli, M. Nunez, Fast testing of critical properties through passive testing, in: Lecture Notes on Computer Science, No. 2644/2003, Springer, 2003, pp. 295–310.

[15] ISO/IEC, ISO/IEC 9646-1. Information Technology - open Systems Interconnection - Conformance Testing Methodology and framework part 1: General concepts.

[16] A. Cavalli, D. Lee, C. Rinderknecht, F. Zaidi, Hit-or-jump: An algorithm for embedded testing with applications to IN services., in: J. Wu, S. T. Chanson, Q. Gao (Eds.), Formal Methods for Protocol Engineering and Distributed Systems, FORTE XII / PSTV XIX'99, Vol. 156 of IFIP Conference Proceedings, Kluwer, 1999, beijing, China.

[17] D. Lee, M. Yannakakis, Principles and methods of testing finite state machines - a survey, IEEE Transactions on Computers 84 (1996) 1090–1123.

[18] ITU-T, Recommandation Z.100: CCITT Specification and Description Language (SDL), Tech. rep., ITU-T (1999).

[19] D. Johnson, D. Maltz, Y.-C. Hu, The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR) - Experimental RFC, IETF MANET Working Group, http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt (July 2004).

[20] S. Maag, C. Grepet, A. Cavalli, Un Modèle de Validation pour le Protocole de Routage DSR, in: Hermes (Ed.), CFIP 2005, 2005, pp. 85–100, bordeaux, France.

[21] C. Grepet, Methodology pour la validation de protocoles de routage ad hoc, Ph.D. thesis, Evry Val d'Essonne University (January 2007).

[22] F. Zaidi, Contribution à la génération de tests pour les composants de service. application aux services de réseau intelligent., Ph.D. thesis, Evry Val d'Essonne University (November 2001).

[23] T.Clausen, P.Jacquet, Optimized Link State Routing Protocol (OLSR) - RFC3626, INRIA, ietf Edition (October 2003).

[24] C.Grepet, S.Maag, A testing methodology for a manet routing protocol using a node self-similarity approach, in: The 3rd IEEE PerCom Workshop on Pervasive Wireless Networking, 2007.

[25] K. Merouane, C. Grepet, S. Maag, A methodology for interoperability testing of a manet routing protocol, in: The Third IEEE International Conference on Wireless and Mobile Communications, 2007.

[26] S.Maag, F.Zaidi, Testing methodology for an ad hoc routing protocol, in: ACM PMšHWšN 2006,, 2006.

[27] C.Djouvas, N.D.Griffeth, N.A.Lynch, Using self-similarity for efficient network testing, Tech. rep., Lehman College (2005).

[28] D.B.Johnson, Validation of wireless and mobile network models and simulation, monarch Project (1999).

[29] Ns-2 emulator, `http://www.isi.edu/nsnam/ns/ns-emulation.html`.

[30] E. Nordstrom, Dsr-uu v0.1, http://core.it.uu.se/core/index.php/DSR-UU, uppsala University.

[31] J. Dike, user-mode-linux, `http://user-mode-linux.sourceforge.net/`.

[32] B. Wehbi, Dynamic remote access solution to a hot-zone, Master's thesis, Université Pierre et Marie Curie (2005).