Model-Based Testing for MANETs

S. Maag <Stephane.Maag@telecom-sudparis.eu>
Institut Mines-Telecom/Telecom SudParis, CNRS UMR 5157,
9 rue Charles Fourier, 91011 Evry Cedex, France

Abstract. Conformance testing in network engineering is a crucial phase in the development of complex communicating systems. Model-based testing allows to automatize the testing process by generating test suites from a formal specification and to execute them on a real IUT. While many techniques have been developed, their application to test wireless routing ad-hoc protocols still raises many issues. The paper objective paper is to present the node self-similarity reducing the number of inconclusive verdicts often met in traditional MBT.

Keywords: model based testing; manet; nodes self-similarity; routing protocols

1. Introduction

Conformance testing in network engineering is a crucial phase in the development of complex communicating systems. Among the different testing steps, the development and execution of test cases based on a formal model is an important issue for testing communication protocols and other reactive systems. The purpose of these tests is to determine whether a protocol implementation conforms to its specification. Usually a conforming implementation is required to have the same input/output behavior as defined by the specification. In various application domains, such as telecommunication systems, communication protocols and other reactive systems, the specification can be represented in the form of an extended finite state machine (EFSM). In particular, EFSMs are the underlying models for formal description techniques, such as SDL and SysML. Several model based testing (MBT) techniques have been proposed to generate and execute test cases from formal specification [21]. While most of these specification techniques allow to efficiently design these above mentioned systems, there are currently new environments that bring their own inherent constraints. This is notably the case when considering wireless Mobile ad-hoc Networks (MANET).

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. Besides, conformance testing for ad hoc routing protocols is crucial to the reliability of those networks. 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. Andel et al. [5] tries to illustrate some comparisons between NS-2 [1], OpNet [2] and Glomosim [4]. Results providing by the simulation testing is sometimes far from the one obtained in a real case study.

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. Our work focuses on a testing technique based on algorithms to check the conformance of ad hoc routing protocols. Nevertheless, the execution of these test sequences 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. We illustrate 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.

The remainder of the paper is as it follows. In Section 2, we present some related works. In Section 3, basics on MBT are presented. Then, we describe the node self-similarity and its application on an emulated testbed in Section 4 and 5 respectively.

2. Related work

Verisim [6] is a model combining NS-2 and the trace verification component provided by the Monitoring and Checking system namely MAC [7]. 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*. The authors have shown that the AODV implementation in NS-2 was false regarding some properties. This first work was very interesting, disturbing and raised several issues regarding the efficiency/reliability of the simulation/emulation. These works have known recently a certain resonance by the publication of [23] in which MANET simulations results still reveal pitfalls. The authors describe "design space" of MANET routing in terms of its basic dimensions and corresponding parameters.

While Verisim performs a validation through invariants in a simulator, [8] 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. However, it does not allow studying functional properties neither the nodes' interactions.

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 [9] developed a formal model namely Distribued Abstract State Machines (DASM). While this model allows verifying the behavior of a node in a functional way, unfortunately the model is non-executable and does not allow to observe the nodes' interoperability.

[10] 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 like the required knowledge of the network, are still present.

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.

corresponding context variables.

3. Conformance testing and formal models

3.1 Basics

The conformance testing usually relies on the comparison between the behavior of an implementation and the formal specification of a given protocol. 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. Design 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 is the definition of the test purposes.

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 from the specification.

Step 6. Format the test sequences i.e. to produce test sequences in some accepted formalism as Test Description Language (TDL) [13] or in Testing and Test Control Notation (TTCN3), the ITU-TS standard language used for test specification.

As above mentioned, our approach is based on formal model. In our work we define and use as a formal specification the Extended Finite State Machine (EFSM).

Definition 1. An EFSM M is defined as: $M = (I, O, S, x, T)$ with I, O, S, 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, $P_t(x)$ a predicate on the values of the variables, $A_t(x)$ an action on the variables.

Our protocols are specified using an EFSM based language, named Specification Description Language (SDL) standardized by ITU-T [22]. 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 describing the architecture of the system i.e. the connection and organization of the elements (blocks, processes, etc.) with the environment and between them.

3.2 Test case generation

There exist several techniques to generate test suites from a formal specification and especially from an EFSM [3]. In this work, we used the TESTGEN-SDL approach. For a given EFSM, $M = (I, O, S, x, T)$, each combination of a state in S and variable values of x consists of a configuration. The initial state $s(0)$ with the initial variable values $x(0)$ forms an initial configuration. On the other hand, we are only interested in the configurations that are reachable from the initial configuration, and they can be represented by a reachability graph as follows. It is costly to take the system to the initial configuration, whereas the test sequence length makes little difference. Specifically, a test is a path from $s(0)$ in the reachability graph. Assume that we are given a desired fault coverage, i.e., a set of system entities that we want to cover, such as transitions and variable values. We assign a distinct color to each such entity and we have a set C of k=|C| distinct colors. Each state and transition is associated with a subset of colors from C, denoted by $c(v) \subseteq C$ and $c(u, v) \subseteq C$, respectively. The function $c(.)$ designates the required coverage associated with a state or transition. For instance, if a transition t is assigned a distinct color c_t, which is to be covered. Then each edge (u, v), which is from the transition t, has $c_t \in c(u, v)$.

We are interested in a complete test set that covers all the colors. That is, the union of the color sets of the nodes and edges on the paths (tests) is the set of all colors C. Note that they are not necessarily the conventional covering paths that cover all the edges. Further, more details on TESTGEN-SDL are provided in [11].

3.3 Test case execution

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. 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 of Points of Control and Observation (PCO) that are connected with the Upper and Lower Testers (UT and LT) controlled by the Test Coordination Procedure (TCP). 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 [12] different conformance testing architectures. The main differences between the presented architectures rely on the position, communication and synchronization of the components.

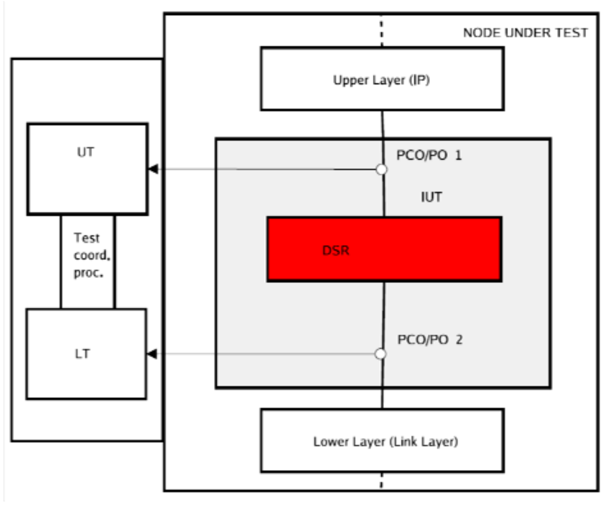

Fig. 1. Our testing architecture

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 communications 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.

Generated test suites from SDL models have been applied on several implementations under test specifically in wired networks. However, interesting results have also been obtained for wireless routing protocols rising at the same time novel issues [14]. Indeed, we performed applications of these above model based testing techniques for protocols in Mobile ad-hoc Networks (MANET). Many inconclusive verdicts were obtained compared to the PASS ones. The reasons were that the SDL specification was unable to consider topological changes due to packet losses, radio disconnections, etc. We therefore defined a novel approach based on nodes self similarity. This is what we describe in the next section.

4. 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.

4.1 Self similarity notions

The self similarity notion is presented and applied in [17] 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 $A0$ 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 2. Nodes combination.

Let $\{N_i\}_{i\in E}$ where $E \in [1\ldots n]$ and $n \in N$ be a collection of models that can be described as EFSMs. We note $N_1 \circ \ldots \circ N_n$ the combination of all N_i defined as:

$O(N) = \bigcup_{i\in E} O(N_i)$

$I(N) = \bigcup_{i\in E} I(N_i) - \bigcup_{i\in E} O(N_i)$

$S(N) = \prod_{i\in E} S(N_i)$

$x(N) = \prod_{i\in E} x(N_i)$

$T(N) = (s, s', e, o, P_i(x), A_i(x))$

if $(s_i, s'_i, e, o, P_i(x), A_i(x)) \in T(N_i)$ where $P_i(x) \equiv P_i(x_i)$, $A_i(x) \equiv A_i(x_i)$, $(e, o) \in I(N_i) \times O(N_i)$.

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 3. Nodes self similarity.

Let two possible actions for a node be *send*(*Message*, *n*, *m*) and *receive*(*Message*, *n*', *m*') where *n* (respectively *m*') is the observed node, *m* (respectively *n*') the *destination* of the packet (respectively *sender*), and *Message* is 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. Besides, $Tr(N)$ 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:

$\boldsymbol{Tr(ActHide_\Phi(N_1 \circ N_2)) \subseteq Tr(N)}$, where $\Phi = \{send(Message, N_1, N_2), send(Message, N_2, N_1), receive(Message, N_1, N_2), receive(Message, N_2, N_1)\}$.

Due to the inherent constraints, we use the self similarity considering:

1. The self similarity is applied from the viewpoint of a single node, 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 between the IUT and another node.

4.2 Nodes self similarity through MANETs

As the links in those kinds 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 viewpoint, three kinds 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, $i \in [1\ldots n]$, and D.

We consider the nodes in the route from the viewpoint 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 Fig. 2.

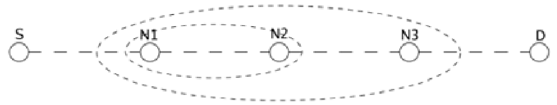

Fig. 2. Combination by self similarity when all communications succeed.

- 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* (Fig. 3).

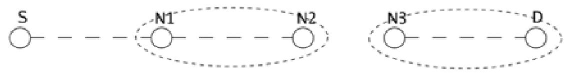

Fig. 3. Combination by self similarity when a communication fails.

With these definitions, the IUT behavior is not impacted by the path length when the test sequences are executed (except 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. Due to the lack of space, we herein do not detail this relationship but the interested reader may have a look to [15].

The node self-similarity enables to represent a large class of topologies with a small number of nodes and to execute test sequences regardless of the number of intermediate nodes. Thereby we can reduce the number of nodes used in our specification in order to generate test scenarios.

5. An experimental study

In order to generate more accurate and shorter test scenario, we minimize our specification. Due to node self-similarity, we can decide to keep only the smallest number of nodes required to generate a test sequence according to specific test objectives. To test functional properties of the DSR protocol (Dynamic Source Routing), we did not find, from the requirements, test objectives requiring more than 2 routes into the network. Then, our specification is reduced to 4 nodes, *S*, *N*0, *N*1 and *D* which compose 2 routes [*S*,*N*0,*D*] and [*S*,*N*1,*D*] as represented in Fig. 4.

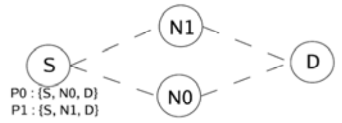

Fig. 4. Specification topology

This topology could represent a large class of real networks by node self-similarity and using two sets of routes during the execution of the test. Our main idea here is to create a relation between the specification and the implementation defined as:

Let S_{spec} and D_{spec} be respectively the representation of S and D in the specification $Spec$ and S_{imp}, D_{imp} their representation in the implementation Imp. Let $(p_n(x) \mid \{x, n\} \in \mathbb{N})$ be the n^{th} route chosen by S_{imp} to reach D_{imp} and composed by x nodes.

- In $Spec$:
  - $P0_{spec} = \{(S_{spec}, N0, D_{spec})\}$
  - $P1_{spec} = \{(S_{spec}, N1, D_{spec})\}$
- In Imp : $p_n(x) \in P(n \; mod \; 2)_{imp}$

All along the test execution, a *Test Coordination Procedure* (*TCP*) will preserve a relation between $P0_{spec}$ and $P0_{imp}$, and also between $P1_{spec}$ and $P1_{imp}$ assuming it as an elected strategy. Both sets save the theoretical RouteCache in the *TCP*. With respect to *Spec*, $P0_{imp}$ and $P1_{imp}$ match possible routes described in the specification. For instance, if a test sequence implies that *P*0*spec* disappears: the *TCP* will detect the RouteError packet as an input, will erase the first element of $P0_{imp}$, $p0(x)$ and will select $p1(y) \in P1_{imp}$ as the new route that *IUT* must use.

5.1. Dynamic Source Routing

Dynamic Source Routing (DSR) is a reactive protocol that discovers and maintains routes between nodes on demand [16]. 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.

5.2. Experiments

Once the *RouteDiscovery* has been performed for *S*, the IUT has a representation of the topology. From the viewpoint of S, the selected route can be simplified by self-similarity for each subsequent I/O as above explained. In order to insert the test scenario in the IUT, we use a UP and a LT linked to the TCP. In our example, a test scenario containing 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.

We detail hereafter the different steps of the execution in the network shown in Fig. 5. We decided to apply a direct emulation technique based on: (i) 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, (ii) 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), and (iii) virtual machines: virtual nodes (simulated ones) communicating with the focal machines.

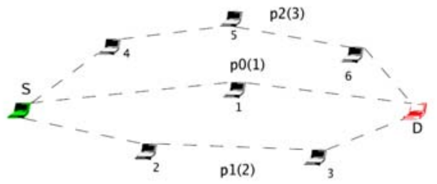

Fig. 5. An example of network.

A discrete events simulator such as NS-2 is sufficient and we will thus use NS-2e (NS-2 patched as emulator) [18]. Four virtual machines are executed on the focal machine that also runs the simulation side. Our approach is applied on experimentation through the DSR-UU implementation [19]. The test sequences are provided by one of our tools TESTGEN-SDL and some test purposes. Direct emulation is used. It allows using 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 [20] to create virtual machines with existing prepared kernel and file system. DSR-UU was added in the kernel. NS-2 patched for emulation was performed to manage mobility and wireless communication between the virtual machines. If we want a large 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 Fig. 6.

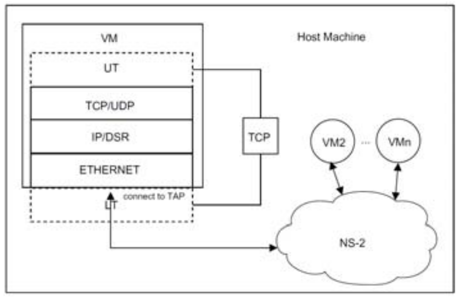

Fig. 6. Direct emulation and testing architecture.

We give next an example of the application of our approach through a testbed. This example illustrates the execution of a test scenario obtained from a formal specification on a real network taking into account the dynamic topologies. While we obtained ~95% of inconclusive testing verdicts without the nodes self similarity, we here reduce this rate to 5%.

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 p0(1) = (S, 1,D), p1(2) = (S, 2, 3,D) and p2(3) = (S,4,5,6,D) which are put in the sets P0imp et P1imp as defined by the ‘‘shortest path’’ strategy. A route pointer defines which route the TCP expects the IUT to use. This pointer indicates the first element of P0imp, p0(1) = (S,1,D). The sets P0imp and P1imp in Imp are equivalent to P0spec and P1spec for the test scenario.
8- LT observes an output SrcR(p0(1)) and p0(1) $\in$ P0imp – 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 inputRErr(1,D), TCP checks P0 et P1 and removes p0(1) from P0. The pointer moves on P1(1) = p1(2) = (S, 2, 3,D) – pass.
10- LT observes an outputSrcR(p1(2)) and p1(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, each expected I/O has been observed.

6. Conclusion

We have presented the node self similarity (NSS) approach adapted and applied in a MANET. This approach allows to reduce the formal specification of the protocol by considering functional similarity of the nodes in the network. By the same way, we reduced the number of test cases but particularly, we did face to the changing topologies and the dynamicity of certain nodes. This approach has been applied to an implementation of the DSR protocol. Compared to experiments without our NSS TCP strategy, we note that we reduced the number of inconclusive verdicts obtained from the execution of test suites.

References

[1]. NS2, The network simulator. Available from: <http://www.isi.edu/nsnam/ns>, 2004.
[2]. OPNet, The opnet modeler. <http://www.opnet.com/products/modeler/home.html>, 2005.
[3]. Rita Dorofeeva, Khaled El-Fakih, Stephane Maag, Ana R. Cavalli, Nina Yevtushenko, FSM-based Conformance Testing Methods: a Survey annotated with Experimental Evaluation, in Elsevier Information and Software Technology, Vol. 52, p.1286-1297, 2010.
[4]. L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, M. Gerla, Glomosim: a scalable network simulation environment, Technical Report 990027, UCLA Computer Science Department, May 1999.
[5]. Andel, Todd R., and Alec Yasinsac. On the credibility of manet simulations. Computer 39 (7) (2006): 48-54.
[6]. K. Bhargavan, C. Gunter, I. Lee, O. Sokolsky, M. Kim, D. Obradovic, M. Viswanathan, Verisim: formal analysis of network simulations, IEEE Transactions on Software Engineering, 28 (2) (2002) 129−145.
[7]. M. Kim, M. Viswanathan, H. Ben-Abdallah, S. Kannan, I. Lee, O. Sokolsky, Formally specified monitoring of temporal properties, in: Euromicro Conference on Real-Time Systems, 1999.
[8]. T. Lin, S.F. Midkiff, J.S. Park, A framework for wireless ad hoc routing protocols, Wireless Communications and Networking, 2, 2003 1162−1167.
[9]. U. Glasser, Q.-P. Gu, Formal description and analysis of a distributed location service for mobile ad hoc networks, Theoretical Computer Science, 2003
[10]. I. Zakkuidin, T. Hawkins, N. Moffat, Towards a game theoretic understanding of ad hoc routing, Electronic Notes in Theoretical Computer Science, 2005, 119.
[11]. Besse, Cédric, Ana Cavalli, and David Lee. An automatic and optimized test generation technique applying to TCP/IP protocol. 14th IEEE International Conference on Automated Software Engineering, 1999.
[12]. Willcock, Colin, et al. Multi Component TTCN-3. An Introduction to TTCN-3, Second Edition, 77-98, 2011.
[13]. Yu, Wenjing, et al. TDL: a transformation description language from feature model to use case for automated use case derivation. Proceedings of the 18th International Software Product Line Conference-Volume 1. ACM, 2014.
[14]. Koceilah Merouane, Cyril Grepet, Stephane Maag, A Methodology for Interoperability Testing of a MANET Routing Protocol , The Third IEEE International Conference on Wireless and Mobile Communications ICWMC 2007.
[15]. S. Maag, C. Grepet and A. Cavalli, A formal validation methodology for MANET routing protocols based on nodes' self similarity, Computer Communications Journal, Vol.31:4, pp. 827-841, 2008
[16]. D. Johnson, D. Maltz, Y.-C. Hu, The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR) − Experimental RFC, IETF MANET Working Group. July 2004.
[17]. Djouvas, C., Griffeth, N. D., & Lynch, N. A. Testing Self-Similar Networks. Electronic Notes in Theoretical Computer Science, 164 (4), 67-82, 2006.
[18]. NS-2 emulator. Available from: <http://www.isi.edu/nsnam/ns/ns-emulation.html>.
[19]. E. Nordstrom, Dsr-uu v0.1. Available from: <http://core.it.uu.se/core/index.php/DSR-UU>, Uppsala University
[20]. J. Dike, user-mode-linux. Available from: <http://user-mode-linux.sourceforge.net/>.

[21]. R.Hierons et al., Using formal specifications to support testing. ACM Computing Surveys, page 41(2):176, 2009.
[22]. ITU-T, Recommendation Z.100: CCITT Specification and Description Language (SDL), Technical Report ITU-T, 1999.
[23]. Daniel Hiranandani, Katia Obraczkaand J.J Garcia-Luna-Aceves, Manet protocol simulations considered harmful: The case for benchmarking. IEEE Wireless Communications, vol. 20, no 4, 2013.

Тестирование в беспроводных самоорганизующихся сетях на основе формальных моделей

Стефан Мааг <Stephane.Maag@telecom-sudparis.eu>
Institut Mines-Telecom/Telecom SudParis, CNRS UMR 5157,
9 rue Charles Fourier, 91011 Evry Cedex, France

Аннотация. Тестирование является одним из ключевых этапов разработки сложных взаимодействующих систем. Использование формальных моделей при тестировании позволяет автоматизировать процесс генерации тестовых последовательностей по формальному описанию спецификации и дальнейшего тестирования реальной системы. Несмотря на то, что тестирование на основе формальных моделей достаточно хорошо развито, использование этого подхода при тестировании протоколов маршрутизации в беспроводных самоорганизующихся сетях (ad-hoc) требует решения ряда специальных возникающих проблем, поскольку отличительной чертой беспроводных мобильных самоорганизующихся сетей (MANET) является отсутствие предопределенной инфраструктуры и отсутствие управляющих узлов. В данной работе, для формального описания спецификации используется модель расширенного конечного автомата, описанная в терминах языка SDL. Для уменьшения числа неопределенных вердиктов, часто возникающих в традиционном тестировании на основе моделей, рассматривается самоподобие узлов сети. Практическая значимость предложенного подхода иллюстрируется эмуляцией тестирования протокола DSR (Dynamic Source Routing).

Ключевые слова: тестирование на основе формальных моделей; мобильные самоорганизующиеся сети; самоподобие узлов; протоколы маршрутизации.

Список литературы

[1]. NS2, The network simulator. Available from: <http://www.isi.edu/nsnam/ns>, 2004.
[2]. OPNet, The opnet modeler. <http://www.opnet.com/products/modeler/home.html>, 2005.
[3]. Rita Dorofeeva, Khaled El-Fakih, Stephane Maag, Ana R. Cavalli, Nina Yevtushenko, FSM-based Conformance Testing Methods: a Survey annotated with Experimental Evaluation, in Elsevier Information and Software Technology, Vol. 52, p.1286-1297, 2010.
[4]. L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, M. Gerla, Glomosim: a scalable network simulation environment, Technical Report 990027, UCLA Computer Science Department, May 1999.
[5]. Andel, Todd R., and Alec Yasinsac. On the credibility of manet simulations. Computer 39 (7) (2006): 48-54.
[6]. K. Bhargavan, C. Gunter, I. Lee, O. Sokolsky, M. Kim, D. Obradovic, M. Viswanathan, Verisim: formal analysis of network simulations, IEEE Transactions on Software Engineering, 28 (2) (2002) 129−145.

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

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

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

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

[11]. Besse, Cédric, Ana Cavalli, and David Lee. An automatic and optimized test generation technique applying to TCP/IP protocol. 14th IEEE International Conference on Automated Software Engineering, 1999.

[12]. Willcock, Colin, et al. Multi Component TTCN-3. An Introduction to TTCN-3, Second Edition, 77-98, 2011.

[13]. Yu, Wenjing, et al. TDL: a transformation description language from feature model to use case for automated use case derivation. Proceedings of the 18th International Software Product Line Conference-Volume 1. ACM, 2014.

[14]. Koceilah Merouane, Cyril Grepet, Stephane Maag, A Methodology for Interoperability Testing of a MANET Routing Protocol , The Third IEEE International Conference on Wireless and Mobile Communications ICWMC 2007.

[15]. S. Maag, C. Grepet and A. Cavalli, A formal validation methodology for MANET routing protocols based on nodes' self similarity, Computer Communications Journal, Vol.31:4, pp. 827-841, 2008

[16]. D. Johnson, D. Maltz, Y.-C. Hu, The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR) − Experimental RFC, IETF MANET Working Group. July 2004.

[17]. Djouvas, C., Griffeth, N. D., & Lynch, N. A. Testing Self-Similar Networks. Electronic Notes in Theoretical Computer Science, 164 (4), 67-82, 2006.

[18]. NS-2 emulator. Available from: <http://www.isi.edu/nsnam/ns/ns-emulation.html>.

[19]. E. Nordstrom, Dsr-uu v0.1. Available from: <http://core.it.uu.se/core/index.php/DSR-UU>, Uppsala University

[20]. J. Dike, user-mode-linux. Available from: <http://user-mode-linux.sourceforge.net/>.

[21]. [21] R.Hierons et al., Using formal specifications to support testing. ACM Computing Surveys, page 41(2):176, 2009.

[22]. ITU-T, Recommendation Z.100: CCITT Specification and Description Language (SDL), Technical Report ITU-T, 1999.

[23]. Daniel Hiranandani, Katia Obraczkaand J.J Garcia-Luna-Aceves, Manet protocol simulations considered harmful: The case for benchmarking. IEEE Wireless Communications, vol. 20, no 4, 2013