Conformance Testing of Network Simulators Based on Metamorphic Testing Technique

Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu*, and Shengqiong Wang

Centre for Software Analysis and Testing, Swinburne University of Technology
`{tychen, dkuo, hliu, shengqiongwang}@swin.edu.au`

Abstract. Network simulators, which implement network protocols under some simulated conditions, have been widely used to analyze the feasibility of network protocols. Conformance testing of the simulator against the protocol is a very important task in the community of telecommunications. However, many current conformance testing methods face a problem of finding a systematic mechanism to verify the test outputs. This paper proposes to use an innovative testing approach, metamorphic testing (MT), to alleviate such a problem. We select one ad-hoc on-demand distance vector (AODV) simulator for study and test its conformance against the AODV protocol by the MT technique. Through our experiments, we illustrate the applicability of MT in the protocol conformance testing, confirm the reliability of the selected AODV simulator, and demonstrate the cost-effectiveness of MT using the mutation analysis technique.

1 Introduction

A network protocol specifies a set of mechanisms for exchanging messages among communication entities in a network system. A protocol must be feasible and its implementation must also conform to the protocol in order to deliver expected services in the network system [2]. Simulation is an important method for analyzing the feasibility of a protocol [7]. The network simulator, a protocol implementation under simulated network environments, is particularly useful to identify potential problems of the implemented protocol. It is important to ensure conformance between the simulator and the protocol.

Protocol conformance testing tests a protocol entity against the protocol specification. It aims to gain confidence in the correctness of the implementation with respect to a given specification [7]. International Organization for Standardization (ISO) has defined a framework and common terminologies for conformance testing of Open Systems Interconnection (OSI) systems [6]. Many approaches have been proposed to conduct conformance testing under various circumstances, such as unique input/output sequences generation method [7] and model-based approaches [13].

Many current conformance testing methods only work well when the network protocol can be modeled as a fully specified finite state machine (FSM). For such

* Corresponding author

protocols, we can always expect the correct output given any testing input [7]. For many other network protocols, which are not completely modeled by FSM, it is normally difficult to find a systematic verification mechanism for test results [8]. Such a verification mechanism is normally called the testing oracle [4] in the context of software testing. If there does not exist an testing oracle (known as the oracle problem in software testing), it is then very difficult to verify the correctness of the simulator's output.

Metamorphic testing (MT) [4] is an innovative testing method to alleviate the oracle problem. MT first identifies some properties from the specification of the software under test. These properties, which are known as metamorphic relations (MRs), are then used to generate some test cases. MT verifies the outputs of test cases based on MRs. Besides the alternative verification mechanism, MT has many other advantages. For example, it can be effectively applied by end users without too much knowledge of software testing. MT can also automatically generate a large number of test cases at a low cost, and MT test outputs can be verified by some simple script. MT have been successfully applied to detecting bugs in various programs [3, 9].

MT is basically a general technique used in the testing of software with any form of specification. In other words, no matter whether the software specification can be modeled by FSM or not, MT can always provide a mechanism to verify the test results. MT can identify some key properties from the specification, and generate test cases based on these properties. When a network protocol specification or a network simulator is updated, regression testing [12, 15] is always conducted to re-run the testing to ensure the correctness of the updates. Provided that the key properties identified by MT remain unchanged during the updates, all associated MT test cases can be re-used in the regression testing. In the paper, we attempt to apply MT into the conformance testing of a network simulator against its network protocol. A case study is conducted on an ad-hoc on-demand distance vector (AODV) simulator to illustrate the applicability and effectiveness of MT in conformance testing.

2 AODV protocol and its simulator

Ad-hoc on-demand distance vector (AODV) routing protocol [14] is a reactive protocol, that is, it establishes a route from a source node to a destination node only "on demand". AODV avoids the counting-to-infinity problem by using "sequence numbers" mechanism on route updates. It also uses a so-called "blacklist" mechanism to avoid invalid connection attempt. Concer [5] has developed an AODV simulator based on OMNeT++ [11], a discrete event simulation environment. One of us has worked on this simulator and has extensive domain knowledge of it, so we selected this specific simulator as the target program of our case study. The simulator depicts the AODV protocol with a number of nodes in a simulated field without obstacles. Each node in the field is comprised of five layers, namely, the application, network, date link, physical, and mobility layers. Inside a node, a higher layer (such as the application layer) consumes certain services offered by the lower layer (such as the network layer), but all layers is designed to be invisible to the implementation details of other layers.

Network simulators usually return some outputs relating to network protocol attributes, which are generally presented as the forms of network performance, such as, latency and throughput. Latency in a packet switched network is measured by the time from the source node sending a packet to the destination node receiving it, and throughput is the amount of digital data per time unit that pass through a certain node in the network. However, it is difficult to verify the correctness of these outputs, because the values of these outputs depend on various simulation environments, such as CPU and memory.

3 Metamorphic testing

Most software testing techniques (such as random testing and branch testing [10]) assume that the oracle exists. However, the oracle may not exist in some practical situations. For example, given a program for finding the shortest path in an undirected graph, when the graph is nontrivial, there is no oracle to effectively verify whether the returned outcome is really the shortest path between two nodes.

Metamorphic testing (MT) was proposed to test programs when oracle problem occurs [4]. MT requires domain knowledge to identify some important properties from the specification. These properties are called metamorphic relations (MRs). Some traditional testing techniques are first used to generate some *source test cases*. MRs are then applied to construct some *follow-up test cases* from source test cases. Both source and follow-up test cases are executed on the program under test. The test results are checked against MRs. If a relation is violated, a fault is said to be detected. For example, in the shortest path program, there is a permutation property: the program can produce the same outcome for a graph and the graph's permutation. Let a graph G be the source test case. We can generate G', a permutation of G, as the follow-up test case. The MR is that the program should produce the same output for G and G'.

4 Metamorphic testing on AODV simulator

As mentioned in Section 2, it is very difficult to verify the correctness of the test outputs of the AODV simulator. In this study, we attempt to use MT to test the conformance between the simulator and the AODV protocol. Our testing is mainly focused on two main outputs, latency and throughput, and two key mechanisms, the "sequence numbers" and "black-list" mechanisms. In the case study, in order to simplify the testing environment, we have modified the source codes in the application and mobility layers of the AODV simulator. Since the AODV protocol is implemented in the network layer, the modifications will not affect the conformance testing. The simplified testing environment we use in this study is a simulated network which contains a fixed number of nodes. Our testing is conducted mainly on a pair of randomly selected nodes (denoted by A and

B), which are randomly moving inside the network. We identify the following 11 MRs. Among them, MRs 1 to 7 have an additional prerequisite that there is always a connection between A and B; while MRs 1 to 5 and 8 to 10 further requires that the network's topology remains unchanged.

MR1: The source test case is that A sends a data packet P to B. The resultant latency and throughput are l_1 and r_1, respectively. The follow-up test case is that the locations of A and B are changed, and then A sends the same packet P to B. The resultant latency and throughput are l_2 and r_2, respectively. We should have the relations $l_2 \approx l_1$ and $r_2 \approx r_1$.

MR2: The source test case is that A sends a data packet P to B. The resultant latency and throughput are l_1 and r_1, respectively. The follow-up test case is that B sends the same data packet P to A. The resultant latency and throughput are l_2 and r_2, respectively. We should have the relations $l_2 \approx l_1$ and $r_2 \approx r_1$.

MR3: The source test case is that A sends a data packet P to B with channel delay c_1. The resultant latency is l_1. The follow-up test case is that A sends the same data packet P to B with a different channel delay c_2. The resultant latency is l_2. We should have the relation $\frac{l_1}{c_1} \approx \frac{l_2}{c_2}$.

MR4: The source test case is that A sends a data packet P_1 with packet size s_1 to B. The resultant throughput is r_1. The follow-up test case is that A sends a different data packet P_2 with packet size s_2 to B. The resultant throughput is r_2. We should have the relation $\frac{r_1}{s_1} \approx \frac{r_2}{s_2}$.

MR5: The source test case is that the routing table of A contains the route p_1 to B. The follow-up test case is that all route entries in A's routing table are deleted, and then A requests to identify a new route p_2 to B. We should have the relation $p_1 = p_2$.

MR6: The source test case is that A sends a data packet P to B via a route with h_1 hops. The resultant latency is l_1. The follow-up test case is that the route from A to B is changed to a new one with h_2 hops, and then A sends the same data packet P to B. The resultant latency is l_2. We should have the relation that if $h_1 > h_2$, then $l_1 > l_2$.

MR7: The source test case is that A sends a data packet P to B via a route with h_1 hops. The resultant sequence number is q_1. The follow-up test case is that the route from A to B is changed to a new one with h_2 hops, and then A sends the same data packet P to B. The resultant sequence number is q_2. We should have the relation that if $h_1 > h_2$, then $q_1 > q_2$.

MR8: The source test case is that A requests to transmit a data packet P to B (first transmission). The follow-up test case is that after a while, A requests to transmit the same packet P to B again (second transmission). We should have the relation (1) if the first transmission is successful, the second transmission should also be successful, or (2) if A only broadcasts a Route Request (RREQ) packet for searching a route to B but does not forward the packet at the first transmission, A should buffer the data packet at the second transmission.

MR9: The source test case is that A requests to transmit a data packet P to B (first transmission). The follow-up test case is that A's neighbor node C requests to transmit the same packet P to B (second transmission). We should have the relation (1) if the first transmission is successful, A should reply to C a Route Reply (RREP) packet at the second transmission, or (2) if A only broadcasts RREQ for searching a route to B but does not forward P at the first transmission, it should broadcast RREQ again for searching a route to B at the second transmission.

MR10: The source test case is that A's neighbor node C requests to transmit a data packet P to B (first transmission). The follow-up test case is that A requests to transmit the same packet P to B (second transmission). We should have the relation (1) if A replies to C an RREP packet at the first transmission, the second transmission should be successful, or (2) if A only broadcasts RREQ for searching a route to B and C does not forward the packet at the first transmission, A should broadcast RREQ again for searching a route to B at the second transmission.

MR11: The source test case is that A is put into the black list of B, and then A requests to transmit a data packet P to B. B will reply with a certain number (n_1) of RREP packets to A. The follow-up test case is that A is deleted from the black list of B, and then A requests to transmit the same packet P to B. B will reply with a certain number (n_2) of RREP packets to A. We should have the relation $n_1 < n_2$.

For each MR, we generated a certain number of source test cases by random testing technique [10], and at least one follow-up test case is generated based on the source test case and according to MR. In our experiment, we applied all these *MT test cases* to test the original version of the AODV simulator. In order to further investigate the effectiveness of MT in protocol conformance testing, we also used mutation analysis technique [1] to randomly seed some faults into the target program. We generated six mutants whose faults are related to key attributes of the simulator. All MT test cases were also applied to test these mutants. The experimental results showed that MT did not detect any fault in the original simulator, but the fault in each mutant has been revealed by at least one MR. With respect to the effectiveness of MT technique, we found that the success rates in detecting faults of our MRs and MT test cases are about 26% and 17%, respectively. Such results are very impressive in terms of the cost-effectiveness of a testing method.

5 Conclusion

Network simulator is an important tool for analyzing the network protocol. It is critical to ensure the conformance between the simulator and the protocol. However, protocol conformance testing is sometimes faced with an oracle problem, that is, there does not exist a systematic mechanism to verify the correctness of the test output given any possible program input. We are not aware of any systematic work dealing with the oracle problem in protocol conformance testing. In this paper, we proposed to apply metamorphic testing (MT), an innovative approach to alleviating the oracle problem, into the protocol conformance testing of network simulators. We selected ad-hoc on-demand distance vector (AODV) protocol and one of its simulators as our case study. Some key attributes are identified from the AODV protocol, and 11 metamorphic relations (MRs) are defined based on these attributes. We generated a large number of MT test cases based on these MRs, and checked the test results against these MRs. Our experimental results showed that the selected simulator conforms to the AODV protocol with respect to the chosen MRs and the used MT test cases. We also used MT to test some fault-seeded mutants of the simulator. The results of the mutation analysis showed that MT is very effective in detecting faults.

In this pilot study, we only conducted MT under a simplified testing environment. Some of our MRs may not be valid when the testing environment becomes more complicated. It is of great importance to identify more MRs that can be used in more general scenarios. It is also worthwhile to apply MT to test various applications of different network protocols.

Acknowledgment

This project is supported by an Australia Research Council grant (DP0771733). We are grateful to Chi Zhang for conducting some preliminary experiments.

References

1. J. H. Andrews, L. C. Briand, and Y. Labiche. Is mutation an appropriate tool for testing experiments? In *Proc. of ICSE2005*, 402–411, 2005.
2. T. P. Blumer, D. Sidhu, and A. Chung. Experience with formal methods in protocol development. *ACM Comput. Commun. Rev.*, 21(2):81–101, 1991.
3. T. Y. Chen, J. W. H. Ho, H. Liu, and X. Xie. An innovative approach for testing bioinformatics programs using metamorphic testing. *BMC Bioinform.*, 10:24, 2009.
4. T. Y. Chen, T. H. Tse, and Z. Q. Zhou. Fault-based testing without the need of oracles. *Inform. Softw. Tech.*, 45(1):1–9, 2003.
5. N. Concer. Ad-hoc network simulator. `http://www.omnetpp.org/filemgmt/singlefile.php?lid=87`, 2005.
6. ISO. Information technology - open systems interconnection - conformance testing methodology and framework. ISO/IEC 9646.
7. R. Lai. A survey of communication protocol testing. *Journ. Syst. Softw.*, 62(1):21–46, 2002.
8. P. D. L. Machado and W. L. Andrade. The oracle problem for testing against quantified properties. In *Proc. of QSIC2007*, 415–418, 2007.
9. C. Murphy, G. Kaiser, L. Hu, and L. Wu. Properties of machine learning applications for use in metamorphic testing. In *Proc. of SEKE2008*, 867–872, 2008.
10. G. J. Myers. *The Art of Software Testing.* John Wiley and Sons, 2004.
11. OMNeT Community. OMNeT++ system. `http://www.omnetpp.org`.
12. A. K. Onoma, W.-T. Tsai, M. H. Poonawala, and H. Suganuma. Regression testing in an industrial environment. *Commun. ACM*, 41(5):81–86, 1998.
13. A. M. Paradkar. Towards model-based generation of self-priming and self-checking conformance tests for interactive systems. In *Proc. of SAC2003*, 1110–1117, 2003.
14. C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector routing. RFC3561, 2008.
15. G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold. Prioritizing test cases for regression testing. *IEEE Trans. Softw. Eng.*, 27(10):929–948, 2001.