Testing of SDN Applications for Design Flaws and Implementation Bugs

Jiangyuan Yao[1], Zhiliang Wang[1], Xia Yin[2], Jun Bi[1], Xingang Shi[1], Jianping Wu[2], and Yahui Li[2]

[1] Institute for Network Sciences and Cyberspace, Tsinghua University
[2] Department of Computer Science and Technology, Tsinghua University

1 Introduction

Software-Defined Networking (SDN) is extremely hot in both academia and industry. The programmability of SDN challenges the reliability of SDN networks. The SDN applications play the role of SDN network' brains. Therefore, testing of SDN applications become a hot topic in research.

In the most studies about testing of SDN applications, researchers mainly focus on finding design flaws [1–4]. There may be both design flaws and implementation bugs in SDN applications. To expose implementation bugs, the black-box testing with formal model is a powerful candidate.

In this paper, we use a group of parallel state machines and a abstract topology to describe the SDN applications' behaviors and working environments. Based on this formal model, we propose a new test process combining the model checking and model-based testing. The former can reveal design flaws and the latter can expose implementation bugs.

2 Test Process

Fig.1 shows the overview of our test process. As centralized controller, the SDN application may collect and store information from the networks. We use a group of parallel state machines to specify the data structure and the applications' behaviors. However, we notice that the SDN applications may modify many nodes' configurations in the topology to change the functions of the networks. As a result, we add abstract topology into our formal model.

For design flaws, we use the model checking aided generation. First, we can extract some constraints from general network requirements. Then we employ model checking tools to check the formal model against the constraints. If the design flaws are found, we can get counter examples and translate them into executable test cases with our converter.

For implementation bugs, we propose the generation based on partial composition and topology. First, we simplify the topology with its symmetry. Second, we only select some related state machines for partial composition to alleviate the risk of state space explosion. Third, we can employ existing generation methods to derive test sequences from the composition. These test sequences are so

called single test sequences because they only contain the inputs/outputs of single node in the network. Consequently, we use the simulation execution on the topology to generate the extended test sequences from the single sequences.

Finally, we execute the test cases with the test system against SDN applications. Both design flaws and implementation bugs can be exposed.

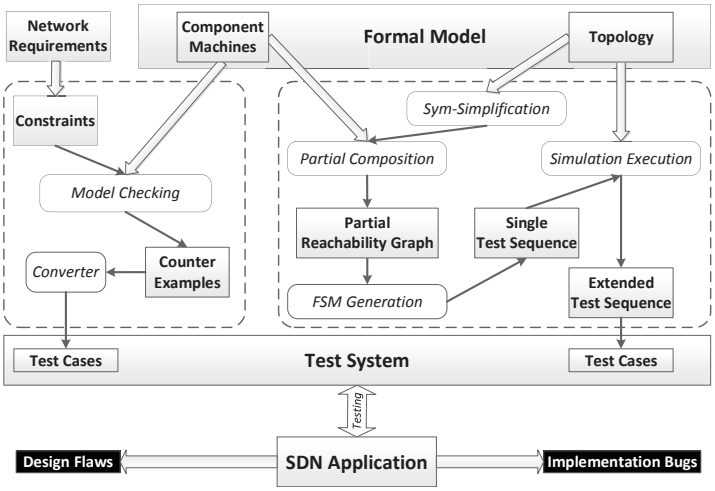

Fig. 1. Overall Test Process

Acknowledgment

This work is partially supported by the National High Technology Research and Development Program of China (863 Program, No. 2015AA016105), the National Natural Science Foundation of China (Grant No. 61202357), the Project for 2012 Next Generation Internet technology research and development, industrialization, and large scale commercial application of China (No. 2012 1763).

References

1. Canini, M., Venzano, D., Perešíni, P., Kostić, D., Rexford, J.: A nice way to test openflow applications. In: NSDI 2012
2. Scott, C., Wundsam, A., Raghavan, B., Panda, A., Or, A., Lai, J., Huang, E., Liu, Z., El-Hassany, A., Whitlock, S., Acharya, H., Zarifis, K., Shenker, S.: Troubleshooting blackbox sdn control software with minimal causal sequences. In: SIGCOMM 2014
3. Nelson, T., Ferguson, A.D., Scheer, M.J., Krishnamurthi, S.: Tierless programming and reasoning for software-defined networks. In: NSDI 2014
4. Ball, T., Bjørner, N., Gember, A., Itzhaky, S., Karbyshev, A., Sagiv, M., Schapira, M., Valadarsky, A.: Vericon: Towards verifying controller programs in software-defined networks. In: PLDI 2014