National Conference on 'Innovative Research on Robotics, Circuits and Technology' (IRCT 2018)
Organized By : CIrcuit Branches of SCSVMV, (EEE, ECE, EIE & Mechatronics) , Kanchipuram, Tamil Nadu, India
In Assotiation with International Journal of Scientific Research in Science, Engineering and Technology
© 2018 IJSRSET | Volume 5 | Issue 1 | | Print ISSN: 2395-1990 | Online ISSN : 2394-4099

Mathematical Modeling for Software Defined Networks

S. Chandramohan
Assistant Professor, Department of ECE, SCSVMV, Kanchipuram, Tamil Nadu, India

ABSTRACT

Software-defined networking (SDN) makes it possible to control an entire network in software, by writing programs that tailor network behavior to suit specific applications and environments. Unfortunately, developing correct SDN programs is easier said than done. SDN programmers today must deal with several complications. Our goal is to provide a mathematical foundation for software-defined networking that can be used to build and verify high-level SDN tools.

Keywords: SDN, Open flow architecture, COQ

I. INTRODUCTION

Two-tiered architecture

An SDN "program" has two distinct components: the controller program itself and the packet-processing rules installed on switches. These pieces have intricate dependencies that make reasoning difficult—e.g., installing or removing a rule can prevent the controller from receiving future network events. Hence, a programmer must reason about the behavior of the controller program, the rules on switches, and the interactions between the two via asynchronous messages.

Low-level operations

SDN platforms such as Open Flow force programmers to use a low-level API to express high-level intentions, which makes reasoning about SDN unnecessarily hard. Recent revisions of Open Flow expose even more hardware details, such as multiple typed tables, port groups, and vendor-specific features, which makes the problem worse.

II. EVENT REORDERING

Hardware switches employ a number of techniques to maximize performance, including reordering controller messages. This makes the semantics of SDN programs highly non-deterministic, further complicating reasoning. For example, in the absence of barriers, a switch may process messages from the controller in any order.

A programmer who uses these tools will be assured that certain specified formal guarantees will not be violated. To this end, we have developed a low-level model of SDN, called Featherweight OpenFlow. This model is based on the informal OpenFlow specification, but has a precise mathematical definition that makes it suitable for formal reasoning. We have implemented Featherweight OpenFlow in the COQ theorem prover as an executable artifact that can be used to build practical, high-level tools.

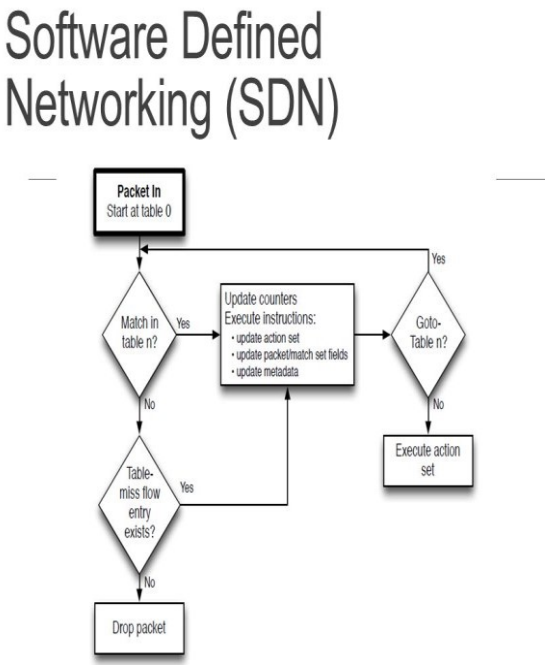

Figure 1. Flowchart of SDN

Our vision is a mathematical foundation for SDNs that enables and facilitates formal network reasoning. Recent advances in for-mal methods have made it possible to precisely model systems of realistic size. In particular, operational semantics have been used to model the behavior of complex systems such as the C programming language, x86 processors, and even whole operating systems.

We seek to develop detailed models of SDNs that support reasoning about essential network functionality such as forwarding, as well as complex features such as bandwidth, queues, controller resources, and failures. With these models, researchers can communicate their ideas concisely and unambiguously; developers of SDN controller platforms and tools can verify that their features are implemented correctly and users

III. SIMPLE CONTROLLER CORRECTNESS PRINCIPLES

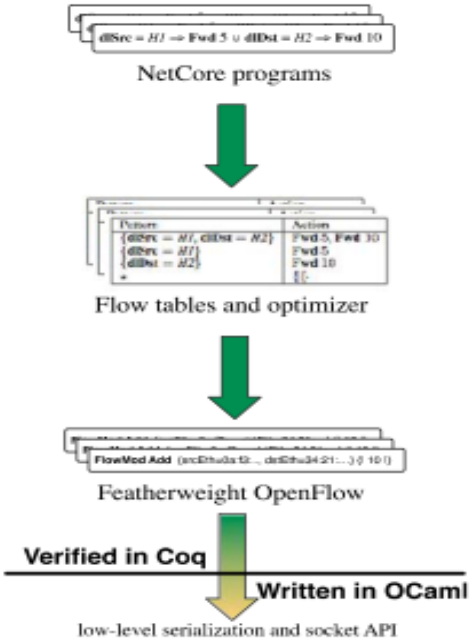

Figure 2. Controller Stack Diagram

Proving from scratch that a given controller correctly implements a given packet-processing function is a formidable task. Doing so requires reasoning about intricate details such as asynchrony in the network and the possibility of message reordering. We have developed a generic reasoning technique that dramatically simplifies the proof task. To verify a controller, it is only necessary to prove two natural properties: (i) the controller program must implement the packet-processing function, and (ii) each switch must approximate the packet-processing function and otherwise send packets to the controller. For most controllers, proving these properties is straightforward.

This result encapsulates a large amount of intricate reasoning about OpenFlow pro-grams and packages it up into a generic controller-correctness theorem. This is a powerful result: to establish correctness for a new controller, we do not have to start from scratch; we only have to prove two simple properties. Thus, controllers that use our technique can safely provide high-level abstractions to SDN applications.

IV. CONCLUSION

We hope that our SDN model will serve as a useful foundation for building other tools. For example, the model could be used as a test-oracle for OpenFlow switches, or as an engine for an OpenFlow software model-checker, in the style of NICE. The model could also be used to develop property-checking tools for high-level abstractions. We have built such a tool for NetCore based an encoding in first-order logic extended with fixed points.

V. REFERENCES

[1]. L. Girish and S. K. N. Rao, "Mathematical tools and methods for analysis of SDN: A comprehensive survey," 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, 2016, pp. 718-724.

[2]. M. Canini, D. Venzano, P. Peresını, D. Kostic, and J. Rexford. Automating the testing of OpenFlow applications. In ´ NSDI, 2012.

[3]. Guha, M. Reitblatt, and N. Foster. Machine-verified network controllers. In PLDI, 2013.

[4]. C. Monsanto, N. Foster, R. Harrison, and D. Walker. A compiler and run-time system for network programming languages. In POPL, 2012.

[5]. T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu,R. Ramanathan, Y. Iwata, H. Inoue, T. Hama et al., "Onix: a distributed control platform for large-scale production networks," in Proceedings of the 9th USENIX conference on Operating systems design and implementation, 2010, pp. 1-6.

[6]. A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford,G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4d approach to network control and management," SIGCOMM Comput. Commun.Rev., vol. 35, no. 5, pp. 41-54, Oct. 2005.

[7]. M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, "Design and implementation of a routing control platform," in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2, ser. NSDI'05,Berkeley, CA, USA, 2005, pp. 15-28.

[8]. M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," in Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications, ser. SIGCOMM '07, New York, NY, USA, 2007, pp. 1-12.

[9]. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69-74, Mar. 2008.

[10]. S. Azodolmolky, R. Nejabati, E. Escalona, R. Jayakumar, N. Efstathiou, and D. Simeonidou, "Integrated openflow-gmpls control plane: an overlay model for software defined packet over optical networks," Opt.Express, vol. 19, no. 26, pp. B421-B428, Dec 2011.

[11]. M. Channegowda, R. Nejabati, M. R. Fard, S. Peng, N. Amaya, G. Zervas, D. Simeonidou, R. Vilalta, R. Casellas, R. Mart´ınez, R. M.noz, L. Liu, T. Tsuritani, I. Morita, A. Autenrieth, J. Elbers, P. Kostecki, and P. Kaczmarek, "Experimental demonstration of an openflow based software-defined optical network employing packet, fixed and flexible dwdm grid technologies on an international multi-domain testbed," Opt.Express, vol. 21, no. 5, pp. 5487-5498, Mar 2013.

[12]. M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. TranGia,"Modeling and performance evaluation of an openflow architecture,"in Teletraffic Congress (ITC), 2011, Sept., pp. 1-7.

[13]. Y. Luo, P. Cascon, E. Murray, and J. Ortega, "Accelerating openflowswitching with network

processors," in Proceedings of the 5thACM/IEEE Symposium on Architectures for Networking and Communications Systems, ser. ANCS '09, New York, USA, 2009, pp. 70-71.

[14]. A. Bianco, R. Birke, L. Giraudo, and M. Palacin, "Openflow switching:Data plane performance," in Communications (ICC), 2010 IEEE International Conference on, May, pp. 1-5.

[15]. A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: scaling flow management for high-performance networks," in Proceedings of the ACM SIGCOMM 2011 conference, ser.SIGCOMM '11, New York, NY, USA, 2011, pp. 254-265.

[16]. 12] R. Pries, M. Jarschel, and S. Goll, "On the usability of openflow in data center environments," in Communications (ICC), 2012 IEEE International Conference on, June, pp. 5533-5537.

[17]. C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. Moore, "Oflops:An open framework for openflow switch evaluation," in Passive and Active Measurement, ser. Lecture Notes in Computer Science, N. Taft and F. Ricciato, Eds. Springer, 2012, vol. 7192, pp. 85-95.

[18]. Openflow controller performance comparison. Online]. Available:

[19]. http://www.openflow.org/wk/index.php/Controller Performance Comparisons

[20]. (last access 29 March 2013).