OFf: Bugspray for Openflow

[Extended Abstract]

Ramakrishnan Durairajan
University of Wisconsin
rkrish@cs.wisc.edu

Joel Sommers
Colgate University
jsommers@colgate.edu

Paul Barford
University of Wisconsin
pb@cs.wisc.edu

ABSTRACT

The increasing complexity of software-defined (SDN) applications requires comprehensive methods and tools for debugging and analyzing program and network behavior. A key challenge in SDN application development is that programs can interact with network devices and configurations in unexpected ways, depending on traffic and application mix. In this paper, we describe *OFf*, a debugging and test environment for SDN program development. *OFf* leverages the *fs-sdn* simulator, which was designed to offer simple-to-use, accurate, and scalable evaluation of OpenFlow-based SDN applications. *OFf* offers a variety of commonly available debugging features such as stepping, breakpoints, watchpoints, and inspection and modification of program state. It also offers SDN-specific capabilities that facilitate network behavior analysis including packet tracing and replay, visualization features, and alerts that are triggered when, for example, configurations change.

Categories and Subject Descriptors: C.2.3 [Network Operations]: Network Management; I.6.3 [Simulation and Modeling]: Applications

Keywords: Debugging; OpenFlow; Software-Defined Networks

1. INTRODUCTION AND MOTIVATION

Development of complex SDN configurations, like other software, requires tools and systems for facilitating debugging and program analysis. While standard debugging capabilities such as step-by-step execution, inspecting and modifying program state, and tracking changes to variables are often sufficient for ordinary host-based software, the distributed nature of SDN applications and configurations significantly complicates debugging and testing. Moreover, the fact that a particular SDN configuration meets a design specification may be inadequate for ensuring that it behaves in a predictable and stable manner when deployed in a live environment.

SDN deployments must handle a wide range of operating conditions, including the possibility of unanticipated traffic flows, and interactions with other deployed applications. Such unexpected conditions can lead to a variety of consequences including performance degradation, exposure of security vulnerabilities, and application failures. The potential severity and significance of these behaviors demands robust testing capabilities that go far beyond standard debugging and testing techniques, and includes the ability to assess configurations across a spectrum of operating conditions.

In this paper, we describe *OFf* [1], a debugging and test environment for SDN applications. *OFf* is designed to support debugging and testing controller applications by enabling standard debugging capabilities (*e.g.*, stepping, breakpoints, and watch variables) as well as a set of advanced capabilities to provide visibility into network behavior. *OFf*'s SDN-specific capabilities enable comprehensive testing of SDN applications in a representative, controlled and repeatable fashion through key capabilities including packet tracing, packet replay and visualization features, and alerts. *OFf*'s unique capability to simultaneously trace program execution and network state enables unwanted behavior in the network to be associated with the control program. *OFf* is designed to be used via a simple command line interface and support is enabled by simply including a library when coding the application.

2. BACKGROUND

In this section we give a brief overview of the *fs-sdn* tool on which *OFf* is built, and describe prior studies that influence and inform the design and implementation of *OFf*.

2.1 *fs-sdn* overview

fs-sdn [1] is a simulation-based tool that is designed to facilitate prototyping and evaluating new SDN applications. It is based on the *fs* [2] tool that was designed to efficiently generate realistic network measurements such as flow records and SNMP-like counters for use in different types of networking studies. *fs*, and by extension *fs-sdn*, use discrete event simulation techniques to generate network measurements and simulate network conditions. Unlike other simulation-based systems, its core abstractions are based on network flows and as a result it achieves much better performance than packet-based simulators. *fs-sdn* extended the *fs* engine by transparently incorporating the POX [3] OpenFlow controller framework and API, including switch components that can be controlled and configured through the OpenFlow control protocol.

2.2 Related work

OFf is most closely related to research on tools to expose and trace program and network state in SDN settings. In particular, ndb [4] and its successor NetSight [5] offer some similar features as *OFf*. A key difference, however, is that *OFf* offers capabilities not only to trace network state, but also to trace controller program

[1]Source code for OFf is openly available to the community and can be found at: `https://github.com/52-41-4d/fs-master`

HotSDN'14, August 22, 2014, Chicago, IL, USA.
ACM 978-1-4503-2989-7/14/08.
http://dx.doi.org/10.1145/2620728.2620776.

execution state, thus tying together observed network behavior with the control program that induced that behavior. *OFf* is also related to OFRewind [6], which enables replaying packets collected in an SDN setting in order to understand the effect of different control programs on traffic flows. Less directly related to *OFf* are efforts to verify that certain invariants hold. In particular, the Anteater system [7], Veriflow [8], and NICE [9] each seek to verify that certain network properties are never violated.

3. OVERVIEW OF OFf

In this section we provide an overview of *OFf*, including its architecture and features provided.

3.1 OFf Overview

OFf is a comprehensive source-level debugger that allows a developer to debug SDN applications. *OFf* does not require any special effort from the developer before debugging an application: the library can simply be included when coding the application like any other standard library. *OFf* also does not require any additional hardware and does not affect the program execution unless the developer issues a debugging command. Therefore, OFf can be used only when needed during the application development.

3.2 OFf Architecture

OFf consists of two parts: the *OFf Debugging Unit* and *OFf interfaces* that connect to the *fs-sdn* simulator and the SDN controller platform. We describe each of these below.

3.2.1 OFf Debugging Unit

The debugging unit is composed of four components. First, a *UI wrapper* provides a text-based interface that dispatches commands from the developer to one of the three other units and prints any output to a display.

Second, the *Debugger* component provides an abstract interface to a language-level debugger, running it as a child process. In our prototype, it runs the Python PDB [10] debugger, building onto and extending its features. It contains separate modules (with enhanced features) to deal with all specialized *OFf* commands such as enable and disable watch points, tracking variables, etc., that are not recognized by PDB. It also adds many features to the basic PDB command set by providing the ability to *(i)* longlist and shortlist source code during debugging, *(ii)* pretty print expressions, *(iii)* hide and unhide hidden code frames during debugging, *(iv)* interactive interpreter with all variables in scope, *(v)* track, watch, or unwatch variables, *(vi)* edit source files during debugging, *(vii)* enable or disable break points on the fly, and *(viii)* sticky mode to visualize code during debugging session.

The third component—*Trace Replay*—has the ability to reproduce network activity that has been captured in a trace and replay it later. Finally, the *Diff Report Generator* component helps detect changes in topology, mutations in rules/actions across switches, and performance variations from previous runs (or across configuration changes) of *fs-sdn*, then generates a report to help assess implications of configuration changes.

3.2.2 OFf Interfaces

The *OFf* Debugging Unit described above can be linked to the development and debugging environment for applications by adding one line of code to the controller application in the same way that any other standard library would be included. The debugging library can be included in *fs-sdn* or in a controller module to help debug either one.

Our current implementation of *OFf* is targeted towards debugging applications written in Python, specifically for the POX controller. However, *OFf* (as well as *fs-sdn*) is not limited to this software platform. The *debugger* component can be amended with a GDB [11] wrapper and support for NOX controller-based applications can be added. A GDB wrapper would also enable OFf to support multiple programming languages and hardware platforms, such as Java-based controllers and applications, as a virtue of multi-language/multi-platform support provided by GDB.

4. SUMMARY

Ensuring that SDN configurations behave as expected is predicated on careful and comprehensive debugging and testing. In this work, we describe *OFf*, an SDN debugging and testing tool that provides standard debugging capabilities such as stepping and watch variables, as well as SDN-specific capabilities to assess details of network interactions and changes over iterations of the same program. *OFf* is built on top of *fs-sdn*, which provides accurate and scalable simulation of OpenFlow-based SDN configurations. *OFf* is openly available to the community and development of additional features and capabilities is ongoing. Specifically, we intend to examine commonly reported SDN-related bugs and misconfigurations to understand better how to best design features in *OFf* to facilitate debugging.

Acknowledgments

This work was supported in part by NSF grants CNS-1054985, CNS-0831427, CNS-0905186, ARL/ARO grant W911NF1110227 and the DHS PREDICT Project. Any opinions, findings, conclusions or other recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF, ARO or DHS.

5. REFERENCES

[1] M. Gupta, J. Sommers, and P. Barford. Fast, Accurate Simulation for SDN Prototyping. In *Proceedings of ACM HotSDN*, 2013.

[2] J. Sommers, R. Bowden, B. Eriksson, P. Barford, M. Roughan, and N. Duffield. Efficient network-wide flow record generation. In *Proceedings of INFOCOM*, 2011.

[3] POX, Python-based OpenFlow Controller. `http://www.noxrepo.org/pox/about-pox/`.

[4] N. Handigol, B. Heller, V. Jeyakumar, D. Maziéres, and N. McKeown. Where is the Debugger for My Software-defined Network? In *Proceedings of ACM HotSDN*, 2012.

[5] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown. I Know What Your Packet Did Last Hop: Using Packet Histories to Troubleshoot Networks. In *Proceedings of the 11th USENIX NSDI*, 2014.

[6] A. Wundsam, D. Levin, S. Seetharaman, and A. Feldmann. OFRewind: Enabling Record and Replay Troubleshooting for Networks. In *Proceedings of the USENIX ATC*, 2011.

[7] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. B. Godfrey, and S. T. King. Debugging the Data Plane with Anteater. In *Proceedings of the ACM SIGCOMM Conference*, 2011.

[8] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey. VeriFlow: Verifying Network-wide Invariants in Real Time. In *Proceedings of the 10th USENIX NSDI*, 2013.

[9] M. Canini, D. Venzano, P. Perešíni, D. Kostić, and J. Rexford. A NICE Way to Test Openflow Applications. In *Proceedings of the 9th USENIX NSDI*, 2012.

[10] Pdb: The Python Debugger. `http://docs.python.org/2.7/library/pdb.html`.

[11] GDB: The GNU Project Debugger. `http://www.gnu.org/software/gdb/`.