VeriFlow: Verifying Network-Wide Invariants in Real Time (Extended Abstract)

Ahmed Khurshid, Xuan Zou, Wenxuan Zhou, Matthew Caesar, P. Brighten Godfrey
University of Illinois at Urbana-Champaign
{khurshi1, xuanzou2, wzhou10, caesar, pbg}@illinois.edu

Packet forwarding in modern networks is a complex process, involving codependent functions running on hundreds or thousands of devices, such as routers, switches, and firewalls from different vendors. As a result, a substantial amount of effort is required to ensure networks' correctness, security and fault tolerance. However, faults in the network state arise commonly in practice, including loops, suboptimal routing, black holes and access control violations that make services unavailable or prone to attacks (e.g., DDoS attacks). Software-Defined Networking (SDN) promises to ease the development of network applications through logically-centralized network programmability via an open interface to the data plane, but bugs are likely to remain problematic since the complexity of software will increase. Moreover, SDN allows multiple applications or even multiple users to program the same physical network simultaneously, potentially resulting in conflicting rules that alter the intended behavior of one or more applications [15].

One solution is to rigorously check network software or configuration for bugs prior to deployment. Symbolic execution [6] can catch bugs through exploration of all possible code paths, but is usually not tractable for large software. Analysis of configuration files [7, 17] is useful, but cannot find bugs in network software, and must be designed for specific configuration languages and control protocols. Another approach is to statically analyze snapshots of the network-wide data-plane state [4,5,10,13,16]. However, these previous approaches operate offline, and thus only find bugs after they happen.

Our work studies the question, *Is it possible to check network-wide correctness in real time as the network evolves?* If we can check each change to forwarding behavior before it takes effect, we can raise alarms immediately, and even prevent bugs by blocking changes that violate important invariants. For example, we could prohibit changes that violate access control policies or cause forwarding loops. However, existing techniques for checking networks are inadequate for this purpose as they operate on timescales of seconds to hours [5, 10, 13].

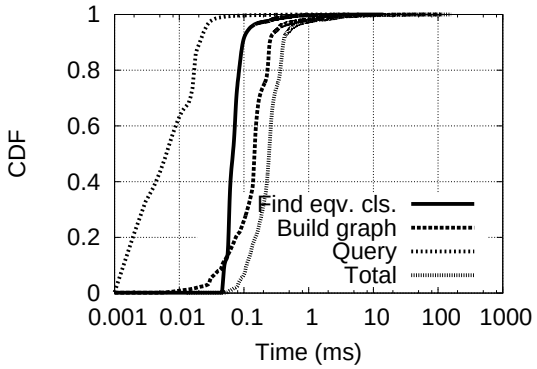

Figure 1: Microbenchmarks using a realistic trace of forwarding table updates show that VeriFlow's total verification time remains below 1 millisecond (ms) for 97.8% of the updates.

We present a design, VeriFlow, which demonstrates that the goal of real-time verification is achievable. VeriFlow provides an efficient technique for reasoning about network-wide properties using a low-level view of the network (the data plane), as close as possible to the network's actual running behavior. This technique, which we call *data plane verification*, allows VeriFlow to catch bugs that other tools miss, and provides a framework for a unified analysis of heterogeneous, multi-protocol networks.

VeriFlow leverages SDN to obtain a picture of the network as it evolves by sitting as a layer between the SDN controller and the forwarding devices, and checks validity of invariants as each rule is inserted, modified or deleted. However, SDN alone does not make the problem easy. In order to ensure real-time response, VeriFlow introduces novel incremental algorithms. First, we slice the network into a set of *equivalence classes*

(ECs) of packets based on the new rule and the existing rules that overlap with the new rule. Packets belonging to an EC experience the same forwarding actions throughout the network. Here, we find the set of ECs whose operation could be altered by a rule, and verify network invariants only within those classes. Second, VeriFlow builds individual *forwarding graphs* for every EC found in the previous step using the current network state. Third, VeriFlow traverses these graphs to *query* the status of one or more invariants.

To evaluate performance of our design, we constructed a prototype implementation. Our implementation of VeriFlow supports checking of both OpenFlow [14] version 1.1.0 and IP forwarding rules, with the exception that the current implementation does not support actions that modify packet headers. We microbenchmarked VeriFlow using a stream of updates from a simulated IP network, constructed with Rocketfuel [2] topology data and real BGP traces collected from Route Views [3]. We also evaluated its overhead relative to the NOX controller [8] in an emulated OpenFlow network using Mininet [1]. We find that VeriFlow is able to verify network-wide invariants within hundreds of microseconds as new rules are introduced into the network (Figure 1). VeriFlow's verification phase has little impact on network performance, and inflates TCP connection setup latency by a manageable amount, around 15.5% on average. In summary, our key contribution is to present the first tool that can check network-wide invariants in real time.

We believe the real-time nature of our system, coupled with its ability to provide strong guarantees on the ability to detect and localize complex faults, may enable it to become a standard best practice for future network operations, and an essential tool for highly dependable networks. Our system also provides a general platform for reasoning about the behavior of network protocols, which we believe will make it a valuable tool for building and testing network monitoring and management applications.

This paper summarizes published results in HotSDN 2012 [11] and NSDI 2013 [12]. We note that recently, another group has developed a real-time verification tool [9].

References

[1] Mininet: Rapid prototyping for software defined networks. `http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet`.

[2] Rocketfuel: An ISP topology mapping engine. `http://www.cs.washington.edu/research/networking/rocketfuel`.

[3] University of Oregon Route Views Project. `http://www.routeviews.org`.

[4] Al-Shaer, E., and Al-Haj, S. FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures. In *SafeConfig* (2010).

[5] Al-Shaer, E., Marrero, W., El-Atawy, A., and ElBadawi, K. Network configuration in a box: Towards end-to-end verification of network reachability and security. In *ICNP* (2009).

[6] Canini, M., Venzano, D., Peresini, P., Kostic, D., and Rexford, J. A NICE way to test OpenFlow applications. In *NSDI* (2012).

[7] Feamster, N., and Balakrishnan, H. Detecting BGP configuration faults with static analysis. In *NSDI* (2005).

[8] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. NOX: Towards an operating system for networks. In *SIGCOMM CCR* (2008).

[9] Kazemian, P., Chang, M., Zeng, H., Varghese, G., McKeown, N., and Whyte, S. Real time network policy checking using header space analysis. In *NSDI* (2013).

[10] Kazemian, P., Varghese, G., and McKeown, N. Header space analysis: Static checking for networks. In *NSDI* (2012).

[11] Khurshid, A., Zhou, W., Caesar, M., and Godfrey, P. B. VeriFlow: Verifying network-wide invariants in real time. In *HotSDN* (2012).

[12] Khurshid, A., Zou, X., Zhou, W., Caesar, M., and Godfrey, P. B. VeriFlow: Verifying network-wide invariants in real time. In *NSDI* (2013).

[13] Mai, H., Khurshid, A., Agarwal, R., Caesar, M., Godfrey, P. B., and King, S. T. Debugging the data plane with Anteater. In *SIGCOMM* (2011).

[14] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., and Shenker, S. OpenFlow: Enabling innovation in campus networks. In *SIGCOMM CCR* (2008).

[15] Sherwood, R., Gibb, G., Yap, K.-K., Appenzeller, G., Casado, M., McKeown, N., and Parulkar, G. Can the production network be the testbed? In *OSDI* (2010).

[16] Xie, G., Zhan, J., Maltz, D., Zhang, H., Greenberg, A., Hjalmtysson, G., and Rexford, J. On static reachability analysis of IP networks. In *INFOCOM* (2005).

[17] Yuan, L., Mai, J., Su, Z., Chen, H., Chuah, C.-N., and Mohapatra, P. FIREMAN: A toolkit for firewall modeling and analysis. In *SnP* (2006).