Performance Verification of Programmable Data Planes at the End-Host

Mina Tahmasbi Arashloo, Cornell University

Today's data-center networks operate at an unprecedented scale. They connect hundreds of thousands of servers with thousands of switches and routers and host a diverse set of application with a wide-range of network requirements. As such, designing, configuring, and operating networks have become increasingly complex and error-prone. As a result, several recent efforts in academia and industry have focused on using formal methods to help verify various network properties [2–5]. These works have successfuly demonstrated the effectiveness of formal methods in verifying the *functional correctness* of network functionalities such as ACLs, BGP, and DNS configurations. With the emergance of programmable data planes and the advances in the performance of SAT and SMT solvers, it is now possible to take the next step towards *performance verification*, i.e., automatic analysis of the throughput, latency, and jitter guarantees of a network for certain workloads. We argue that the first step along this path is verifying the performance of network processing at the end hosts.

The rise of high-speed reconfigurable data paths. To keep up with increasing line rate, end-hosts (servers) in today's data centers are turning into a collection of high-speed reconfigurable data paths: Linux express data path (XDP), programmed using eBPF, is a high-performance software data path that can process packets right after the NIC and before they are processed by the network stack. Moreover, modern NICs themselves have reconfigurable hardware components such as FPGAs that can be programmed using hardware description languages (e.g., Verilog and VHDL). Unlike programmable switching ASICs, these data paths allow direct and low-level programatic control over their underlying hardware. Thus, programs written for these data paths can provide us with enough visibility and details to analyze and reason about their performance.

High-level programming languages. While reconfigurable, directly programming the above data paths can be quite challenging. As such, network operators typically use higher-level languages [1, 6, 7] to describe their desired packet processing functionality, from congestion control, to scheduling, and to network functions such as NATs and firewalls. These programming languages are simple and domain-specific but detailed enough to allow for efficient reasoning about some of their performance properties. For instance, one could verify whether the specified congestion control algorithm provides per-flow throughput guarantees under certain network conditions, or ask whether the specified scheduling algorithm can cause starvation for any flows, or analyze if the specified network function performs too many memory accesses for some packets of a flow and not others, causing jitter.

Moreover, these high-level programs are automatically mapped to low-level programs for the target data path by a compiler, which performs a series of transformations on the input program over multiple passes. Thus, one can design a compiler with passes that verifably preserve the performance guarantees of their input programs at every step, either precisely or with a well-defined approximation. That way, not only network operators can write and modify high-level programs until they satisfy their required performance guarantees, but they can also reason about the performance of the final low-level program that is executed on the data path.

A stepping stone for end-to-end performance verification. This abstract focuses on verifying the performance of packet processing on a single server, which is only one of the many devices in a network that performs packet processing. However, with multiple reconfigurable data paths and complex network functions that switch traffic between virtual machines, today's data-center servers are starting to bear a lot of similarty to a small network. As such, they are an ideal starting point for performance verification, as the principles developed along the way can pave the way for verifying the performance of the entire network.

References

[1] P4 Language Consortium. p4.org.
[2] R. Beckett et al. A General Approach to Network Configuration Verification. In SIGCOMM'17.
[3] R. Beckett et al. Don't Mind the Gap:Bridging Network-Wide Objectives and Device-Level Configurations. In SIGCOMM'16.
[4] K. Jayaraman et al. Validating Datacenters at Scale. In SIGCOMM'19.
[5] S. K. R. Kakarla et al. GRooT: Proactive Verification of DNS Configurations. In SIGCOMM'20.
[6] A. Narayan et al. Restructuring Endpoint Congestion Control. In SIGCOMM'18.
[7] B. Stephens, A. Akella, and M. Swift. Loom: Flexible and Efficient NIC Packet Scheduling. In NSDI'19.