High speed packet forwarding compiled from protocol independent data plane specifications

Sándor Laki, Dániel Horpácsi, Péter Vörös, Róbert Kitlei, Dániel Leskó, Máté Tejfel
Faculty of Informatics, Eötvös Loránd University
Budapest, Hungary
{lakis, daniel-h, vopraai, kitlei, ldani, matej}@elte.hu

ABSTRACT

P4 is a high level language for programming network switches that allows for great flexibility in the description of packet structure and processing, independent of the specifics of the underlying hardware. In this demo, we present our prototype P4 compiler in which the hardware independent and hardware specific functionalities are separated. We have identified the requisites of the latter, which form the interface of our target specific Hardware Abstraction Library (HAL); the compiler turns P4 code into a target independent core program that is linked to this library and invokes its operations. The two stage separation improves portability: to support a new architecture, only the hardware dependent library has to be implemented. In the demo, we demonstrate the flexibility of our compiler with a HAL for Intel DPDK, and show the packet processing and forwarding performance of compiled switches in different scenarios.

CCS Concepts

•**Networks** → **Intermediate nodes;** *Packet-switching networks;* •**Hardware** → **Emerging languages and compilers;** •**Software and its engineering** → *Compilers;*

Keywords

Packet forwarding; SDN; Programmable data plane; P4

1. INTRODUCTION

With the advent of Software Defined Networking (SDN), a new era started in computer networking, providing network operators with a programmatic control over their networks. SDN separates control and data plane functionalities in the sense that a control plane programs multiple forwarding devices through a common, vendor-agnostic interface like OpenFlow [5]. Though OpenFlow and other solutions provide high flexibility in the control plane, the data plane is restricted to a subset of existing protocol headers, inhibiting the introduction of new protocols. Besides different frameworks like Packet Framework [3] and Netmap [6], Domain Specific Languages such as P4 [4, 2] have been proposed to overcome this limitation by enabling to describe packet forwarding in an abstract, protocol independent way.

Compared to OpenFlow, P4 provides a higher level of abstraction for programming the network. In its abstract model, switches parse incoming packets via a programmable parser, and then apply match-action rules in multiple stages arranged in sequence, parallel or a combination of both, where actions are composed of protocol-independent primitives.

In this demo, we present our prototype P4 compiler that, from a P4 program, generates high performance C code utilizing Intel DPDK [1], and we show the packet forwarding performance of the resulting switch.

SIGCOMM'16, August 22–26, 2016, Florianopolis, Brazil
© 2016 ACM. ISBN 978-1-4503-4193-6/16/08...$15.00
DOI: http://dx.doi.org/10.1145/2934872.2959080

2. OUR P4 COMPILER

Our aim was to develop a high-performance P4 compiler with flexible retargetability. To this end, we have identified the essential components of the compiler, and split them into two categories: *hw-dependent* and *hw-independent*. Accordingly, the hardware dependent functionalities are defined by a Hardware Abstraction Library (HAL) that has to be implemented for each target, while the core compiler remains independent of the actual hardware. The core code and the actual target are interconnected through the HAL. Note that our code generator reuses the High Level Intermediate Representation (HLIR) parser of the reference P4 compiler [2]. Currently, the HAL is only implemented for Intel platform and thus our prototype compiler produces Intel DPDK compatible C code, but we are working on the support of other targets including proprietary NPUs.

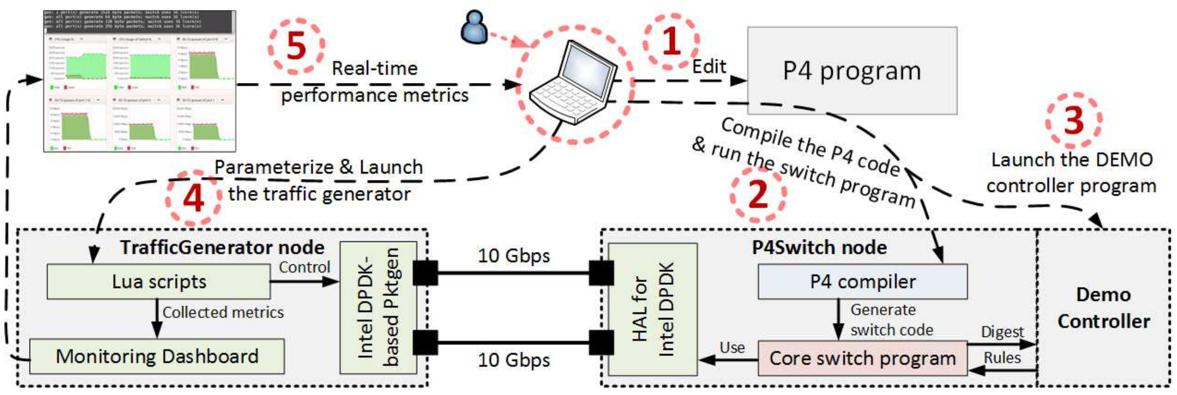

Figure 1: Testbed setup and the key steps of the demonstration showcase.

3. DEMO

In the demo, we present three different application scenarios, using the testbed depicted in Figure 1. It consists of two separate machines (Intel XEON E5-2630 4x8 cores 2.3GHz, 8x4GB DDR3 SDRAM): `TrafficGenerator` (TG node) and `P4Switch` (P4 node). Each one is equipped with a 1 Gbps NIC for management purposes and a dual 10 Gbps NIC (Intel 82599ES) for performance measurements. On both machines, the two 10 Gbps interfaces are used with Intel DPDK drivers. To generate test traffic, the DPDK's PktGen tool [1] is used with custom Lua scripts for parameterization and controlling the measurements on TG node. The collected performance metrics including the observed TX/RX rates are visualized in a simple web-based dashboard. The P4 node is used to demonstrate our P4 compiler and execute the compiled switch program. For the demo, we prepared three P4 programs for scenarios both showing the flexibility of the P4 language and the packet forwarding performance of our solution.In the current setup, the controller is also executed on the P4 node as a separate program. To measure the performance under different conditions, probe traffic is generated by varying the parameters of PktGen.

The demo covers three application scenarios: 1) A simple L2 packet forwarding example with MAC learning feature is presented. The switch contains two lookup tables, one for maintaining the source MAC addresses seen and another for destination MAC address-based packet forwarding. 2) We also show a simple IPv4 forwarding example that reflects the simplified functionalities of a L3 router including the next hop selection based on destination IP address, TTL decrementation and the replacement of source and destination MAC addresses. 3) Finally, we present how an imaginary protocol can be implemented in P4 and show the compiled switch program.

4. PERFORMANCE EVALUATION

As mentioned earlier, the P4 language enables innovation beyond the control plane by significantly increasing the flexibility of the data plane. However, achieving high performance on top-of-the-line hardware via compilation from a P4 code is still a challenging task.

Using our HAL for Intel DPDK, we managed to reach 13.04 and 10.10 Mpps (million packets per second) with a packet size of 64 bytes in a single core setup for the L2 and L3 programs compiled from P4 codes, respectively. For the same test traffic, the standard `l2fwd` example of Intel DPDK results in 14.88 Mpps. Using two cores, the 10 Gbps NIC is saturated even with the minimal packet size.

During the demo,we showcase the steps needed to generate a switch program from a P4 code, and show how it performs under heavy load of network traffic. As depicted in Figure 1, the audience can access our testbed nodes, edit the P4 code, compile it and set the parameters of both the traffic generator and the control plane. More details on the demo and the proposed P4 compiler are available at http://p4.elte.hu/.

Acknowledgement – The authors thank the support of Ericsson Hungary Ltd.

5. REFERENCES

[1] Intel data plane development kit. http://dpdk.org/.

[2] P4 consortium. http://p4.org.

[3] Packet framework. http://dpdk.org/doc/guides/prog_guide/packet_framework.html.

[4] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, July 2014.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[6] L. Rizzo. Netmap: a novel framework for fast packet i/o. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 101–112, 2012.