P4 Language and Programmable Data Planes

Praveen Tammana
IIT Hyderabad, India
Email : praveent@cse.iith.ac.in

Rinku Shah
IIIT-Delhi, India
Email : rinku@iiitd.ac.in

Venkanna U
IIIT Naya Raipur, India
Email : venkannau@iiitnr.edu.in

I. Motivation

To support the ever-growing application demands for high bandwidth, low latency, and high availability, the underlying network architecture has gone through several changes. For instance, in the 5G context [1], we have access-edge networks providing services at the edge and the edges are connected over core networks to talk to their counterparts in the cloud. If we go a bit low-level, the infrastructure behind these novel architectures is also changing fundamentally. Increasingly, many network functionalities are disaggregated and the functions that were traditionally running on special-purpose proprietary hardware targets are now running on general-purpose commodity hardware such as CPUs, SmartNICs [2], [3], [4], [5], [6], and high-speed programmable whitebox switches [7].

Many network targets following Software-Defined Networking (SDN) [8] architecture separate the *control plane* (the brain of the entire network that determines how packets should be forwarded) and the *data plane* (process packets as instructed by the control plane), thus enabling faster innovation in each plane. Today, the SDN control plane software is open-sourced and the data plane hardware is programmable. This means, there is no fixed entity that dictates how the network works, instead it is up to the network owner/operator to define whatever s/he wants and design network protocols that can run on commodity hardware. There are mainly two types of data plane hardware: OpenFlow-based [9] and P4-based [10]. OpenFlow-based SDN followed a bottom-up approach assuming that the data plane (i.e., switches, smartNICs) has a fixed-function behavior supporting a small set of fixed protocols, usually defined in ASIC specs by network equipment vendors (e.g., Cisco, Juniper, Mellanox). In this bottom-up approach, adding a new protocol for measuring or controlling datapath takes years, which implies OpenFlow does not give control over network behavior at packet-level to network operators.

On the other hand, P4-based SDN followed a top-down approach and proposed to use protocol-independent programmable data planes [10] (e.g., smart NICs, Intel Barefoot Tofino). To be specific, P4 is a domain specific language using which a network operator tells switches which packet headers to be recognized and how to process packets, thus being able to implement new protocols (e.g., key-value lookup [11], ML reduce operation [12]), novel network functions (e.g., load balancing [13], NAT, DDoS detection [14]), and simplifies network management (e.g., Google's P4 integrated network stack [15]). To summarize, Software-Defined Networking, programmable data planes, and P4 language, all together have opened up a wide range of opportunities to solve network problems considered difficult and complex in traditional closed and fixed ASIC-based data planes.

II. Outline and topics covered

The objective of this tutorial is to introduce the audience to P4 language [10] and P4-related software tools [16] which can be used to program packet-processing data planes (e.g., eBPF [17], DPDK [18], smartNICs [2], switches [7]). The following topics would be covered:

1) **Introduction to P4 data planes:** This covers different architectures and targets available today, what constitutes a P4 dataplane (i.e., parsers, match-action tables, queues, etc), and then goes deep into two popular architectures: protocol independent switch architecture (PISA) and V1 model.
2) **P4 language basics:** This covers an overview on P4 language constructs, how to install a P4 program on a data plane target by developing two basic applications (basic forwarding, basic tunneling) and one advanced application (stateful firewall).
3) **P4 Runtime and the Control plane:** This covers the existing P4 software tools for controlling P4 data plane (e.g., loading P4 program, updating match-action table rules, etc).

III. Format of tutorial

1) Approximate duration of tutorial is 3 hours.
2) We will utilize 50% of allocated time for presentation, and the remaining 50% for hands-on training.

IV. Requirements for the attendees

- Software and reading material will be provided to participants in USBs for offline participants.
- Attendees will be provided with a Virtual Machine (VM) with pre-installed software (in USBs). They should copy the VM on their laptop/desktop for hands-on training.
- The minimum configuration requirement for the laptop/desktop to run the VM is 4 GB of RAM, 4 CPUs, and 16 GB of free disk space.

V. Indicate if the tutorial is suitable for hybrid in-person and online attendees

The tutorial is suitable for both hybrid in-person and online attendees.

VI. Intended audience of the tutorial

1) UG/PG/PhD students good with UG-level computer networking course.
2) Industry professionals in the networking industry.

VII. Biography of tutorial presenters including their experience with the tutorial topic

Praveen Tammana is currently an Assistant Professor in Computer Science at IIT-Hyderabad. He received his Ph.D. in computer science from The University of Edinburgh, UK. Prior, he was a postdoctoral researcher at Princeton University, USA. His research interests are at the intersection of Systems, Networks, and Security. His current focus is on designing and building networked systems that make networks easy to manage, secure, and robust, by using exciting emerging technologies such as Software-Defined Networking (SDN) and P4-based programmable data planes. Praveen has been primarily working in the SDN domain for the last 8 years, and published at top networked systems venues such as SOSR, NSDI, and OSDI.

Rinku Shah is currently an Assistant Professor in the CSE department at Indraprastha Institute of Information Technology, Delhi (IIITD). She has received her Ph.D. degree in computer science from IIT Bombay. Her research interests lie in the domain of networked systems. She has worked in the intersection domain of Software-Defined Networking (SDN), programmable data planes, and 4G/5G mobile network core. Her current focus is to design flexible, scalable, and fault-tolerant systems by leveraging evolving networking technologies such as SDN and programmable network hardware. Rinku has 5 years of research experience in the SDN domain and 13 years of teaching experience. Her research is published at reputed conference venues such as SOSR, ICNP, and APNet.

Venkanna U. obtained his Ph.D. degree from the National Institute of Technology, Tiruchirappalli (NITT), in 2015. Since 2005, he has been in the teaching profession, and currently, he is an Assistant Professor in the Department of Computer Science and Engineering, IIIT- Naya Raipur. He has 10 years of teaching and research experience. His research interests include Software Defined Networks, Programmable Networks, and Security issues in Internet of Things (IoT). He has the best paper award at IEEE-ANTS 2019 conference. Also, he is elevated as IEEE senior member in August 2021.

References

[1] 3GPP, "5g 3gpp specifications," https://www.3gpp.org/ftp/Specs/archive/23_series/23.502/.

[2] "Agilio CX SmartNIC," https://www.netronome.com/products/agilio-cx/.

[3] "Intel FPGA," https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ds/ds-pac-n3000.pdf.

[4] "NetFPGA SUME," https://netfpga.org/NetFPGA-SUME.html.

[5] "Pensando Distributed Services Card," https://pensando.io/wp-content/uploads/2020/03/Pensando-DSC-25-Product-Brief.pdf.

[6] "Nvidia Bluefield DPU," https://www.nvidia.com/en-in/networking/products/data-processing-unit/.

[7] "Intel Barefoot Tofino switch," https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series.html.

[8] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: An intellectual history of programmable networks," *SIGCOMM Computer Communication Review*, vol. 44, no. 2, p. 87–98, 2014.

[9] N. McKeown *et al.*, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, 2008.

[10] 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 Computer Communication Review*, vol. 44, 2014.

[11] X. Jin, X. Li, H. Zhang, R. Soulé, J. Lee, N. Foster, C. Kim, and I. Stoica, "Netcache: Balancing key-value stores with fast in-network caching," in *Proceedings of the SOSP*, 2017.

[12] A. Sapio, M. Canini, C.-Y. Ho, J. Nelson, P. Kalnis, C. Kim, A. Krishnamurthy, M. Moshref, D. R. Ports, and P. Richtárik, "Scaling distributed machine learning with in-network aggregation," *arXiv preprint arXiv:1903.06701*, 2019.

[13] R. Miao, H. Zeng, C. Kim, J. Lee, and M. Yu, "Silkroad: Making stateful layer-4 load balancing fast and cheap using switching asics," in *Proceedings of the the ACM SIGCOMM Conference*, 2017.

[14] T. Chown, "Telemetry and big data workshop," https://wiki.geant.org/display/PUB/Telemetry+and+Big+Data+Workshop?preview=/148094173/167772344/BD_Telemetry_workshop_sketch_analytics_v6.pdf, 2020.

[15] "P4 Integrated Network Stack (PINS)." https://opennetworking.org/pins/, 2021.

[16] "P4 learning," https://p4.org/learn/.

[17] "extended Berkeley Packet Filter (eBPF)." https://ebpf.io/.

[18] "Data Plane Development Kit (DPDK)." https://www.dpdk.org/.