Enabling P4 Hands-on Training in an Academic Cloud

Jose Gomez, Elie F. Kfoury, Jorge Crichigno
College of Engineering and Computing, University of South Carolina, Columbia, SC, USA.
{gomezgaj, ekfoury}@email.sc.edu, jcrichigno@cec.sc.edu

***Abstract*—This paper describes a cloud infrastructure and virtual laboratories on P4 programmable data plane switches. P4 programmable data planes emerged as a technology that enables innovation in networking. P4 is a programming language used to describe how network packets are processed. This paper explains an entry-level training library on P4. The virtual laboratories introduce the learner to P4 and data plane concepts by providing step-by-step guides and exercises. The virtual laboratories are hosted in the Academic Cloud, a distributed platform that manages and orchestrates computing resources. Additionally, the paper describes a work in progress of P4 virtual laboratories that uses Intel Tofino switches. Lastly, the paper discusses the use of the Academic Cloud as a network testbed.**

***Keywords*—P4; programmable data planes; virtual laboratories; academic cloud; virtual machines; network emulator.**

I. Introduction

The networking industry has been dominated by closed and proprietary solutions. This lack of flexibility caused by standardized requirements makes it difficult to create, modify, or change protocols, thus slowing down innovation [1]. In recent years, data plane programmability has attracted the attention of operators, engineers, and researchers due to its flexibility. In this context, programmable data planes surge as a natural evolution of Software-Defined Networking (SDN), where the software describes the packet processing behavior. Programming Protocol-independent Packet Processor (P4) [2] is the *de facto* programming language for data plane programming. P4 programmable switches have removed the entry barrier to network design, previously reserved to network vendors. With P4, the user can test and deploy novel protocols and applications in a much shorter time span [3]. Although P4 facilitates the design of customized protocols and applications, learning P4 can be challenging. The available open P4 training [4–6] requires the learner to have a background in virtualization (e.g., hypervisors), Linux, and emulation tools. Such requirements limit the audience to graduate students, researchers, and experienced developers.

This paper describes an entry-level training on P4 hosted in the Academic Cloud. The Academic Cloud is a virtual platform that manages and orchestrates computing resources to support virtual laboratories designed for teaching, training, and research. Virtual laboratories on P4 provide the learner with the computing resources and documentation that guide through the basic concepts of programmable data planes and P4. The learner interacts with P4 switches through hands-on experiences and acquires expertise to create, test, and deploy P4 applications. This paper also describes a work in progress consisting of virtual laboratories using physical switches (i.e., Intel Tofino). Finally, it discusses the future use of the Academic Cloud as a network testbed for research.

This work was supported by the National Science Foundation under award numbers 1925484 and 2118311.

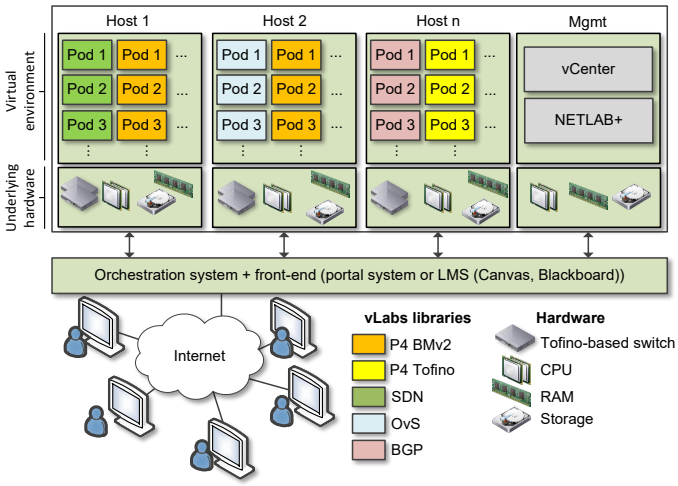

Fig. 1. Academic Cloud.

II. Academic Cloud and Virtual Laboratories

A. Academic Cloud Platform

The Academic Cloud is a distributed platform that provides the computing resources to support virtual laboratories. These computing resources are distributed across four data centers in the United States. The platform dynamically provides the resources that maximize the learner's experience in running a virtual laboratory. Hands-on experience is crucial when learning networking topics. The Academic Cloud platform has been deployed by the University of South Carolina, Stanly Community College, and the Network Development Group (NDG) [7] since January 2020. As of January 2022, it has served over 100,000 learners. Fig. 1 illustrates the Academic Cloud architecture at one of the four data centers. Servers are provisioned with a large number of resources: CPUs (32 to 40 cores per server), RAM (∼1TB), and storage (∼2.5 TB). On average, each data center has approximately 10 servers used to host virtual pods. A pod is a collection of resources

TABLE I. FEATURES OF VIRTUAL LABORATORIES.

Feature	Description
Performance	Virtual laboratories precisely emulate high-performance systems (e.g., high-speed networks running at 50 Gbps).
Functional realism	Virtual laboratories have the same functionality as real hardware in a real deployment, and execute the same code [9].
Traffic realism	Devices within the virtual environment are capable of generating and receiving real, interactive network traffic to and from the Internet, or from other devices within the virtual environment [9].
Presentation layer	Navigating through an experiment is easy for an inexperience learner. Devices within the virtual environment must be accessible by simply clicking on them.
Topology flexibility	It must be easy to create an experiment with any topology, including inter-connecting heterogeneous VMs.

(virtual machines (VMs), virtual switches, virtual links, physical Tofino-based switches, and others) orchestrated by the Academic Cloud platform to deliver the virtual laboratory experiments. Some virtual laboratories require more than one virtual machine to recreate an experiment. The Academic Cloud is implemented with the server virtualization software VMware vSphere [8]. The vSphere components include a collection of bare-metal hypervisors (ESXi), a management server (vCenter), and a VM running NETLAB, a customized management application developed by NDG. The Academic Cloud functions include aggregating resources from the four data centers, implementing a calendar interface for scheduled access to equipment pods, and routing pod reservations to the nearest data center containing the requested pod type.

B. Virtual Laboratories

A virtual laboratory comprises a pod of equipment allocated transparently to the learner. The learner interacts with the virtual laboratory using a web browser. Thus, the learner is not required to install additional software nor possess the computing resources to run the virtual laboratory. During the experiment, the initial configuration of the pod is provided so that the learner only focuses on the content. The main features of the virtual laboratories are summarized in Table I. The learner has access to self-paced training material in networking, virtualization, cybersecurity, Linux, SDN, FRRouting [10], P4 programmable switches, and others.

C. Access to the Academic Cloud from a Learner's Perspective

The system's primary goal is to facilitate the learning experience by providing a user-friendly environment. Fig. 2 shows the steps to reserve a virtual lab in the Academic Cloud. In step (a), the learner accesses the Academic Cloud through a web browser using his/her credentials to login into the platform. In step (b), the learner selects a lab from the library (e.g., Lab 5 from the Introduction to P4 Programmable Switches library). In step (c), the learner enters the lab and has access to the devices in the topology. Then, the learner clicks on a device (e.g., Client) to access the graphical user interface (GUI). In step (d), the learner interacts with the device by issuing the set of commands described in the laboratory manual.

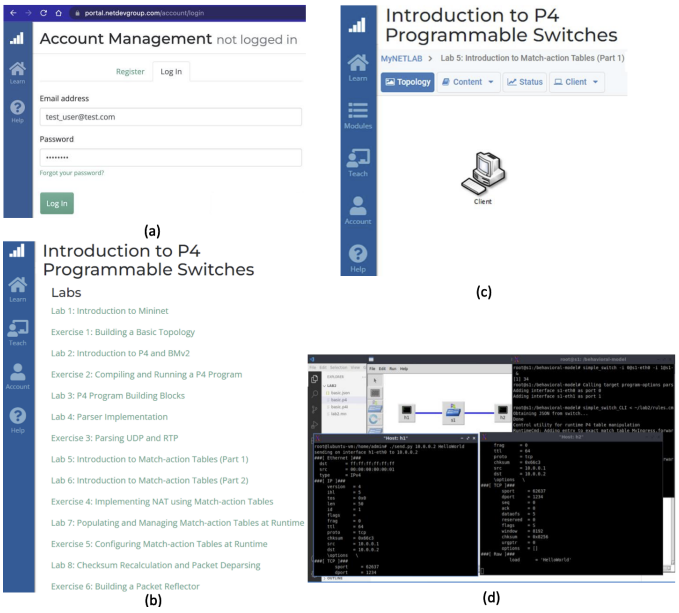

Fig. 2. Accessing the Academic Cloud. (a) A learner enters the system. (b) The learner reserves a virtual laboratory to conduct an experiment. (c) The learner enters the lab, and a scenario is recreated. The scenario consists of a pod of virtual devices. In this example, the user clicked on the "Client" device, which opened a window. (d) The user operates the device via a GUI.

III. VIRTUAL LABORATORIES ON P4 PROGRAMMABLE DATA PLANE SWITCHES

A. P4 Library

The P4 virtual laboratory library developed at the University of South Carolina covers the basic principles of P4 programmable data planes using the Behavioral Model version 2 (BMv2) software switch [11]. The library has eight guided laboratories and six exercises summarized in Table II. The goal is to introduce the learner to programmable data plane switches and the P4 language. The pod uses Containernet [12], a Mininet [13] fork that uses Docker containers [14] and can integrate tools such as Open vSwitch (OvS) [15], SDN controllers, FRR [10], and others. Within the virtual environment, the learner can create complex topologies using MiniEdit, a GUI that facilitates the configuration of elements (routers, switches, containers, and other appliances), save topologies, and run emulations.

The first laboratory introduces the learner to network elements available in Mininet (e.g., hosts, links, legacy switches, P4 switches, and routers) to run an experiment. Lab 2 covers writing, compiling, and loading a P4 program into a switch. Lab 3 introduces the P4 building blocks, which represent the P4 programming abstraction implemented in the software switch [11]. In lab 4, the learner explores the parser's functionalities by defining and processing packet headers. Labs 5 and 6 cover match-action tables using exact and longest prefix match (LPM). Lab 7 navigates through the control plane interface that provides functions to interact with the data plane. Lab 8 explains how to recompute the packet's checksum and emit a modified header using the deparser. Additionally, the library includes exercises to test the learner's knowledge. The learner

TABLE II. DESCRIPTION OF THE VIRTUAL LABORATORIES

Labs. and Exercises	Description
Lab 1: Introduction to Mininet	This lab provides an introduction to Mininet, a network emulator for testing network tools and protocols. It demonstrates how to emulate network topologies using the CLI and GUI.
Exercise 1: Building a Basic Topology	In this exercise the user will build a topology in Mininet and perform a connectivity test.
Lab 2: Introduction to P4 and BMv2	The lab introduces the P4 language and provides a high-level overview of the general lifecycle of programming, compiling, and running a P4 program on a software switch.
Exercise 2: Compiling and Running a P4 Program	This exercise requires the learner to navigate through the P4 programming lifecycle.
Lab 3: P4 Program Building Blocks	This lab describes the building blocks and the general structure of a P4 program. The lab demonstrates how to track an incoming packet as it traverses the switch's pipeline using the switch's logs.
Lab 4: Parser Implementation	This exercise describes how to define custom headers in a P4 program. Then, it explains how to implement a simple parser and shows how to track the parsing states of a packet inside the switch.
Exercise 3: Parsing UDP and RTP	In this exercise, the learner will define and parse UDP and RTP headers. Then, the user will verify the switch's logs to develop debugging skills.
Lab 5: Introduction to Match-action Tables (Part 1)	This lab describes match-action tables and how to define them in P4. Then, the lab explains how to implement a table using exact match.
Lab 6: Introduction to Match-action Tables (Part 2)	This lab describes match-action tables and how to define them in a P4 program. Then, the lab explains how to implement a table using longest prefix match (LPM).
Exercise 4: Implementing NAT using Match-action Tables	This exercise requires the learner to implement Network Address Translation (NAT) with match-action tables.
Lab 7: Populating and Managing Match-action Tables at Runtime	This lab describes how to populate and manage match-action tables at runtime. Then, it explains how to use the simple_switch_CLI tool to interact with the data plane.
Exercise 5: Configuring Match-action Tables at Runtime	In this exercise, the learner will populate the entries of the match-action tables at runtime using the switch's simple_switch_CLI tool.
Lab 8: Checksum Recalculation and Packet Deparsing	This lab describes how to recompute the checksum of an IPv4 header. The lab also explains how a P4 program performs deparsing.
Exercise 6: Building a Packet Reflector	This exercise requires the learner to implement a packet reflector by programming all the P4 pipeline components (i.e., headers, parser, ingress block, egress block, deparser, and checksum computation).

can request access to the P4 library by filling out the form available on the website of the Cyberinfrastructure lab [16].

B. Future Virtual Laboratories with P4 and BMv2

Currently, more advanced virtual laboratories are being developed, covering topics such as advanced P4 constructs, advanced parsing, stateful processing, and data plane/control plane communication protocols. Examples of future virtual laboratories on P4 include monitoring queue statistics using standard metadata, header stacks to implement a custom protocol, and sending digests to notify the control plane about custom events.

IV. WORK IN PROGRESS: TOFINO VIRTUAL LABORATORIES

A. Tofino Pod Description

The Tofino pod is a work in progress that is currently in the testing stage. This pod aims to expose the learner to a production-grade P4 programmable switch, which otherwise can be expensive to deploy. The Tofino pod is hosted in the Academic Cloud and allows the learner to interact with the pod through the web interface.

Fig. 3 shows the components of the Tofino pod: three VMs (PC1, PC2, and Tofino model), an Intel Tofino switch, Network Interface Controllers (NICs), physical fiber links, and virtual links. PC1 and PC2 can communicate through the Tofino model or the Tofino switch. The former runs on a Linux Debian 10 VM, and the latter is a Edgecore Wedge 100BF-32X [17] that runs an Intel Tofino ASIC [18]. The Tofino model has logging enabled, which is unavailable in the hardware switch. This feature allows the programmer to track the packet as it propagates through the data plane pipeline, facilitating debugging and troubleshooting the P4 program.

The PCs communicate via a dual-port NVIDIA Mellanox ConnectX 5 NIC with the Tofino switch using a 100Gbps multi-mode fiber link. The PCs also use the DirectPathI/O functionality available in VMware vSphere [19]. This feature allows VMs to communicate with the hardware directly. On the other hand, the PCs use a virtual Ethernet link to the Tofino Model. Both switches are managed from PC1 using an out-of-band connection.

B. Virtual Laboratories in the Tofino Pod

The Tofino virtual laboratories will cover most of the topics described in Table II. The first lab will cover the design workflow in the Tofino pod, which consists of creating a P4 program in the Tofino model. Once the P4 program is ready, the user loads the P4 program into the Tofino switch. The following labs will explain the Tofino Native Architecture (TNA) [20] blocks, parser implementation, match-action tables, the

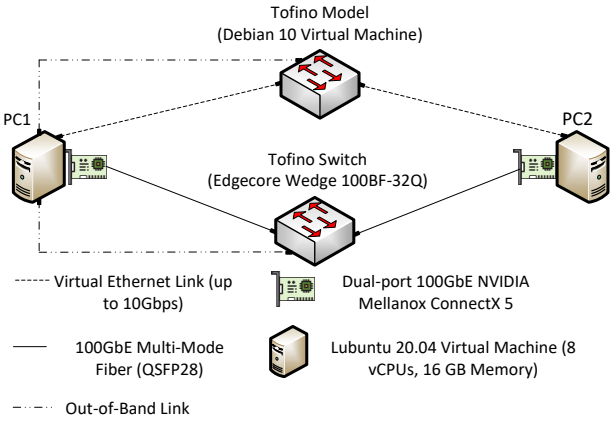

Fig. 3. Topology used for the Tofino virtual laboratories.

runtime environment, checksum calculation, and deparser. The TNA is a programmable switch architecture defined by Intel for their family of Tofino switching ASICs. In contrast to the V1Model, TNA provides its architecture definitions that expose the distinctive capabilities of the ASICs. More advanced labs will include stateful processing, advanced parsing, and the usage of TNA externs such as low-pass filters, packet generators, and hash computation.

The proposed workflow for the Tofino pod starts with creating a P4 program using the Tofino model, then loading the P4 program in the Tofino ASIC to achieve high performance. The learner interacts with the switches using SSH sessions launched from Visual Studio Code, which contains the text editor and the CLI to compile and load a P4 program from the Tofino model to the Tofino switch.

C. The Academic Cloud as a Testbed for Research

Although the Academic Cloud has been used for research purposes already [21–25], its primary purpose has been for teaching and training. The Academic Cloud can support a wide range of equipment (virtual machines, legacy routers, passive taps, mmWave Access Points, smartNICs, firewalls, and other appliances) to conduct research. Moreover, pods in the Academic Cloud can interact with network testbeds, computing facilities, and scientific instruments via an external link. This feature could enable researchers to test their experiments in a research infrastructure such as FABRIC [26].

Preliminary tests in the Tofino pod of Fig. 3 show that a throughput test between PC1 and PC2 achieves up to $\sim$80Gbps via the Tofino switch. Note that the throughput is limited by the traffic generator rather than by the switch. This performance indicates that the Tofino pod has potential as a network testbed for high performance. The Academic Cloud also allows testing applications on a wide area network (WAN) by conducting experiments among its four data centers in South Carolina, North Carolina, Idaho, and Illinois. The authors believe that connecting the data centers to FABRIC may open new research possibilities on programmable data plane applications.

V. Conclusion

This paper describes the implementation of P4 virtual laboratories in the Academic Cloud. The P4 library introduces the learner to data plane concepts and P4 via hands-on experiences. The Academic Cloud is a distributed platform that manages and orchestrates computing resources to enable virtual laboratories. It supports complex network topologies that include real and virtualized hardware. Additionally, this paper presents the Tofino pod, a work in progress that uses Intel Tofino switches. Finally, the paper discusses the potential use of the Academic Cloud to conduct research experiments over WANs. Future virtual libraries will cover network operating systems (NOS) (SONiC, ONL, Stratum) and P4 programmable smartNICs.

References

[1] G. Papastergiou, G. Fairhurst, D. Ros, A. Brunstrom, K.-J. Grinnemo, P. Hurtig, N. Khademi, M. Tüxen, M. Welzl, D. Damjanovic, *et al.*, "De-ossifying the internet transport layer: A survey and future perspectives," *IEEE Communications Surveys & Tutorials*, 2016.

[2] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, 2014.

[3] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An exhaustive survey on P4 programmable data plane switches: Taxonomy, applications, challenges, and future trends," *IEEE Access*, 2021.

[4] The P4 Language Consortium, "P4 Tutorial." [Online]. Available: https://github.com/p4lang/tutorials, Accessed on 03-22-2022.

[5] ETH Zürich - Networked Systems Group (NSG), "P4-Learning." [Online]. Available: https://github.com/nsg-ethz/p4-learning, Accessed on 03-22-2022.

[6] Open Network Foundation (ONF), "Getting Started with P4." [Online]. Available: https://tinyurl.com/2p89237n, Accessed on 03-22-2022.

[7] "Network development group (NDG)." [Online]. Available: https://www.netdevgroup.com/, Accessed on 03-21-2022.

[8] VMware, "vSphere." [Online]. Available: https://www.vmware.com/products/vsphere.html, Accessed on 03-21-2022.

[9] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, 2012.

[10] Linux Foundation, "FRRouting Project." [Online]. Available: https://frrouting.org/, Accessed on 03-22-2022.

[11] The P4 Language Consortium, "Behavioral Model version 2 (BMv2)." [Online]. Available: https://github.com/p4lang/behavioral-model, Accessed on 03-21-2022.

[12] M. Peuster, J. Kampmeyer, and H. Karl, "Containernet 2.0: A rapid prototyping platform for hybrid service function chains," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 2018.

[13] B. Lantz and B. O'Connor, "A mininet-based virtual testbed for distributed SDN development," *ACM SIGCOMM Computer Communication Review*, 2015.

[14] C. Boettiger, "An introduction to docker for reproducible research," *ACM SIGOPS Operating Systems Review*, 2015.

[15] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, *et al.*, "The design and implementation of Open vSwitch," in *12th USENIX symposium on networked systems design and implementation (NSDI 15)*, 2015.

[16] Cyberinfrastructure Lab, "Access request form." [Online]. Available: http://ce.sc.edu/cyberinfra/cybertraining.html, Accessed on 04-24-2022.

[17] Edgecore Networks, "Wedge 100BF-32X." [Online]. Available: https://tinyurl.com/sy2jkqe, Accessed on 03-21-2022.

[18] Intel, "Intel Tofino Ethernet switch ASIC." [Online]. Available: https://tinyurl.com/mry8a8c4, Accessed on 03-21-2022.

[19] VMware, "Direct path I/O." [Online]. Available: https://tinyurl.com/5pvejwbe, Accessed on 03-21-2022.

[20] Barefoot Networks, "P4 Intel Tofino native architecture - public version." [Online]. Available: https://tinyurl.com/5d7nznwd, Accessed on 03-22-2022.

[21] E. F. Kfoury, J. Crichigno, E. Bou-Harb, D. Khoury, and G. Srivastava, "Enabling TCP pacing using programmable data plane switches," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, 2019.

[22] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "Offloading media traffic to programmable data plane switches," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, 2020.

[23] E. F. Kfoury, J. Gomez, J. Crichigno, and E. Bou-Harb, "An emulation-based evaluation of TCP BBRv2 alpha for wired broadband," *Computer Communications*, 2020.

[24] E. Kfoury, J. Crichigno, E. Bou-Harb, and G. Srivastava, "Dynamic router's buffer sizing using passive measurements and P4 programmable switches," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021.

[25] K. Friday, E. Kfoury, E. Bou-Harb, and J. Crichigno, "Towards a unified in-network DDoS detection and mitigation strategy," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020.

[26] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K.-C. Wang, T. Lehman, and P. Ruth, "FABRIC: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, 2019.