P4 Programmable Patch Panel (P7): An Instant 100G Emulated Network on Your Tofino-based Pizza Box

Fabricio Rodriguez
University of Campinas (Unicamp)

Francisco Germano Vogt
University of Campinas (Unicamp)

Ariel Góes De Castro
Federal University of Pampa (Unipampa)

Marcos Felipe Schwarz
Brazilian National Research and Education Network (RNP)

Christian Rothenberg
University of Campinas (Unicamp)

ABSTRACT

Alternatives to run high-fidelity network experiments are traditionally based on virtual and emulation-based environments (e.g., Mininet). While extremely useful for teaching and in support of research practices, existing experimental platforms are commonly limited to transmission speeds of 10Gbps and suffer from performance-fidelity trade-offs as well as inherent scalability constraints. With the programmability that P4 brings to networking researchers and the capabilities of new generation P4 hardware supporting the PSA (Portable Switch Architecture) and TNA (Tofino Native Architecture), it is possible to realize packet processing pipelines that emulate certain network link characteristics and instantiate a network topology to run line-rate traffic using a single physical P4 switch (e.g., Tofino). This is the main contribution of the P7 (P4 Programmable Patch Panel) emulator. In this demonstration, we show how to generate different network topologies starting from a single link to more complex network scenarios featuring various devices and paths, including different link characteristics (e.g., latency, jitter, packet loss, bandwidth) and 100G traffic capacities.

1 INTRODUCTION

With increasingly powerful and complex networking environments being worked out by the industry and academia research efforts, notably fueled lately by network programmability advances (e.g. P4 [1]), the demand for experimental validation before actual deployment becomes paramount. Accessible and affordable user-friendly testbeds providing line-rate and high fidelity performance for evaluation purposes are tricky to achieve. Researchers' budgets are commonly limited and largely impact the quantity and quality of networking devices. In this scenario, preparing an running experiments are commonly limited to small-scale environments (e.g., speeds, number of devices, complexity), emulation/virtualization environments (e.g. Mininet [5], Mininet-WiFi [3]) or simulation-based approaches. At the end, well-known trade-offs of networking experimentation hit the research roadmap and compromise different aspects such as realism, flexibility, scalability, and customizability of the experiments, among others.

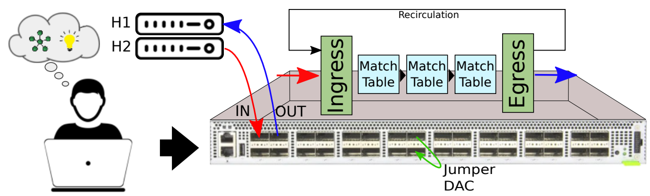

Figure 1: P7 concept and P4 pipeline representation.

In this demo, we present P7 (P4 Programmable Patch Panel)[1] as a high-end yet affordable network emulation platform that overcomes shortcomings from traditional testbed approaches. P7 exploits the capabilities of P4-capable hardware to provide realistic emulation of network topologies using programmable hardware pipeline features such as packet recirculations, port configurations, match+action table abstractions, along a simple path routing solution. Furthermore, the user/experimenter can connect physical servers to inject custom traffic flows (e.g. PCAP-based or Tofino-based) to the emulated networking scenario (see Figure 1). Re-playing link performance behavior based on real traces data sets (e.g. 5G access links) is also on the P7 roadmap.

Our P7 efforts adhere to related work on network emulation platforms such as [5], [4], [6] and [2]. The latter is closest to P7 by also exploiting P4 programmable infrastructure and including control plane support in their scope.

2 P7 ARCHITECTURE & DEMO

With a design that prioritizes simplicity, P7 is an hardware-based emulation testbed that allows users to define a target network topology with annotated link metrics in a user-friendly manner. The user experience is very much similar to defining topologies using the popular Mininet emulator [5]. From the user-defined topology, P7 autogenerates all the necessary files (e.g., P4 code, tables information, chassis configuration) to transform a P4 hardware device into a P4 Programmable Patch Panel capable of realistically emulating different scenarios.

[1]Public available at: https://github.com/intrig-unicamp/p7

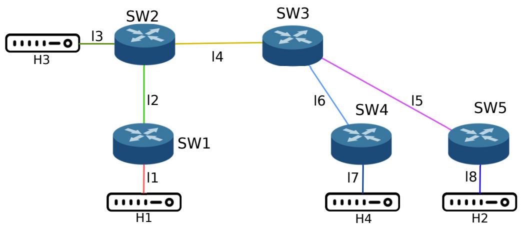

Figure 2: P7 example network topology.

Figure 2 shows an example topology where each link receives a specific annotation (i.e., BW, latency, packet loss, jitter). The topology and link metrics are implemented in a Tofino target to which physical end-hosts or other network nodes (e.g. P4 devices) can be directly attached using 10G, 25G or 100G interfaces as per the hardware specification.

Currently supported link emulation capabilities include simple metrics (e.g., Connectivity, Latency [ms], bandwidth [bps]) as well as advanced ones (e.g., Packet loss [%], Jitter [ms], background traffic [bps], Reordering [%]). A list of characteristics implemented and planned in P7 and the different approaches are detailed in Table 1.

The high-level P7 architecture and workflow are depicted in Fig. 3 where the green and red arrows show the user's input. In the first case, the user defines the topology in the P7 main file following the corresponding syntax and structure. In a second case, the user can import a topology from a Mininet file and add the port configuration of the Tofino target. P7 converts and adapts the imported topology to generate three output files: (*i*) *P4 program*: where all metrics and link connectivity patterns are defined . The metrics are a central part of the P4 program, each link is treated as an individual entity inside the code. Using recirculations and internal temporary tags, it is possible to create the logic of all the different links along the traffic paths; (*ii*) *Interfaces configuration*: the chassis file with all the ports configuration for Stratum [7]; and (*iii*) *Tables information*: comprising all data to fill the P4 tables, including routing information, devices location, and VLAN segmentation.

Table 1: P7 Link characteristics and P4/TNA implementation approaches.

Link Connectivity	Dijkstra algorithm to calculate the routes Internal Recirculation + Temporary P7 Tag
Latency [ms]	Timestamp-based timer Recirculation via internal pipe | external jumper DAC
Jitter [ms]	Random number generator to vary the latency Lookup table with mathematical functions
Packet loss [%]	Random generator for packet discard probability Realistic definition of packet loss
Re-ordering [%]	TNA Traffic Management (TM) features Per-packet probability based recirculation
Bandwidth [bps]	Rate limit TNA TM feature Port configuration and mapping
Background Traffic [bps]	Tofino packet generation engine Up to 100G per pipeline

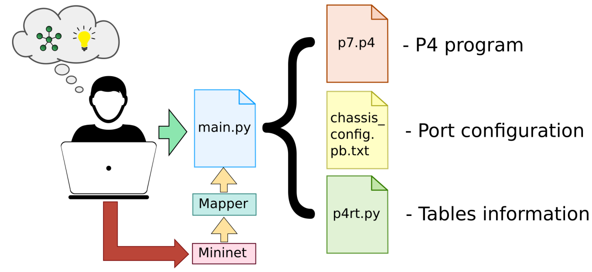

Figure 3: P7 high-level architecture and workflow.

During the demo. Attendees will be asked to choose a topology with specific link metrics (e.g., latency, jitter, packet loss, background traffic, bandwidth) and will see how P7 produces all artifacts to embody the desired experimental environment. We will showcase the ability of P7 to instantiate effective network testbeds capable of reproducing the link metrics with high fidelity. Real-time visualization of network traffic will contribute to validate on-the-fly performance of the link metric and the emulation capabilities. The first public video demonstration of P7 was recently unveiled during the P4 2022 workshop.[2]

3 CONCLUSIONS AND FUTURE WORK

This demo shows how P7 contributes to the ecosystem of affordable 100G experimental platforms with a user-friendly, cost-effective 100G network emulator in support of traditional networking and advanced programmable networking research, as well as teaching purposes. Being a programmable high-fidelity testbed, P7 facilitates repeatable and reproducible research by sharing P7 topology files to be compiled and deployed, resulting in the same output anywhere (along with the specific Tofino target capabilities permit).

Future work. First, we want to formalize the performance/scalability limits of the topologies (i.e. #links, aggregated link capacities and latencies, etc.) depending on the memory, buffers, and stages available in the Tofino target. P7 can be scaled out beyond pizza box to distributed deployments. Also included in the P7 roadmap are: Building an open source community; Integration with Mininet to import and visualize user-defined topologies; Addition of In-band Network Telemetry (INT) features for fine-grained statistics of the emulated network (e.g., queue occupancy, device status); Time-varying link behaviors (e.g., Gilbert-Elliot packet loss model, jitter patterns, 5G access link traces) for further realism and dynamism; Embed P7 (e.g adding P7 hardware-in-the-loop or compiling into P4 SmartNICs) into larger testbeds such as NSF Fabric and disaggregated network initiatives (e.g. RNP-CPQD OpenRAN) to enrich their experimental toolboxes with tailored line-rate network emulation capabilities.

[2] https://www.youtube.com/watch?v=MszZV4cpR3s

4 TECHNICAL REQUIREMENTS

4.1 Equipment to be used for the demo

A single laptop and a remote P4 hardware device (i.e., tofino).

4.2 Space needed

The space provided by SIGCOMM.

4.3 Setup time required

Up to 30 minutes.

4.4 Additional facilities needed, including power and any Internet access requirements

Internet access and power source for laptop are required.

5 DOCUMENTATION, CODE AND VIDEO

P7 is released as an open source project under Apache-2.0 license. It can be downloaded from Github at https://github.com/intrig-unicamp/p7.

The documentation is available at the site https://github.com/intrig-unicamp/p7/wiki which shows all the topology generation options along with the configuration parameters and generated files with their complete usage description.

The link https://youtu.be/aRYxPvlvo_Q shows a video tutorial of the tool.

REFERENCES

[1] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-independent Packet Processors. *SIGCOMM Comput. Commun. Rev.* 44, 3 (July 2014), 87–95. https://doi.org/10.1145/2656877.2656890

[2] Jiamin Cao, Yu Zhou, Ying Liu, Mingwei Xu, and Yongkai Zhou. 2020. TurboNet: Faithfully Emulating Networks with Programmable Switches. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. IEEE, Madrid, Spain, 1–11. https://doi.org/10.1109/ICNP49622.2020.9259358

[3] Ramon R Fontes, Samira Afzal, Samuel HB Brito, Mateus AS Santos, and Christian Esteve Rothenberg. 2015. Mininet-WiFi: Emulating software-defined wireless networks. In *2015 11th International Conference on Network and Service Management (CNSM)*. IEEE, 384–389.

[4] Pravein Govindan Kannan, Ahmad Soltani, Mun Choon Chan, and Ee-Chien Chang. 2018. BNV: Enabling Scalable Network Experimentation through Bare-Metal Network Virtualization. In *Proceedings of the 11th USENIX Conference on Cyber Security Experimentation and Test (CSET'18)*. USENIX Association, USA, 6.

[5] Bob Lantz, Brandon Heller, and Nick McKeown. 2010. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-IX)*. Association for Computing Machinery, New York, NY, USA, Article 19, 6 pages. https://doi.org/10.1145/1868447.1868466

[6] Hongqiang Harry Liu, Yibo Zhu, Jitu Padhye, Jiaxin Cao, Sri Tallapragada, Nuno P. Lopes, Andrey Rybalchenko, Guohan Lu, and Lihua Yuan. 2017. CrystalNet: Faithfully Emulating Large Production Networks. In *26th SOSP*. ACM, NY, USA, 15. https://doi.org/10.1145/3132747.3132759

[7] Open Networking Foundation (ONF). 2019. Stratum. https://github.com/stratum/stratum. (2019).