Network Emulation with P7: A P4 Programmable Patch Panel on Tofino-based Hardware

Fabricio Rodriguez[1], **Francisco Germano Vogt**[1], **Ariel Góes De Castro**[2], **Marcos Schwarz**[3], **Christian Rothenberg**[1]

[1] University of Campinas (UNICAMP)
[2] Federal University of Pampa (Unipampa)
[3]Brazilian National Research and Education Network (RNP)

{frodri,chesteve}@dca.fee.unicamp.br, f234632@dac.unicamp.br, arielcastro.aluno@unipampa.edu.br, marcos.schwarz@rnp.br

Abstract. *The use of virtual and software-based environments, such as Mininet, has become popular for network experimentation. However, these platforms often have limitations, including low transmission speeds and trade-offs between scalability and performance fidelity. Advances in P4 programmability and new P4 hardware that supports Tofino Native Architecture (TNA) have enabled the possibility of emulating various network link characteristics and creating network topologies for running line-rate traffic in a single P4 switch (i.e., Tofino). In this paper, we introduce the P7 (P4 Programmable Patch Panel) emulator, which allows the configuration of network scenarios with different link characteristics, including 100G traffic capacities, using a single P4 switch. We demonstrate the scalability and realism of the P7 emulator, making it an ideal environment for network research and experimentation.*

1. Introduction

With an ever-increasing demand for complex network environments being developed by both the industry and academia, software-based environments struggle to deliver high-fidelity experiments for real scenario instances. This surge in demand has been primarily fueled by advances in network programmability, such as P4 [Bosshart et al. 2014]. However, creating user-friendly testbeds that are accessible, affordable, and can provide line-rate and high-fidelity performance for evaluation purposes can be challenging. Researchers often operate with limited budgets, which greatly affects the quantity and quality of networking devices at their disposal. Consequently, experiments are frequently limited to small-scale environments regarding speed, number of devices, and complexity, or emulation/virtualization environments, such as Mininet [Lantz et al. 2010] and Mininet-WiFi [Fontes et al. 2015], or simulation-based approaches. As a result, networking experimentation often involves common trade-offs that may compromise aspects such as realism, flexibility, scalability, and customizability of experiments, among others.

A first introduction and limited version of the P4 Programmable Patch Panel (P7) was presented in [Rodriguez et al. 2022] as a high-end yet affordable network emulation platform that overcomes shortcomings from traditional testbed approaches. In this work, we aim to detail the P7 implementations and discuss the addition of new link metrics such as custom packet loss models (i.e., Gilbert-Elliot and user-defined) and, moreover, the addition of support for user-defined P4 code to run in the internal switches in a parallel pipeline. P7 is a network emulator for P4-enabled devices. With P7, it is possible to provide realistic emulation of network topologies using programmable hardware pipeline features such as recirculations, port configurations, different match+action tables, and even DAC cables. Furthermore, the user or experimenter can connect physical servers to inject custom traffic (e.g., PCAP-based or Tofino-based) to the emulated networking scenario (see Figure 1). Additionally, we increased the level of customization by supporting user-defined P4 codes on each "emulated" switch.

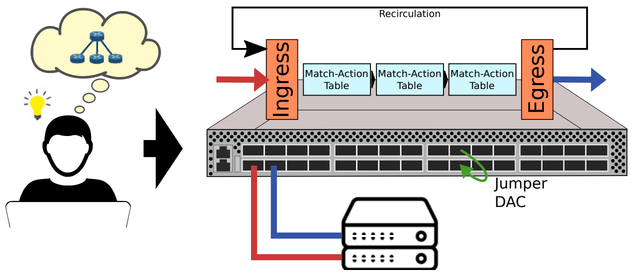

Figure 1. P7 concept and P4 pipeline representation.

The remainder of this paper is organized as follows: motivations and goals for developing our tool are introduced in Section 2. Section 3 summarizes the related works. Section 4 gives an overview of the P7 tool itself, followed by a detailed explanation of its main features. Meanwhile, Section 5 presents a use-case depicting the usage of P7 for performance tests. Section 6 shows the documentation and the demonstration details for the Demo. Finally, future work and conclusion are discussed in Section 7 and Section 8, respectively.

2. Motivation and Goals

Traditional network experiment solutions, such as virtual and emulation-based environments, suffer from performance fidelity, trade-offs, and scalability constraints. Therefore, there is a need for a realistic experimental platform that can provide high-fidelity performance, scalability, and flexibility. We aim to leverage the power of P4-based hardware (Tofino) to provide a realistic experimental platform with high-fidelity performance, scalability, flexibility, and support for data plane programmability using a hardware-based environment.

P7 leverages the programmability and capabilities of new-generation P4 hardware to emulate network links and instantiate a network topology, allowing line-rate traffic to run using a single physical P4 switch.

3. Related Work

This section discusses related works that focus mainly on topology emulation. Existing efforts are implemented on software (e.g., VMs) or hardware.

Mininet [Lantz et al. 2010] is an open-source network emulator for SDN. It introduces the concept of network emulation as a means to test and experiment with SDN architectures in a single machine. Despite that, it has core limitations in providing high-fidelity network experimentation. First, its network emulation capabilities are limited by the characteristics of the host machine's network interface. The emulation accuracy may be affected by the host machine's CPU and memory and the number of virtual hosts and switches being emulated.

Koponen et al. [Koponen et al. 2014] proposes NSX, a network virtualization platform developed by VMware to manage multi-tenant domains in data center environments. It provides a distributed virtualization layer that allows for the creation of virtual networks across multiple physical hosts. Also, NSX includes a network services platform that offers firewall, load balancing, and VPN connectivity services. Similarly, CrystalNet [Liu et al. 2017] consists of three main components to emulate large-scale production networks: (i) the Topology Generator leverages statistical models and real-world network data to create to reflect the characteristics of production networks, such as degree distribution, clustering coefficient, and link capacity; (ii) the Traffic Modeler leverages ML techniques to reflect the behavior of real users and applications and; (iii) the Network Emulator leverages VMs for the testing of different network configurations and policies. However, the work is limited to virtual switches and routers.

The solutions, as mentioned earlier, are software-based and may not be sufficient for testing and evaluating large-scale production networks, as they can suffer from performance and scalability limitations. More recently, BNV [Kannan et al. 2018] leverages hardware virtualization technologies (e.g., Intel VT-d, SR-IOV) to enable the virtualization of network devices (e.g., Open vSwitches). SimBricks [Li et al. 2020] is designed to enable end-to-end simulation of host networking stacks. It simulates components of the host networking stack, such as the network interface card (NIC), driver, kernel, and application, to provide a more comprehensive, flexible, and cost-effective approach to evaluating host networking performance in different scenarios – e.g., packet loss, TCP congestion algorithms, as presented by the authors.

TurboNet [Cao et al. 2020] is the first to leverage the programmability of modern switches to emulate network behavior accurately. Specifically, it uses P4 to implement switch behavior in software and then runs the software on commodity servers to emulate the behavior of the switches on a single commodity server to enable the emulation of large-scale networks. Nevertheless, not all link characteristics and the support of a custom P4 code are present. In this work, we present P7, a simple plug-and-play solution specifically designed to provide a high-fidelity instant 100G emulated network on Tofino-based switches, with various link metrics and the support of a user-defined P4 code.

4. P7 (P4 Programmable Patch Panel)

P7 is a high-end, affordable network emulation platform that realistically emulates network topologies using programmable hardware features. P7 overcomes the shortcomings of traditional testbed emulation approaches limited by small-scale environments, software-based/virtualization environments, or simulation-based approaches, compromising different aspects such as fidelity with real networks, flexibility, scalability, and the customizability of experiments.

P7 allows users to define a network topology, including the link metrics, in a user-friendly script, similar to defining topologies using the popular Mininet. From the user-defined topology, P7 internally generates all the necessary files to transform a single P4 hardware switch into a P7 emulator that can realistically run different scenarios. The design of P7 prioritizes simplicity, making it a hardware-based emulation testbed that allows speeds of 10G, 25G, or 100G.

4.1. Architecture

The P7 architecture presented in figure 2 illustrates the high-level components and their interconnections.

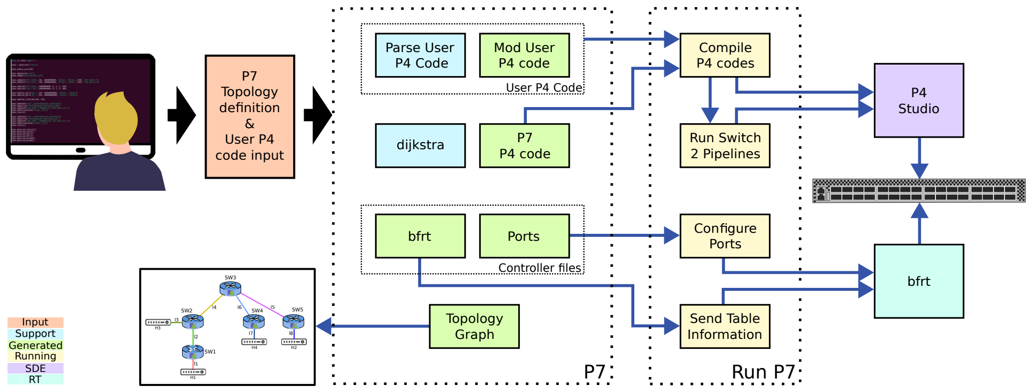

Figure 2. P7 high-level architecture and workflow.

Input: The user defines the topology in the P7 main script. The setup includes the number of links and their characteristics (link metrics), the number of nodes, the custom P4 code (user P4 code), and table configuration.

P7: The central part of our tool, where all the data is processed, and the corresponding files are generated.

- **User P4 Code:** The P4 code is parsed to identify the principal parts of the code such as the parser, ingress, and tables, then the necessary modifications are made, including index in the table and P7 header parser), and the Modified User P4 code is generated.
- **Packet Forwarding:** To perform the routing, we use the Dijkstra algorithm to define the internal forwarding routes based on the shortest path.
- **Controller files:** Using this information, the table information for the P7 P4 code is prepared using the barefoot runtime (bfrt) format. In addition, The bfrt file includes table information from the user's P4 code. Port mapping and configuration files are also generated from the port configuration input in the main P7 file.
- **P7 P4 Code:** In this part of the architecture, the heart of P7 is also built. This file contains all the recirculation, metrics, headers, and forwarding information running in the Tofino switch.
- **Topology Graph:** In addition to the generated files, a topology graph is also created based on the user configuration.

Run P7: To run P7, it is necessary to compile the generated P4 codes (i.e., User, P7) using the SDE tools and run the Tofino switch (with the correct pipeline configuration) from the P4 studio environment. On the other hand, it is necessary to set the port configuration and the table information using the bfrt.

From a hardware perspective, the P4 codes deployment is introduced in figure 3.

To distribute resources, each P4 code (i.e., P7, user) runs in separate pipes (0 and 1). One pipe allocates the P7 P4 code, including all link characteristics logic and forwarding process to the corresponding "emulated" switch. In a different pipe, the user

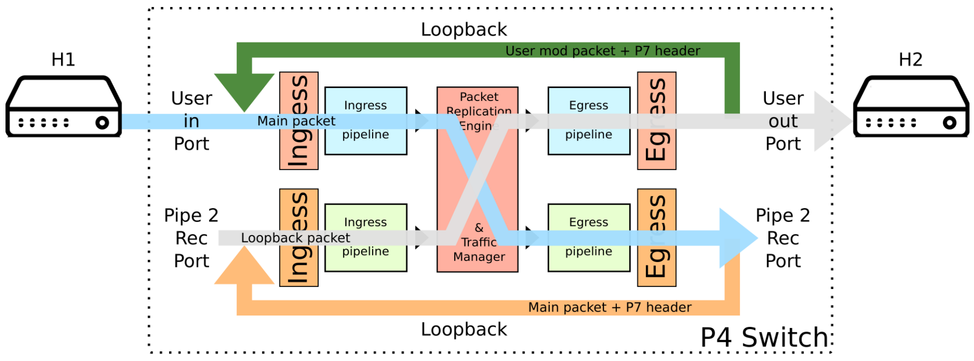

Figure 3. P7 P4 architecture.

P4 code is set. The Tofino Native Architecture (TNA) recirculation feature is used to create internal switch representations. When packets need to be forwarded to a switch, a recirculation + pipeline change occurs. Then, the packet is sent back to the P7 pipe to continue with the topology logic.

4.2. Main features

P7 offers several features and characteristics that make it a reliable and accessible tool for network emulation. These features are summarized below:

P4 programmable: P7 is built on P4 hardware, which provides programmable characteristics such as packet recirculation, port configuration, and match+action table abstractions to offer realistic emulation of network topologies.

Affordable: P7 provides high-end emulation capabilities at an affordable cost, making it accessible to researchers with limited budgets and access to a single P4 hardware.

Realistic emulation: P7 allows users to define network topologies with predefined link metrics (Implementation details in table 1. The following link metrics can be defined:

- **Bandwidth**: Users can define the bandwidth limit (in Mbps) for each individual link in the topology.
- **Latency**: Users can define per-link latency in milliseconds.
- **Packet loss**: Users can define the probability of dropping a packet as a percentage.
- **Jitter**: In addition to latency, users can define per-link jitter (in ms) plus a probability (per packet) that the jitter will be applied.

Programmable data plane emulation: P7 allows users to add custom P4 code to the internal switches. The user can define the table information using the same P7 script.

Custom traffic traces: P7 allows users to inject custom traffic flows from traffic or trace generators, emulating real-world network scenarios.

Simplicity: P7 provides a user-friendly interface for defining network topologies and autogenerates all the necessary files.

High-speed interfaces: P7 supports high-speed interfaces, such as 10G, 25G, and 100G.

Open source: P7 is publicly available under the Apache License 2.0.

5. Use Case

In this section, we first provide an example of a topology that can be instantiated and devices that can be connected to a network. Next, we detail how the topology is configured within the P7 environment and its features, such as user code and network forwarding.

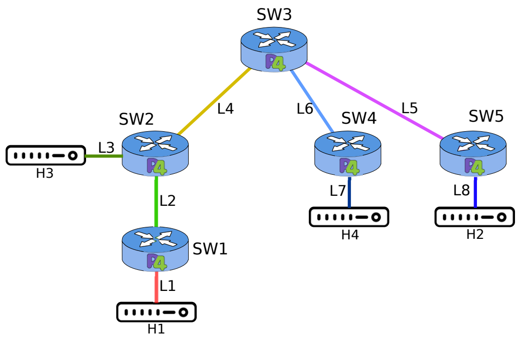

Figure 4. P7 example network topology.

Figure 4 illustrates a high-level overview of a network configuration emulated with P7. In this example, there is a set of interconnected switches (i.e., SW1 to SW5) with different link characteristics (i.e., L1 to L8). Additionally, physical hosts (i.e., H1 to H4) are attached to the switch ports. Custom traffic is sent from a host and passes over the emulated links and switches. Further, Figure 5 describes the data plane approach to achieve the network topology mentioned above in the Tofino switch. In P7, link characteristics are implemented in the P7 P4 code. The P4 tables in this pipe define the routing logic and connection to the switches. Recirculations and pipe changes are used to define the routing rules in the P4 code according to the desired topology. We leverage the available physical ports to connect external devices to the switches to forward the traffic to the corresponding link. The user also has the flexibility to customize the P4 code that can be "emulated" in each internal switch together with the table configuration, thus allowing greater customization of the data plane forwarding logic at each point in the network.

5.1. Experimental Evaluation

To evaluate the proposed metrics (shown in Table 1), we can use custom network topology, for instance, the one represented in Figure 4. End-to-end latency and jitter can be obtained by running a `ping` command between two hosts, such as H1 and H2. Packet loss can be measured either by `ping` or a traffic generator tool by setting independent loss probabilities for each link. Traffic rate and available bandwidth can be estimated by using

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

Link Characteristic	Implementation approach
Link Connectivity	Dijkstra algorithm to calculate the routes Internal recirculation to the same pipe represents a link
Latency [ms]	Timestamp-based timer Recirculation via internal port until the timer reach the desired latency
Jitter [ms]	Random number generator to vary the latency Lookup table with mathematical functions to perform the calculations
Packet loss [%]	Random probability generator to decide the packet discard probability Realistic definition of packet loss to define the good or bad state
Re-ordering [%]	TNA Traffic Management (TM) features Per-packet probability based recirculation
Bandwidth [bps]	Rate limit using the Traffic manager feature Port shaping configuration of the ports
Background Traffic [bps]	Tofino packet generation engine Up to 100G per pipeline of custom traffic profiles

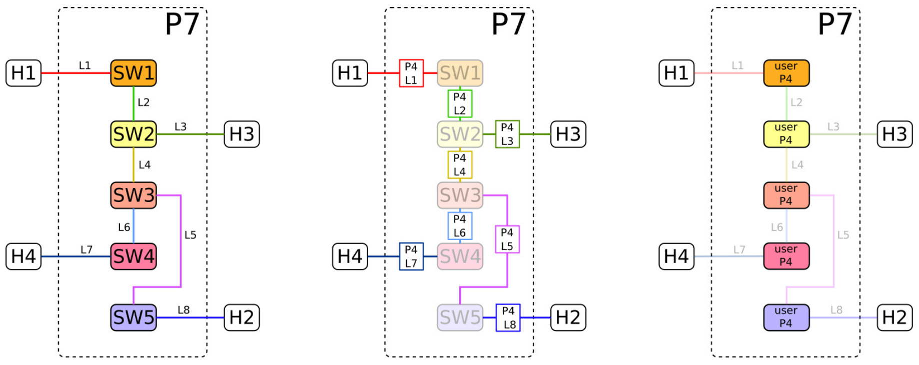

Figure 5. P7 network topology taxonomy.

a traffic generator tool, such as `T-Rex`, to send data between two hosts at a pre-defined bandwidth.

We conducted a series of experiments to evaluate the performance of the proposed metrics, and we can confirm that the experiments are working as expected. We used the topology illustrated in Figure 4 and ran various tests to measure the end-to-end latency, jitter, packet loss, and available bandwidth. We used standard tools like `ping` and `T-Rex` to generate traffic and measure the network parameters.

The results obtained from these experiments were consistent with our expectations and matched the theoretical values we calculated. We verified that the proposed metrics accurately reflect the network performance and can be used to diagnose and optimize it.

6. Documentation, Code, and Demonstration

P7 is an open-source project under the Apache License 2.0, available on GitHub at `https://github.com/intrig-unicamp/p7`. The project welcomes contributions, bug reports, discussions of current functionalities, and proposals for new features.

The documentation is available on the project's wiki at `https://github.com/intrig-unicamp/p7/wiki`, which provides a complete usage description of all features (i.e., link characteristics). Additionally, a video tutorial demonstrating the configuration and execution of the tool is available at `https://youtu.be/dAhy8R34vHU`.

The demonstration will focus on defining network topologies using P7's main script and their link metrics. Additionally, the user can compile and run P7 on remotely available physical Tofino hardware to validate the topology and defined characteristics.

Using a friendly interface similar to Mininet, the user can set all the necessary information (i.e., table information, P4 code) to run experiments on a hardware-based topology and validate adding custom P4 codes in the switches.

7. Future Work

As a next step, we aim to establish the performance and scalability boundaries of the P7 emulator, including the number of links, aggregated link capacities, latencies, and most importantly, the scalability (i.e., number of nodes) that the memory, buffers, and stages available in the Tofino target can support. Furthermore, we plan to extend a comparative analysis with other tools, showcasing the characteristics, limitations, and emulation capabilities. Additionally, We plan to extend the capabilities of P7 beyond a single switch to enable distributed deployments. Additionally, we intend to build an open-source community, integrate P7 with Mininet to facilitate the import and visualization of user-defined topologies, and incorporate In-band Network Telemetry (INT) features to provide detailed statistics of the emulated network's queue occupancy and device status. To enhance the realism and dynamic nature of P7, we plan to introduce jitter patterns and 5G access link traces. Finally, we plan to embed P7 in larger testbeds such as NSF Fabric and disaggregated network initiatives (e.g., OpenRAN Brasil) to expand their experimental toolkits with customized line-rate network emulation capabilities.

8. Conclusions

This demonstration highlights the significance of P7 in affordable 100G experimental platforms. P7 is a user-friendly and cost-effective network emulator that caters to traditional networking, advanced programmable networking research, and teaching purposes. Additionally, it supports scenarios where metric variation is necessary for tests. As a programmable high-fidelity testbed, P7 offers the advantage of facilitating repeatable and reproducible research. Overall, with our experiments, we are confident that we provide reliable and accurate measurements of the network performance, and the proposed metrics can be used to evaluate and improve the performance of real-world networks.

9. Acknowledgment

This work was supported by the Innovation Center, Ericsson S.A., and by the Sao Paulo Research Foundation (FAPESP), grant 2021/00199-8, CPE SMARTNESS. This study was partially funded by CAPES, Brazil - Finance Code 001.

References

Bosshart, P. et al. (2014). P4: Programming protocol-independent packet processors. *SIGCOMM*, 44(3):87–95.

Cao, J. et al. (2020). Turbonet: Faithfully emulating networks with programmable switches. In *IEEE 28th ICNP*, pages 1–11, Madrid, Spain. IEEE.

Fontes, R. R. et al. (2015). Mininet-wifi: Emulating software-defined wireless networks. In *2015 11th CNSM*, pages 384–389, Barcelona, Spain. IEEE.

Kannan, P. G. et al. (2018). Bnv: Enabling scalable network experimentation through bare-metal network virtualization. In *USENIX*, CSET'18, page 6, USA.

Koponen, T. et al. (2014). Network virtualization in multi-tenant datacenters. In *USENIX NSDI*, pages 203–216, Seattle, WA. USENIX Association.

Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: Rapid prototyping for software-defined networks. In *SIGCOMM*, Hotnets-IX, NY, USA. ACM.

Li, H. et al. (2020). Enabling end-to-end simulation for host networking evaluation using simbricks.

Liu, H. H. et al. (2017). Crystalnet: Faithfully emulating large production networks. In *26th SOSP*, NY, USA. ACM.

Rodriguez, F. et al. (2022). P4 Programmable Patch Panel (P7): An Instant 100g Emulated Network on Your Tofino-Based Pizza Box. In *SIGCOMM*, page 4–6, NY, USA. ACM.