Automated Test Bench for High-Performance Network Equipment

Benjamin Steinert[‡§], Gabriel Paradzik[‡§], Michael Menth[‡]
[‡] University of Tübingen, Chair of Communication Networks, Tübingen, Germany
[§] University of Tübingen, Zentrum für Datenverarbeitung, Tübingen, Germany
Email: {benjamin.steinert,gabriel.paradzik,menth}@uni-tuebingen.de

***Abstract*—This paper presents an automated test bench that supports reproducible and holistic benchmarking of data plane and control plane performance. The modular architecture integrates the hardware-based traffic generator P4TG and the software-based traffic generator iperf3 for precise control over test traffic. Additionally, it supports automated Device under Test (DuT) reconfiguration between test runs and metric collection. A case study demonstrates the feasibility of the approach by measuring the performance of a modern P4-based COTS data center switch.**

I. Introduction

Network equipment such as routers or switches are traditionally built with fixed-function ASICs that define the data plane capabilities of the device. The ASICs are controlled by a management CPU that implements the control plane logic. In recent years, network equipment vendors have adopted programmable ASICs to construct high-performance devices with flexible feature sets. Examples include AMD Pensando Elba Data Processing Unit (DPU) [1], Intel Tofino [2], or Cisco Silicon One [3], among others. Such devices promise advantages over products with fixed-function ASICs in terms of flexibility and performance.

However, the performance limits stated in data sheets are not always explicit regarding specific features. Moreover, control plane functions and automation interface performance – such as the time needed for a configuration to take effect on the data plane – are often not part of the data sheet reports. For network operators, it is therefore difficult to reliably assess if the performance of a certain device is sufficient for the intended use case without manual evaluation and measurement.

Performance measurements are a substantial part of many research articles. In the existing literature, different testbeds have been proposed that are tailored to specific use cases, e.g., for IoT [4], TSN [5], MPTCP [6], or SDN [7]. However, existing approaches are often based on emulation or simulation and often do not allow to evaluate a physical network device. With the NetBOA framework [8], the authors propose an approach that allows to automatically perform benchmarks with flexible traffic generation and Bayesian optimization for finding weak spots in the parameter space. Automated DuT configuration is not in scope of NetBOA. For measuring data plane performance, several tools are available for high-speed traffic generation. Examples include the P4-based P4TG [9], DPDK-based Cisco TRex [10], or software-based iperf3 [11]. However, the measurement of control plane or API performance is usually not possible with such tools.

This work was supported by the bwNET 2.0 project which is funded by the Ministry of Science, Research and the Arts Baden-Württemberg (MWK). The authors alone are responsible for the content of this paper.

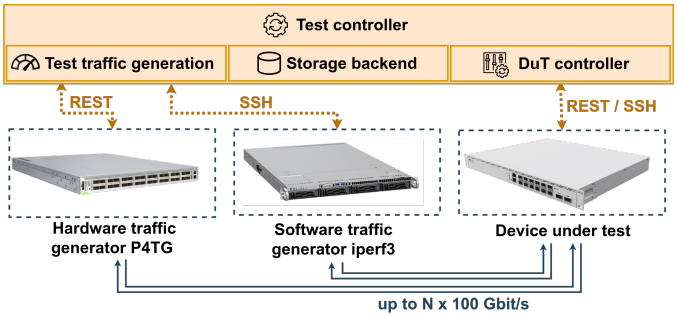


Fig. 1. Architecture and components of the automated test bench. The modular test controller orchestrates the hardware-based traffic generator P4TG, the software-based traffic generator iperf3, and the device under test.

This paper presents an automated test bench for network equipment benchmarking that can measure data plane performance as well as control plane performance across different device types. The proposed modular test bench supports declarative configuration of experiment parameters and automated execution of test runs. This includes the automated generation of custom traffic patterns, automated vendor-specific configuration of the DuT, and automated data gathering for metrics of interest. Such features enable reproducible test conditions that can be tailored to the target network.

II. Test Bench Architecture and Components

The test bench consists of different components that enable reproducible experiments with fine-grained control over test traffic and DuT configurations. Figure 1 displays the overall architecture, including the test controller software, the P4-based traffic generator P4TG, a software-based traffic generator, and a DuT controller. The central test controller software controls the traffic generators and it interfaces with the DuT for automated configuration and metric gathering. Test traffic is generated using the P4-based traffic generator P4TG and the software-based iperf3. The DuT controller implements vendor-specific automation interfaces that are used for configuration and metric monitoring of different network appliances.

A. Test Controller

The test controller is the central software component responsible for executing experiments based on a declarative experiment configuration. It is written in Python and uses automation interfaces exposed by the traffic generators and the DuT to automatically push configuration between test runs and gather metrics during the test runs. The configurations and gathered metrics are stored in JSON format for each test run in a NoSQL MongoDB database. This allows for checkpointing between runs and enables automated analytics or visualizations of the gathered data. The design of the test controller software is highly modular such that the storage backend can be easily replaced, new traffic generators can be included, or new vendor-specific modules for DuT configuration or data gathering can be added.

B. Test Traffic Generation

The hardware-based high-performance traffic generator P4TG enables configuration of up to 10 different UDP traffic streams with customizable properties. This includes features like address randomization, adjustable packet sizes or precise throughput setting. While P4TG works well for UDP-based high-throughput tests, stateful TCP connections are not part of the feature set of P4TG. For generating congestion-controlled traffic, the software traffic generator iperf3 is used. With this combination, fine-grained control over test traffic is possible. The P4TG provides a REST API for configuration of traffic streams and for gathering of measurement data like round-trip times (RTTs), data rates, inter-arrival times (IATs), or packet drops. The iperf3 software traffic generator runs on a separate machine and is controlled remotely.

C. DuT Controller

The DuT controller is a module of the test controller that implements vendor-specific automation interfaces of the DuT. With such a module, it is possible to automatically configure the DuT between experiments and to monitor performance metrics and device parameters. In the presented test bench, three different vendor modules are implemented, i.e., for MikroTik RouterOS, Aruba AOS-CX, and AMD Pensando Policy and Services Manager (PSM). More device modules can be added by implementing desired configuration tasks using the device's automation interfaces. For automatic configuration of device parameters, the DuT controller translates abstract configuration tasks into into device-specific commands that are executed on the DuT. The DuT controller also polls the DuT for device metrics. Thereby the polling interval and metrics of interest are configured in the test controller software.

D. Workflow

Conducting an experiment with the automated test bench starts with a declarative definition of experiment parameters in the test controller configuration. The test controller translates them into device-specific configurations for the traffic generators or the DuT. When starting an experiment, the automatic configuration of the DuT is performed, then the traffic generators are configured. After successful configuration, the experiment runs are executed automatically. The data of interest is gathered automatically from the traffic generators and the DuT, and is stored persistently in a database for each run. This mechanism allows for checkpointing, automated analysis, and automated visualization of the results.

III. Case Study: Hardware Evaluation - L4 Packet Filtering on Aruba CX 10000

Modern network devices are built with programmable ASICs that promise advanced features with high performance. In this case study, we used the automated test bench to benchmark the packet filtering feature including the performance of the REST API automation interface on an Aruba CX 10000 device. While the test bench is capable of many different benchmarks and experiments, we demonstrate the general feasibility of the approach by presenting selected aspects.

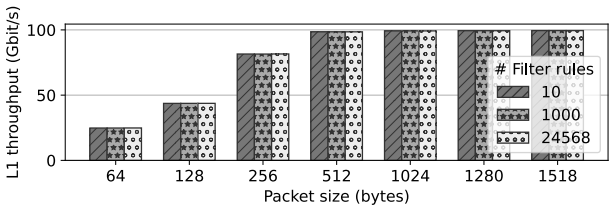
A. Evaluated Device

The evaluated device is a modern data center switch with 48x SFP28 25 Gbit/s interfaces and 6x QSFP 100 Gbit/s interfaces. The physical interfaces are connected to a fixed-function Broadcom Trident-3 ASIC for general L2/L3 packet processing. Additionally, the switch has two built-in P4-programmable AMD Pensando Elba ASICs that are each connected to the Broadcom Trident-3 ASIC with a 400 Gbit/s lane. The Elba chips enhance the overall functionality of the device by enabling L4 packet processing. They allow filtering a large number of L4 traffic flows with up to 100 Gbit/s on the device. The different ASICs are connected to a CPU for management and control plane functions, where the Aruba AOS-CX operating system controls the Trident-3 chip and a software called PSM controls the Elba chips. The chips complement each other, delivering L2/L3 processing and L4 packet filtering capabilities, a feature that is further evaluated in the present case study.

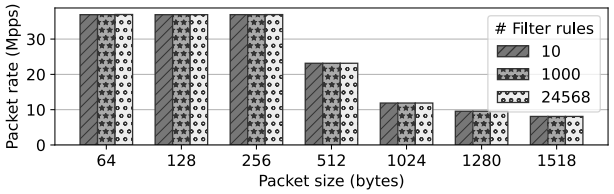
B. Data Plane Performance

For measuring data plane performance, we generate artificial test traffic and measure packet drops on the DuT. In each experiment run, the generated throughput is increased until packet drops are observed. The highest throughput without packet drops is reported as maximum possible throughput based on certain traffic and DuT configurations.

Existing studies suggest that firewall performance deteriorates as more filter rules are installed [12]. To measure the impact of L4 filter rules on the Aruba CX 10000, we install varying numbers of L4 filter rules and measure the maximum possible throughput using so-called Internet Mix (IMIX) traffic. IMIX traffic consists of packets of different sizes with certain proportions and it is a common traffic configuration for network performance tests. We configure the P4TG traffic generator to produce three different constant bitrate (CBR) streams such that 12% of the total throughput consists of 64 B packets, 54% of 512 B packets, and 34%



(a) Maximum L1 throughput in Gbit/s.



(b) Maximum packet rate in million packet per second (Mpps).

Fig. 2. Maximum data plane throughput of the AMD Pensando Elba ASIC depending on packt size and the installed number of filter rules.

of 1518 B packets. We measure that the Elba ASIC is able to process 100 Gbit/s IMIX traffic in line rate without packet drops while the number of active filter rules does not have an impact on the measured throughput.

While IMIX traffic is not always representative of the specific target network, we repeat the experiment with CBR traffic consisting of only one packet size. To measure the impact of different packet sizes on the maximum throughput, we vary the packet size of the generated traffic from 64 B to 1518 B. To evaluate if the number of filter rules does have an impact on the maximum throughput, we install varying numbers of filter rules and measure the maximum possible throughput. Figure 2 depicts the results. The maximum data plane throughput of the chip seems to be slightly below 40 Mpps which can be seen for small packets (64 B to 256 B). For larger packet sizes, the line rate of 100 Gbit/s is achieved. We observe that the number of active filter rules does not have an impact on the maximum throughput.

C. Control Plane Performance

Control plane performance is often not reported in data sheets, but it is important for network operators to know how fast configuration changes are active on the data plane. In this case study we focus on L4 filter rule installation speed. Thereby, we generate a low-volume test stream and craft a specific filter rule that blocks this stream. To measure how long it takes for a set of L4 filter rules to be active on the data plane, we configure filter rules on the DuT such that the specifically crafted rule for the test stream is configured last. Then we measure the time when the test stream is interrupted. On some devices, filter rules have to be installed subsequently, i.e., one rule at a time. On other devices, it is possible to perform batch-based installation, i.e., multiple rules at once.

Using the PSM software REST API to configure the AMD Pensando Elba chip, it is possible to install up to 24568 stateful L4 filter rules with batch-based installation. We install different numbers of filter rules and measure the filter rule

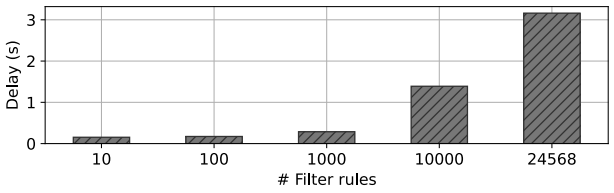


Fig. 3. Installation time for different number of filter rules on the AMD Pensando Elba ASIC using batch configuration with the PSM REST API.

installation time. The results are shown in Figure 3, it can be observed that the maximum number of filter rules are active within 3.2 s which corresponds to an installation rate of more than 10k rules/s. We repeat the experiment with the Broadcom Trident-3 ASIC for comparison. Using the Aruba AOS-CX REST API, it is possible to install up to 1536 ACL rules on the Trident-3 chip, one rule at a time. We measure that it takes around 50 s for the 1536 ACL rules to be active on the data plane which corresponds to an installation rate of around 30 rules/s.

IV. Conclusion

In summary, the proposed test bench provides an automated, reproducible framework for benchmarking network devices. It integrates hardware- and software-based traffic generators, and automates traffic generation, device configuration, and metric collection based on declaratively defined parameters. The test bench can also evaluate control plane performance, e.g., filter rule application speed. A case study on a modern device demonstrated the feasibility of the approach, while showing that programmable ASICs can enable high-performance L4 packet filtering with fast filter rule installation.

References

[1] "AMD Pensando® 2nd Generation ("Elba") Data Processing Unit," 2024. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/pensando-technical-docs/product-briefs/pensando-elba-product-brief.pdf

[2] "Intel® Tofino™ Series." [Online]. Available: https://www.intel.com/content/www/us/en/products/details/network-io/intelligent-fabric-processors/tofino.html

[3] "Cisco Silicon One Product Family White Paper," 2021. [Online]. Available: https://people.ucsc.edu/w̃arner/Bufs/white-paper-sp-product-family.pdf

[4] M. Vučinić *et al.*, "OpenBenchmark: Repeatable and Reproducible Internet of Things Experimentation on Testbeds," in *IEEE Conference on Computer Communications Workshops*, 2019.

[5] J. Hanke *et al.*, "Software-Defined Testing Facility for Component Testing with Industrial Robots," in *IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2022.

[6] M. Michalski, "The Hardware and Software Test Bed for MPTCP," in *IEEE 33rd International Conference on Computer Communications and Networks (ICCCN)*, 2024.

[7] L. Zhu *et al.*, "SDN Controllers: Benchmarking & Performance Evaluation," *arXiv preprint arXiv:1902.04491*, 2019.

[8] J. Zerwas *et al.*, "NetBOA: Self-Driving Network Benchmarking," in *ACM SIGCOMM Workshop on Network Meets AI & ML*, 2019.

[9] S. Lindner *et al.*, "P4TG: 1 Tb/s Traffic Generation for Ethernet/IP Networks," *IEEE Access*, vol. 11, 2023.

[10] "TRex – Realistic Traffic Generator." [Online]. Available: https://trex-tgn.cisco.com/

[11] "iperf3: A TCP, UDP, and SCTP Network Bandwidth Measurement Tool." [Online]. Available: https://github.com/esnet/iperf

[12] L. Wüsteney *et al.*, "Impact of Packet Filtering on Time-Sensitive Networking Traffic," in *17th IEEE International Conference on Factory Communication Systems (WFCS)*, 2021.