Fridolin Siegmund, Matthias Hollick, Ralf Kundel. *Instant P4STA: Beyond Tbit/s Network Function Evaluation with P4 Programmable Hardware* To appear in the Proceedings of IEEE Conference on Network Operations and Management Symposium 2025, IEEE, 2025.

Instant P4STA: Beyond Tbit/s Network Function Evaluation with P4 Programmable Hardware

Fridolin Siegmund[§], Matthias Hollick[§], Ralf Kundel[§]
[§]Technical University of Darmstadt
fridolin.siegmund@tu-darmstadt.de

***Abstract*—Cloud data center, backbone, and access networks constantly push the boundaries towards lower latencies, jitter, and scalable throughput. Evaluating data plane devices, *i.e.*, switches, routers, and complex network functions, by developers and service operators under demanding settings is imperative to ensure service resilience in real-world deployments. Our proposed prototype, *Instant P4STA*, extends a packet timestamping framework for programmable hardware by combining a hardware packet generator with a uniform browser-based packet editor for dynamic packet generation. The user can specify the packet template bit-by-bit, utilizing the Python library *Scapy* with a vast variety of packet templates. This way, our prototype combines the best features of software and hardware-based packet generators. We demonstrate packet generation up to 3.2 Tbit/s on eight egress ports with up to four packet types in parallel. More packet generation throughput is possible with more egress ports, capped only by the number of physical ports in the programmable hardware.**

***Index Terms*—P4, Tofino, Packet Generator, SDN, QoS, Latency, Throughput, Timestamping, Packet Crafting**

I. Introduction

The current development of networks with ever-increasing link speeds and forwarding capacities providing digital services around the globe leads to a growing demand for network benchmarking equipment with packet generators that reach more than 1 Tbit/s of generated throughput. With the latest forwarding devices on the market, *e.g.*, switches and routers, supporting link speeds at multiple interfaces with each up to 800 Gbit/s Ethernet, advanced benchmarking tools supporting the generation of multiple Tbit/s and enabling evaluation with nanosecond accuracy are required. Also, other packet forwarding and processing functionalities such as firewalls or subscriber termination in future 6G and home fiber access networks are build taking high throughput and advanced Quality of Service (QoS) demands in account. Benchmarking and evaluation of such devices is crucial for developers, vendors and customers and requires the capability to generate packets at multiple ports with line rates up to 800 Gbit/s. At the same time, the packet generator needs to offer flexible templates for different protocols to support a heterogeneous set of devices under test. Typically, software-based packet generators offer high flexibility for packet generation but may be limited in packet generation at high rates; for instance, libraries such as DPDK are limited at around 10-100 Gbit/s, depending on the packet sizes. Also, measuring QoS metrics, *i.e.* one-way latency and jitter, in software leads to unacceptable low time resolutions in the range of microseconds. Additionally, software stacks add some latency to the packet processing after timestamping. Hardware-based packet generators offer line rate packet generation per port. However, they are less flexible in the type of generated packets, *e.g.*, with a limited set of packet templates and potential statelessness. Also, hardware-based packet generators offer a best-in-class time accuracy for timestamping generated packets with nanosecond time resolution.

In this demonstration, we combine the integrated packet generator in the Tofino P4 programmable ASIC with the P4STA framework, which enables disaggregated hardware-based network function evaluation with nanosecond accuracy. We solve the lack of flexibility in hardware-based packet generators and integrate a versatile user interface for packet template editing. This interface is based on Scapy* as a packet creation and editing framework, allowing the users to specify every bit of the packet templates.

The proposed benchmarking framework *Instant P4STA* offers support for the following features, all included in a rack-mounted Tofino switch with one height units:

- 400 Gbit/s packet generation per port (i.e. up to line rate)
- Simultaneous generation of up to 4 different packet types
- Per packet timestamping with nanosecond accuracy
- Dual staged evaluation of timestamps in hardware and with the x86 monitoring instance

II. Background

Programmable switch-based traffic generators such as P4STA [1], [2] and P4TG [3] utilize, *e.g.*, P4 programmable switches such as the Intel Tofino to generate load and measure Device Under Tests (DUTs) characteristics with nanosecond precision. In Table I, the features of P4TG and P4STA are differentiated. While P4TG utilizes the integrated packet generator in the Tofino, the packet generation and timestamping capabilities are limited to UDP packets. P4STA, on the other hand, generates TCP and UDP packets by deploying the external iPerf3 software-based load generator but is therefore limited in throughput. P4STA timestamps both TCP and UDP headers. Both frameworks implement data plane duplication features for enhanced egress throughput. P4TG utilizes mul-

*https://scapy.net/

Fridolin Siegmund, Matthias Hollick, Ralf Kundel. *Instant P4STA: Beyond Tbit/s Network Function Evaluation with P4 Programmable Hardware* To appear in the Proceedings of IEEE Conference on Network Operations and Management Symposium 2025, IEEE, 2025.

	Measurement		Packet Generation		
	Stamped Protocols	Timestamps: Accuracy & Placement	Protocols	Multiple Types	DP Dup
P4TG [3]	UDP	[ns] in P4 Pipe	UDP (internal)	No	Mcast Recirc
P4STA [2]	TCP/UDP	[ns] in MAC	TCP/UDP (external)	Yes (external)	Mcast
Instant P4STA	TCP/UDP/ ICMP/Encap.	[ns] in MAC	Any (internal)	Yes (4)	Mcast

TABLE I: Load generation and measurement frameworks P4TG, P4STA and Instant P4STA in comparison.

ticasting and packet recirculation for data plane duplication, while P4STA relies on multicast groups exclusively.

III. Prototype and Demonstration

We present our prototype *Instant P4STA*, which extends the P4STA framework by developing load generation in the integrated hardware-based packet generator in the Tofino ASIC instead of external software-based packet generators. Figure 1 shows the prototype's architecture in our testbed. The P4STA Management Virtual Machine (VM) runs the P4STA core and webserver, though both could run on the x86 CPU in the white box Tofino directly for a portable setup. In the web GUI, the user sets the port configuration of the connected Devices Under Test (DUT), which can be every Ethernet capable network function from 10 Gbit/s to 400 Gbit/s per port. The packets are processed by the P4 pipeline while timestamps are set in the MAC layer. The timestamp is placed either in the TCP header or UDP payload, not limited by additional lower headers such as VLAN. We also included General Packet Radio Service Tunneling Protocol User Plane (GTP-U) timestamping support for 5G and future 6G user plane packets in *Instant P4STA*, supporting TCP and UDP as GTP-U payload.

As Figure 1 shows, each test run can generate four different types of packet templates at a line rate up to 400 Gbit/s at multiple ports. If the packet generation engine is not producing sufficient throughput in high-load scenarios, user-defined multicast groups allow for heavy throughput upscaling

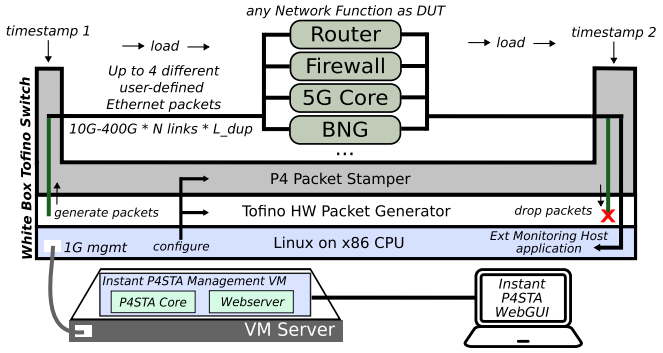

Fig. 1: Proposed architecture of *Instant P4STA* with integrated packet generation and nanosecond timestamping.

```
packets = {}

# adds packet payload "0" after scapy headers until total packet length is reached
def add_payload_0(scapy_headers, total_paket_len):
    return scapy_headers/Raw(load="0" * (total_paket_len - len(scapy_headers)))

# UDP Packet
udp_packet_headers = Ether(src="00:6d:52:e0:22:2e")/IP(src="10.11.12.1")/UDP(dport=42424)
packets["udp_packet"] = add_payload_0(udp_packet_headers, 1500)
# TCP Packet
tcp_packet_headers = Ether()/IP()/TCP(dport=24242)
packets["tcp_packet"] = add_payload_0(tcp_packet_headers, 1500)
# ICMP Ping Request
ping_headers = Ether(src="00:6d:52:e0:22:2e", dst="00:6d:52:e0:22:2f")/IP(dst="10.10.20.1", src="10.10.20.42", ttl=20)/ICMP(type=8)
packets["ping_packet"] = add_payload_0(ping_headers, 64)
# ARP Request
arp_headers = Ether(src="00:6d:52:e0:22:2e", dst="ff:ff:ff:ff:ff:ff")/ARP(pdst="10.10.20.1", hwsrc="00:6d:52:e0:22:2e", psrc="10.10.20.42")
packets["arp_packet"] = add_payload_0(arp_headers, 128)
# 5G GTP-U Packet
udp_base = Ether(src="00:6d:52:e0:22:2e", dst="00:6d:52:e0:22:2f")/IP(src="10.99.99.199", dst="10.99.99.99")/UDP(sport=2152, dport=2152)
gtp_base = gtp.GTPHeader(gtp_type=255, teid=846, E=1, next_ex=0x85)/gtp.GTPPDUSessionContainer(ExtHdrLen=1, type=1, QFI=1)
gtp_tcp_headers = udp_base/gtp_base/IP(src="10.45.0.2", dst="10.77.77.42")/TCP(sport=1234, dport=1234, flags="PA")
packets["gtpu_tcp"] = add_payload_0(gtp_tcp_headers,1444)
gtp_udp_headers = udp_base/gtp_base/IP(src="10.45.0.2", dst="10.77.77.42")/UDP(sport=1234, dport=1234)
packets["gtpu_udp"] = add_payload_0(gtp_udp_headers, 1460)
```

Update Packets | Load Default

Fig. 2: Web GUI: packet template editing with Scapy.

and usage of all egress ports. Compared to software-based load generators, the flexibility in our prototype is limited to uni-directional flows, *e.g.* no bidirectional Transmission Control Protocol (TCP) session can be established. Though generating TCP payload packets at one port and acknowledgment packets at a second port is easily possible, it lacks the implementation of a TCP algorithm.

During the benchmark, packets are timestamped before and after the DUT. Packets containing the second timestamp are duplicated to the external monitoring host, running on the x86 CPU in the white box switch.

After the test run, measurements are evaluated in two stages. Every packet's minimum, maximum, and average latency is stored as the first stage in the stamper. Due to the external monitoring host's packet processing limit, not all packets are duplicated for monitoring, depending on whether the GoLang or DPDK version is configured. The configurable sampling factor balances accuracy, processing delay, and DUT load to prevent packet loss at the monitoring host.

As shown in Figure 2, the benchmarking run configuration includes a packet template editor with Scapy. In Scapy, all kinds of packets can be created. Also, if the packet type is not included in Scapy, packets can be created with raw bytes. After saving the current Scapy Python code, the defined packets appear in the drop-down menu shown in Figure 3 ready to be configured for the generation ports. In Figure 3, the generation ports, packet templates, throughput, upscale factors, and the duration of the load generation can be configured.

We demonstrate the *Instant P4STA* prototype by benchmarking multiple DUTs for 60 seconds. We use both Tofino 2

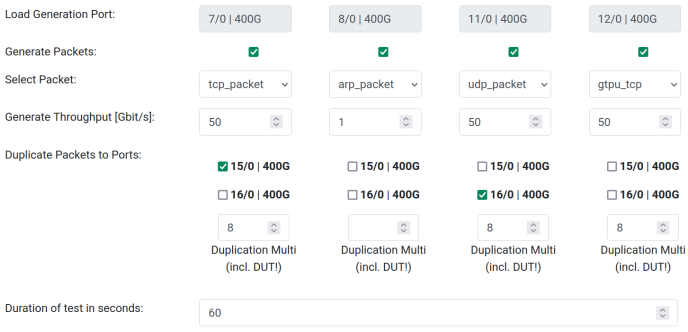

Fig. 3: Web GUI: configuration of the integrated packet generator.

Fridolin Siegmund, Matthias Hollick, Ralf Kundel. *Instant P4STA: Beyond Tbit/s Network Function Evaluation with P4 Programmable Hardware* To appear in the Proceedings of IEEE Conference on Network Operations and Management Symposium 2025, IEEE, 2025.

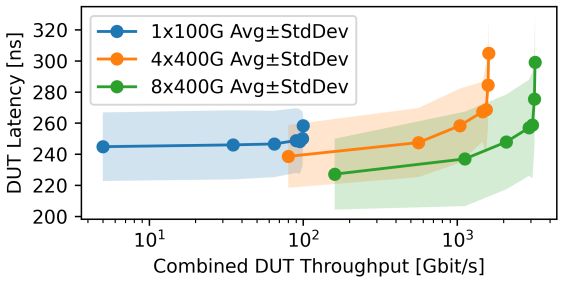

Fig. 4: Measured Direct Attach Cables (DACs) as DUTs with up to 3.2 Tbit/s total DUT load (1500-byte packets).

(Edgecore DCS810) with 400 Gbit/s links as well as Tofino 1 (Edgecore Wedge100BF) with 100 Gbit/s links.

In Figure 4, the average latency and standard deviation of networking cables with up to 400 Gbit/s per link as DUT is shown. Links with 400 Gbit/s operate with a slightly higher average latency when fully loaded than 100 Gbit/s links. Also, the standard deviation increases for 400 Gbit/s links. There is no measurable influence on DUT latencies by the number of used links at the packet generator. With a maximum cable length of 1 m, the actual transmission of electromagnetic waves takes only a few nanoseconds, and the rest is spent on *e.g.* the physical layer processing and transceiver traversing. As expected, no packet loss is observable for the cables.

Software-based network functions such as Virtual Network Functions (VNFs) run on an AMD Ryzen 5800X with an Intel E810 network card with two 100 Gbit/s interfaces in our testbed. Therefore, the benchmark is limited to 100 Gbit/s per link as 400 Gbit/s network cards are not widely available yet. All evaluated network functions as DUT are forwarding packets from the ingress to the egress port, while the packets can be modified in the network function, *e.g.* decapsulation.

As Figure 5 shows, the measurements of an OpenVSwitch-DPDK L1 packet forwarder, the routing stack in the Linux kernel as well as a 5G User Plane Function (UPF) by Open5GS with one established cellular subscriber session. DPDK naturally outperforms the other DUTs regarding throughput, exceeding the 100 Gbit/s link, and low latency. The Linux kernel routing is limited in about 10 Gbit/s of forwarding capacity with 1500-byte packets. As the UPF is decapsulating GTP-U packets, which are then routed by the Linux kernel,

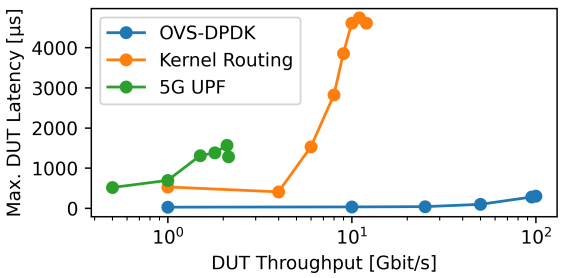

Fig. 5: Measured maximum latency of software-based network functions connected by a 100 Gbit/s link (1500-byte packets).

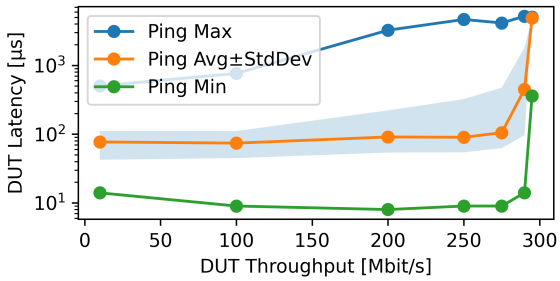

Fig. 6: Measured ping response of Linux kernel by timestamping ICMP echo request and corresponding response packets. The blue area shows the standard deviation (64-byte packets).

the maximum throughput is about 2 Gbit/s with similar latency behavior as kernel routing only.

Figure 6 shows another of the many *Instant P4sta* use cases by generating timestamped ICMP echo request packets, answered by a Linux kernel as DUT. With a maximum around 300 Mbit/s of 64 byte ICMP packets, the Kernel is capable of responding to around 4.7 million ping packets per second. Expectedly, the measured latency increases from a minimum of 8 μs exponentially to up to 5 ms under heavy load.

IV. Conclusion and Future Work

Our demonstration shows how commodity P4 programmable white box switches can generate multiple Tbit/s at 400 Gbit/s line rate. Simultaneously, packets are timestamped with nanosecond accuracy, allowing the evaluation of network functions down to the measurement of packet cable transit delays. We have shown how packet templates can be dynamically generated in the user-friendly web GUI, enabling endless evaluation scenarios for many network functions. At the conference venue, we demonstrate *Instant P4STA* live in a browser with a remote-controlled testbed.

Besides increasing the link speeds to 800 Gbit/s or even 1.6 Tbit/s, more evaluation of, *e.g.*, checksums in the data plane could open new opportunities for network function evaluation. However, our research has shown that integrating more data plane features in *Instant P4STA* requires more hardware resources than the Tofino 2 ASIC currently offers.

Acknowledgement

This work has been supported by the LOEWE initiative (Hesse, Germany) within the emergenCITY center [LOEWE/1/12/519/03/05.001(0016)/72]. Furthermore, we thank our colleagues and reviewers for their valuable input.

References

[1] R. Kundel, F. Siegmund, J. Blendin, A. Rizk, and B. Koldehofe, "P4STA: High Performance Packet Timestamping with Programmable Packet Processors," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS)*. IEEE, 04 2020, pp. 1–9.

[2] R. Kundel, F. Siegmund, R. Hark, A. Rizk, and B. Koldehofe, "Network Testing Utilizing Programmable Network Hardware," *IEEE Communications Magazine*, pp. 12–17, 02 2022.

[3] S. Lindner, M. Häberle, and M. Menth, "P4TG: 1 Tb/s Traffic Generation for Ethernet/IP Networks," *IEEE Access*, vol. 11, pp. 17 525–17 535, 2023.