Ralf Kundel, Fridolin Siegmund, Boris Koldehofe: "How to measure the speed of light with programmable data plane hardware?" @ 2nd P4Europe Workshop - ACM/IEEE ANCS 2019.
2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)

How to measure the speed of light with programmable data plane hardware?

Ralf Kundel
Multimedia Communications Lab
TU Darmstadt, Germany
ralf.kundel@kom.tu-darmstadt.de

Fridolin Siegmund
Multimedia Communications Lab
TU Darmstadt, Germany
fridolin.siegmund@kom.tu-darmstadt.de

Boris Koldehofe
Multimedia Communications Lab
TU Darmstadt, Germany
boris.koldehofe@kom.tu-darmstadt.de

***Abstract*—Driven by real-time applications such as IIoT, TSN and vehicular networks, the optimization of networks and its elements regarding latency and throughput becomes more and more important. With this demo we show how latencies of network components can be identified within nanosecond accuracy by use of commodity P4 hardware. We show a measured propagation speed of $5ns/m$ in fiber optical cables. Besides that, our approach scales up to $100Gbit/s$ link speed by the aggregation of many low-cost load generators to a flexible software-based load generation.**

***Index Terms*—P4, Hardware Timestamping, 100G, Latency, speed of light**

I. Introduction

The technical evolution of computer networks and its applications, e.g. real-time video streaming, causes higher and higher throughput and low latency requirements. Optimization and investigation of state-of-the-art networking devices, which should fulfill these requirements, becomes increasingly difficult as the latency decreased and the bandwidth has risen strongly in the last years. Modern switches have a forwarding delay of a few hundreds of nanoseconds paired with $100Gbit/s$ and more per port. Measuring performance characteristics of those devices with software solutions becomes very hard, as timestamping with microsecond accuracy requires the use of Network Interface Cards (NIC) with hardware timestamping (e.g. IEEE 1588-2002) and multiple Rx-queues [1]. Furthermore, it must be ensured that losses are not caused by the load generator, e.g. caused by the NIC, PCIe, x86-system. Hardware based solutions, which are specially built to test network devices under high load, are very costly and their flexibility, e.g. adding new protocols or congestion-control mechanisms, is very limited.
Programmable data plane hardware, mainly driven by the programming language P4, opens new capabilities to create a low cost load generator with high accuracies. With our work P4STA[1] we created a framework, including a data plane implementation, for load generation, aggregation, hardware timestamping and evaluation, which is mainly based on the advantages of P4-programmable hardware and FPGAs.

This timestamping platform can measure the characteristics of a so called *device under test* (DUT), which can be anything like a cable, switch, router or network function. How the DUT is connected with the P4-switch and the software load generators is depicted in Figure 1. For this demo we consider a fiber optical cable as DUT. By varying the length of the fiber, we will show additional latency caused by the length of the fiber. Measuring multiple fiber length enables the computation of speed of light and the measurement error by linear regression. Highly optimized chip designs of programmable data plane hardware, running on clock frequencies above $1GHz$ allows timestamping packets with a clock precision of $1ns$. Even modern FPGAs have a clocking period of $3ns$ which is enough for most use-cases. With this demo we want to show the benefits of:

- aggregation of multiple load servers,
- timestamping packets with ns-accuracy under load and
- zero loss detection with hardware counters.

[1] https://github.com/ralfkundel/P4STA

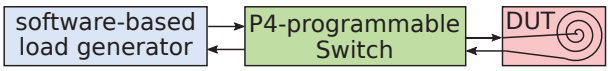

Fig. 1. P4-based timestamping concept. Time measurements are performed inside the P4-switch, the load generator creates and validates packets.

II. Preliminaries and Demo Setup

This demo consists of a P4STA-based testbed setup, which consists of the following components:

- **P4 programmable switch:** 65 (or 32) port Barefoot Tofino. Netronome SmartNICs would work similar.
- **Load generator servers:** 2 Server with Intel Xeon D-1541, 64 GB Ram and Intel X710 NIC (2x10Gbit/s).
- **Breakout-cable:** *1x40Gbit/s -¿ 4x10Gbit/s* for interconecting the load generators with the P4 switch.
- **DUT:** 2 x Transceiver *Flexoptix 40G QSFP+ SR4* and MTP-capable fibers (1m, 2m, 3m, 10m).
- **Management switch:** *TP-LINK AC1750* wireless access point with 4 Gigabit Ethernet Ports.
- **Laptop/PC:** Any Computer or Tablet, supporting Ethernet or WiFi, with an Internet browser.

The components will be connected as depicted in Figure 2, fitting into a small size 19" rack with at least 4U height. The two *10Gbit/s* ports per server are connected by a breakout cable with one port of the P4 switch. Two fiber optical QSFP+ transceiver are plugged into the switch and connected with the

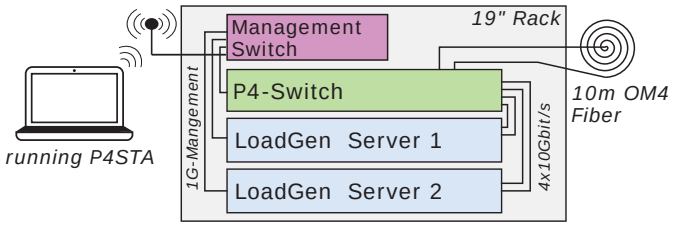

Fig. 2. Setup of the Demo in a 4U rack. Two common servers with 10G-NICs, a P4-programmable Tofino and a management switch.

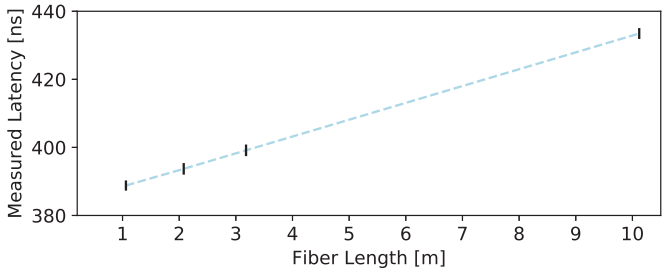

Fig. 3. Measured latency for 1m, 2m, 3m, 10m fiber, linear regression and standard deviation bars.

1m optical cable (the first DUT of this Demo). All devices are connected by an *1Gbit/s* management switch which is required to access the devices via ssh to start the experiment and retrieve the results. All servers run on Ubuntu and one of them executes the P4STA-core software which manages the execution of the experiment. The installation scripts of P4STA ensure that all needed configurations (e.g. permissions, iperf3 installation) are done. The laptop is used for accessing the web-application via http from the P4STA server. The frontend provides all needed functionality for configuration, test execution and evaluation.

III. Measure the speed of light

With this demo we want to highlight the following features of P4 programmable hardware: (1) load aggregation, (2) timestamping with ns granularity and (3) loss detection. Initially, valid IP-addresses must be assigned to all load generator ports which belongs to the same IP-subnet. This enables the use of a kernel-based load-generator and no DPDK-capable network interface cards are required. For that, the first load generator server starts two iPerf3 servers and the second load generator two iPerf3 clients, which transmit data to server with each (up to) $10Gbit/s$. Using MoonGen would lead to very similar results but requires compatible NICs. TCP flow control and packet validation is done in software. The P4 switch aggregates these two flows on a single link which is sent through the DUT. With more servers an aggregated load of up to $100Gbit/s$ is possible together with the Linux TCP/IP stack and congestion control. Running load generation with multiple threads for each of the two ports has converged to a load of narrowly $20Gbit/s$.

The timestamps are stored inside the packets. This could be done either in the payload of UDP packets or as an additional TCP option. We decided to use TCP options as these do not influence the application layer. For that, a TCP option header will be added by the P4 program which carries the two 48-bit timestamps. The first timestamp is taken before the DUT and immediately added to the corresponding header field. The second timestamp is added after returning from the DUT and stored in the option field as well. Besides that, all packets are counted before and after the DUT in order to detect packet loss. The packets, including the additional TCP header, are duplicated in the P4 switch and forwarded to a control plane application over PCIe which captures the two timestamps per packet.

The average measured latency for a time period of $10s$ and different fiber length is depicted in Figure 3. Each of these measurements consists of around $16,000,000$ stamped and captured packets. The given latency is the average delay with a standard deviation of $1.5ns-1.7ns$ (see error bars in Figure 3). Linear regression of these points leads to a calculated speed of light in the fiber cables of $4.94ns/m$, validated by related work [2], and a constant offset of $383.49ns$. The average absolute residuum, representing the error of this approximation, is only $0.06ns$. Using MAC-timestamping instead of P4 pipeline timestamps would minimize this error further. The number of timestamped packets before and after the DUT is the same and by that the correctness of the loss detection is proven. With this demo we have shown how to (1) aggregate load of common servers, running as software load generators, to higher bandwidths, (2) how to measure latency with nanosecond accuracy and (3) to prove zero packet loss of a DUT on commodity hardware.

IV. Conclusion

With this work we have demonstrated the capabilities and flexibility of P4 hardware to support packet timestamping with nanosecond accuracy. On the example of fiber optical cables we are able to measure the speed of light with a measurement standard deviation of $1.5ns$ independent on the fiber length. Future work will be a fully-fledged load generator and timestamper based on P4 programmable standard hardware enabling flexible measurements under $100Gbit/s$ and above.

Acknowledgment

This work has been supported by Deutsche Telekom through the Dynamic Networks 8 project, and in parts by the German Research Foundation (DFG) as part of the project C2 within the Collaborative Research Center (CRC) 1053 MAKI. We thank our colleagues and reviewers for their valuable input.

References

[1] M. Primorac, E. Bugnion, and K. Argyraki, "How to measure the killer microsecond," in *Proceedings of the Workshop on Kernel-Bypass Networks*, ser. KBNets '17. New York, NY, USA: ACM, 2017, pp. 37–42. [Online]. Available: http://doi.acm.org/10.1145/3098583.3098590

[2] A. Singla, B. Chandrasekaran, P. B. Godfrey, and B. Maggs, "The internet at the speed of light," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XIII. New York, NY, USA: ACM, 2014, pp. 1:1–1:7. [Online]. Available: http://doi.acm.org/10.1145/2670518.2673876