Affordable Measurement Setups for Networking Device Latency with Sub-Microsecond Accuracy

Alexej Grigorjew[1], Philip Diederich[2], Tobias Hoßfeld[1], Wolfgang Kellerer[2]
[1]University of Würzburg [2]Technical University of Munich
{alexej.grigorjew, tobias.hossfeld}@uni-wuerzburg.de
{philip.diederich, wolfgang.kellerer}@tum.de

Abstract—**This document presents a networking latency measurement setup that focuses on affordability and universal applicability, and can provide sub-microsecond accuracy. It explains the prerequisites, hardware choices, and considerations to respect during measurement. In addition, it discusses the necessity for exhaustive latency measurements when dealing with high availability and low latency requirements. Preliminary results show that the accuracy is within ±0.02 µs when used with the Intel I350-T2 network adapter.**

I. Introduction

As Time-Sensitive Networking (TSN) and similar efforts for Quality of Service (QoS) have become more critical, the need for accurate latency measurements is also advancing to many new fields that were previously satisfied with raw throughput or statistical properties. Real-Time stream reservations are essentially requests for latency guarantees with a 100% certainty, which means that all sources of latency in the network must be well known and understood. Similarly, applications commit to a certain maximum burst size and traffic rate during reservation, which they must keep with a 100% certainty as well. This leads to a problem with black-box systems, for both network switches and end device equipment, as their manufacturers usually do not disclose their architecture and performance characteristics in the necessary detail. For a reliable implementation of hard QoS guarantees, it is often necessary to measure the internal latency and traffic characteristics of these components and to rely on these measurements during stream reservation.

One challenge of these measurements is that, due to the unknown underlying architecture, it is often unknown how networking devices react to a change of the configuration or traffic mix. Due to crowded lookup tables, new stream reservations may suddenly trigger a slower path in the switch fabric. A sudden population of high priority queues may come with a short-term higher delay due to low internal sampling rates of empty queues. Therefore, for each unique set of parameters (including traffic patterns and switch configurations), it is necessary to perform new, careful measurements that reflect the actual utilization pattern in practice, rather than being limited to artificial lab samples. New utilization patterns require new measurements frequently, but dedicated measurement hardware – as often used in literature – may quickly become very expensive. In this paper, the contribution is a general setup for latency and traffic measurements with affordable components. This paper includes a short guide for hardware selection, a few pointers on necessary considerations, and helpful tools.

II. Previous Measurements and related work

There are several works [1]–[6] that already take a look at different aspects of hardware switch performance. They consider different measurement and traffic scenarios. Furthermore, the measurements in the paper [1]–[6] mostly consider OpenFlow [7] setups and OpenFlow capable hardware.

The works in [8], [9] already evaluate the performance of different Openflow Hardware switches. The authors evaluate the raw processing delay of these hardware switches with different traffic loads and device configurations. Carefully chosen traffic allows [9] to calculate internal delays and parameters such as the overhead for the priority queueing implementation or buffer sizes of the switches. Both works [8], [9] use a measurement setup based on Layer 1 taps and a purpose-built measurement card.

In contrast to [8], [9], we adapt the setup and do not rely on purpose build measurement cards. Additionally, we plan to extend the measurements with more devices and other device types, such as TSN-capable switches and DPDK-based switch implementations.

III. Use Cases

The existing measurement studies on the performance of hardware switches [1]–[6], [8], [9], raise the question: Why are new measurements still necessary?

First, it is crucial for real-time reservation applications to exactly know what delays the hardware introduces. Only the hardware developers know the interplay of firmware and hardware to the necessary degree. Therefore, use-cases consider switches and networking devices as black-box devices. Different combinations of actions and configurations can produce vastly different performance results with black-box devices.

Second, manufacturers continuously release new devices with different features. Whenever a use case considers a new switch in a real-time network, performance benchmarks are necessary.

Third, recent use cases reveal an interest in new performance indicators other than maximum latency. For example, the latency variance (delay jitter) can be more relevant than the maximum value to some applications. Further, the influence of

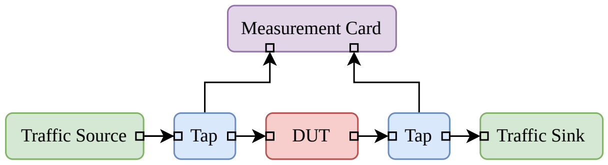

Fig. 1. Test setup to measure the delay that is added to packets by a Device Under Test (DUT) on their path.

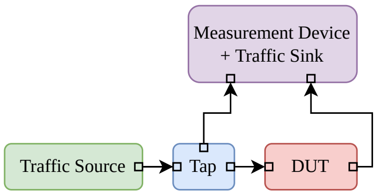

Fig. 2. Adjusted test setup where measurement is performed by the traffic sink device. This allows to use only a single tap.

higher priority real-time transmissions on lower priority Best-Effort traffic is an interesting new research opportunity, as converged networks with both types of traffic become more common.

In addition to latency measurements of switches, resource allocation requires accurate information about the traffic specification of each stream. The burst size and data rate typically determine a stream's traffic specification. While many applications can configure the burst and data rate of a stream, the end device equipment, e.g., network interface cards (NICs), can be just as intransparent as switches, and operating systems can add further delays. For example, the NIC could perform batch processing and change the burstiness of the traffic [8]. The resulting traffic specification after leaving the device is essential to know for resource reservation.

Finally, measurements are essential during the development of new types of forwarding devices. For example, there is still no hardware available that supports Asynchronous Traffic Shaping (ATS). A careful software implementation with network acceleration techniques such as DPDK can bridge this gap and enable flexible prototyping – which must also be measured accurately. Further, Linux's kernel Traffic Control (TC) module can be used for traffic shaping of end devices and prototype switches. The TC module supports many algorithms already, such as priority queuing (tc-prio, tc-mqprio) and token bucket shaping (tc-htb). Furthermore, the TC module can offload some operations to capable NICs for significant performance improvements. Software implementations can present viable alternatives, especially in cloud environments, but only after their performance and accuracy were evaluated thoroughly.

IV. Measurement Setup

This section describes the measurement setup commonly used for latency measurements, as illustrated in Figure 1. In general, a traffic source sends packets towards a traffic sink. These packets traverse the Device Under Test (DUT, e.g., a switch) on their path. The setup uses network taps before and after the DUT to mirror traversing packets to a measurement card. The measurement card receives the packets from both taps and notes their time stamps. Their accuracy is improved by leveraging the hardware time stamping capabilities of the measurement card. Note that Figure 1 shows two different devices for traffic source and traffic sink, but a single device can also be used for both.

Figure 2 illustrates a second measurement setup that requires only one network tap. The second setup colocates the traffic sink with the measurement device. Since hardware time stamping is independent of the underlying software, it is possible to use the same device for measuring and sinking traffic without losing accuracy. However, note that hardware time stamping is commonly only available for received packets. Therefore, the network tap before the DUT is necessary to achieve full accuracy. Without the first tap, the measurement would rely on the accuracy of the TX time stamp of an outgoing packet, which usually references the system clock in software. Dedicated measurement equipment may overcome this limitation but at a much higher cost.

In addition to device latency, it is important to measure the traffic specification (i.e., burst size and data rate) for traffic sources. Here, the traffic source is connected directly to the measurement card, which records the arrival times of the packets with hardware time stamps. Accurate results are essential to calculate the true rate and burstiness of the observed traffic.

V. Hardware Selection

In this section, important considerations during hardware selection are discussed, as well as some example devices, and useful workarounds.

In general, the traffic source, traffic sink, DUT, and the measurement computer can be chosen almost arbitrarily. The only restriction here is that the motherboard of the measurement device must be able to host the selected measurement NIC, e.g., it needs a free PCI-E slot and must have enough PCI-E lanes available on the CPU. During measurements, dedicated load generators can be used as the traffic source, such as MoonGen [10], iperf [11], and sockperf [12]. However, as mentioned earlier, the traffic characteristics can influence the performance of the black-box DUT. It is encouraged to use the actual applications or traffic traces from real deployments for measurements, if they are already available during testing.

A. *Measurement Card (NIC)*

While the measurement computer itself can be chosen almost freely, the measurement card on that computer must be chosen carefully to include certain characteristics. In general, we require a NIC with hardware time stamping capabilities to ensure that the operating system and the measurement software have no influence on the measured timings.

Typically, NICs that are advertised to support the Precision Time Protocol (PTP, IEEE 1588 and IEEE 802.1AS) generally support hardware time stamping as a prerequisite. However, time stamping is generally applied together with a

TABLE I
EXAMPLE NICS AND THEIR TIME STAMPING CAPABILITIES.

Hardware	BW	Driver	HWTSTAMP_FILTER_ALL
Intel i210	1 Gbit/s	igb	yes
Intel I350-T2	1 Gbit/s	igb	yes
Intel X520-DA2	10 Gbit/s	ixgbe	no (only PTP)
Intel X710	10 Gbit/s	i40e	no (only PTP)
Chelsio BT-520	10 Gbit/s	cxgb4	no (only PTP)
Mellanox ConnectX-4 Lx	10 Gbit/s	mlx5	yes

filter in the NIC drivers. Some devices are only capable of hardware-receive time stamps for PTP packets, most likely due to performance limitations. Table I lists six tested cards and their capabilites, as reported by the respective drivers. It can be difficult to identify cards that are capable of `HWTSTAMP_FILTER_ALL` a priori. In general, the safest approach is to check the respective driver's source code for hardware limitations. For example, for `ixgbe`, the supported controllers are explicitly stated in the source for `ethtool`'s info[1] and when actually setting the time stamp modes[2]. If the cards are already physically available, Section VI includes pointers to test for time stamping capabilities without looking at source codes. Note that, at the time of writing, the information is inconsistent in the `cxgb4` driver source. `HWTSTAMP_FILTER_ALL` is not listed as a capability, but still accepted as a socket option.

In addition to the general time stamping capabilities, it is important that both measurement points are based on the same time reference. This means that both interfaces of the measurement card must either be synchronized, or they must be controlled by the same local clock. The latter is preferred here, as using the same clock for both measurements gives more accurate results, and it avoids the necessity for time synchronization altogether. With most NICs, this is the case when two interfaces are controlled by the same controller chip on the card. For example, the I350-T2 can have up to four ports with two controllers on the card. For delay measurements, it is necessary that both network taps are connected to the ports of the same controller.

Onboard NICs: In contrast to the full measurement setup with taps, the traffic specification (burst, rate) of an application can be measured by a single interface that is connected directly to the traffic source. It is important to note that the accuracy of such measurements can be improved significantly if hardware time stamping is used. Note that no additional hardware must be purchased for this measurement. Many onboard Ethernet interfaces already support hardware time stamping out of the box, especially motherboards with Intel chips.

B. Network Taps

Network taps are used to replicate all packets going out of one (net) port to a second (tap) port. This is mostly used for monitoring purposes, including latency measurements. Unlike mirror ports in Ethernet switches, fast network taps save some latency either (i) by physically replicating the electrical signals on layer 1, or (ii) by performing cut through switching instead of store and forward switching, essentially beginning to forward packets already before they are fully received by the ingress. In that regard, having consistent delays (i.e., no jitter) is more important than having the lowest delay taps, as measurements can be calibrated to adjust their results to a constant delay. In that regard, we successfully tested Dualcomm's ETAP-XG, providing constant latencies after start-up.

Compared to hardware time stamping NICs, network taps can be more expensive and are less often already available. However, building your own 100 Mbit/s Ethernet tap is rather simple. Unlike 1G and 10G Ethernet which operate all phases in full duplex, 10Base-T and 100Base-T Ethernet use dedicated receive (RX) and transmit (TX) wires inside the RJ45 connector. Using RJ45 breadboard adapters and some resistors, it becomes simple to connect the RX wires of one interface to a second interface for mirroring. It is possible to implement a 100 Mbit/s tap using breadboard equipment worth less than 20 €. Note that such a setup may require some tinkering until the signal quality is sufficiently robust for 100 Mbit/s. In addition, note that it might be possible to build your own 1G tap by implementing the telephone hybrid circuit for all four wire pairs, but this has not been attempted yet.

VI. USEFUL TOOLS AND COMMANDS

This section includes useful tools to check the capabilities of available hardware, and it describes how to start a measurement by very simple means.

First of all, `ethtool` can be used to display many capabilities of a NIC. Most notably, `ethtool -T <interface>` shows all supported time stamping capabilities as reported by the driver. If this list includes `hardware-receive`, the device *should* be able to perform hardware measurements. The output of `ethtool` also includes the section `Hardware Receive Filter Modes` with two interesting values. First, `HWTIMESTAMP_FILTER_ALL` indicates that the card can time stamp every packet it receives. This is the desired filter value, which is also used by `tcpdump` and `libpcap`. Second, there can be many different `HWTIMESTAMP_PTP_*` filters. If the output only includes PTP filters, the driver will most likely not accept the `ALL` filter as socket option. For some notable examples, please refer to Table I.

Hardware time stamping can be enabled by setting socket options[3] in C/C++. But instead of writing your own program, `tcpdump` can be used to capture hardware time stamps as well, by adding the arguments `--time-stamp-type adapter_unsynced` and `--time-stamp-precision nano` to the command. The output of tcpdump can directly be processed further, or the resulting `pcap` file can be read and analyzed afterwards, for example by python/scapy.

As mentioned earlier, it should be verified that both interfaces used for measurement belong to the same controller.

[1]https://github.com/torvalds/linux/blob/a32e7ea362356af8e89e67600432bad83d2325da/drivers/net/ethernet/intel/ixgbe/ixgbe_ethtool.c#L3152

[2]https://github.com/torvalds/linux/blob/a32e7ea362356af8e89e67600432bad83d2325da/drivers/net/ethernet/intel/ixgbe/ixgbe_ptp.c#L1041

[3]`SIOCSHWTSTAMP` and `SOF_TIMESTAMPING_RX_HARDWARE`

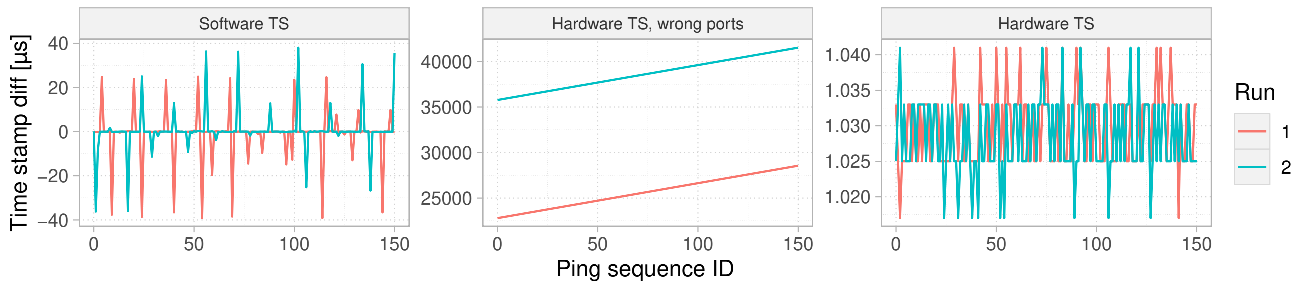

Fig. 3. Time difference of packet arrivals between the two taps for different time stamping modes.

For a PCI-E connected NIC, this can be verified directly with the interface name. For example, `enp5s0f0` and `enp5s0f1` belong to the same controller, as both have the same major ID 5, while `enp3s0f0` would be a different controller. In addition, `lspci` can be used to check the IDs of the PCI-E entries directly. Major IDs indicate the controller, while the minor IDs typically represent ports connected to those controllers.

Finally, the code snippets that have been used for the testing and results in this paper are available on GitHub[4].

VII. Preliminary Results

This section describes preliminary measurement results to show the difference between the time stamping options. The setup is similar to the measurement setup in Fig. 1. The only change is that there is no DUT, only a cable connecting two Dualcomm ETAP-XG network taps. The taps connect to two ports of an Intel I350-T2, working as the measurement device. During the measurement, the traffic source sends ICMP pings to the sink. Since there is a direct cable between the two taps, the measurement card should receive both tapped packets nearly at the same time. This way, the general accuracy of the setup can be measured for calibration.

The evaluation includes three time stamping modes: software time stamps, hardware time stamps with interfaces connected to different controllers, and hardware time stamps with both interfaces connected to the same controller on the Intel I350-T2. Figure 3 shows the measured differences between both receive time stamps for 150 consecutive measurements from two different runs for each case.

First, the plot on the left illustrates the difference between the packet arrival times with software time stamps. Most packets have a difference of approx. ±20 ns. When the reported delay is this low, the packets were likely dequeued in the same batch and time stamped immediately after each other, but their true delay may be higher. Further, the results include both positive and negative numbers, which is caused by the jitter introduced by the NIC and the operating system. Sometimes, the delay varies up to ±40 µs. The high variance in the measurements renders this time stamping method useless for precise delay measurements of DUTs.

[4]https://github.com/lsinfo3/hwtstamp-snippets

Second, the plot in the middle shows the results for hardware time stamps with different controllers. In this test case, the network taps connect to interfaces on the same card. However, two different controllers on the I350 serve these interfaces. The resulting slope represents the relative drift of the two clocks in the controllers. The offset between runs 1 and 2 supports this thesis since the second run was measured afterwards, and the clocks drifted further apart. By comparing the absolute time stamps for an extended period of time, it was apparent that one clock was slightly slower than intended, while the other one was slightly faster. Both clocks drifted by approx. ±20 µs per second, which is an inaccuracy of 0.002%.

Third, the plot on the right visualizes the time stamp difference with hardware time stamps on the same controller. This plot shows a small and constant delay of approx. 1.03 µs. The results vary up to ±16 ns in discrete steps of 8 ns. Power-cycling the taps changes the delay randomly, but afterwards, it remains within ±16 ns during the measurements. This means that prior to measuring the latency of the DUT, a calibration measurement is always necessary with directly connected taps in order to assess this random offset and adjust the measurement accordingly. Note that, when measuring with 100 Mbit/s, the observed jitter increased to a discrete ±80 ns, most likely due to adjusted internal polling rates in the NIC. The results show that accurate delay measurements are possible with commercial NICs instead of dedicated measurement cards.

VIII. Conclusion

This document presented a short overview of affordable network latency measurement setups by using hardware time stamping NICs instead of dedicated measurement cards. It explained the relevant hardware considerations and useful tools to test and select measurement hardware. In addition, it emphasized the necessity of exhaustive latency measurements in the field of hard QoS guarantees and stream reservation. The preliminary results indicate that, after a short calibration phase, accurate one-way latency measurements of black-box devices are possible within ±0.02 µs accuracy.

Acknowledgment

This work was partly funded by the Deutsche Forschungsgesellschaft DFG (German Reasearch Agency) under Grant No 316878574.

References

[1] M. Appelman, M. De Boer, and R. Van Der Pol, "Performance Analysis of OpenFlow Hardware," University of Amsterdam, Tech. Rep., Feb. 2012.

[2] D. Chefrour, "One-Way Delay Measurement From Traditional Networks to SDN: A Survey," *ACM Computing Surveys*, vol. 54, no. 7, pp. 156:1–156:35, Jul. 2021. [Online]. Available: http://doi.org/10.1145/3466167

[3] L. C. Costa, A. B. Vieira, E. De Britto e Silva, D. F. Macedo, G. Gomes, L. H. Correia, and L. F. Vieira, "Performance evaluation of OpenFlow data planes," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, May 2017, pp. 470–475.

[4] R. Durner, A. Blenk, and W. Kellerer, "Performance study of dynamic QoS management for OpenFlow-enabled SDN switches," in *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)*, Jun. 2015, pp. 177–182.

[5] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an OpenFlow architecture," in *2011 23rd International Teletraffic Congress (ITC)*, Sep. 2011, pp. 1–7.

[6] P. Rygielski, M. Seliuchenko, S. Kounev, and M. Klymash, "Performance Analysis of SDN Switches with Hardware and Software Flow Tables," in *proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools on 10th EAI International Conference on Performance Evaluation Methodologies and Tools*, ser. VALUETOOLS'16. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), May 2017, pp. 80–87. [Online]. Available: https://doi.org/10.4108/eai.25-10-2016.2266540

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.org/10.1145/1355734.1355746

[8] N. Derić, A. Varasteh, A. Van Bemten, C. Mas-Machuca, and W. Kellerer, "Towards Understanding the Performance of Traffic Policing in Programmable Hardware Switches," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, Jun. 2021, pp. 70–78, iSSN: 2693-9789.

[9] A. van Bemten, N. Derić, A. Varasteh, A. Blenk, S. Schmid, and W. Kellerer, "Empirical Predictability Study of SDN Switches," in *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, Sep. 2019, pp. 1–13.

[10] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "MoonGen: A Scriptable High-Speed Packet Generator," in *Proceedings of the 2015 Internet Measurement Conference*, ser. IMC '15. New York, NY, USA: Association for Computing Machinery, Oct. 2015, pp. 275–287. [Online]. Available: http://doi.org/10.1145/2815675.2815692

[11] iperf3: A tcp, udp, and sctp network bandwidth measurement tool. https://github.com/esnet/iperf. Accessed: 2022-06-01.

[12] Mellanox: Network benchmarking utility. https://github.com/Mellanox/sockperf. Accessed: 2022-06-01.