Behind the scenes: what device benchmarks can tell us

Simon Bauer, Daniel Raumer, Paul Emmerich, Georg Carle
Technical University of Munich, Chair of Network Architectures and Services
Garching b. München, Germany
{bauersi|raumer|emmericp|carle}@net.in.tum.de

ABSTRACT

While software-based packet forwarding devices and middleboxes are gaining momentum, also device diversity increases. This also challenges the area of device benchmarking. In this paper, we present our new benchmarking framework for OpenFlow switches that is running just on commodity server hardware. We present results of benchmarks and discuss how benchmarks reveal information about inner details of devices. As case study we implemented a new test to determine queue sizes and service rates based on a simple queuing theory model.

CCS CONCEPTS

• **Hardware** → *Networking hardware*; • **Networks** → **Network experimentation**; **Network performance analysis**; *Network performance modeling*; *Network measurement*; *Packet-switching networks*;

KEYWORDS

OpenFlow benchmarking, SDN-CERT, performance benchmarks, feature tests, OpenFlow switches, performance measurements, queuing theory

1 INTRODUCTION

The IETF benchmarking methodology working group (bmwg) was founded in 1989 [3]. Since then, devices not just achieved a manifold increase in performance and in the range of functionality, but also internal complexity increased. Especially the trend towards softwarization also found its way into network interconnect devices [14]. One example for this trend is OpenFlow (OF). However, devices that provide functionality in software, which may be provided in order to make them a fully OF conform device, may perform worse in those features then software OF switches on commodity servers [42]. Also the internal design that often explains performance limits is usually unknown to the users. In this paper, we describe different OF benchmarks and discuss what benchmarks can tell beyond pure measurement results intended for comparison of OF switches.

The remainder of this paper is structured as follows. In Section 2, we briefly sketch the state of the art for OpenFlow switch benchmarking. Section 3 describes our software framework for OpenFlow switch benchmarking running on commodity hardware. In Section 4, we discuss benchmarking results of different hardware switches, software switches and hybrid switch implementations. Section 5 introduces OpenFlow-specific tests regarding feature matches and actions, as well as flow table size. Afterwards, we describe a case study in which we implemented a test that assumes an internal queuing model in order to estimate the buffer size of a device. In Section 6, We evaluate the test by controlled changes in the queue size of tested devices. We conclude our discussion in Section 7.

2 BACKGROUND

General performance indicators for network interconnect devices were defined in RFC 1242 [4]. In 1999, the bmwg published RFC 2544 [6]. RFC 2544 has developed into the standard for the benchmarking of network interconnect devices (e.g. [7, 31, 44]). Since then, the bmwg has added recommendations for benchmarking of special functionalities, e.g. RFC 2889 [25] for switching devices, RFC 2647 [30] and RFC 3511 [19] for firewall benchmarks, RFC 3222 [45] for FIB performance, RFC 6201 for reset characterization, and RFC 5180 [37] for IPv6 protocol support, or to clarify, e.g. RFC 6815 [5] concerning the requirement for isolated test environments. Documents from other organizations, such as ITU-T Y.1564 [20] (EtherSAM), MEF 14 [26], or ETSI TST WG [13], provide additional comments on methodology or partially redefine these benchmarks.

Methodologies, like the aforementioned RFC 2544, RFC 2889, RFC 3222, and RFC 6201, presume the view of data plane devices as boxes that provide services in hardware. Today,

ANRW '18, July 16, 2018, Montreal, QC, Canada

ACM ISBN 978-1-4503-5585-8/18/07…$15.00
https://doi.org/10.1145/3232755.3232776

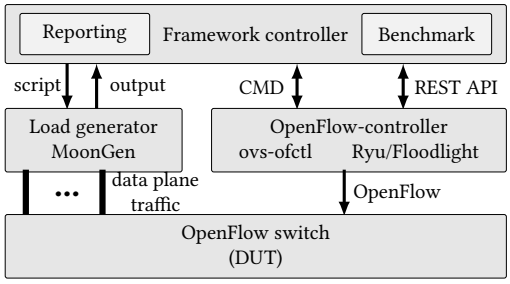


Figure 1: Framework architecture

members in the bmwg see need for benchmarks of upcoming components implemented in software. In order to benchmark SDN components e.g., there is ongoing work on recommendations for controller benchmarks [1]. But also other software based network components including VNFs, the underlying infrastructure, like the virtualization platform, and virtual switches are addressed.

3 OPENFLOW SWITCH BENCHMARK

3.1 Architecture

The framework has a modular design to mitigate bad maintainability and debugging properties of our monolithically designed previous benchmarking framework that was also capable of running on commodity hardware (cf.[39]). Architecture and configuration is inspired by the Testing and Test Control Notation Version 3 (TTCN-3) which is a standard managed by the European Telecommunications Standards Institute (ETSI).

The key components of the framework are shown in Figure 1. The framework controller orchestrates the benchmarks. Test series are defined via configuration files which also specify conditions for tests based on outputs of previous tests, as well as OF rules required for a certain benchmark. Each test consists of two steps: a feature test tests for support of the feature under test and the actual performance test. The controller provides two ways of configuration via OF: active (Ryu and Floodlight) and passive mode (ovs-ofctl). While the use of a single controller would be sufficient in theory, we e.g. were required to use Floodlight in order to program the tested MikroTik CCR1036 OF switch [38]. The ability to run on commodity hardware is supported by the load generator MoonGen [12]. SDN controller, benchmark controller and load generator can be run on different physical machines. A reporting module generates reports in LaTeX/PDF with visual representations of the results.

3.2 Comparison with other frameworks

No tests for performance, but only tests for functionality are defined by the Open Network Foundation (ONF) in context of the OpenFlow Conformance Certification [32]. Also from other organizations, until today, no recommendations for performance tests of OF switches exist. Vendors such as Ixia [21], Spirent [43], and Xena [46] offer benchmarks for OF switches. In the last years, also software based approaches running on commodity hardware were shown to produce benchmarks at comparable quality [39]. This has made it feasible for academia and other institutions with limited technical resources to contribute and to create their own solutions for network device benchmarking. Rotsos et al. [41, 42] use a NetFPGA for OFLOPS – an evaluation framework for OF devices. By using NetFPGA's capabilities flow installation time can be measured in the sub-millisecond range. Lin et al. [23] created OFBench a framework that uses metrics like action time, pipeline time, buffer size, pipeline efficiency, and timeout accuracy to benchmark OF switches. Rosa et al. [40] described Gym, a benchmarking framework with focus on VNFs deployed on top of Open vSwitch. In the most recent version our framework stands out by tests including OF groups, VLAN, and OF meters, as well as a test for buffer size. Beyond these solutions to benchmark OF switches, frameworks for other devices (e.g. routers [39]) or manual executed tests (e.g. OF for amplification of load [11], or a study of flow table sizes on OF switches [22]) exist and propose valuable methodology, tests, and metrics. However, by 2018 there is still no commonly accepted view on how to benchmark OF switches. The created isle solutions contradict the general benchmark goals to be of value: the benchmark outcome reflects real-world device behavior, reproducibility, i.e., other organizations can reproduce and verify published benchmarks, and comparability, i.e., we can directly compare the benchmarks executed by different organizations or testers.

4 FIELD STUDY

This section introduces benchmarking results generated with our framework. First, we introduce tested switch implementations. Afterwards, we present results from performance measurements including the impact of varying packet sizes. Results of performed measurements are publicly available [2].

4.1 Tested devices

Measurements are done with different types of OF switches, i.e., hardware switches, software switches and hybrid approaches. Tested hardware switches are a NEC PF5240 [29], a HP 3800 [18], a Dell S3048-ON [8], and its more powerful pendant S4048-ON [9]. All mentioned HW switches are capable of using an active OF controller. The tested NEC PF5240 and HP 3800 each offer 48 1 Gigabit Ethernet (GbE)

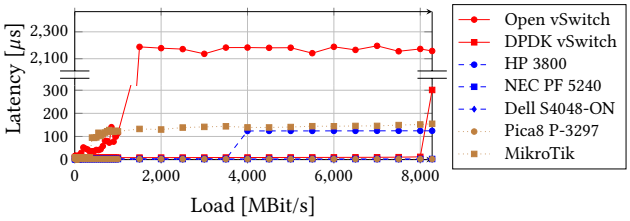


Figure 2: Latency comparison

ports and four 10 GbE ports. The Dell S4048-ON provides 48 10 GbE and 6 40 GbE ports. Software switches are represented by the open source implementation Open vSwitch, with DPDK (DPDKvS [34]) and without DPDK (OvS [33]). OvS is frequently used in cloud scenarios as cost-effective replacement of expensive hardware appliances. DPDKvS replaces the less efficient NAPI and kernel network stack by the high performance packet processing framework DPDK. DPDK has been shown to provide very efficient packet I/O, respectively packet processing in the past [15] and speeds up Open vSwitch significantly [16]. OvS and DPDKvS are tested on a server equipped with a 2.00 GHz Intel Xeon E5-2640 CPU, 16 GB DDR3 memory and an Intel X540-AT2 10 GbE dual port NIC. The host provides 32KB L1 cache, 256KB L2 cache, and 20 MB LLC. The used operating system is Debian based on kernel version 3.16. The used OvS is version 2.3.0, the used DPDKvS is version 2.5.2 based on DPDK version 2.2. No further modifications were made.Hybrid switch approaches promise to combine high flexibility by accessible software with performance characteristics of hardware switches. We tested two Pica8 switches, model P-3290 [35] and model P-3297 [36], and a MikroTik CR1036 [27]. The Pica8 P-3297 is based on a virtual application specific integrated circuit (vASIC) feature. vASIC allows to abstract the hardware for the software. This abstraction provides hardware independence. The MikroTik CR1036's approach is to parallelize packet processing without special hardware features, based on the RouterOS operating system. The Linux-based RouterOS allows to add functionality by modules. It has to be noted, that the OF module by the MikroTik CR1036 is only considered as experimental [27]. The used OpenFlow version for all tested switch implementations can be found in Table 1. As far as no other installed rules are mentioned, measurements are performed with a simple port forwarding rule between one input and one output port.

4.2 Benchmarking results

First, we analyze maximum throughput and average latency with minimum packet size, i.e., 64 B. Minimum packet size is assumed to be the heaviest load for packet switching, as matching header fields is independent of the packet size. Regarding maximum throughput, we measure switches with

Table 1: Maximum throughput of tested switches

Device	Device class	OF version	Max. throughput
Open vSwitch	software	1.5	1.95 Mpps
DPDK vSwitch	software	1.5	14.88 Mpps
HP 3800	hardware	1.0	6.44 Mpps
NEC PF 5240	hardware	1.0	14.88 Mpps
Dell S3048-ON	hardware	1.3	14.88 Mpps
Dell S4048-ON	hardware	1.3	14.88 Mpps
Pica8 P-3290	hybrid	1.4	14.88 Mpps
Pica8 P-3297	hybrid	1.4	14.88 Mpps
MikroTik CCR1036	hybrid	1.0	14.88 Mpps

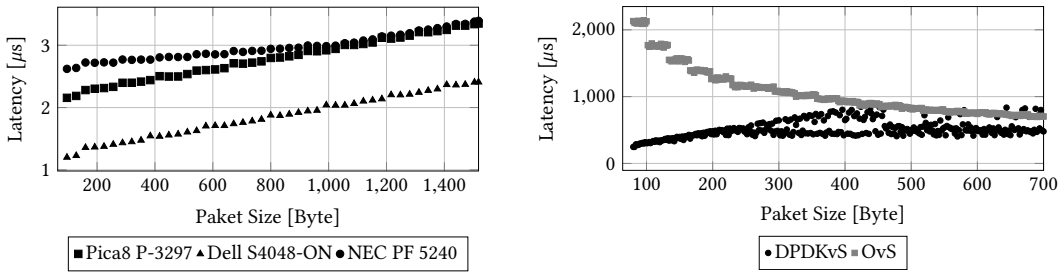
increasing load. The last load processed without packet loss is defined as maximum throughput [6]. We measure on 10 GbE ports. Except the HP 3800 and OvS all tested switches achieved line rate. In difference to OvS, DPDKvS is capable to process line rate due to its efficient packet I/O. Table 1 shows maximum throughput for all tested switches.

The measured average latencies for tested switch implementations differ significantly. The hardware switches forward with lowest latency. This HP 3800's overload correlates to significantly increased latency. The Pica8 P-3297 is capable to process packets much faster compared to the tested MikroTik switch, which results in significantly higher latencyaccording to our measurements. OvS results in higher latency for lower and higher offered loads, compared to HW switches and DPDKvS. The latency measured for OvS peeks out as soon as the DUT gets overloaded. Figure 2 shows measured latencies for all switches at varying offered data rates.

4.3 Impact of varying packet sizes

To determine the impact of packet size, we measure throughput and latency with increasing packet size. We start measuring with 64 B, and increase packet size for each measurement iteration by 1 B until maximum packet size is reached. Steps of 1 B are smaller then e.g. suggested by RFC 2544 [6] Measurements are done at line rate, i.e., 10 Gbit/s.

Regarding throughput of software switches, no direct impact of packet size is expected. Extracting header fields and looking up flow table entries is independent of packet size. This is confirmed by all measurements. However, packet size directly limits maximum throughput by the quotient from line rate and packet size. As mentioned before, OvS and the HP 3800 are overloaded by small packet sizes. As soon as the packet size increases and the number of packets per second decreases accordingly, all switches are able to process line rate. For the HP 3800 this applies for packets larger than 256 B. OvS running on our commodity server is able to process at line rate for packets larger than 832 B. As expected,



(a) Latency of selected hardware switches, 32 B steps

(b) Latency of tested software switches, 1 B steps

Figure 3: Latency with varying packet size

all other switches are capable to process packets at line rate for all packet sizes.

Regarding latency, we found that the latency of hardware switches increases with packet size, as long as the DUT is not overloaded. The results of the hybrid switch implementations differ significantly from each other. The Pica8 P3297 shows similar performance as hardware implementations do. In contrast, the MikroTik switch results in significant higher latency. This can be explained by MikroTik's still-in-development OF module. A closer analysis latency shows a gradual increase of latency for the Dell S4048-ON and S3038-ON, the NEC PF5240, the Pica8 P-3290, and the Pica8 P-3297. This behavior is shown in Figure 3a. This gradual increase is explained by the ratio between CPU word length, respectively cache line size, and packet size. When packet size exceeds a multiple of CPU word length an additional cache access is required for each packet, compared to smaller packet sizes. Each additional cache access increases latency.

Results of the latency measurements for OvS and DPDKvS can be found in Figure 3b. The latency of DPDKvS increases with packet size and shows an increasing spread of measured latency, while the majority of results stays on a constant level. This spread of latency measurements can be traced back on different arrival times of packets that are processed in one batch and an increased usage of the maximum batch size per poll. A packet arriving early has to wait longer, compared to a packet arriving just before the whole batch is processed. However, DPDKvS results in significantly lower latency compared to OvS, especially for small packets. For OvS, latency decreases with increasing packet size. The DUT is overloaded for the offered load of 10 Gbit/s and small packets. Larger packets result in fewer packets to be processed per second. Fewer packets decrease computational load and mitigate overload. This causes lower latency.

5 OPENFLOW-SPECIFIC TESTS

5.1 Matches and actions

Next to packet size, the matches and actions a switch has to perform influence performance. Matching different header fields and combination of fields may result in different internal processing costs. The costs to switch a packet depend on the protocol, on the number and kind of featured header fields, and on the comparison of extracted header fields. Therefore, throughput and latency with different OF rules can give hints on the implementation of different matching algorithms in software and the support of rules as hardware implementations. Baseline measurements reveal that none of the tested switches suffers under poor performance for different combinations of considered header fields, except the HP 3800. The HP 3800 needs to perform comparisons in software as soon as MAC addresses are involved. An internal rate limit of 10 kpps is implemented on the HP 3800 to limit packet processing in software. This is purposed to avoid overload of the CPU, since this leads to unresponsiveness to control commands.

Analogous to matches, per packet processing costs change with performed actions. Additional workload arise by write accesses to header fields and the calculation of checksums which are required after modifying a packet header. We run measurements for the software switches OvS and DPDKvS with different combinations of modified header fields. Test load is generated in shape of Ethernet frames containing an IPv4 packet with an UDP packet as payload. These protocols and corresponding protocol headers imply different costs to recalculate checksums. Dependencies between checksums and packet headers of the different protocols can be found in Table 2. Figure 4 shows throughput and latency for different test cases which are represented in Table 3. Our measurements show that throughput and latency of software switches suffer under additional effort due to expensive

Table 2: Dependencies between protocol checksums

Modified header	Recalculated checksums		
	Ethernet	IPv4	UDP
Ethernet	x		
IPv4	x	x	x
UDP	x		x

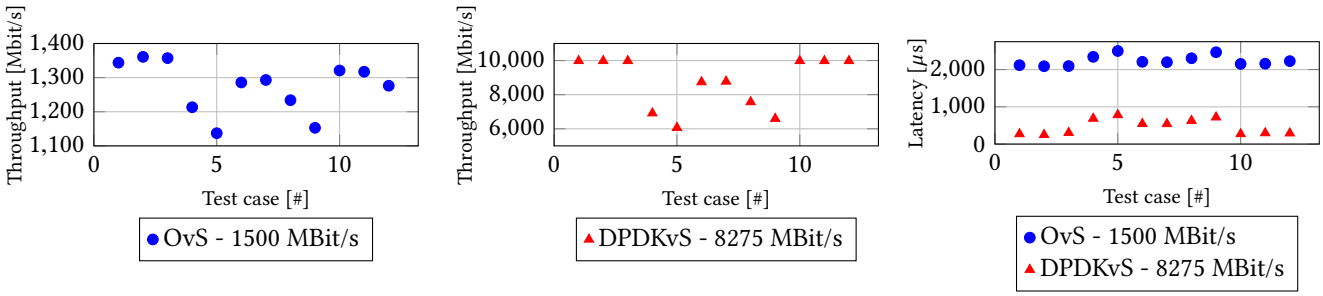


Figure 4: Impact of different combinations of matches on tested software switches

Table 3: Tests for measurement of impact of matches

Test case	Fields	Test case	Fields
1	SRC MAC	7	DST IPv4
2	DST MAC	8	SRC & DST IPv4
3	SRC& DST MAC	9	IPs+Ports
4	MACs+IPs	10	SRC Port
5	MACs+IPs+UDPs	11	DST Port
6	SRC IPv4	12	SRC & DST Port

Table 4: Measured flow table sizes

Switch	Measured flow table size
HP 3800	ca. 1027 rules
NEC FP 5240	ca. 1798 rules
Pica P-3290	ca. 2052 rules
Pica P-3297	ca. 3588 rules
Dell S3048-ON /S4048-ON	ca. 488 rules
MikroTik Router	ca. 80 rules
Open vSwitch	> 10K rules

checksum calculations. The calculation of the IPv4 checksum is particularly expensive, as all other headers are affected.

5.2 Flow table size

Another aspect of performance is the implementation and size of flow tables. Kuźniar et al. [22] already described that update time of flow tables increases with its size. However, here, we describe a test to determine the maximum size, i.e., the maximum number of installed rules. It is important to mention that a single OF rule may be realized internally via numerous entires in the flow table depending on the switch's implementation. When the need of flow table entries exceeds the flow table size entries get dropped (either old ones or the new one is not installed). If a rule has to be applied that is not installed, a request has to be sent to the OF controller or packets not matching the current set of rules get dropped. In both cases additional resources are consumed and packets get dropped. To determine the maximum flow table size we measure with a low data rate, i.e., 100 MBit/s. This is done to ensure that packet drops are caused due to limited flow tables. The workload generator is configured to increment the packet's destination IP addresses after each generated packet. We then increment the amount of installed rules for each measurement iteration. All rules are written to the same table and are based on destination IP addresses. If a switch's flow table is overloaded, packets will get dropped due to missing flow table entries. This allows a calculation of the flow table size by the formula: $s = r \cdot (1 - l)$, with s as the maximum number of flow table entries, r as the number of installed rules, and l as the rate of lost packets. This approach is based on the assumption that each rule has the same memory requirements.

Results show that software implementations support significantly more flow table entries. All measured maximum table sizes can be found in Table 4. It has to be considered, that measured flow table sizes are influenced by switch-specific interpretation and translation of configured rules. I.e., one installed rule can result in a different number of required flow table entries. To determine potential impact of the number of installed rules on throughput and latency, we measure switches with a data rate slightly higher than the determined maximum throughput. At the same time we increase the number of installed rules for each measurement run. This way performance deltas between single measurement points can be assigned to the number of installed flow table entries. Larger tables potentially consume more resources to look up entries in the flow table. Our measurements show that the performance of hardware switches is not affected by the amount of deployed rules. In contrast, OvS (with and without DPDK) suffers under increasing flow table entries. Throughput decreases most for the first added rules, while the additional resource consumption decreases with more rules.

6 QUEUE SIZE TEST

Models are used to estimate performance metrics like throughput, worst case or average latency, or energy consumption based on given configuration and input parameters. Such examples are network calculus [17] or queuing theory [24]). By

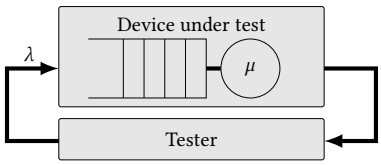


Figure 5: Simple -/-/1/K/∞/FIFO queuing model

simplifying real systems, models provide an affordable tool for the planning and the analysis of network components or infrastructures.

6.1 Queue size and service rate

Kendall's extended notation is used to describe queuing systems by $A/S/c/K/N/D$. A and S are used to describe the arrival process the service time distribution. For our test we assume a device consisting of one server, i.e., $c = 1$. Our test is purposed to determine the queue size K. The number of processed packets is unlimited, i.e., $N = \infty$, and packets are processed in FIFO order.

6.2 Test procedure

In the following we describe how to measure queue size. The test parametrizes a -/-/1/K/∞/FIFO queuing model where average service rate μ and queue size K have to be determined. Figure 5 shows an illustration of the introduced queuing model. To determine needed parameters, we assume that measured latency L consists of $L = T_{queue} + T_{process} + T_{other}$, with T_{queue} as the waiting time a packet incurs before being processed. Processing requires the time $T_{process}$. T_{other} contains delays like transmit or serialization delays. The average service rate μ can be directly measured as throughput, while the maximum queue size has to be determined by a test consisting of two measurements. First, we measure the latency L_{low} at an offered rate below maximum processing rate, this implies that the queue is empty, i.e., $T_{queue,low} = 0$. Therefore, we derive L_{low} from $L_{low} = T_{process} + T_{other}$. The second run measures the device with a filled queue. this second measurement results in L_{high}. Based on measured data, we calculate the waiting time with filled queue by $T_{queue,high} = L_{high} - L_{low}$, based on the assumption that $T_{process} = const$ With the waiting time T_{queue} we compute the queue size by $T_{process} = \frac{1}{\mu_{high}}$ and $K = T_{queue} * \mu_{high}$.

6.3 Evaluation

Table 5 shows the measured queue sizes for OvS with differently configured RX ring_sizes in the ixgbe driver. The expected value for the buffer size excluding the ring_size $E[K - ring\_size] = 2497$; The deviation from $E[K] + ring\_size$ is listed in the last column of Table 5 as error.

Table 5: Queue size test for Open vSwitch

RX ring $[\#pkts]$	L_{low} $[\mu s]$	L_{high} $[\mu s]$	μ $[pkt/\mu s]$	Measured queue $[\#pkts]$	Error
64	16	1731	1.50	2572	+ 11
512	16	2056	1.48	3019	+ 10
1024	16	2422	1.47	3536	+ 15
2048	16	3104	1.47	4539	- 6
4096	16	4541	1.45	6561	- 32

We also applied our queue size test to the switches introduced in Section 4. The test determined a queue size of 3268 packets for OvS DPDK, 550 packets for the HP3800, and 347 packets for the tested MikroTik device. For the other switches we considered for out performance measurements, the test could not be applied, as overloading test load could not be produced, which is required for the introduced queue size test.

6.4 Discussion

The applied model is a simplification of real systems. Therefore, results have to be interpreted carefully, e.g. for a system with parallel processing paths and load balancing to parallel queues. Also $T_{process} = \frac{1}{\mu_{high}}$ may be inaccurate, for example for systems which process packets in batches.

Beside a new and easy comparable metric, test data may be used as foundation for model based considerations. Our prototype test measures queues behind the primary processing bottleneck, i.e., the CPU. It differs from the approach discussed in the IETF bmwg [28] and by Lin et al. [23] which are based on burst size tests. The precise generation of a burst can be hard to implement, e.g., for software based load generators, and results of the back-to-back frame benchmark showed to be unstable for certain devices: especially software devices waking up from energy saving states.

7 CONCLUSION

This paper shows that benchmarks of OpenFlow switching devices are able to reveal more than meets the eye. We showed measurements that provide insights into performance aspects of switches like the cache line size and batching behavior, the impact of different match combinations, or to determine flow table size. In addition, our tests show performance deltas between different switch implementation approaches, i.e. hardware, hybrid and software switches. We contribute a test to determine a switch's queue size based on queuing theory. To support reproducible research, we contribute our framework in the used version [10] and all raw data to the community [2]. Our methodology, as well as our benchmarking framework, can be applied and extended for new devices and upcoming benchmark requirements.

REFERENCES

[1] Anton Basil, Mark Tassinari, Vishwas Manral, and Sarah Banks. 2018. Terminology for Benchmarking SDN Controller Performance - draft-ietf-bmwg-sdn-controller-benchmark-term-09. (2018).

[2] Simon Bauer, Daniel Raumer, Paul Emmerich, and Michael Remmler. 2018. Collection of OpenFlow-Switch Performance Test Data. https://mediatum.ub.tum.de/1439119. (2018).

[3] bmwg 2018. BMWG history. https://datatracker.ietf.org/doc/charter-ietf-bmwg/history/. (2018).

[4] S. Bradner. 1991. Benchmarking Terminology for Network Interconnection Devices. RFC 1242. (1991).

[5] S. Bradner, K. Dubray, J. McQuaid, and A. Morton. 2012. Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful. RFC 6815. (2012).

[6] S. Bradner and J. McQuaid. 1999. Benchmarking Methodology for Network Interconnect Devices. RFC 2544. (1999).

[7] Cisco. 2012. Cisco Nexus 3548 Switch Performance Validation. http://www.cisco.com/c/dam/en/us/products/collateral/switches/nexus-3548-switch/white_paper_c11-716751.pdf. (2012).

[8] DELL EMC. 2017. NETWORKING S3048-ON SWITCH - Spec-Sheet. (2017).

[9] DELL EMC. 2017. NETWORKING S4048-ON SWITCH - Spec-Sheet. http://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Dell-EMC-Networking-S4048-ON-Spec-Sheet.pdf. (2017).

[10] Paul Emmerich. 2018. SDN-Cert. https://github.com/bauersi/SDN-Cert. (2018).

[11] Paul Emmerich, Sebastian Gallenmüller, and Georg Carle. 2016. FLOWer – Device Benchmarking Beyond 100 Gbit/s. In *IFIP Networking 2016*. Vienna, Austria.

[12] Paul Emmerich, Sebastian Gallenmüller, Daniel Raumer, Florian Wohlfart, and Georg Carle. 2015. MoonGen: A Scriptable High-Speed Packet Generator. In *Internet Measurement Conference (IMC) 2015, (IRTF Applied Networking Research Prize 2017)*. Tokyo, Japan.

[13] ETSI. 2016. GS NFV-TST 002: Report on NFV Interoperability Testing Methodology. (2016).

[14] Nick Feamster, Jennifer Rexford, and Ellen Zegura. 2013. The Road to SDN. *Queue* 11, 12 (Dec. 2013).

[15] Sebastian Gallenmüller, Paul Emmerich, Florian Wohlfart, Daniel Raumer, and Georg Carle. 2015. Comparison of Frameworks for High-Performance Packet IO. In *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*.

[16] Robin Giller. 2016. Open vSwitch with DPDK Overview. https://software.intel.com/en-us/articles/open-vswitch-with-dpdk-overview. (2016).

[17] Peter Heise, Fabien Geyer, and Roman Obermaisser. 2015. Deterministic OpenFlow: Performance evaluation of SDN hardware for avionic networks. In *Network and Service Management (CNSM), 2015 11th International Conference on*. IEEE, 372–377.

[18] Hewlett Packard Enterprise. 2016. HPE 3800 switch series - Data sheet. https://h20195.www2.hpe.com/v2/GetDocument.aspx?docname=4AA3-7115ENW&doctype=data%20sheet&doclang=EN_US&searchquery=&cc=ie&lc=en. (2016).

[19] B. Hickman, D. Newman, S. Tadjudin, and T. Martin. 2003. Benchmarking Methodology for Firewall Performance. RFC 3511. (2003).

[20] ITU-T. 2011. Y.1564: Ethernet service activation test methodology Recommendation. (2011).

[21] IXNETWORK. 2017. Datasheet: SOFTWARE DEFINED NETWORK (SDN) TEST SOLUTION. https://www.ixiacom.com/sites/default/files/2017-08/Ixia-T-DS-IxNetwork-SDN.pdf. (2017).

[22] Maciej Kuźniar, Peter Perešíni, and Dejan Kostić. 2015. What You Need to Know About SDN Flow Tables. *Lecture Notes in Computer Science (LNCS)* (2015), 13.

[23] Ying-Dar Lin, Yu-Kuen Lai, Chen-You Wang, and Yuan-Cheng Lai. 2017. OFBench: Performance Test Suite on OpenFlow Switches. *IEEE Systems Journal* (2017).

[24] Kashif Mahmood, Ameen Chilwan, Olav Østerbø, and Michael Jarschel. 2015. Modelling of OpenFlow-based software-defined networks: the multiple node case. *IET Networks* 4, 5 (2015), 278–284.

[25] R. Mandeville and J. Perser. 2000. Benchmarking Methodology for LAN Switching Devices, Address caching capacity. RFC 2889. (2000).

[26] Metro Ethernet Forum. 2005. MEF 14: Abstract Test Suite for Traffic Management Phase 1. (2005).

[27] MikroTik. 2015. Cloud Core Router CCR1036-8G-2S+ Brochure. https://i.mt.lv/routerboard/files/CCR1036-8G-2Splus-131030144844.pdf. (2015).

[28] A. Morton. 2018. Updates for the Back-to-back Frame Benchmark in RFC 2544. Internet-Draft (individual). (2018).

[29] NEC Group. [n. d.]. ProgrammableFlow PF5240 Switch - Product Information Slides. https://www.nec.com/en/global/prod/pflow/images_documents/ProgrammableFlow_Switch_PF5240.pdf. ([n. d.]).

[30] D. Newman. 1999. Benchmarking Terminology for Firewall Performance. RFC 2647. (1999).

[31] ntop. 2012. Whitepaper: RFC-2544 performance test - PF_Ring-DNA VS Standard network Driver. http://www.ntop.org/wp-content/uploads/2012/04/DNA_ip_forward_RFC2544.pdf. (2012).

[32] ONF. [n. d.]. OpenFlow®Conformance Certification. https://www.opennetworking.org/training-certification/product/. ([n. d.]).

[33] Open vSwitch. [n. d.]. - Production Quality, Multilayer Open Virtual Switch. http://openvswitch.org/. ([n. d.]).

[34] Open vSwitch. [n. d.]. Open vSwitch with DPDK. http://docs.openvswitch.org/en/latest/intro/install/dpdk/. ([n. d.]).

[35] Pica8 P3290. 2014. - Data sheet. http://www.blucorona.com/solutions/pica8/p-3290.shtml. (2014).

[36] Pica8 P3297. 2014. - Data sheet. https://www.pica8.com/wp-content/uploads/pica8-datasheet-48x1gbe-p3297.pdf. (2014).

[37] C. Popoviciu, A. Hamza, G. Van de Veldeand, and D. Dugatkin. 2008. IPv6 Benchmarking Methodology for Network Interconnect Devices. RFC 5180. (2008).

[38] Pull request. 2016. Floodlight controller, fix unreadable HelloMessage OpenFlow 1.3 with switches of older OpenFlow versions. https://github.com/floodlight/floodlight/pull/639. (2016).

[39] Daniel Raumer, Sebastian Gallenmüller, Florian Wohlfart, Paul Emmerich, Patrick Werneck, and Georg Carle. 2016. Revisiting Benchmarking Methodology for Interconnect Devices. In *The Applied Networking Research Workshop 2016 (ANRW '16)*. Berlin, Germany.

[40] Raphael Vicente Rosa and Christian Esteve Rothenberg. 2017. Taking Open vSwitch to the Gym: An Automated Benchmarking Approach. In *The Applied Networking Research Workshop 2017 (ANRW '17)*. Prague, Czech Republic.

[41] Charalampos Rotsos, Gianni Antichi, Marc Bruyere, Philippe Owezarski, and Andrew W. Moore. 2015. OFLOPS-Turbo: Testing the next-generation OpenFlow switch.. In *ICC*. IEEE.

[42] Charalampos Rotsos, Nadi Sarrar, Steve Uhlig, Rob Sherwood, and Andrew W Moore. 2012. Oflops: An Open Framework for OpenFlow Switch Evaluation. In *PAM*. Vienna, Austria.

[43] Spirent. 2015. Whitepaper: OpenFlow Performance Testing. https://www.spirent.com/~/media/White%20Papers/Broadband/PAB/OpenFlow_Performance_Testing_WhitePaper.pdf. (2015).

[44] The European Advanced Networking Test Center (EANTC). 2008. Whitepaper: Huawei Technologies Service Activation Using RFC 2544 Tests. (2008).

[45] G. Trotter. 2001. Terminology for Forwarding Information Base (FIB) based Router Performance. RFC 3222. (2001).

[46] Xena Networks. 2015. Whitepaper: OpenFlow Performance Testing. https://xenanetworks.com/openflow-performance-testing-white-paper/. (2015).