What You Need to Know About Optical Circuit Reconfigurations in Datacenter Networks

Johannes Zerwas, Wolfgang Kellerer, Andreas Blenk
Chair of Communication Networks, *Technical University of Munich*, Munich, Germany
{johannes.zerwas,wolfgang.kellerer,andreas.blenk}@tum.de

***Abstract*—Increasing demand for flexibility in datacenter networks has led researchers to propose various designs of adaptable topologies using optical circuit switching. However, reconfigurations interrupt data transmissions that can diminish the benefit of adapting physical networks. The state-of-the-art lacks detailed models of end-to-end reconfiguration characteristics of involved networking components like end-host NICs and switches. The measurements of five commercially available programmable NICs and switches demonstrate that end-to-end reconfiguration delays indeed exhibit variability across devices and settings. The results suggest, however, that their behavior can be modeled, which opens new opportunities for scheduling algorithms.**

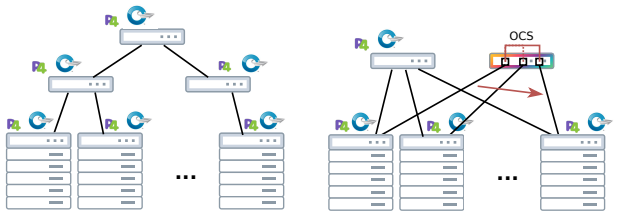
***Index Terms*—Programmable Networks, Optical Circuit Switching, Reconfiguration, Benchmarking, Model**

I. Introduction

The past decade brought up many proposals for flexible datacenter networks (DCN). One driver of flexibility are programmable switches and NICs in end-hosts, e.g., using OpenFlow [1] or P4 [2]. Yet another driver to cope with the increased demand for flexibility are adaptive topologies that use optical circuit switching [3]–[13]. Reconfigurable optical devices complement the programmability on control and data plane by a third dimension: the topology. An example is shown in Fig. 1(a) where a static topology uses multi-hop paths to provide connectivity between any rack pair. In contrast, the adaptive topology temporarily sets direct connections between the racks shifting whole link capacities at runtime (Fig. 1(b)).

Unfortunately, reconfiguring topologies comes at a cost: the time it needs to reconfigure the optical hardware. Meanwhile, data rates in DCNs are increasing to hundreds of gigabits. Hence, even interruptions of hundreds of milli-seconds can diminish the benefit of adapting networks. Indeed, optical hardware exists that can cycle through given configurations with reconfiguration times in the order of micro-seconds (e.g., Mordia [9], RotorNet [6]) or even nano-seconds (e.g., Sirius [10]). However, this kind of hardware is specialized and not widely available. Due to its cycling speeds, such hardware is also limited in its programmability. The schedules are fixed or can only be changed on a coarse time-scale [6], [10].

As an alternative to specialized hardware, milli-second switching times can also be realized with commodity-off-the-shelf (COTS) hardware. Optical circuit switches (OCS) provide reconfigurations in order of milli-seconds and are reconfigurable on-demand, such that a scheduling or matching algorithm can determine the next configuration based on the network state. Several prototypes based on COTS OCS were



(a) Programmable DCN with static topology. (b) Programmable DCN with adaptive topology.

Fig. 1. Programmable DCNs. (a) only control and data plane of packet switches are programmable. (b) control and data plane and topology are programmable.

built and their advantages are demonstrated, e.g., [3], [4], [12], [14]. In order to facilitate the demanded planning and to operate such reconfigurable DCNs (RDCNs) efficiently, solid understanding of the reconfiguration behavior, its stability, predictability and influence factors is necessary. Moreover, having access to a performance model of reconfiguration times can open possibilities to consider specific reconfiguration costs and, thereby, reduce negative impacts.

The state of the art provides only singular measurements of reconfiguration times for COTS devices [3]–[9]. Still, it has not been thoroughly explored if and how existing programmable networking equipment can cope with such reconfiguration. In particular, a benchmark across different devices and varying reconfiguration scenarios has not been done yet. This is critical as previous evaluations of programmable networking devices showed that they behave differently, e.g., with respect to OpenFlow updates [15]. Moreover, many theory-oriented works make varying assumptions about reconfiguration times and often base them on previous measurements or the data sheets of the OCSs [16]–[18]. Yet, these figures do not always consider end-to-end reconfiguration delay of optical links, i.e., the time until packets are received via the new circuit, and the delay on the control plane. In addition, they neglect potential variability, i.e., rely on the assumption that end-to-end reconfigurations are deterministic.

This paper evaluates the reconfiguration behavior of programmable optical networks, i.e., networks consisting of programmable switches, programmable end-devices and programmable optical hardware. To this end, a meta-analysis of existing work on topological reconfigurations in DCNs draws the landscape of such measurements. Based on the structure of a reconfiguration command, the paper elaborates potential

ISBN 978-3-903176-43-0

factors of variability for the reconfiguration delay such as number and frequency of reconfigurations. It contributes three measurement procedures that are tailored for two classes of devices: programmable switches and programmable NICs. They leverage the potential of programmable COTS equipment, i.e., no specialized measurement equipment is needed. The evaluation provides insights into the behavior of five programmable COTS networking devices under optical circuit reconfiguration and verifies the hypotheses about the influence factors. Thereby, it extracts the significant components of end-to-end reconfiguration delay. Indeed, the measurements demonstrate that end-to-end reconfiguration delays vary across data plane devices. While the effect on the data plane interruptions is limited, several factors, such as the number of circuits, affect the control plane delays. However, they can be used to predict reconfiguration times.

II. Related Work

Related work spans two domains. First, we review existing measurements of end-to-end reconfiguration delays and also compare these to delays used in more theory-oriented works. Second, a selection of performance modeling in the area of programmable networks is listed.

A. Existing Measurements with Optically-switched RDCNs

While there is no dedicated measurement study for circuit switching in RDCNs, different prototypes exist and are evaluated with respect to their reconfiguration delay. These measurement results can be classified into two groups: RDCNs that use (pre-)calculated schedules and RDCNs with programmable, on-demand reconfigurations. Table I overviews the measurements and the involved network devices.

1) RDCNs with pre-calculated schedules: Mordia [9] proposes a custom-built OCS that supports arbitrary reconfigurations. By opting for 2D-MEMS and WSSs instead of 3D-MEMS, Mordia achieves reconfiguration times of $11.5\,\mu$s on average. The prototype represents ToRs by hosts, which are equipped with Myricom 10G-PCIE-8B dual port NICs and DWDM transceivers. An FPGA-based controller communicates the active circuits and synchronizes the hosts via an out-of-band channel. The reconfiguration times are not further decomposed. ReacToR [7] extends the Mordia OCS prototype and emulates ToRs with FPGAs (HTG-V6HXT-100GIG-565). The reconfiguration times are in line with those from Mordia. A deeper analysis of potential influence factors on reconfiguration time is not given.

RotorNet [6] is another custom circuit switch implementation. Also here, host-based measurements of end-to-end reconfiguration times are performed using Myricom 10G NICs. They show delays of $150\,\mu$s. Out-of-band notifications synchronize the hosts and the circuit switch. RotorNet follows a demand-oblivious reconfiguration approach, i.e., reconfigurations are pre-determined and cannot be varied at runtime. Sirius [10] uses an optical fabric with passive components. Circuits are set up using arrayed-waveguide gratings and transceivers with tunable lasers that use certain wavelengths according to a pre-determined schedule. Connections are realized via FPGAs (Xilinx UltraScale VCU108). Tuning the lasers can operate at nano-seconds speed so that end-to-end reconfiguration times of $3.84\,\mathrm{ns}$ are achieved.

The aforementioned prototypes provide all end-to-end reconfigurations $<$ 1ms. However, they address a different use-case in which circuit scheduling is not performed on-demand but uses pre-defined schedules. Those schedules are either fixed or changed on a coarse time-level.

2) RDCNs with on-demand reconfigurations: The majority of programmable RDCNs with on-demand reconfigurations uses full crossbar, 3D-MEMS-based OCS or Wavelength Division Multiplexing-based (WDM) switching. The Helios prototype [3] uses a commercially available Glimmerglass OCS with 64 ports and connects Fulcrum Monaco switches with 10G ports. The authors report on several configuration changes and firmware modifications necessary on the switch such as deactivating debouncing and electronic dispersion compensation (EDC), which were conducted with support from the manufacturer. They drill down the reconfiguration time into two components: The reconfiguration time on the data plane is $27\,\mathrm{ms}$; the control plane adds another $170\,\mathrm{ms}$ in case of synchronous requests to the OCS and $30\,\mathrm{ms}$ in case of asynchronous requests. However, the measurements do not investigate how reconfiguration parameters or the networking devices affect end-to-end reconfiguration delay.

OSA [4] and WaveCube [12] use the same testbed. It consists of a Polatis Series 1000 32-port OCS and CoADna Wavelength Selective Switches (WSS). The measurements of the optical receive power report reconfiguration times of $14\,\mathrm{ms}$ for the WSSs and $9\,\mathrm{ms}$ for the OCS. The authors do not provide further insights into the end-to-end reconfiguration times nor details about the used NICs that connect to the optical fabric. Flat-tree [8] uses a commercially available 192-port OCS. Xia et al. estimate the end-to-end reconfiguration time based on observed TCP throughput to be between $2-2.5\,\mathrm{s}$. This includes also forwarding rule updates on the ToR switch. The authors do not report how individual components contribute to the end-to-end delay. Similarly coarse observations are provided in xWeaver [14]. Here, the observed time for the TCP throughput to restore after reconfiguration is around $300\,\mathrm{ms}$. Again, details of the used equipment are not provided.

In constrast to this, MegaSwitch [11] provides a detailed overview of the used devices. The prototype bases on Broadcom Pronto-3922 switches, running in Open vSwitch mode, with InnoLight 10GBASE-ER DWDM SFP+ transceivers. The custom-built optical fabric uses WSSs and is controlled by a Raspberry Pi. The authors split the total reconfiguration time into three components: the optical reconfiguration delay ($3\,\mathrm{ms}$), the added overhead due to flow updates on the packet switches ($13\,\mathrm{ms}$), and the control plane delay ($7\,\mathrm{ms}$). While this provides an initial model for the total reconfiguration time, it does not cover potential influence factors of the individual components.

ProjecToR [5] relies on free-space optics for switching and uses TI DLP Discovery 4100 kits with 0.7 XGA chipsets

TABLE I

OVERVIEW OF EXISTING RECONFIGURATION MEASUREMENTS: NAME, IF CIRCUIT SWITCHING IS PROGRAMMABLE, IF (TOR) SWITCHES ARE USED FOR AGGREGATION, THE ENDPOINTS OF THE LINKS, IF ONLY COMMERCIALLY AVAILABLE EQUIPMENT IS USED, MEASURED DATA PLANE RECONFIGURATION DELAY, AND IF RECONFIGURATION DELAYS ARE DRILLED DOWN.

Name	Programmable	Connects via switch	Data plane device (**S**witch/**F**PGA/**N**IC)	Uses COTS OCS	Delays	Drills down
Mordia	✗	✗	Myricom 10G-PCIE-8B (N)	✗	11.5μs	✗
ReacToR	✗	(✓)	HTG-V6HXT-100GIG-565 (F)	✗	11.5μs	✗
RotorNet	✗	✗	Myricom 10G NICs (N)	✗	150μs	✗
Sirius	✗	✗	Xilinx UltraScale VCU108 (F)	✗	3.84ns	✓
Helios	✓	✓	Fulcrum Monaco 24-port 10G (S)	✓	27ms	✓
OSA/WaveCube	✓	✗	some 1G NICs (N)	✓	9ms	✗
Flat-tree	✓	✓	Not specified	✓	$2 - 2.5$s	✗
xWeaver	✓	✓	Not specified	✓	300ms	✗
MegaSwitch	✓	✓	Broadcom Pronto-3922 (S)	✗	3ms	✓
ProjecToR	✓	✗	TI DLP Discovery 4100 (F)	✗	12μs	✗
This paper	n/a	✓/✗	multiple	✓	$7.5\text{ms} - 1.5$s	✓

as forwarding equipment. The reconfiguration measurements focus on loss of light duration due to mirror adjustments and are in the order of $12\,\mu$s. Measurements of end-to-end reconfiguration delays are not provided. Due to the distributed nature of ProjecToR, it differs from our use-case.

In conclusion, existing measurements for RDCNs with on-demand reconfigurations that rely on commercially available equipment consistently achieve reconfiguration times in the order of milli-seconds. The modeling and evaluation of impact factors is on a basic level and only individual points are provided. A comprehensive investigation and model are missing.

B. Reconfiguration Delays Used in Theory Papers

Prior algorithmic or theoretically-focused studies are oriented at measurement results such as listed above. Depending on which particular approach is targeted, the used reconfiguration delays vary. Works that aim at pre-calculating schedules consider the micro-second scale values. For instance, Solstice [16] uses $20\,\mu$s; Sunflow [18] considers delays between $10\,\mu$s and $100\,$ms. Similarly, Vargaftik et al. [17] use $20\,\mu$s and $20\,$ms depending on the evaluated switch. C-Share represents an approach which calculates circuits on demand. The used delay is $20\,$ms. In summary, these works are based on measurement results of other works and do also not consider potential variability of the reconfiguration delay in their evaluations.

C. Performance Models for Programmable Network Devices

Predictability and evaluation of performance models as a general concept has already been explored for other aspects of programmable networks. Harkous et al. [19] analyze the influence of P4 constructs on packet processing latency. Scholz et al. [20] model packet rates, latencies and resource consumption of two classes of P4 targets in dependence of matching types and table sizes of P4 programs. Katsikas et al. [21] provide such evaluations for programmable NICs and further evaluate impacts of reconfiguring such devices. Performance modeling has also been conducted for programmable control planes. For instance, van Bemten et al. [15] assess the predictability with respect to correctness and latency of reconfigurations of commercially available OpenFlow switches. Other works provide specific benchmarking and modeling tools for such software defined networks [22], [23].

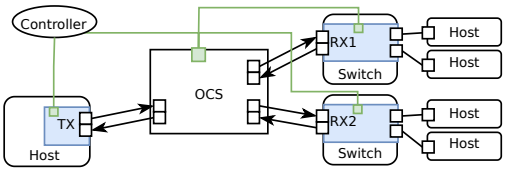


Fig. 2. Example of a programmable optical link. Three programmable networking devices (1 host, 2 switches) are connected via an on-demand reconfigurable OCS. An external controller handles all the components.

III. MEASURING END-TO-END RECONFIGURATION DELAY OF PROGRAMMABLE OPTICAL LINKS

The review of programmable, on-demand RDCN prototypes shows consistent values for the reconfiguration delays. But the end-to-end reconfiguration delay of programmable optical links has not been modeled in detail yet.

Fig. 2 shows a programmable optical link. The link is terminated by control- and/or data-plane programmable interfaces, e.g., an end-host-based (Smart-)NIC or FPGA (left), or an OpenFlow or P4-capable switch (right). The NICs are equipped with duplex optical transceivers to send and receive data via separate circuits. The optical link traverses an on-demand configurable (programmable) OCS. An external controller handles the components.

A reconfiguration takes several steps. First, the controller sends the new configuration to the OCS. After processing the command, the OCS sets the new circuit in place. The transmitter and receiver of the new circuit need to synchronize their clocks so that the receiver can correctly recover the signal [10]. Link connectivity is then communicated to higher layers and ultimately to the (forwarding) application. Eventually, the controller updates forwarding rules or configuration of the programmable devices. The exact process depends on the specific firmware and operating system of the device.

A. Modeling

This paper focuses on the optical switching part. Reconfiguration delays due to forwarding rule updates are out of scope here. Similarly to previous proposals [3], [11], the model contains two components for the end-to-end reconfiguration delay: data plane downtime t_o and control plane delay t_c.

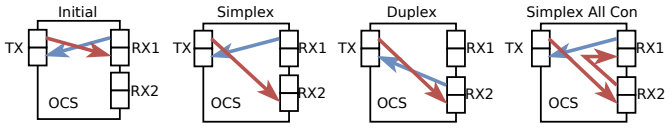


Fig. 3. Circuits and reconfigurations on the OCS. Ports TX, RX1 and RX2 connect to programmable networking devices. Each port consists of two fibers (in/out). The figure left (Initial) shows the initial circuits. The other three parts show the reconfiguration cases: *Simplex*, *Duplex* and *Simplex All Con.*

Since processing on the control plane happens before the data plane is interrupted, the sum of both provides the total delay:

$$t_r = t_o + t_c. \tag{1}$$

In programmable RDCNs, the reconfigurations can vary in several dimensions with potential impacts on the reconfiguration time. Influence factors can be all attributes of a reconfiguration command that can be modified by the controller. We consider as candidates the number of modified ports n and the distances between the ports d, which are reconfigured.

Moreover, the data plane downtime might depend on the used devices x (categorical). The review of related work has shown that two classes are being used: host-based and switch-based interfaces (cf. Fig. 2 left and right). Many proposed designs conceptually rely on a (programmable) top-of-rack switch as an aggregation point towards the circuit-switched fabric, e.g., [3], [11], [14]. Interestingly, directly connecting an OCS to the hosts can provide benefits in certain situations, e.g., for distributed machine learning [24].

Since the deployed transceivers are bidirectional, i.e., use two circuits, there are three ways of reconfiguration: *Simplex* (SI), *Duplex* (DU) and *Simplex All Con.* (SAC) (Fig. 3). The ports TX, RX1 and RX2 connect to optical transceivers, e.g., as shown in Fig. 2. SI changes only the circuit, which carries the traffic to RX2. This represents the case of setting unidirectional links, a widely adopted assumption in theoretic work [16]. Moreover, it is the least demanding reconfiguration as it requires only one change. Thereby, TX always receives optical power on its receiver. As a consequence, link establishment or failure routines as defined by IEEE 802.3 are not triggered on TX. However, some NICs might run these routines at RX1 leading to unintended behavior. For instance, shutting off the transmitter of RX1 results in loss of signal at TX which in turn stops transmitting [3]. To avoid such disconnects, DU configures both circuits to RX2. This is also a natural choice if traffic is transmitted bidirectionally. Finally, SAC constructs a ring over all involved transceivers with the same intention as DU, i.e., to avoid triggering link failure routines. However, it creates unidirectional circuits and thereby, provides more flexibility [3]. The reconfiguration type c is the fourth impact factor. Thus, we hypothesize as relation for the data plane downtime:

$$t_o = D(n, d, x, c). \tag{2}$$

The control plane delay might also depend on the modified ports n. Moreover, COTS OCSs usually come with a number

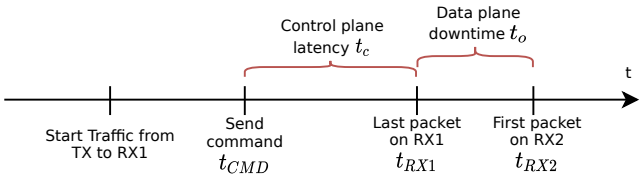


Fig. 4. Idealized timeline of the measurement. Characteristic duration of reconfiguration (red braces). RX1 and RX2 are the ports between the circuit is changed.

of control plane protocols p (categorical), e.g., TL1 [25] or Netconf [26]. We obtain for the control plane delay:

$$t_c = C(n, p). \tag{3}$$

The following measurements evaluate the hypothesized relationships above.

B. Measuring data plane downtime and control plane delay

1) Data plane measurements: To assess the hypothesized relations for the data plane interruption, the measurement applies the process as shown in Fig. 4. A traffic source sends a continuous stream of packets over an existing link from TX to RX1. The controller sends a command to reconfigure the circuit to another receiving port (RX2) at time t_{cmd}. The downtime of the data plane is given as

$$t_o = t_{\mathrm{RX2}} - t_{\mathrm{RX1}} \tag{4}$$

where t_{RX1} is the time of the last packet received by RX1 and t_{RX2} is the time of the first packet received by RX2.

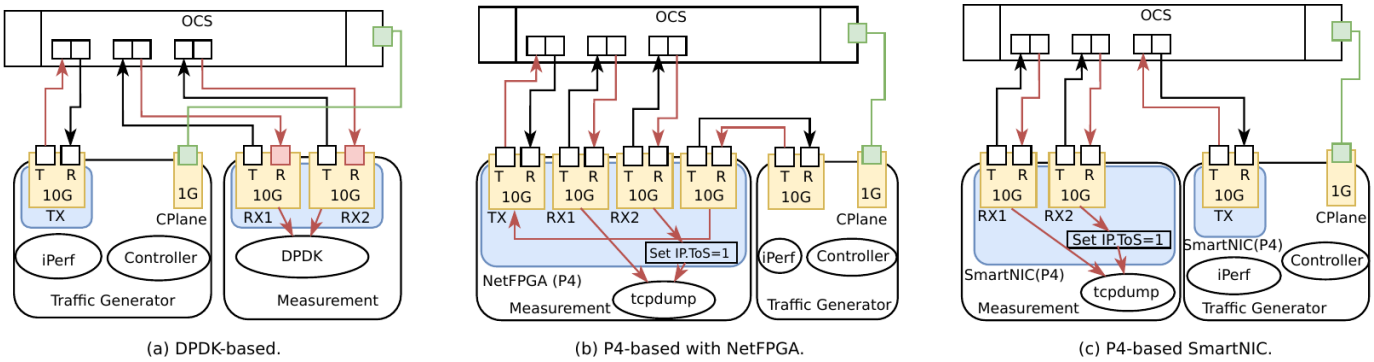
2) Control plane measurement: The measurement of t_c uses a similar approach as the data plane one. We exclude any timings needed calculating the new configuration, e.g., for collecting demand statistics or executing matching algorithms. Accordingly, we define t_c as shown in Fig. 1, from sending the command message t_{CMD} to the OCS until reception of the last packet on RX1:

$$t_c = t_{\mathrm{RX1}} - t_{\mathrm{CMD}}. \tag{5}$$

C. Testbed

The testbed is built around a Polatis Series 6000n OCS [27] with 32 in- and 32 out-ports. The OCS runs firmware version 6.7.5.34. The devices are use FS.com 10GBASE-LR SFP+ 1310nm (SMF) transceivers [28]. The measurements compare two classes of data plane devices: programmable NICs deployed in hosts and programmable switches. For the host-based cases, we consider three NICs with different expected levels of programmability (from low to high): a regular NIC (Intel X710-DA4 [29]) with DPDK [30] support, a SmartNIC (Agilio CX 2x10G [31]) with support for P4 and DPDK and an FPGA (NetFPGA-SUME [32]) with prepared support for P4 and the full programmability of FPGAs.

The switch-based measurements are conducted using two OpenFlow-based switches: Pica P3297 [33] and Dell S4048-ON [34]. The switches run in default configuration and only make simple forwarding decisions. The exact settings of the



(a) DPDK-based. (b) P4-based with NetFPGA. (c) P4-based SmartNIC.

Fig. 5. Setups for host-based measurements: Setups consist of a traffic generator, which includes also the OCS controller, and a measurement server. The measurement server hosts the NICs under test. Arrow heads indicate direction of optical light and packets. Red arrows indicate the packet streams that are to be measured.

measurement cases are detailed later (cf. Sec. IV-A1 for host-based and Sec. IV-A2 for switch-based).

Traffic and packet dumps are generated on two servers running Ubuntu 18.04 (4.15.0-47-generic kernel) with 128 GB of RAM and an Intel Xeon Silver 4114@2.2 GHz (20 cores). The traffic generating server also runs the controller of the OCS, which connects via a dedicated management network. It uses iPerf 2.0.10 to generate a single UDP flow consisting of 1000 B packets at a rate of 2 Gbps ($\approx$ 250 Kpps). This translates to a packet inter arrival time of $\approx 4\,\mu$s, which provides a sufficient temporal resolution for the reconfigurations, which are expected to be in the order of milli-seconds.[1] Presented results are based on 30 runs of at least 10 s duration.

IV. Evaluating Data Plane Downtime

We start with evaluating the data plane component. After introducing the detailed configurations, results for the data plane devices are compared and potential influence factors from Sec. III-A are analyzed.

A. Detailed Setup

The settings to measure data plane reconfiguration behavior in the host-based and switch-based cases vary slightly due to the capabilities of the considered devices. The general assumption is that the links are homogeneous, i.e., that both endpoints use the same device model. Combining different devices makes attribution of effects harder and is left for future work. Overall, this leads to three setups:

1) Host-based: The two data plane programming abstractions DPDK and P4 provide both sufficient precision for time measurements in the order of milli-seconds [35], [36]. However, neither DPDK nor P4 is supported by all of the devices under test. Hence, we propose two approaches and also evaluate the impact of using DPDK or P4 on the measurements. The SmartNIC supports both of them, which helps to put the obtained results into relation.

[1]Specifically, the data sheet of the OCS states a switching time of 25 ms.

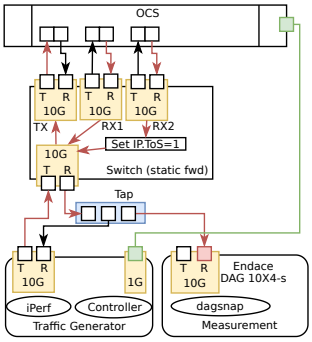
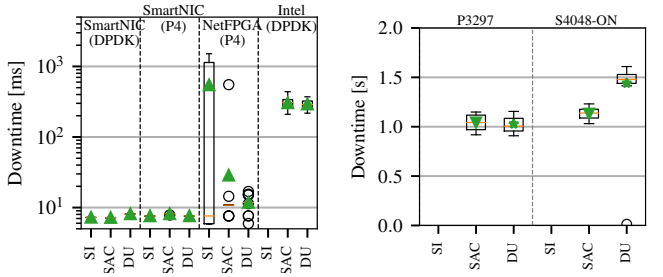


Fig. 6. Setup for switch-based measurements: It consists of the switch under test, a traffic generator, which includes also the OCS controller, and a measurement server. Arrow heads indicate direction of optical light and packets. Red arrows indicate the packet stream that is dumped by an Endace DAG 10X4-S.

a) DPDK (Fig. 5(a)): The first approach uses a custom DPDK [30] application to measure the data plane downtime. The application consists of two threads. Each thread polls bursts of packets from its assigned port, i.e., `RX1` or `RX2`. When packets from the traffic generator are received, the current cycle count (from the CPU) is read and saved as the last time a packet was received. Similarly, the timestamp of the first received packet per port is also stored. The data plane downtime is directly calculated from these two collected timestamps. The DPDK application uses the available poll mode driver of the NICs. It is available for the Intel NIC and the SmartNIC. However, it relies on SW timestamping since HW timestamping is not supported by the used Intel NICs.

b) P4 (Fig. 5(b) and Fig. 5(c)): The second approach uses P4. Thus, it is limited to the NetFPGA and the SmartNIC. The setups between these two differ due to the different number of ports of the cards. A simple application forwards packets from the traffic generator (from virtual port with SmartNIC or physical port with NetFPGA) via port `TX` towards the OCS. Packets received on either `RX1` or `RX2` are forwarded to a virtual port that is exposed to the operating system. The IP type of service field (ToS) is set to 1 when the packet is received on `RX2`. In addition, the application adds an In-band Network Telemetry (INT) header containing the time of the packet



(a) Host-based. (b) Switch-based.

Fig. 7. Data plane downtime. Comparison of the five devices, the reconfiguration cases (SI,DU,SAC) and measurement approaches (P4, DPDK). Only SmartNIC shows low downtimes of $\approx 7.5\,\text{ms}$ and low variance. Both switches show downtimes > 1s. SI did not succeed for the switches and the Intel NIC.

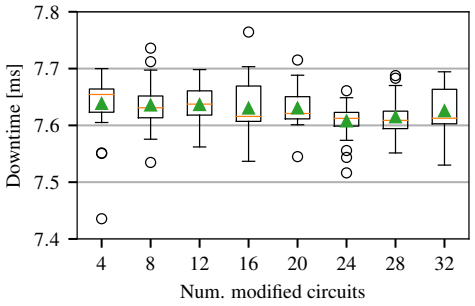


Fig. 8. Data plane downtime against num. of modified circuits. Downtime is measured with SmartNIC (P4), SI case. No significant impact is observable.

reception. Precision of the timestamp is $5\,\text{ns}$ on the NetFPGA[2] and $20\,\text{ns}$ on the SmartNIC[3]. Finally, the packets are collected using *tcpdump*. This setup provides two ways to obtain the downtime: from the timestamps in the INT data and from the timestamps in the packet dump. The downtime is given as the difference of the timestamp of first packet with $\text{IP.ToS} = 1$ and the timestamp of last packet with $\text{IP.ToS} = 0$. If not stated otherwise, the INT data is used.

2) Switch-based (Fig. 6): The setup for the switches follows a similar idea as that of the P4-based measurements with NetFPGA. The traffic generator sends traffic to the switch under test. The switch forwards the packets without modification on another port (`TX`) to the OCS. The OCS loops back to a third port (`RX1`) or fourth port (`RX2`). For packets received on `RX2`, the switch sets $\text{IP.ToS} = 1$. Independent of the port, packets are forwarded to the initial port. The outgoing direction of this port connects to a fiber tap and eventually to an Endace DAG 10X4-S to dump the packets. P3297 and S4048-ON realize this forwarding using OpenFlow rules.

B. Results: Comparing devices and reconfiguration cases

1) Host-based: Fig. 7(a) illustrates the data plane downtime for the host-based devices and the different measurement approaches. It further compares the three reconfiguration cases SI, SAC, DU, which results in 12 settings. Green triangles indicate the average values. We first note that only SmartNIC provides consistent values of $\approx 7.5\,\text{ms}$ for all three reconfiguration settings and also has low variance across the measurement runs. If only one circuit is modified (SI), Intel NIC loses connectivity, i.e., no packets are received on `RX2`. Hence, the downtime cannot be calculated. SmartNIC and NetFPGA have downtimes in the order of $10\,\text{ms}$. However, NetFPGA's behavior is hardly predictable. It shows high variability with values $> 100\,\text{ms}$.

In the DU case, all candidates manage the reconfiguration. SmartNIC achieves $7.6\,\text{ms}$ on average while NetFPGA is around $11.8\,\text{ms}$ on average. For SmartNIC, the difference between INT-based and DPDK-based values is about $0.5\,\text{ms}$ on average. Thus, both measurement approaches are viable here. For Intel, the average downtime is around $292\,\text{ms}$. This is significantly larger than the expected $25\,\text{ms}$ of the OCS. An explanation might be given by the specific firmware or driver, e.g., with slow implementations of link set up routines.[4]

2) Switch-based: Fig. 7(b) shows boxplots of the data plane downtimes for the switch-based measurements. We note that the SI case does not succeed for any switch. The switches turn off `RX1` and subsequently also `TX` so that no further packets are transmitted after the reconfiguration has finished. For SAC and DU, both switches show a data plane downtime > 1s on average and median. While for P3297 both timings are around $\approx 1.1\,\text{s}$, S4048-ON has larger downtimes for DU compared to SAC. The behavior is consistent over the runs but the reasons could not be clarified.

The conclusions for t_o of the switches are negative. Data plane downtimes above 1s translate to extreme losses of network capacity over time hindering the use of COTS packet switches on the edge of a circuit-switched or hybrid fabric. These observations confirm the ones from [3] and [11]. As suggested by [3] and also confirmed in discussions with two industry partners, this behavior is not related to the switching ASIC but roots in the firmware of PHY and MAC layer components, e.g., stems from tuning of physical link parameters during link setup.

C. Results: Analyzing Influence Factors

After evaluation of reconfiguration behavior of the five networking devices, we turn our focus on how the number of modified circuits and port distance impact the data plane downtime. Since SmartNIC performed best, we use it for the following analysis.

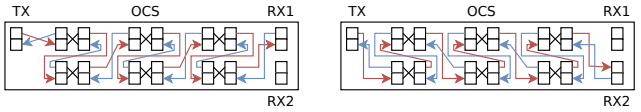
1) Number of modified circuits: Fig. 8 shows boxplots of the downtime against the number of modified circuits. The number of circuits ranges up to 32, which marks the maximum number of circuits that can be set simultaneously with a 32x32 OCS. The values of dataplane downtime range from $7.5\,\text{ms}$ to $8\,\text{ms}$. Moreover, the median value decreases slightly with the number of modified circuits. Fitting the following linear model

$$D(n) = \delta + \gamma \cdot n \tag{6}$$

[2]The value was retrieved from https://github.com/NetFPGA/P4-NetFPGA-public/wiki/Workflow-Overview#p4-netfpga-extern-library

[3]This value was received via correspondence with Netronome.

[4]For verification, we run the measurements with two firmware versions and contacted Intel to obtain explanations about this behavior, however without any success.



(a) Initial. (b) After reconfiguration.

Fig. 9. Setup for shoelace measurement. Black lines indicate fibers. Colored arrows are circuits. The optical link uses all ports of the OCS. The figure shows a sub-set of ports due to space limitations.

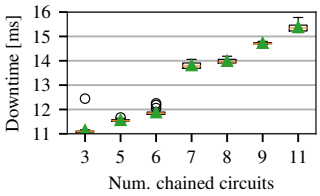


Fig. 10. Dataplane downtime against number of chained circuits in the shoelace setup. Downtime is measured with SmartNIC (P4), SI case. Downtime increases with number of chained circuits.

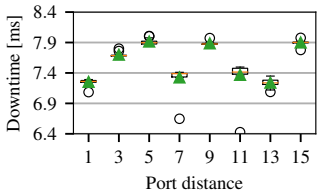


Fig. 11. Data plane downtime against the port distance. Downtime is measured with SmartNIC (P4), SI case. Port distance is viewed from outside perspective. No clear pattern is visible.

shows a slope of $\gamma = -8.39 \cdot 10^{-7}$ and an intercept of $\delta = 0.0076$s with $p = 0.0032$, which is significant for $\alpha = 0.05$. However, this slope is below the temporal resolution which is $\approx 4\,\mu$s according to the packet sending rate. Thus, this decrease has to be considered as measurement noise.

A natural question that arises when changing multiple circuits is whether these reconfigurations happen at the same time. The current setup can only measure downtime of a single optical link. Therefore, a setup similar as the shoe-lace proposed in [15] is used. Fig. 9 illustrates the circuits. The other components are the same as for the SmartNIC setup in Fig. 5(c). `TX` is connected to the traffic generator and `RX1` and `RX2` are connected to the respective ports of the SmartNIC on the measurement host. The idea is to redirect the signal through multiple ports with fiber loopbacks, and to request to reconfigure these ports simultaneously. Thereby, the downtime would increase if the reconfigurations happen sequentially in the OCS. Specifically, this kind of measurement provides the time between the first port being disconnected and the last port being connected.

Fig. 10 shows boxplots for the data plane downtime against the number of chained circuits, i.e., the length of the shoelace. For all settings, the variance in the downtimes is low, which states a deterministic behavior given the number of changed circuits. Moreover, an increase of the downtime with the number of chained circuits is evident. Specifically, chaining only three circuits the average value is $11.1\,$ms while the value is $15.3\,$ms for chaining 15 circuits. Considering also the number of modified circuits does not significantly impact the downtime for a single optical circuit, this means that the changes are applied sequentially.

2) Port Distance: A second parameter that can impact the downtime is the (physical) distance between the ports `RX1` and `RX2` at the OCS. Intuitively, one would assume a linear

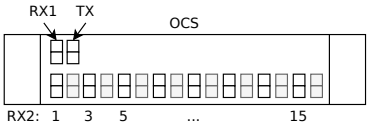


Fig. 12. Port numbers as used for the port distance measurement.

relationship between distance d and the data plane downtime:

$$D(d) = \delta + \gamma \cdot d. \tag{7}$$

Fig. 12 indicates `RX2` with increasing distances to `RX1` $(1, 3, 5...)$. The distances are given as the number of ports between `RX1` and `RX2` on the horizontal axis from an outside view of the OCS. The results (Fig. 11) show no clear pattern. The major part of the values lies between 7 and $8\,$ms. Fitting the model gives a slope of $\gamma = 6.81 \cdot 10^{-6}$ with $p = 0.273$ which is not significant. Thus, we cannot conclude if the port distance affects the data plane downtime.

Takeaway: Data plane devices react differently to OCS reconfigurations. While the NICs achieve small downtimes, the tested switches with default configurations are less suited for such scenarios. The considered influence factors do not show any significant impacts on the data plane downtime. The data plane component is considered as constant. However, we note that multiple reconfigurations happen sequentially.

V. Evaluating Control Plane Delay

While the downtime of the dataplane affects the actual resource usage, it is also important to consider the behavior of the OCS's control plane. Slow reaction here leads to delayed setting of the new circuits or requires issuing the command ahead of time so that changes are in place at the right time.

A. Detailed Setup

The delay on the control plane is measured using the same setup as for the P4-based measurements with the SmartNIC. In addition to the data plane, also the control plane traffic is captured using a 1G network tap and *tcpdump* (Fig. 13). The control plane delay as described in Sec. III-B can directly be calculated from the timestamps in the dump file. Comparing measurements of the SmartNIC showed only a small difference ($< 1\%$) between *tcpdump* and the P4 program. The evaluated protocols are TL1 [25], a widely used management protocol for the optical domain and the more generic network management protocols Netconf [26] and Restconf [37].

B. Results: Single Reconfiguration

1) Control Plane Protocols: Fig. 14 shows empirical CDFs of the obtained delays for the three control plane protocols. It also compares the results for running the OCS for > 180 days and for running it only a few days (< 5 days). In both cases, TL1 reacts significantly faster than Netconf and Restconf. After the restart (< 5 days), TL1 takes $85\,$ms on average to apply the changes while Netconf and Restconf take $245\,$ms and $257\,$ms. These values are in the same range as observed in prior studies [3]. Netconf and Restconf obtain almost the same timings. The reason for this is that both use the YANG

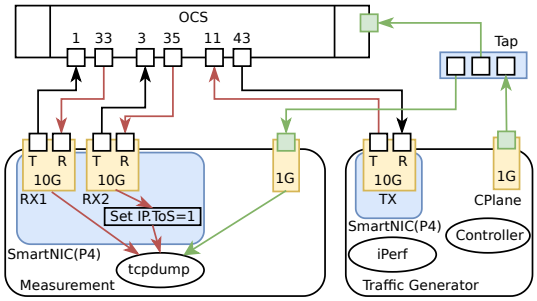


Fig. 13. Measurement setup for control plane case. The setup is the same as for the SmartNIC case. The control plane traffic is tapped and also dumped.

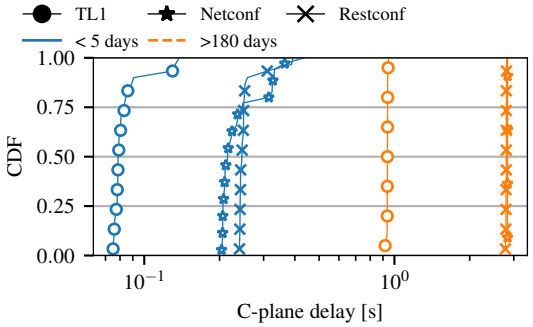


Fig. 14. CDF of control plane delay for three control plane protocols. TL1 performs significantly better than Netconf and Restconf.

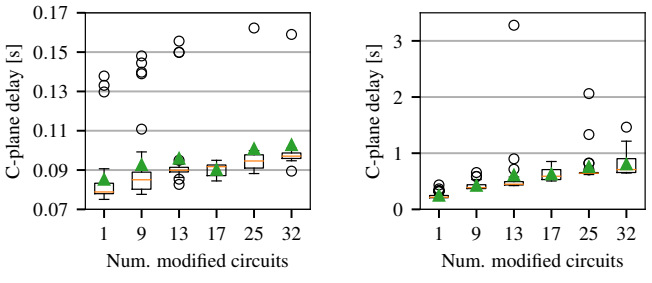


Fig. 15. Control plane delay against number of modified circuits. For TL1 and Netconf a linear relationship is observable.

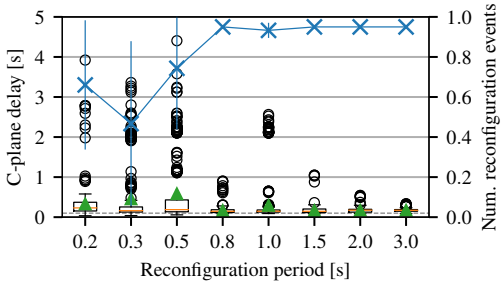


Fig. 16. Control plane delay against reconfiguration period. For each period, 20 consecutive reconfigurations of one circuit are executed. TL1 is used.

data model of the OCS and apart from the connections, *ssh* for Netconf and *https* for Restconf, are handled by the same code in the OCS's firmware. All control plane interfaces show low standard deviations of 16.6 ms for TL1, 57.7 ms for Netconf and 37.6 ms for Restconf.

Comparing the two temporal cases, there is a strong software aging behavior of the OCS: over 180 days of operation, the control plane delay increases by one order of magnitude. While this is mainly a quality management issue, operators need to account or monitor such aspects when deploying demand-aware RDCNs [38].

2) Number of Modified Circuits: As for the data plane considerations, the number of modified circuits might also impact the control plane delay. Fig. 15 shows boxplots of the control plane delay against the number of modified circuits, i.e., the control plane message size. For TL1, the mean values indicate a steady increase in the delay from around 84 ms for only one circuit being changed to around 102 ms when 32 circuits are changed. The behavior seems to be linear, so we assume the following relationship:

$$C(n) = \delta + \gamma \cdot n. \tag{8}$$

Performing linear regression results in values $\gamma = 0.000561$ and $\delta = 0.086393$ with a confidence of $p = 0.00341$ which is significant for $\alpha = 0.05$. For Netconf, a similar behavior is observable. The obtained parameters from the linear regression are $\gamma = 0.01779$ and $\delta = 0.279702$ with $p < 10^{-10}$. This relationship seems to be intuitive. However, knowing how the control plane behaves opens new possibilities for scheduling reconfigurations and for using available resources more efficiently. We also note some severe outliers in the data. These indicate unpredictable behavior, which might hinder application of today's commercially available OCS for frequent reconfigurations, which is evaluated in the following.

C. Results: Multiple reconfigurations

The obtained models for the control plane delay provide a lower bound on the reconfiguration period that is possible with the evaluated OCS. However, up to here, the reconfigurations have been performed in a "one-shot" style. Thus, the question arises whether the OCS behaves consistently with periodic reconfigurations, i.e., under constant load.

To assess this, the OCS is stressed with 20 consecutive reconfigurations via the TL1 interface with inter reconfiguration periods between 0.2 s and 3.0 s. Fig. 16 illustrates control plane delays and the fraction of successful reconfigurations. First, we note that for low reconfiguration periods about 30% of the reconfiguration requests are not successfully executed (blue line). This number steadily decreases along with the variance over the runs with higher periods. For large reconfiguration periods, the control plane delay is in the expected range around 100 ms. With decreasing reconfiguration period, the delay and also the number of outliers, i.e., values that exceed 1.5 times the inter-quartile range, increases. Thus, stable operation with low reconfiguration periods is hardly possible with the current firmware of the OCS.

Takeaway: The control plane component of for reconfiguration times is less deterministic than the data plane one. We observe significant influence of the command parameterization on the control plane delay. Furthermore, the measurements have larger variances and also show a significant number of outliers. Putting modest stress to the control plane results in loss of reconfigurations. The behavior is explainable with the reliance on software for processing.

VI. Conclusion

Optical circuit switching in DCNs has received more and more attention in the past years. While topology adaptation enhances the network's flexibility, reconfigurations lead to interruptions that can diminish the advantages of such adaptive topologies. This paper studies the end-to-end reconfiguration delays of programmable optical links using a commodity OCS. It presents a measurement methodology and investigates the behavior of five programmable networking devices.

The measurements reveal indeed varying performance across the devices. Furthermore, we observed that the specific reconfiguration request impacts the delay on the control plane. Interestingly, many theory-oriented works neglect this variability, i.e., rely on the assumption that end-to-end reconfigurations of optics is deterministic. The measurement results report that reconfiguration times are predictable, but not constant – a fact that should be considered in future work.

Acknowledgment

This work has received funding by the Bavarian Ministry of Economic Affairs, Regional Development and Energy as part of the project 5G Testbed Bayern mit Schwerpunktanwendung eHealth. This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 438892507.

References

[1] 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 CCR*, vol. 38, no. 2, pp. 69–74, 2008.

[2] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM CCR*, vol. 44, no. 3, pp. 87–95, 2014.

[3] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers," in *Proc. ACM SIGCOMM 2010*, pp. 339–350.

[4] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen, "Osa: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Trans. on Networking*, vol. 22, no. 2, pp. 498–511, 2013.

[5] M. Ghobadi, R. Mahajan, A. Phanishayee, N. Devanur, J. Kulkarni, G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper, "Projector: Agile reconfigurable data center interconnect," in *Proc. ACM SIGCOMM '16*, pp. 216–229.

[6] W. M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A. C. Snoeren, and G. Porter, "Rotornet: A scalable, low-complexity, optical datacenter network," in *Proc. ACM SIGCOMM '17*, pp. 267–280.

[7] H. Liu, F. Lu, A. Forencich, R. Kapoor, M. Tewari, G. M. Voelker, G. Papen, A. C. Snoeren, and G. Porter, "Circuit switching under the radar with reactor," in *Proc. 11th USENIX NSDI '14*, pp. 1–15.

[8] Y. Xia, X. S. Sun, S. Dzinamarira, D. Wu, X. S. Huang, and T. S. E. Ng, "A Tale of Two Topologies: Exploring Convertible Data Center Network Architectures with Flat-tree," in *Proc. ACM SIGCOMM '17*, pp. 295–308.

[9] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating microsecond circuit switching into the data center," *ACM SIGCOMM CCR*, vol. 43, no. 4, pp. 447–458, 2013.

[10] H. Ballani, P. Costa, R. Behrendt, D. Cletheroe, I. Haller, K. Jozwik, F. Karinou, S. Lange, K. Shi, B. Thomsen, and H. Williams, "Sirius: A Flat Datacenter Network with Nanosecond Optical Switching," in *Proc. ACM SIGCOMM '20*, pp. 782–797.

[11] L. Chen, K. Chen, Z. Zhu, M. Yu, G. Porter, C. Qiao, and S. Zhong, "Enabling Wide-spread Communications on Optical Fabric with MegaSwitch," in *14th USENIX NSDI '17)*, pp. 577–593.

[12] K. Chen, X. Wen, X. Ma, Y. Chen, Y. Xia, C. Hu, and Q. Dong, "WaveCube: A scalable, fault-tolerant, high-performance optical data center architecture," in *IEEE INFOCOM*, 2015, pp. 1903–1911.

[13] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. S. E. Ng, M. Kozuch, and M. Ryan, "C-Through: Part-time Optics in Data Centers," in *ACM SIGCOMM '10*, pp. 1–12.

[14] M. Wang, Y. Cui, S. Xiao, X. Wang, D. Yang, K. Chen, and J. Zhu, "Neural Network Meets DCN: Traffic-driven Topology Adaptation with Deep Learning," in *Proc. ACM SIGMETRICS '18*, pp. 1–25.

[15] A. van Bemten, N. Deric, A. Varasteh, A. Blenk, S. Schmid, and W. Kellerer, "Empirical Predictability Study of SDN Switches," in *Proc. ACM/IEEE ANCS '19*, pp. 1–13.

[16] H. Liu, M. Kaminsky, G. Porter, A. C. Snoeren, M. K. Mukerjee, C. Li, N. Feltman, G. Papen, S. Savage, S. Seshan, G. M. Voelker, and D. G. Andersen, "Scheduling techniques for hybrid circuit/packet networks," in *Proc. ACM CoNEXT '15*, pp. 1–13.

[17] S. Vargaftik, K. Barabash, Y. Ben-Itzhak, O. Biran, I. Keslassy, D. Lorenz, and A. Orda, "Composite-Path Switching," in *Proc. ACM CoNEXT '16*, pp. 329–343.

[18] X. S. Huang, X. S. Sun, and T. E. Ng, "Sunflow: Efficient Optical Circuit Scheduling for Coflows," in *Proc. ACM CoNEXT '16*, pp. 297–311.

[19] H. Harkous, M. Jarschel, M. He, R. Priest, and W. Kellerer, "Towards Understanding the Performance of P4 Programmable Hardware," in *Proc. ACM/IEEE ANCS '19*, pp. 1–6.

[20] D. Scholz, H. Stubbe, S. Gallenmuller, and G. Carle, "Key Properties of Programmable Data Plane Targets," in *Proc. ITC 32*, 2020, pp. 1–9.

[21] G. P. Katsikas, "What You Need to Know About (Smart) Network Interface Cards," in *Proc. 22nd PAM*, 2021, pp. 319–337.

[22] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore, "OFLOPS: An Open Framework for OpenFlow Switch Evaluation," in *Proc. PAM*, 2012, pp. 85–95.

[23] A. Blenk, A. Basta, L. Henkel, J. Zerwas, W. Kellerer, and S. Schmid, "Perfbench: A Tool for Predictability Analysis in Multi-Tenant Software-Defined Networks," in *Proc ACM SIGCOMM '18 Posters and Demos*, pp. 66–68.

[24] M. Khani, M. Ghobadi, M. Alizadeh, Z. Zhu, M. Glick, K. Bergman, A. Vahdat, B. Klenk, and E. Ebrahimi, "A Network Architecture for Fast Training of Distributed Machine Learning with Silicon Photonics," pp. 1–21, 2019.

[25] Telcordia. GR-831. [Online]. Available: https://telecom-info.njdepot.ericsson.net/site-cgi/ido/docs.cgi?ID=SEARCH&DOCUMENT=GR-831

[26] M. Björklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," RFC 6020, Oct. 2010. [Online]. Available: https://rfc-editor.org/rfc/rfc6020.txt

[27] Polatis Series 6000. [Online]. Available: https://www.polatis.com/

[28] FS.com 10GBASE-LR SFP+ 1310nm (SMF). [Online]. Available: https://img-en.fs.com/file/datasheet/10g-base-lr-1310nm.pdf

[29] Intel X710-DA4. [Online]. Available: https://www.intel.de/content/www/de/de/products/docs/network-io/ethernet/network-adapters/ethernet-x710-brief.html

[30] Data Plane Development Kit. [Online]. Available: https://www.dpdk.org/

[31] Netronome - Agilio CX SmartNICs. [Online]. Available: https://www.netronome.com/products/agilio-cx/

[32] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, "Netfpga sume: Toward 100 gbps as research commodity," *IEEE micro*, vol. 34, no. 5, pp. 32–41, 2014.

[33] Pica P3297 Datasheet. [Online]. Available: https://www.pica8.com/wp-content/uploads/pica8-datasheet-48x1gbe-p3297.pdf

[34] Dell S4048-ON Datasheet. [Online]. Available: https://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Dell-EMC-Networking-S4048-ON-Spec-Sheet.pdf

[35] R. Kundel, F. Siegmund, J. Blendin, A. Rizk, and B. Koldehofe, "P4STA: High Performance Packet Timestamping with Programmable Packet Processors," in *Proc. IEEE/IFIP NOMS '20*, pp. 1–9.

[36] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "MoonGen: A Scriptable High-Speed Packet Generator," in *Proc. ACM IMC '15*.

[37] A. Bierman, M. Björklund, and K. Watsen, "RESTCONF Protocol," RFC 8040, Jan. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8040.txt

[38] M. Grottke, R. Matias, and K. S. Trivedi, "The fundamentals of software aging," in *Proc. IEEE International Conference on Software Reliability Engineering Workshops (ISSRE Wksp)*, 2008, pp. 1–6.