HARMLESS: Cost-Effective Transitioning to SDN for Small Enterprises

Levente Csikor*, László Toka†, Márk Szalay†, Gergely Pongrácz‡, Dimitrios P. Pezaros*
and Gábor Rétvári†
*School of Computing Science, University of Glasgow, Email: {firstname.lastname}@glasgow.ac.uk
†High Speed Networks Lab, Budapest University of Technology and Economics, Email: {lastname}@tmit.bme.hu
‡Ericsson Research, TrafficLab Hungary, Email: gergely.pongracz@ericsson.com

***Abstract*—Software-Defined Networking (SDN) offers a new way to operate, manage, and deploy communication networks and to overcome many long-standing problems of legacy networking. However, widespread SDN adoption has not occurred yet due to the lack of a viable incremental deployment path and the relatively immature present state of SDN-capable devices on the market. While continuously evolving software switches may alleviate the operational issues of commercial hardware-based SDN offerings, namely lagging standards-compliance, performance regressions, and poor scaling, they fail to match the cost-efficiency and port density.**

In this paper we propose HARMLESS, a new SDN switch design that seamlessly adds SDN capability to legacy network gear, by emulating the OpenFlow switch OS in a separate software switch component. This way, HARMLESS enables a quick and easy leap into SDN, combining the rapid innovation and upgrade cycles of software switches with the port density and cost-efficiency of hardware-based appliances into a fully dataplane-transparent and vendor-neutral solution. HARMLESS incurs an order of magnitude smaller initial expenditure for an SDN deployment than existing turnkey vendor SDN solutions while it yields matching, or even better data plane performance for smaller enterprises.

***Index Terms*—SDN, Migration, OpenFlow, Switch design**

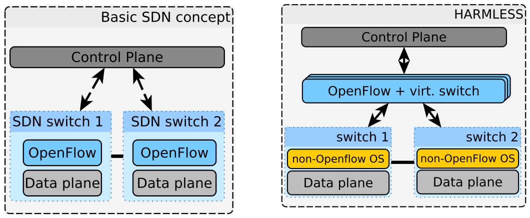
I. Introduction

SDN offers a radical break with traditional ways of building, operating and managing networks. By the physical and logical separation of the network control plane from the packet processing functionality, SDN exposes new levels of *abstraction* to the operator. SDN hides the specifics of the underlying data plane technologies from the network control applications behind a standardized southbound interface, and unprecedented network and service *programmability*, since the network is now controlled by an adaptable software functionality via an open API. Migration to SDN architecture improves network operations by eliminating the need for box-by-box management and troubleshooting, eases to create network functions and services due to the flexibility of the global network view in the centralized SDN control plane, and allows the operator to easily buy into new models for network management and operations, like automated orchestration, on-the-fly chaining of services, and "as-a-service" schemes [1]–[4].

Despite the fact that many large corporations (e.g., Google [5]) and telcos (e.g., Deutsche Telekom [6]) have already gained significant foothold in SDN, *smaller businesses, campus network operators, and service providers* without substantial in-house expertise and select IT staff ("organizations that aren't called Google" [7]) face significant business, economic, and technical deployment barriers, since migration to SDN requires a nontrivial amount of forward planning, an extensive investigation of vendor offerings/options, and a fairly radical change in the mental model, producing a typical chicken and egg problem [1]–[4], [7].

First, there is a broad selection of SDN migration strategies: incremental deployment strategies may offer the smoothest upgrade path and the least interference with daily network operations [8], yet managing heterogeneous network architectures may prove challenging [2], [4]. Jumping outright to full-blown SDN by swapping all legacy network gear to SDN-capable devices overnight may mitigate this pain factor, but greenfield migration is hardly an option for small businesses due to the huge capital expenditure, the flag-day deployment, and the induced service downtime. *A lightweight SDN migration combining the smoothness and reversibility of incremental upgrades with the swiftness and transparency typical to greenfield deployments is still largely missing* [9].

Second, there is the paradox of choice inherent to the booming and immature state of the SDN market today, with a breadth of vendor SDN offerings, turnkey solutions, technology trends, and marketing hype, that a less informed operator may find very challenging to explore and evaluate [1], [6]. For a starter, to obtain an SDN-enabled appliance a network operator has essentially two nontrivial choices: buying commercial off-the-shelf (COTS) and white-box (i.e., generically branded switches with no default network operating system) hardware switches or relying on, possibly already purchased and deployed, general-purpose servers and running a virtual software switch on top (e.g., Open vSwitch, OVS [10]). Thanks to the use of special-purpose ASICs and network processors, hardware SDN network gear has traditionally been praised for providing high port density at a reasonable price tag, albeit notorious for lacking standards-compliance, limited TCAM sizes (typical 1^{st} and 2^{nd} generation devices can have at most 100s and couple of 1000s of flow rules in TCAM), performance regressions, and unscalability [9], [11], [12]. In particular, users are widely complaining about missing OF features (even as elementary as IP address rewrite [9]), switch control plane performance and delays in updating the data



(a) SDN with OpenFlow (b) HARMLESS

Fig. 1: HARMLESS: SDN with an extra level of separation.

plane, atomic flow modification commands not being applied atomically or at all [11], [13], etc. Note that some newer generation OF switches (e.g., from `NoviFlow`, `Corsa`, `Barefoot`) offering high performance with up to 1 million flow rules have recently become available in the market, however not just their costs hinders smaller enterprises to obtain one, but due to the way ASICs are designed an arbitrary forwarding pipeline cannot be applied without fulfilling certain requirements [14] (e.g., wire-speed VLAN handling can only be done in table 0).

Software SDN switches, on the other hand, struggle to match the port density of hardware switches due to the physical space limits of blades and the steep price of multiport NICs, but at the same time excel at programmability, extensibility, rapidly evolve with new standards and receive bug fixes very fast [10], [15]. While, thanks to recent advances in softswitch design and implementation (e.g., multi-threaded switch design, hierarchical flow caching, custom-compiled OF datapaths [10], [15]–[19]) the performance tax of software switching has greatly decreased, programmability and port density are still competing, if not mutually exclusive, goals in current SDN networking equipment.

In this paper, we propose HARMLESS, the *Hybrid ARrchitecture to Migrate Legacy Ethernet Switches to SDN*, to foster SDN migration for smaller enterprises. HARMLESS leverages the current trend for "software-defined-everything", but takes this idea to the extreme: it applies an additional level of abstraction on top of the conventional control plane–data plane separation by *further decoupling the packet processing hardware from the switch's operating system*, which are today integrated in COTS devices in a single box, *and implementing the OpenFlow (OF) OS in a dedicated software switch* (see Fig. 1). This makes it possible to add SDN capability to plain Ethernet switches, or to any legacy network device for that matter, through bypassing the legacy switch OS. Thanks to the additional level of virtualization, *HARMLESS realizes a delicate sweet spot between hardware and software SDN switching*. In particular, *it combines the advantages of software and hardware switching, whereas the hardware component delivers the high port density and raw packet processing functionality, and the softswitch adds programmability, adaptability, and standards-compliance*. Using extensive measurements with a HARMLESS prototype, we show that the benefits of HARMLESS are realized with no significant impact on raw packet processing performance, latency, and dataplane transparency (note that the performance is upper bounded by the used software switch).

From an economical aspect, *HARMLESS offers a viable migration strategy to smaller enterprises.* Since HARMLESS leverages the *existing* network infrastructure it offers distinct price advantages over SDN alternatives available on the market (see details in Sec. III). Crucially, in cases where legacy switches and bare-metal servers for running the OF component are readily available, like in smaller private clouds, enterprise and research networks, *HARMLESS makes it possible to get into SDN instantly, incurring zero expenditure for a partial or even a complete deployment.* And even if equipment must be purchased anew, HARMLESS can save up to an order of magnitude investment. In a broader perspective, HARMLESS sheds a fresh new light on the ages-old, and often highly contentious, "hard switch vs softswitch" debate and presents an interesting new dimension in switch architectures [20]–[25].

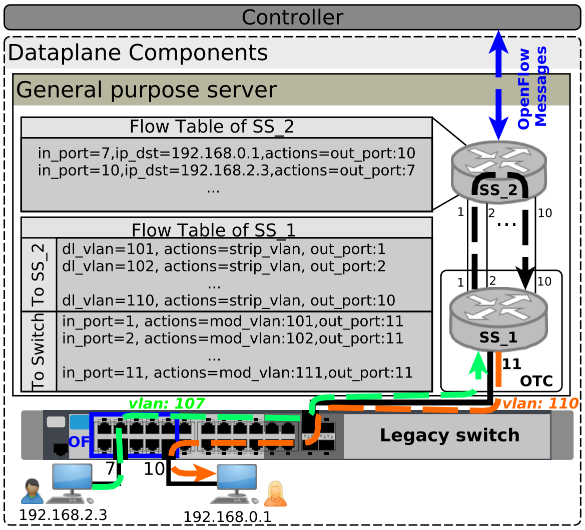
Roadmap: in Sec. II we present the HARMLESS architecture, in Sec. III we give a cost analysis, in Sec. IV we evaluate HARMLESS in many aspects, in Sec. V we summarize related work, and finally Sec. VI concludes the paper.

II. The HARMLESS Architecture

So how to magically turn a legacy device, say, a dumb Ethernet switch, into an OpenFlow-speaking one? After all, this would require to open up what is traditionally a closed black box and substitute the legacy switch OS with an SDN-capable one, something that has proved notoriously difficult to do so far. Instead, we adopt a more viable and backward compatible approach for the purposes of HARMLESS by extending the "Tagging and Hairpinning" technique (also called "anything-on-a-stick" [26], [27] or distributed switch design [28]), originally advocated for hypervisor switches by Cisco and HP, to the general context of SDN [24].

The idea behind "Tagging and Hairpinning" is to offload VM-to-VM communication from the hypervisor to the first hop switch. When a VM sends a packet it is marked by a unique VLAN id ("tagging") and forwarded to the access switch, which will then do a forwarding/policy lookup to decide whether to loop the packet back to another VM, in which case it is marked with the unique VLAN id of the target VM ("hairpinning"), or send it further along the data center fabric, or drop it right away. The rationale for this technique is that packet processing is done on efficient special purpose hardware at the first hop switch instead of a potentially less powerful hypervisor switch, while the downsides are doubling bandwidth utilization and increased latency. The main contribution of this paper is the observation that, when cast in the general context of SDN switching, *the "Tagging and Hairpinning" technique yields a uniquely cost-efficient organization of packet processing functionality and forwarding intelligence, and presents an appealing incremental SDN deployment path.*

For the purposes of presenting HARMLESS, suppose that an operator is given a manageable Layer-2 (L2) 802.1Q Ethernet switch with free high-speed trunk ports, a general



Flow Table of SS_2

```
in_port=7,ip_dst=192.168.0.1,actions=out_port:10
in_port=10,ip_dst=192.168.2.3,actions=out_port:7
...
```

Flow Table of SS_1

To SS_2

```
dl_vlan=101, actions=strip_vlan, out_port:1
dl_vlan=102, actions=strip_vlan, out_port:2
...
dl_vlan=110, actions=strip_vlan, out_port:10
```

To Switch

```
in_port=1, actions=mod_vlan:101,out_port:11
in_port=2, actions=mod_vlan:102,out_port:11
...
in_port=11, actions=mod_vlan:111,out_port:11
```

Fig. 2: Transparent HARMLESS: Software Switch Twice (S4).

purpose server that has spare NICs and available capacity to run an OpenFlow vswitch, and adequate cabling, backplane capacity, or other means for interconnecting the switches' trunk ports with the softswitch NICs (see bottom of Fig. 2). Suppose further that a host with IPv4 address `192.168.2.3` connected to port 7 on the legacy switch wishes to send packets to another host with address `192.168.0.1` connected to port 10 on the same switch, and suppose that a security policy is in place according to which these two hosts are permitted to exchange traffic only between each other. Accordingly, suppose that the operator wants to control the switch through OpenFlow and so wishes to program this forwarding behavior as given by the *Flow table of SS_2* in Fig. 2. This would handle communication between the two hosts adequately, except that it would be impossible to control the legacy switch through OpenFlow due to the black box nature of legacy COTS appliances.

This is where the "Tagging and Hairpinning" technique comes in: let the legacy switch tag each packet with a unique VLAN id that identifies the access port it was received from, forward the tagged packet to the software switch along the trunk-port–softswitch interconnect (the *uplink*) to enforce the OpenFlow forwarding and security policies, and send the packet back to the legacy switch via another uplink "hairpinned", i.e., tagged with the unique VLAN id of the proper outgoing port as per the specified flow table. (If the packet was already VLAN-tagged, the legacy switch can use VLAN Q-in-Q tunneling [29] or any other tagging scheme.)

In a strawman's approach, the controller would need to program a slightly modified flow table into the software switch, whereas the "match on ingress port X" rules are converted to "match on VLAN id X" rules and the "output to port Y" actions are rewritten to "modify VLAN id to Y and output to default port" actions. However, to avoid having to tailor controller programs to the underlying HARMLESS layer, we introduce a transparent HARMLESS setup, where the novel idea is to carefully extend the framework with an additional *OpenFlow Translator Component (OTC)* to serve as an adaptation layer. OTC is realized by an additional softswitch [30], which is connected to the OF component by as many patch ports (10 in our example) as the number of the access ports managed and dispatches packets to and from patch ports based on the VLAN ids (see Fig. 2). In our particular example, upon receiving a packet on port 7 from the first host, the legacy switch would tag it with a unique VLAN id, say, 107, and send it along the uplink. The OTC then dispatches the packet (with the VLAN tag removed) to patch port 7 towards the the softswitch SS_2 (managed by the controller), which in turn identifies the original input port based on the patch port and makes sure that the originating host is allowed to communicate with the destination host by matching the first flow entry in the flow table (see Flow Table of SS_2 in Fig. 2). Then, SS_2 sends out the packet on its patch port 10 towards the OTC, which pushes a unique VLAN id, say, 110 onto the packet, then it is looped back to the legacy switch, which, after doing a VLAN-to-port translation, forwards it to the destination host (on physical port 10). Note that the only requirements for the legacy switch are manageability (to setup the VLANs on the access ports), support for 802.1Q (to do the VLAN un/tagging), and free trunk ports to be used for an uplink (later, in Sec. IV we will deal with the "loss" of trunks), which allows to use HARMLESS over basically any legacy Ethernet switch on the market today [31], [32].

These modifications render HARMLESS data plane-transparent enabling to write controller programs ignoring the fact that the underlying data plane is realized with HARMLESS, to make controller programs portable between deployments, and to allow to invoke higher-level languages and policies to setup the data plane [33], [34].

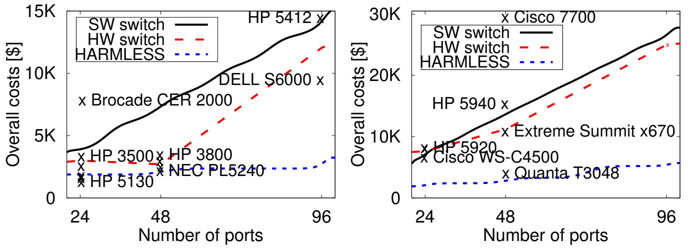
III. Cost Analysis

Below, we argue that HARMLESS strikes a fine balance between the cost-efficiency of COTS switches and the flexibility of softswitches in terms of deployment costs.

A. Hard vs. Softswitch

The great majority of COTS and white-box SDN switch market options come with 24 or 48 ports at 1G (or 10G), supplemented with 2–4 uplink ports operating at 10G (or 40G, respectively) in 1U or 2U form-factor. In case of carrier grade port rates, e.g., 40G, the vendor offerings are even more diverse, often having no designated uplink ports. Prices vary in a wide range; the cheapest OpenFlow-capable 24x1G-port switch (HP-2920) can be purchased at $\$1,200$ while the most expensive 48x10G offering comes at a whopping $\$30,000$. The 48-port form factor seems to provide the most economic per-port prices, with the average price tag of $\$2,700$ for a 48x1G+4x10G OF switch and $\$11,200$ for a 48x10G+4x40G device. Based on these considerations, the following formulas estimate the CAPEX of hardware switch deployment with a total of x ports:

$$C_{HW}^{1G} = \$2700 \left\lceil \frac{x}{48} \right\rceil , \qquad C_{HW}^{10G} = \$11200 \left\lceil \frac{x}{48} \right\rceil .$$



(a) CAPEX per 1G port density (b) CAPEX per 10G port density

Fig. 3: CAPEX for a software, COTS/white-box hardware, and HARMLESS switch as the function of the number of (a) 1G access ports and (b) 10G access ports (prices from 2017).

In case of softswitches, the main CAPEX factor is purchasing servers with a sufficient number of NICs and CPUs. We consider x86-based 1U servers at a bulk price of $\$1,400$ on average, including the motherboard with 1 CPU, 4x1G built-in ports, 3 PCIe (3.0 x8 or x16) slots, memory, disk, power, etc. A server can host up to 3 additional NICs, costing $\$160$ for 4x1G (Intel i350), $\$480$ for 4x10G (Intel X710), and $\$500$ for 2x40G (Intel XL710), which total up to 16 ports per server at $1G$, 12 ports at 10G, and 6 ports at 40G. Note that when a single server cannot provide the required port density another server must be purchased. Furthermore, in most cases the third PCIe slot is hardwired to the second CPU socket, therefore a single CPU server can host up to 12 1G (8 at 10G, 4 at 40G) ports; for more ports per server a second CPU must be installed (hence the last negative term in the below formulas, where the variable $\#extraCPU$ indicates in both the 1G and 10G cases whether the last server is sufficient to serve the rest of the ports w/o an additional CPU). The price of a server CPU ranges between $\$200$ and $\$7,000$ depending on the CPU class, cache size, clock rate, and power consumption; we calculate with the price of a 6-core Intel E5-2620v3 CPU at $\$400$.

With this in mind, the following formulas estimate the CAPEX of a software switch deployment with x ports:

$$\begin{aligned} C_{SW}^{1G} &= \$1400 \left\lceil \frac{x}{16} \right\rceil + \$160 \left(\left\lceil \frac{x}{4} \right\rceil - \left\lceil \frac{x}{16} \right\rceil \right) \\ &\quad + \$400 \left(\left\lceil \frac{x}{16} \right\rceil - \#extraCPU_{SW}^{1G} \right) \\ C_{SW}^{10G} &= \$1400 \left\lceil \frac{x}{12} \right\rceil + \$480 \left\lceil \frac{x}{4} \right\rceil \\ &\quad + \$400 \left(\left\lceil \frac{x}{12} \right\rceil - \#extraCPU_{SW}^{10G} \right) \end{aligned}$$

$$\#extraCPU_{SW}^{1G} = \begin{cases} 1, & \text{if } 0 < (x \bmod 16) < 13 \\ 0, & \text{otherwise} \end{cases}$$

$$\#extraCPU_{SW}^{10G} = \begin{cases} 1, & \text{if } 0 < (x \bmod 12) < 9 \\ 0, & \text{otherwise} \end{cases}$$

For the sake of simplicity, in the price analysis for softswitches and HARMLESS we do not account for OPEX components (e.g., energy consumption, cooling, cabling, rack space occupancy); we return to further CAPEX/OPEX issues later. Fig. 3 shows the CAPEX analysis for the purchase of a single hardware-based and software-based SDN switch. The price figures show that hardware switches are more economical at port density 24 and 48, due to software switches needing many expensive NICs and additional servers to match the same number of ports (even though the price advantages level off at high-end switches with 96 ports).

In summary, current market trends suggest that *hardware SDN switches provide high port density at moderate prices but lag behind software switches in scalability, standards-compliance and programmability.*

B. HARMLESS: High port density at low cost

Next, we evaluate HARMLESS as a cost-efficient middle-ground between the two extremes. HARMLESS unifies the advantages of hardware and software switches, by decoupling the raw forwarding functionality from the OpenFlow glue, resulting in an optimal separation of concerns: high port density by legacy devices, while the softswitch component enables implementing the packet processing intelligence in a clear and extensible way. Observe that the softswitch does not need to match the port density of the legacy switch effectively removing the major cost component, NIC prices.

HARMLESS leverages the existing and deployed computing and networking infrastructure, readily available in many prospective SDN deployments like SOHO networks, small and medium sized enterprises, private data centers, campus networks and private clouds. For these use cases, HARMLESS offers an instantaneous SDN transition path with *zero* initial expenditure, apart from the usual practice of server relocation, cabling, etc. Note, however, that the server running the OpenFlow component and the legacy switches do not even need to be co-located; in fact, the OpenFlow logic can be virtualized at a remote site or even delegated to a public cloud at the price of proper traffic forwarding and slightly increased latency.

Even in cases where spare servers or Ethernet switches are not available, purchasing them anew in a HARMLESS setup is still much less costly than purchasing equivalent SDN network gear from commercial suppliers. For the below CAPEX analysis, we assume that the legacy 48x1G+4x10G (or 48x10G+4x40G at 10G access) Ethernet switches are already on stock (if not, add another couple of hundred USD per switch), so only bare-metal servers for the OpenFlow components and 10G NICs (respectively, 40G) need to be purchased. We aggregate 12 access ports to each trunk port, over-committing the uplink at a similar rate as plain Ethernet networks [35], multiplexing up to 144 access ports (72 at 10G) to a single OpenFlow component. Accordingly, the CAPEX for a HARMLESS (HL for short) deployment with x ports are as follows:

$$\begin{aligned} C_{HL}^{1G} &= \$1400 \left\lceil \frac{x}{144} \right\rceil + \$480 \left\lceil \frac{x}{48} \right\rceil \\ &\quad + \$400 \left(\left\lceil \frac{x}{144} \right\rceil - \#extraCPU_{HL}^{1G} \right) \\ C_{HL}^{10G} &= \$1400 \left\lceil \frac{x}{72} \right\rceil + \$500 \left\lceil \frac{x}{24} \right\rceil \\ &\quad + \$400 \left(\left\lceil \frac{x}{72} \right\rceil - \#extraCPU_{HL}^{10G} \right) \end{aligned}$$

$$\#extraCPU_{HL}^{1G} = \begin{cases} 1, & \text{if } 0 < (x \bmod 144) < 97 \\ 0, & \text{otherwise} \end{cases}$$

$$\#extraCPU_{HL}^{10G} = \begin{cases} 1, & \text{if } 0 < (x \bmod 72) < 49 \\ 0, & \text{otherwise} \end{cases}$$

Similar to software switches, the first term accounts for the server, the second term for the NICs, and the third for the additional CPUs. Using this estimate, the CAPEX analysis in Fig. 3 shows that HARMLESS is by a large margin the most cost-efficient SDN migration option, thanks to the high level of aggregation that reduces the number of costly servers and NICs as compared to pure softswitches, providing one order of magnitude more economical option.

IV. Evaluation

Next we turn to the practical aspects of HARMLESS and compare it against market alternatives. We evaluate the *scalability, standards compliance, data plane performance, latency* in diverse use cases taken from practical networking applications and under different workloads, and we present the results side by side with the *SDN migration cost analysis (CAPEX/OPEX)* for each possible SDN switch option (softswitch, COTS and white-box switches, and HARMLESS). First, we describe our testbed and the measurement methodology, then we present the specific use cases, and finally we present the evaluation results.

A. Testbed and methodology

Our testbed includes two IBM x3550 M5 servers with Intel Xeon E5-2620v3 processors and 16GB of memory running Debian Linux Jessie 8.0/kernel 4.9, each server equipped with an Intel X710 NIC (10G) and Intel XL710 (40G) NIC, respectively. The setup also contains two legacy switches, a `Cisco 3750X` (24x1G + 4x10G) and an `Arista 7048T` (48x1G + 4x10G), three COTS OpenFlow switches, an `HP 3500` (24x1G), an `Extreme X440` (28x1G + 2x10G), and a `Brocade ICX6610` (28x1G + 4x10G), and two white-box switches (`Quanta T1048` and `Edge-Core AS4610`) all supporting OpenFlow v1.3. The COTS switches represent the state-of-the-art in COTS SDN switching as of 2015, while the white-box switches are from the low-end market of 2016. The switches have the following flow table size limitations:

- `HP 3500`: 1 flow table in TCAM with max. 1500 rules, and 4 further logical tables (processed in software);
- `Extreme X440`: 1 flow table in TCAM with max. 255 flow rules, assuming each one has limited length in terms of match fields, and another table for MAC and VLAN matching rules (also in TCAM);
- `Brocade ICX6610`: 1 flow table with max. 3000 rules in TCAM (half if rules match on both L2 and L3 headers), and no further tables.
- `Quanta T1048`: 1 flow table in TCAM with roughly 2000 flow rules and 6 logical tables for tens of thousands of flow rules and different layer matching (e.g., matching on both source and destination MAC address can only be implemented in a logical table).
- `Edge-Core AS4610`: supports multiple flow tables, one of them containing actions, carrying maximum only 3840 flow rules in TCAM (plus $24,576$ and $32,768$ flow rules for exact matching on destination MAC address and destination IP address, respectively)

In each experiment, one of the servers was configured to run NFPA [36], an Intel DPDK *pktgen*-based benchmarking tool, back-to-back with the system-under-test (SUT). For the software switch evaluations the SUT was provisioned on the other IBM server, running a stable version of OVS (v2.7.0) and ESwitch [15], both compiled with a stable Intel DPDK v16.11.1. The hardware switch option was evaluated on each of our COTS switches, while for HARMLESS the OpenFlow component was again configured on the IBM server running OVS (HARMLESS-OVS) or ESwitch (HARMLESS-ESwitch), connected to one of the legacy switches.

The measurements were conducted over synthetic traffic traces, specially tailored to each use case (see below) to contain a configurable number of flows. Note that packets were never dropped intentionally, instead the OpenFlow pipelines contain default catch-all rules to forward unmatched/dropped packets to the external port; our aim was to measure raw throughput and not whether the switches can filter traffic adequately (they can). With this configuration, packet loss only occurs when the SUT becomes a physical bottleneck and therefore the packet rate received at the packet generator is representative of the raw performance. Packets were minimum-sized (i.e., 64 bytes) and Receive Side Scaling (RSS) was turned on in multi-core setups [37]. All measurements were conducted at 40G for at least 60 seconds [38]. At first the *packet rates were measured in a single-core setup*; note that the attainable throughput using a single core and PCIe x8 v3 bus speed is 15 Gbps (22 Mpps) with 64-byte packets; multi-core scalability is studied in a separate measurement round.

1) Use cases: We considered 4 realistic use cases [36], from private data centers to telco gateways. All scenarios will be cast in a single hypothetical service provider's legacy network (see Fig. 4). The setup contains a smaller data center (DC) with 4 racks connected into a CLOS topology with separate L2 domains at the leaves and an L3 domain as the spine [39], an industrial-scale load-balancer [40], and a telco access gateway [41] that aggregates subscribers located behind Customer Endpoints (CEs) [42].

The lower layer of the DC topology represents the *L2* use case, with each top-of-rack (ToR) switch provisioned as a separate *L2* domain; a sample L2 traffic flow is marked with orange in Fig. 4. While certain data centers may differ in the configuration of L2/L3 domains [43] this use case describes large L2 networks illustratively, to be migrated from traditional 802.1 to SDN with the aim of eliminating dependency on spanning trees and benefiting from centralized control [44].

The *L3* use case embodies the upper layer of the CLOS network interconnecting the L2 islands, a common setup in DCs [39]; a sample traffic is marked with cyan in Fig. 4.

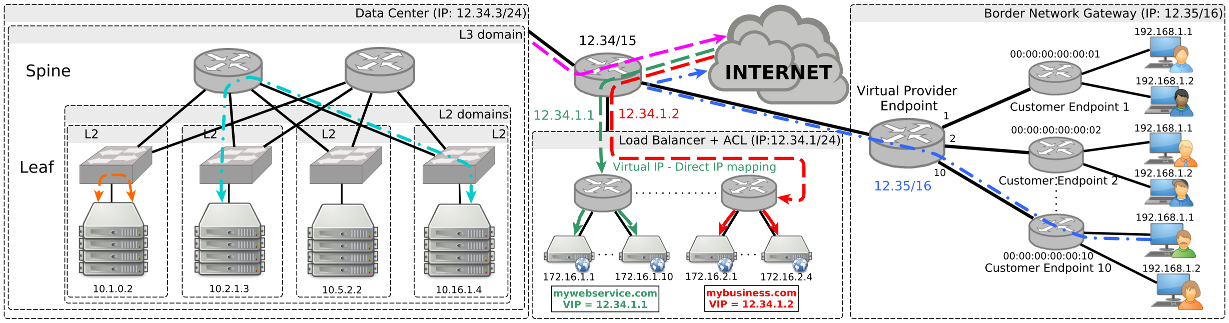


Fig. 4: Use cases in a service provider network.

The *load balancer and access control list* use case in the middle of Fig. 4 captures the functionality of a web frontend, balancing incoming web traffic (TCP port 80) for 100 different web services, each available at a unique IP address [36]. During the measurements, traffic traces were crafted so that 70% of packets go to a randomly chosen web service while the rest is filtered at the ACL.

The telco *access gateway* use case [36] on the right hand side of Fig. 4 is the most complex one consisting of a Virtual Provider Endpoint (VPE) that serves Internet access to subscribers located behind Customer Endpoints (CEs). For brevity, we identify CEs with the MAC address and we assume that the operator sets 10 CEs, each serving 20 users provisioned with unique private IP addresses.

B. Measurement results

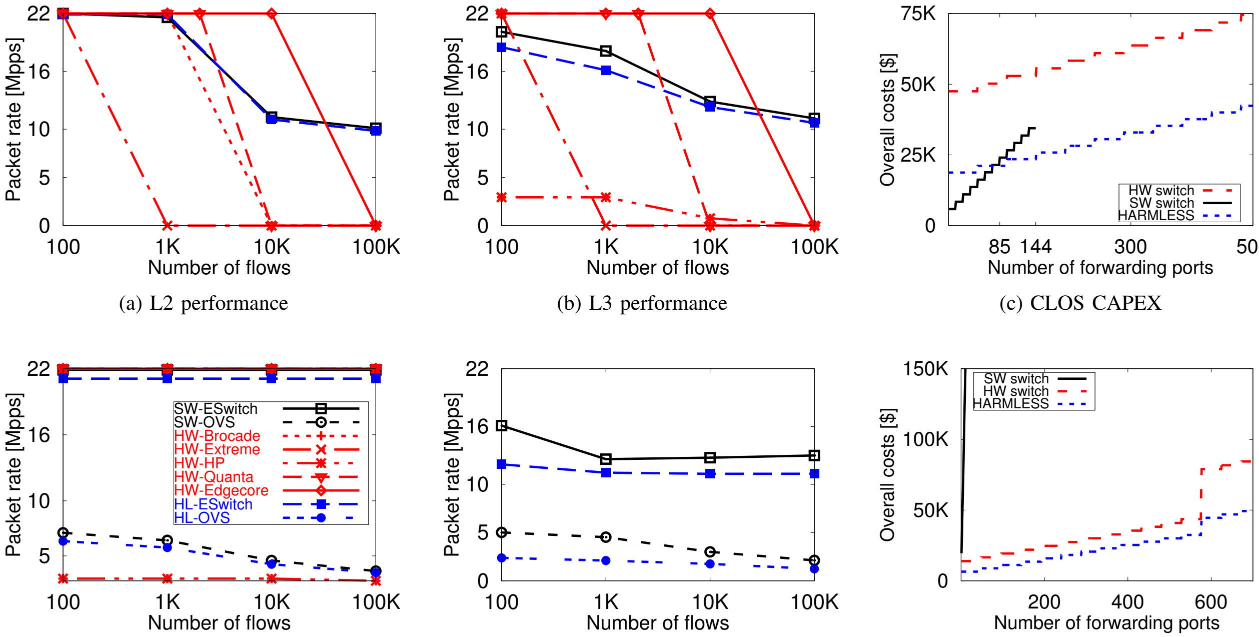
1) Scalability and standards-compliance: Configuring the use cases on COTS and white-box switches proved far from trivial, due to the prohibitive flow table sizes and subtle restrictions on flow matching rules. The hardware switches support only a single flow table in TCAM and may or may not provide additional tables in software. Thus, multi-table OpenFlow pipelines had to be tediously collapsed into a single table by hand; in case of the white-box devices their software, e.g., Pica8 PicOS on `Quanta` and ONL+Indigo on `EdgeCore`, do this automatically. Unfortunately, even then the switches rapidly run out of TCAM space because of the flow-state explosion effects for which table collapsing is notorious [10]. In the *access gateway* use case for instance a separate flow entry must be created for every (user, CE, IP prefix) tuple, yielding so many entries that none of the hardware offerings could implement this use case. Furthermore, one has to take into account the ramifications of the chip in each individual switch, e.g., the `HP` switch does not support static matching on MAC addresses, the `Brocade` switch does not support MAC rewrite, the `EdgeCore` switch cannot modify IP fields, only on slow-path. *Current hardware switches do not scale beyond small and medium workloads, and even in that case may require hand-tweaking the OpenFlow pipeline, while softswitches and HARMLESS support very large deployments.*

2) Performance: Fig. 5a, 5b, 5d, and 5e compare the raw packet rate with the hardware switches, the software switches, and HARMLESS measured in the *L2*, *L3*, *load balancer* and *access gateway* use case, respectively. Recall that due to the attainable packet rate of a single CPU core the y axes are scaled up to 22 million packets per seconds (Mpps). Note that for each use case results are reported only for the hardware switches that could handle the use case.

Our observations are as follows. First, as long as hardware OpenFlow switches manage to forward packets purely in the fast path they perform at wirespeed. However, as soon as a hardware switch runs out of TCAM space and forwarding falls back to the software slow path performance plummets. Note, however, that among the devices providing logical tables only `HP` can use TCAM and logical tables at the same time for the same scenario; `Quanta` installs as many flow rules in its TCAM as it can ($2K$), and silently ignores the rest without notifying the controller. For instance, in the *L2* use case the `Extreme` switch could handle 100 flows at line rate ($1K$ and $2K$ flows with the `Brocade` and the `Quanta`, respectively) but could not tackle $1K$ flows at all ($10K$ for the `Brocade` and the `Quanta`). On the other hand, all hardware switches can support the relatively small flow table of the *load balancer* use case adequately (even though the `HP` proved to be slower).

Meanwhile the ESwitch-based OpenFlow softswitch performs close to line rate at small and medium sized workloads and only becomes worse at very large flow tables. Depending on the workload, *the HARMLESS-ESwitch combination attains a performance very close to that of the TCAM-based fast path of hardware switches and the best softswitches*, in the majority of the cases reaches up to 90–95% *and it robustly outperforms the hardware's slow-paths and the* `HP` *switch*. Results with OVS (only presented for the *load balancer* and the *access gateway* use cases for brevity) are much worse, but then again the HARMLESS-OVS mix is very close to pure OVS. This suggests that *the performance of HARMLESS is eminently conditioned on the OpenFlow softswitch component*; here, the HARMLESS-ESwitch combination seems very appealing.

We measured the throughput on multiple CPU cores (Fig. 6) under the larger workloads (namely, in the L3/100K and LB/10K use cases [36]); this time, we use 128-byte packets as the Intel XL710 NIC cannot be saturated with smaller packets [45]. The results indicate that HARMLESS scales to multiple cores linearly, however the HARMLESS-ESwitch mix already achieves its maximum performance (much higher than OVS can attain with 6 cores) with only 4 cores.



(a) L2 performance

(b) L3 performance

(c) CLOS CAPEX

(d) LB performance

(e) GW performance

(f) GW CAPEX

Fig. 5: Raw packet throughput as the function of the workload size in (a) the *L2*, (b) the *L3*, (d) the *load balancer* (LB) and (e) the *access gateway* (GW) use cases, and the CAPEX at different deployment scales for (c) a full CLOS topology that integrates the *L2* and *L3* use cases, and (f) the *access gateway*. Note the common legend for panel (a), (b), (d), and (e) in plot (d): SW: software switches, HW: hardware switches, HL: HARMLESS.

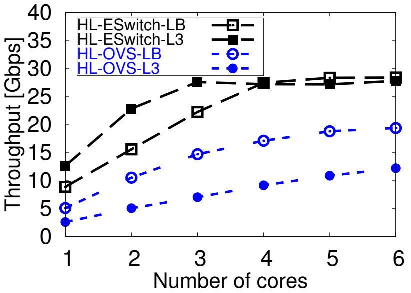


Fig. 6: HARMLESS Multi-core scalability: throughput (Gbps) with ESwitch and OVS (128-byte).

3) CAPEX: Fig. 5c compares the CAPEX for a greenfield deployment in the CLOS-based telco DC (the *L2* and *L3* use cases combined) as the function of access port density supported at the ToR switches. Note that due to the different form factors the spine layer scales differently: purchasing four 48x10G hardware switches for the spine incurs a huge initial investment but can then scale to 48 leaf switches economically; in case of software switching, one leaf switch, offering similar aggregation ratios as a typical hardware device provides (e.g., 48x1G vs. 4x10G), mounts 12x1G + 1x10G, thus we only need *one server* with 12x10G capacity as the spine resulting in a small CLOS topology (providing 144 access ports at the most). Observe in Fig. 5c that in contrast to COTS devices and HARMLESS, where we are given 4 spine switches (the most expensive parts of the CLOS topology), the necessity of only one spine server is the only reason why software switches involve less initial investment.

Since in case of HARMLESS the trunk ports of the legacy devices are used to provide the OpenFlow capability, special attention is needed in order to preserve the non-blocking 1:1 over-subscription ratio of the CLOS topology. Therefore, in HARMLESS a spine switch is comprised of a 48x10G+4x40G legacy switch plus a server with two 4x10G NICs for the softswitch component, and 2x100G NIC with an additional CPU for compensating the inherent “loss” of uplink ports resulting in a sum of $4,100 per spine switch (for the NIC we considered the average price of a Mellanox ConnectX-5 NIC of $1,300). Nonetheless, a legacy leaf switch of 48x1G + 4x10G only requires a HARMLESS server with two 4x10G NICs resulting in an average price of $2,350 per leaf switch.

Fig. 5f gives the CAPEX for a telco *access gateway* greenfield SDN deployment in a simple tree topology (Fig. 4) with a depth of 3 consisting of 48x1G forwarding ports at the leafs, 48x10G aggregation switches in the middle, and one switching gear with 40G forwarding ports as the core (we considered the average price of $21,500 USD) offering an overall 1:1 over-subscription ratio. One can observe that when relying merely on software switches, expenses can easily reach high even for fewer number of ports. On the other hand, in case of the hardware devices and HARMLESS the steep cost steps arrive at 576 forwarding ports: those indicate the price of the 32x40G OpenFlow-enabled core switch, and the three 2x100G NICs for HARMLESS, respectively. Crucially, *in all*

TABLE I: Latency over in a bridge and in the L2/1K use cases, energy consumption, and rack space.

SDN switch	Latency [μs]		Power [W]		Rack
	Bridge	L2	Min	Max	
SW-ESwitch	238	233	70	230	3U
HW-HP HW-HP-SW	138 697	NA 730	142	616	1U
SL-ESwitch	268	265	164	350	1.3U

cases HARMLESS is the most cost-efficient option, supporting roughly the same performance at the fraction of the price: on average HARMLESS is 2–4 times less expensive than a softswitch-, a COTS-, or a white-box-based deployment, but the price difference can even reach to an order of magnitude. Here, we assume that the legacy Ethernet switches for HARMLESS are on stock; if not, HARMLESS is still 1.5–3 times more cost-efficient due to the economical price tag of commodity Ethernet switches; however, if Ethernet switches and spare servers are available in adequate number then, recall, SDN migration with HARMLESS incurs zero cost.

4) Latency: In order to check whether the additional softswitch in the loop increases the latency of HARMLESS prohibitively (extra latency occurs for inter-port communication only, but not for out-of-switch traffic), we conducted a series of latency measurements in various setups; Table I gives the results for a single port-forwarding rule in the OpenFlow pipeline and for the L2/1K workload. The HP switch, when using the TCAM-based fast path, yields roughly 130 μsec delay, but it is much less efficient when it falls back to the software datapath (around 700 μsec). ESwitch's delay is around 230 μsec reliably, with HARMLESS only 10% more thanks to the fast underlying plain Ethernet switch (adding roughly 30–50 μsec to the softswitch latency), but still much faster than the software fallback of the COTS switch. The results indicate that *the additional softswitch does not introduce prohibitive latency in HARMLESS*, just the contrary, its latency is on par, and in some cases even better than, alternatives; accordingly, HARMLESS latency seems sufficient for anything but the most delay-critical applications.

5) OPEX: Below, we extend our analysis with operational costs, which can constitute a significant factor in the total spending. Table I shows an evaluation of two important OPEX components. The energy consumption is estimated from the datasheets of the switches and the CPUs (note that the legacy Ethernet switch used in HARMLESS consumes less power than a full-scale SDN switch). The rack space occupancy is normalized for the standard 48x1G form factor: 1U for a hardware switch, 3U for the three 16x1G servers needed for a 48-port software switch, and for HARMLESS $1U$ for the legacy 48x1G switch and $1U$ for the 12x10G server, but the latter can handle 2 additional legacy switches as well which gives 1.3U normalized to 48 ports overall. Cabling might be more difficult though, since some high-speed uplinks that could otherwise be used for aggregation are allocated for HARMLESS; yet, the flexibility of access port assignment in HARMLESS may be exploited to optimize cabling costs. Overall, *the costs for operating HARMLESS are at the same level as that of the alternatives*.

V. Related Work

SDN migration. The new levels of abstraction, programmability, and logically centralized control are important motivators for deploying SDN in operational networks [1], like enterprise networks [46], DC fabrics [47], transport networks [48], [49], WANs [3], [5], and Internet exchanges [50]. However, most deployments involve the complete and irreversible overhaul of the existing legacy networking infrastructure. Incremental deployment strategies [8] seek to find a smoother migration path than a flag-day greenfield upgrade [1], [2]. Managing a heterogeneous network, however, can become rather unwieldy due to the potential interference between coexisting legacy and SDN control planes. For example, forwarding loops may be formed due to the legacy control plane masking certain forwarding decisions from the SDN controller [4]. HARMLESS fits into any of these SDN migration paths smoothly, thanks to its dataplane transparency, fine-grained upgrades, and vendor neutrality.

Hybrid SDN. Similar to HARMLESS, the hybrid SDN scheme Panopticon [2] connects legacy device ports to SDN-capable switches using VLAN tagging. The objective in Panopticon is different though: guaranteeing that each forwarding path traverses at least one SDN switch that can exert control over the traffic along that path. On the contrary, HARMLESS is dataplane-transparent and can accommodate any SDN policy without special tweaking. Furthermore, Panopticon needs a nontrivial number of newly purchased SDN switches, while HARMLESS can introduce the *existing* legacy network infrastructure to under SDN control and hence is more economical.

Fibbing [3], [49] is also similar to HARMLESS in that it endues a legacy network employing a distributed routing protocol with SDN control. However, it is bound by the limitations of destination-based routing, while HARMLESS opens up the full power of SDN to realize any forwarding and security policy.

VI. Conclusion

Recently, SDN has grown out of the "niche status" and found important use in communication networks. However, there still exist areas it has not penetrated, mainly service provider networks and smaller businesses with less technically savvy IT staff. In this paper, we presented HARMLESS, a new SDN switch design to offer an attractive deployment path for such cases.

The main idea in HARMLESS is opening up traditional black-box network gear and virtualizing the switch OS in a separate softswitch component. HARMLESS allows an operator to start experimenting with SDN instantaneously: by connecting the trunk port of a legacy Ethernet switch to a spare x86 server and firing up HARMLESS, an operator can immediately engage with OpenFlow controller programs with zero initial investment. Later, any combination of legacy ports and switches can be connected to the HARMLESS software switch to incrementally reach a full SDN deployment.

HARMLESS realizes an appealing combination of hardware and software switching, with the hard switch providing the port density and the softswitch delivering programmability. Our comprehensive CAPEX analyses on realistic SDN migration scenarios indicate that HARMLESS attains the most economic SDN migration strategy today, with performance close to (90–95%), and in some cases even higher than that of the alternatives. Crucially, HARMLESS is exempt from the dataplane quirks and performance regressions experienced with COTS OpenFlow appliances. With the continuous evolution of software-based switching and general-purpose packet processing solutions, throughput achieved with HARMLESS will likely further improve in the future.

Acknowledgements

The work has been supported in part by the European Cooperation in Science and Technology (COST) Action CA 15127: RECODIS - Resilient communication and services; and by the UK Engineering and Physical Sciences Research Council (EPSRC) projects EP/L026015/1, EP/N033957/1, and EP/P004024/1.

References

[1] Open Networking Foundation, "SDN Migration Considerations and Use Cases," ONF Solution Brief, Nov 2014.

[2] D. Levin, M. Canini, S. Schmid, F. Schaffert, and A. Feldmann, "Panopticon: Reaping the benefits of incremental SDN deployment in enterprise networks," in *USENIX ATC*, 2014, pp. 333–345.

[3] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central control over distributed routing," in *SIGCOMM*, 2015, pp. 43–56.

[4] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," *SIGCOMM CCR*, vol. 44, no. 2, pp. 70–75, 2014.

[5] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined wan," in *SIGCOMM*, 2013.

[6] Brocade, "Migrating to SDN: planning for a smooth transition," [Online: https://goo.gl/cgPFTs], 2014.

[7] M. McNickle, "With hybrid SDN deployment, no need for network forklift," TechTarget SearchSDN, 2013.

[8] M. K. Mukerjee, D. Han, S. Seshan, and P. Steenkiste, "Understanding tradeoffs in incremental deployment of new network architectures," in *CoNEXT*, 2013, pp. 271–282.

[9] I. Pepelnjak, "Q&A: Vendor Openflow Limitations," http://blog.ipspace.net/2016/12/q-vendor-openflow-limitations.html, Dec 2016.

[10] B. Pfaff *et al.*, "The design and implementation of open vswitch," in *NSDI*, 2015.

[11] M. Kuźniar, P. Perešíni, and D. Kostić, "What you need to know about SDN flow tables," in *PAM*, 2015, pp. 347–359.

[12] I. Pepelnjak, "Table Sizes in OpenFlow Switches," http://blog.ipspace.net/2016/03/table-sizes-in-openflow-switches.html, March 2016.

[13] M. Kuzniar *et al.*, "A SOFT way for OpenFlow switch interoperability testing," in *CoNEXT*, 2012.

[14] A. Pavlidis *et al.*, "Overview of SDN Pilots Description and Findings: Part A," Deliverable D7.1, 2017.

[15] L. Molnár *et al.*, "Dataplane specialization for high-performance openflow software switching," in *Proceedings of ACM SIGCOMM*, 2016, pp. 539–552.

[16] Intel, "Guide: Data plane development kit for linux," Guide, April 2015.

[17] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, F. Huici, and L. Mathy, "Towards high performance virtual routers on commodity hardware," in *CoNEXT*, 2008, pp. 1–12.

[18] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "RouteBricks: Exploiting parallelism to scale software routers," in *SOSP*, 2009, pp. 15–28.

[19] M. Shahbaz, S. Choi, B. Pfaff, C. Kim, N. Feamster, N. McKeown, and J. Rexford, "PISCES: A programmable, protocol-independent software switch," in *SIGCOMM*, 2016, pp. 525–538.

[20] M. Casado, T. Koponen, D. Moon, and S. Shenker, "Rethinking packet forwarding hardware," in *HotNets*, 2008.

[21] D. Moon, J. Naous, J. Liu, K. Zarifis, M. Casado, T. Koponen, S. Shenker, and L. Breslau, "Bridging the software/hardware forwarding divide," uC Berkeley.

[22] K. Argyraki, S. Baset, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, E. Kohler, M. Manesh, S. Nedevschi, and S. Ratnasamy, "Can software routers scale?" in *PRESTO*, 2008, pp. 21–26.

[23] G. Pongrácz, L. Molnár, Z. L. Kis, and Z. Turányi, "Cheap silicon: A myth or reality? picking the right data plane hardware for software defined networking," in *HotSDN*, 2013, pp. 103–108.

[24] J. Gross, A. Lambeth, B. Pfaff, and M. Casado, "The rise of soft switching, Part I, II, III," Network Heresy, 2011.

[25] G. Ferro, "Soft switching fails at scale," EtherealMind, 2011.

[26] N. Gaur, "Fundamentals of vlans: Router on a stick," CCENT/CCNA R&S Study Group, 2014.

[27] Cisco, "Network address translation on a stick," Technical study, Document ID: 6505, 2008.

[28] F. F. Andrew Lunn, Vivien Didelot, "Distributed switch architecture," Netdev Conf 2.1, 2017.

[29] IEEE, "Std 802.1ad - 2005 IEEE Standard for Local and metropolitan area networks – virtual Bridged Local Area Networks, Amendment 4: Provider Bridges"," 2005.

[30] M. Szalay, L. Toka, G. Rétvári, G. Pongrácz, L. Csikor, and D. P. Pezaros, "Harmless: Cost-effective transitioning to sdn," in *Proceedings of the SIGCOMM Posters and Demos*, 2017, pp. 91–93.

[31] Cisco, "Campus network for high availability design guide," Design Guide, 2008.

[32] Juniper, "Campus networks reference architecture," 2010.

[33] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: a network programming language," in *ICFP*, 2011, pp. 279–291.

[34] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker, "Modular SDN programming with Pyretic," *USENIX Mag.*, vol. 38, no. 5, 2013.

[35] H. Hudson, "Extending access to the digital economy to rural and developing regions," The MIT Press, 2002.

[36] L. Csikor, M. Szalay, B. Sonkoly, and L. Toka, "NFPA: Network function performance analyzer," in *Proc. of NFV-SDN*, 2015, Pipelines available: http://nfpa.tmit.bme.hu/?page=usecases.

[37] Microsoft, "MSDN: Introduction to Receive-Side Scaling," [Online: http://goo.gl/BpoErm, accessed 05-07-2016].

[38] S. Bradner *et al.*, "Benchmarking methodology for network interconnect devices," RFC 2544, 1999.

[39] T. Koponen *et al.*, "Network virtualization in multi-tenant datacenters," in *NSDI*, 2014.

[40] R. Gandhi and othersg, "Duet: Cloud scale load balancing with hardware and software," in *SIGCOMM*, 2014, pp. 27–38.

[41] Intel, "Network function virtualization: Virtualized BRAS with Linux and Intel architecture," https://goo.gl/TVj8co.

[42] S. K. N. Rao, "SDN and its USE-CASES-NV and NFV," White Paper, NEC technologies India Limited, 2014.

[43] Cisco, "Cisco Data Center Infrastructure 2.5 Design Guide," https://goo.gl/kW78VM, Nov 2011.

[44] C. Kim *et al.*, "Floodless in Seattle: a scalable Ethernet architecture for large enterprises," in *SIGCOMM*, 2008, pp. 3–14.

[45] Intel Corporation, "Intel Ethernet Converged Network Adapters XL710 10/40 GbE," Datasheet, 2015.

[46] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards programmable enterprise WLANS with Odin," in *HotSDN*, 2012, pp. 115–120.

[47] N. Mysore *et al.*, "PortLand: a scalable fault-tolerant Layer 2 data center network fabric," *SIGCOMM CCR*, vol. 39, no. 4, pp. 39–50, 2009.

[48] N. Kang, Z. Liu, J. Rexford, and D. Walker, "Optimizing the "one big switch" abstraction in software-defined networks," in *CoNEXT*, 2013, pp. 13–24.

[49] M. Chiesa, G. Rétvári, and M. Schapira, "Lying your way to better traffic engineering," in *CoNEXT*, 2016.

[50] A. Gupta, M. Shahbaz, L. Vanbever, H. Kim, R. Clark, N. Feamster, J. Rexford, and S. Shenker, "SDX: a software defined Internet Exchange," in *SIGCOMM*, 2014, pp. 551–562.