C-Share: Optical Circuits Sharing for Software-Defined Data-Centers

Yaniv Ben-Itzhak¹, Cosmin Caba², Liran Schour³, Shay Vargaftik¹³
¹IBM Research Lab, Haifa, Israel ²DTU Fotnik ³Technion
{yanivb, lirans}@il.ibm.com  cosm@fotonik.dtu.dk  shayvar@tx.technion.ac.il

Abstract
Integrating optical circuit switches in data-centers is an ongoing research challenge. In recent years, state-of-the-art solutions introduce hybrid packet/circuit architectures for different optical circuit switch technologies, control techniques, and traffic rerouting methods. These solutions are based on separated packet and circuit planes which do not have the ability to utilize an optical circuit with flows that do not arrive from or delivered to switches directly connected to the circuit’s end-points. Moreover, current SDN-based elephant flow rerouting methods require a forwarding rule for each flow, which raise scalability issues. In this paper, we present C-Share – a practical, scalable SDN-based circuit sharing solution for data center networks. C-Share inherently enable elephant flows to share optical circuits by exploiting a flat upper tier network topology. C-Share is based on a scalable and decoupled SDN-based elephant flow rerouting method comprised of elephant flow detection, tagging and identification, which is utilized by using a prevalent network sampling method (e.g., sFlow). C-Share requires only a single OpenFlow rule for each optical circuit, and therefore significantly reduces the required OpenFlow rule entry footprint and setup rate rule. It also mitigates the OpenFlow outbound latency for subsequent elephant flows. We implement a proof-of-concept system for C-Share based on Mininet, and test the scalability of C-Share by using an event driven simulation. Our results show a consistent increase in the mice/elephant flow separation in the network which, in turn, improves both network throughput and flow completion time.

1. INTRODUCTION
In recent years, optical circuit switching has emerged as a promising solution for scaling data center networks. Current optical-circuit-switch/electrical-packet-switch (referred to as OCS/EPS) solutions, e.g., [1] [17] [27] [35], are based on separated OCS and EPS planes, employing the OCS for high-bandwidth, slowly varying, and long-lived flows (elephant flows), and the EPS for fast varying and short-lived flows (mice flows). Accordingly, each solution presents a method for detecting and rerouting elephant flows.

In the following we explain the lack of mice/elephant flow separation and scalability issues in current solutions. First, OCS can create low-latency high-bandwidth circuit using a relatively slow reconfigurable cross-board. OCS reconfiguration penalty, which is the time to establish a circuit, is tens of µs for 2D MEMS wavelength selective switches, e.g., [27, 30], and tens of ms for 3D MEMS optical circuit switches, e.g., [1, 4, 11, 17, 35]. Despite this penalty, previous solutions utilize a given optical circuit by transmitting only elephant flows that arrive from and delivered to switches directly connected to the optical circuit’s end-points – referred to as a private circuit. Therefore, other elephant flows that are not assigned to an optical circuit are transmitted through the EPS plane. These elephant flows are usually high persistent TCP flows, which tend to fill the network buffers end-to-end. In turn, both elephant and mice flows that share these buffers are introduced with a non-trivial queuing delay. Therefore, delay sensitive mice flows and especially coflow flows, [12, 13, 31, 37], are adversely affected.

Second, state-of-the-art-solutions, e.g., [1, 17], introduce a coupled architecture in which both the detection and rerouting of elephant flows are employed over the switches directly connected to the OCS plane. In particular, for OpenFlow (OF) based solutions [1], such coupling dictates the installation of an OpenFlow rule for each detected elephant flow in order to reroute it to the OCS plane – referred to as per-flow setup. This approach results in a significant OpenFlow entry footprint [14]. Furthermore, the OF rule setup rate is usually limited to tens of rules per second [21], and the OF rule installation requires outbound latency to take effect in the data-plane.

In this paper, we present a different approach for integrating OCS in DCN. C-Share inherently enables sharing of optical circuits, leading to better mice/elephant flow separation, by introducing a scalable OpenFlow-based solution.

In recent years, data-centers have been evolving towards a flatter aggregation/core hierarchy with more densely interconnected switches, also known as spine-leaf topologies. Such topologies can deploy and adjust capacity more easily, with better manageability, and offer more deterministic network performance, particularly in latency [32]. C-Share takes this trend one step further, and presents flat topology

¹ In this paper we use circuit and optical circuit interchangeably.
² Collection of flows with a shared completion time that depends on completion time of the last-flow.
for the upper data-center packet tier by exploiting high-radix packet switches, as depicted in Figure 1. The OCS is used to transmit elephant flows by creating network path shortcuts over the flat topology of the upper tier, hence dynamically allocating bandwidth between the packet switches. The flat upper tier topology used by C-Share inherently enables sharing of optical circuits by elephant flows which do not arrive from or delivered to switches directly connected to the circuit’s end-points.

Figure 1 presents an example of private optical circuit between S2 and S3, which transmits only elephant flows that arrive from S2 and delivered to S3, through the optical circuit. Therefore, elephant flows from S1 to S3 are transmitted through the S2→S3 packet link by S2. In turn, these elephant flows share the network buffers with mice flows between S2 and S3, which increases the end-to-end latency of both elephant and mice flows. On the other hand, the shared optical circuit between S6 and S4 transmits any elephant flow that is delivered to S4 regardless its origin switch (by a corresponding S6 switch configuration). Therefore, elephant flows from S7 to S4 are transmitted through the shared circuit by S6. Hence, a better mice/elephant flow separation is obtained in the network, significantly reducing the load over the packet links between S4, S5, and S6, and resulting in better network performance for all flows.

C-Share introduces SDN-based scalable elephant flow rerouting method supporting optical circuit sharing. C-Share exploits the servers to detect and tag elephant flows by setting the DSCP IP field, which is usually used for packet classification. Then, the DCN orchestrator identifies the elephant flows by sampling the upper tier packet switches. Therefore, in order to redirect all elephant flows to a given optical circuit by a packet switch, a single OF rule is required that matches the elephant flow DSCP tag and its destination. Hence, the OF rule footprint and OF flow setup rate are significantly reduced; and the outbound latency is mitigated for subsequent elephant flows after the circuit has been established.

The contributions of this work include:
1) New topology concept for EPS/OCS DCN that further separates mice and elephant flows, thus improves network performance.
2) Scalable SDN-based architecture that reduces the OF rule footprint and setup rate. It also mitigates the outbound latency problem of OF switches.

2. C-Share TOPOLOGY

In this section, we present the concept of C-Share topology without delving into design and options of the upper packet tier topology.

Current DCN switches offer up to 128 ports of 25Gbps [2]. In the near future, switches with 256 ports of 25Gbps are expected and apparently will be followed by switches with 256 ports of 50Gbps. As the port density increases, data-center networks become flatter with flat upper tier topology, such that the packet switches are intra-connected, thus omitting the need for an additional network layer above it. There are several well-known topologies, such as, multi dimensional torus or mesh, Flattened Butterfly [23], Dragonfly [24], and HyperX [6] that can be used to that end. In C-Share topology (Figure 1), the OCS plane is connected to all of the packet switches at the upper tier, and employs network path shortcuts and dynamic bandwidth allocation among them. We introduce two types of optical circuits that can be used in C-Share topology.

**Private Circuit** is utilized only by elephant flows that arrive from and delivered to switches directly connected to the optical circuit’s endpoints, e.g. [1][17].

**Shared Circuit** is inherently supported by C-Share topology, and can be utilized also by elephant flows that are transmitted through switches connected to the circuit’s endpoints, but arrive from or delivered to other switches.

For private circuit configuration, elephant flows which are not assigned to an optical circuit are transmitted through the
packet switches, thus might overload them. This, in turn significantly degrades the mice flows performance [10]. However, as opposed to previous solutions, C-Share topology dictates that some of these elephant flows are transmitted through switches which are already connected to an optical circuit. Therefore, by using shared optical circuits, better mice/elephant flow separation is obtained, which results in lower congestion over the upper packet tier links, leading to better network performance.

3. C-Share ARCHITECTURE

Figure 2 depicts block diagram of C-Share architecture, which decouples the elephant flow detection and rerouting phases to elephant flow detection and tagging, observation, and rerouting phases. First, the egress network traffic of each server is sampled and tracked by the Elephant Flow Detector (step 1 in Figure 2). Each flow that exceeds a given threshold for the transferred bytes and/or the flow duration (according to the criteria initialization in step 0) is detected as elephant flow, similar to [7, 14, 15]. Then, each detected elephant flow is tagged by setting a predefined value to the IP DSCP field 3, notated by DSCP_e (steps 2 and 3, over the Server 4 and the Packet Network, respectively). The Upper Tier Packet Switches (which are directly connected to the OCS plane) are monitored by the Network Observer plane to observe only the tagged elephant flows and track their bandwidth and duration 5 (step 4). Studies on live DCN traffic [22] show that elephant flows account for less than 10% of all flows. Therefore, tagging the elephant flows in advance by the servers and only tracking them over the packet switches significantly reduces the number of tracked flows by the Network Observer, which reduces CPU, memory and network usage. On the contrary, detecting the elephant flows over the packet switches require significantly more network and compute resources since all flows should be monitored.

The Network Scheduler decides which circuits to establish according to the current flow demand in the network (step 5), and informs the Infrastructure Controller (step 6). In turn, the Infrastructure Controller configures the data-plane accordingly (step 7). Then, each pair of packet switches connected to a circuit’s endpoints are installed with an OF rule to reroute matched elephant flows through this circuit. The OF rule matches the DSCP_e value in the IP header and the destination subnet connected to the switch at the other end-point of the circuit. Private circuit is configured by matching only flows ingress from ports connected to the lower tier. Shared circuit is configured by matching also ports connected to packet switches at the upper packet tier (section 3.1).

C-Share architecture requires only a single flow rule in order to transmit all of the elephant flows through a given optical circuit, either shared or private. Furthermore, subsequent elephant flows, which are generated and tagged after the corresponding optical circuit has been established, are also matched by the flow rule over the packet switches to be redirected through the optical circuit. Hence, the outbound latency is mitigated, and the required OF rule footprint and OF rule setup rate are reduced (section 3.2).

3.1 Private / Shared Circuit Configuration

Private and shared optical circuits are differed by setting which of the switch’s input ports are matched by the rerouting rule of elephant flows through the optical circuit. Therefore, different metadata values are assigned to packets from input ports connected to the lower and the upper packet tiers. Then, by mask matching on the metadata value of an ingress packet, one can configure the switch either to use the optical circuit as private by serving only packets from the lower tier, or shared by serving packets also from the upper tier.
3.2 Scalable Elephant Flow Rerouting

By leveraging the DSCP tagging of the elephant flows and the packet metadata assignment according to their corresponding input ports, C-Share results in a single OF rule for each switch that is connected to an optical circuit’s endpoint. Therefore, C-Share significantly reduces the OF footprint, as compared to previous works which requires an OF rule for each rerouted flow. Let’s consider the per-flow setup time for each switch; it results in average flow table consumption of 50%-67% for elephant flows rerouting. Since C-Share requires only single OF rule for each circuit, it results in a significantly smaller OF footprint, as we demonstrated in our evaluation (section 4). Furthermore, OF switches have limited OF rule setup rate. For instance, HP ProCurve 5400zl switches support up to 1.7K OpenFlow entries [14]: HP ProCurve J9451A supports 1.5k OF entries [21]; HP ProCurve 5406zl, Pica8 P-3290, and Dell PowerConnect 8132F support up to 1.5k, 2k and 750 rules, respectively [25]. Hence, the currently used per-flow setup approach results in average flow table consumption of 50%-67% for elephant flows rerouting. Since C-Share requires only single OF rule for each circuit, it results in a significantly smaller OF footprint, as we demonstrate in our evaluation (section 4). Furthermore, OF switches have limited OF rule setup rate. For instance, [21] indicates that flow rule setup rate of OF switches is limited to approximately 40 flow/sec. Clearly, C-Share significantly reduces the required OF setup rate; hence, proposes feasible solution for current OF switches.

Once an optical circuit is configured, subsequent ingress elephant flows arriving to the switch ports are matched by the OF rule and then transmitted through the optical circuit. Consequently, C-Share mitigates the OF outbound latency[6] for such subsequent flows. The OF outbound latency has been measured by previous works: [20] reports that the outbound latency can be as high as 30ms, [33] measures the outbound latency of two switches by using OFLOPS. They report ranges of 50-1000ms and 8-2000ms depending on the number of inserted flow entries. [25] measures outbound latency of up to 400ms. The outbound latency is at the same order of the 3D MEMS OCS reconfiguration penalty or even higher. However, the OCS reconfiguration penalty affects the network only once for each optical circuit configuration. Whereas, the outbound latency penalty has a larger network degradation potential. Therefore, by avoiding this additional latency for each subsequent elephant flow served by an optical circuit, C-Share results in better network performance.

4. EVALUATION

Topologies: we evaluate C-Share for two flat upper tier topologies: Ring and Flattened Butterfly [23]. Ring topology offers a simple wire connectivity, and is used by industrial DCNs. Facebook [16] presents a DCN architecture which uses Ring topology to connect the cluster and aggregation switches; and Google [54] uses Ring topology to connect cluster routers. Flattened butterfly (FBFly) takes advantage of high-radix switches to create a scalable, yet low-diameter network. Google [5] show that FBFly is a power efficient topology for high-performance datacenter networks. For both topologies, the bandwidth of the packet and circuit links are set for 1/10 ratio, as used by [27]. We use two DCN traces to simulate aggregated traffic to the upper packet tier, with skewed and uniform traffic patterns.

Traces from the University of Wisconsin (UNI) are presented in [9], which contain recorded traffic among approximately 2900 servers for a one hour duration. Analysis of this trace by [23] shows mostly sparse and skewed traffic. We analyze UNI ppcap traces and extract the TCP sessions properties, and their start time. Then, in order to simulate DCNs with different number of hosts, we consolidate the hosts by subnets, and merge the traffic for each subnet to represent a node in our modified trace. The subnet sizes are chosen accordingly to meet the required number of hosts. In addition, we reduce the time intervals between the sessions to obtain moderate network load.
Synthetic Data Center Trace (Uniform) is created based on traffic characteristics from [7] [8] [22] [27], such that elephant flows are 10% of the number of flows and accommodate 90% of the demand. We generate traffic with random distribution of sessions between mice flow traffic (2KB to 32KB) and elephant flows (up to 100MB) [8] [19], with uniform traffic distribution [7].

**4.1 Emulation**

We develop an emulated environment of C-Share by using Mininet [25] version 2.2.1 running over an IBM x3550 M4 server with 196GB of RAM, 24 Xeon-E5-2630@2.3GHz CPUs (with six cores each), and Ubuntu 14.04 with Linux 3.19 kernel. We use sFlow [36] to sample the egress flows of the hosts by the Elephant Flow Detector, and to sample the Open vSwitches by the Network Observer. The OCS is emulated by a constrained Open vSwitch to employ optical circuits, such that only one input port can be configured to transmit to any given output port. Each OCS reconfiguration is emulated by first removing the colliding optical circuits, and configuring the new requested optical circuits after a 20ms delay to emulate 3D MEMS OCS typical reconfiguration penalty, e.g. [4]. We evaluate an upper tier Ring with 10 packet switches and 3-ary-3-flat FBFly (9 packet switches) with packet and circuit links of 10Mbps and 100Mbps, respectively. The network traffic is generated by iperf3 [3] according to UNI1 and uniform traces configured for moderated network load without hitting the CPU-bound of the server that running Mininet.

![Average Mice Throughput](image1)

(a) Average Mice Throughput

![Average Elephant Throughput](image2)

(b) Average Elephant Throughput

Figure 4: Average throughput as reported by iperf3 for Mininet environment under moderate network load.

| Trace | Method | Topology | Private Circuits | Shared Circuits |
|-------|--------|----------|-----------------|-----------------|
| UNI1  | Per-flow setup | Ring (10 switches) | 445 | 449 |
|       | C-Share | Ring (10 switches) | 26 | 31 |
| Uniform | Per-flow setup | Ring (10 switches) | 563 | 588 |
|        | C-Share | Ring (10 switches) | 45 | 52 |

Table 1: OF rule footprint for elephant flow rerouting during one minute of trace, under moderate network load. C-Share significantly reduces the OF footprint.

Figure 4 presents a comparison of the average throughput as reported by iperf3 between mice and elephant flows, for both network traces over the Ring and FBFLy topologies. In general, shared circuits improve the throughput of both elephant and mice flows as compared to private circuits. In particular, we observe that: (1) Skewed traffic (UNI1 trace) introduces patterns which can be exploited by shared circuits, such as many elephant flows from different sources to the same destination. Therefore, shared circuits further improve the network performance of skewed traffic, for instance by 29% for mice flows over FBFLy, and 57% for elephant flows over Ring; whereas, uniform traffic is improved by 9% and 26%, respectively. (2) The connectivity of Ring topology is limited, which results in degraded performance as compared to FBFLy. Therefore, the connectivity and network throughput of Ring topology can be further improved by the shared circuits. In particular, the shared circuits improve the network throughput of Ring topology by 21%-57%, as compared to FBFLy which is improved by 8%-28%.

In addition, Table 1 presents the OF rule footprint of UNI1 and uniform traces, under moderate network load.

**4.2 Simulation**

We use an event driven simulation to evaluate the completion time of mice coflows and elephant flows, and measure the corresponding OF rule footprint for rerouting elephant flows by the packet switches through private or shared circuits. We use the synthetic uniform traces to demonstrate the scalability of C-Share under intensive network load. Specifically, we generate network traffic comprised of mice coflows and elephant flows. The mice coflows are 90% of the number of flows, and accommodate 10% of the total demand. We simulate Ring and Flattened Butterfly upper packet tier topologies, with varied number of packet switches. Each packet switch serves 40 hosts. The packet and circuit links are set to 10Gbps and 100Gbps, respectively.

Figure 4 presents the average completion run-time of mice coflows and elephant flows for private and shared circuit configurations over 60 trials. The shared circuits improve the average completion time by 20% for a Ring with 10 switches and up to 30% for a Ring with 16 switches. The Ring topology is unscalable in terms of connectivity. Therefore, the shared circuits can significantly increase the topology connectivity and mice/elephant flow separation, which results in increased improvement of the completion time as the Ring size increases. On the other hand, since FBFLy is scalable, the improvement of the completion time by shared circuits
equals 15%-20% for all FBFly sizes; hence, the shared circuits results in relatively constant mice/elephant flow separation degree. The same applies for the OF rule footprint presented in Figure 6. The required OF rules of per-flow setup for Ring remains constant and higher than 1.7k (prevalent OF table size [14, 21, 25]). On the other hand, due to the scalability of FBFly, as the size of FBFly increases, less OF rules are required for rerouting the elephant flows through private or shared circuits. However, the OF footprint of per-flow setup is still high and is significantly reduced by C-Share.

5. RELATED WORK

EPS/OCS DCN solutions, e.g. [17, 27, 35], present different approaches for integrating OCS in DCN. The control planes presented in these works are based on non-SDN methods, thus limited as compared to SDN-based solutions. c-Through [35] uses predefined VLANs for static EPS/OCS planes, and tags elephant flows with the corresponding VLAN, without the ability to dynamically configure the network. Helios [17] implementation consists of Monaco packet switches and sets its forwarding table to reroute all flows that are delivered to a specific destination pod, without the ability to separate among mice and elephant flows to EPS and OCS planes, respectively. REACToR [27] presents state-of-the-art FPGA-based solution; however, do not propose an elephant tagging and rerouting technique. Furthermore, these works are based on separated EPS and OCS planes, which restrict the separation of mice and elephant flows in the network. On the other hand, C-Share inherently supports such mice/elephant flow separation.

ProjecToR [18] presents a free-space optics (FSO) solution for DCN, composed of dedicated and opportunistic optical links. C-Share can be employed over such solution, and offer optical circuit sharing over the dedicated optical links.

SDN-based works present elephant [14] and network-limited [2] flow scheduling for EPS-only DCNs. [1] presents SDN-based solution for OCS/EPS DCN. These works use a specific OF rule for each rerouted flow. Hence, they introduce the aforementioned OF scalability issues. Namely, table rule footprint, setup rate, and outbound latency. All are mitigated by C-Share.

6. CONCLUSIONS AND FUTURE WORK

In this paper we proposed C-Share, a new approach for integrating OCS in DCN. We demonstrated how C-Share inherently supports circuit sharing that further separates mice and elephant flows leading to increased performance. We presented a scalable SDN-based solution including elephant flow rerouting that requires only a single OF rule per circuit.

This work is a starting point on a way towards a full-fledged implementation of C-Share. We list two of C-Share advanced architectural aspects.

Advanced Circuit Sharing: We presented shared circuits only for last hop routing. Namely, the sharing is employed for elephant flows delivered to one of the circuit’s endpoints. By advanced configuration, we can enable circuit sharing with elephant flows at any hop along their routes.

Upper Tier Topology: C-Share is evaluated for Ring and FBFly upper tiers. Other topologies might offer better mice/elephant flow separation by exploiting circuit sharing.
7. REFERENCES

[1] Calient. 3Dms optical circuit switching for software defined data centers and metro networks.

[2] High-Density 25/100 Gigabit Ethernet StrataXGS Tomahawk Ethernet Switch Series. https://www.broadcom.com/products/ethernet-communication-and-switching/switching/bcm56960-series

[3] iPerf - The TCP, UDP and SCTP network bandwidth measurement tool. https://iperf.fr/

[4] Polatis 6000n Protection Services Switch Data Sheet. http://www.polatis.com/datasheets/products/Polatis_6000n_Protection_Services_Switch_Data_Sheet.pdf

[5] D. A. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and A. Regev. ElephantFlow: scalable flow management for high-performance networks. In ACM SIGCOM Computer Communication Review, volume 41, pages 254–265. ACM, 2011.

[6] N. Farrington and A. Andreyev. Facebook data center network architecture. In IEEE Optical Interconnects Conf. Citeseer, 2013.

[7] 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. ACM SIGCOMM Computer Communication Review, 41(4):339–350, 2011.

[8] M. Ghobadi, R. Mahajan, A. Pandishayee, N. Devanur, J. Kulkarni, G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper. Agile reconfigurable data center interconnect. In Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference, pages 216–229. ACM, 2016.

[9] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Labirini, D. A. Maltz, P. Patel, and S. Sengupta. VIT: a scalable and flexible data center network. In ACM SIGCOMM computer communication review, volume 39, pages 51–62. ACM, 2009.

[10] K. He, J. Khalid, S. Das, A. Gember-Jacobson, C. Prakash, A. Akella, L. E. Li, and M. Thottan. Latency in software defined networks: Measurements and mitigation techniques. In Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pages 435–436. ACM, 2015.

[11] D. Y. Huang, K. Yocum, and A. C. Snoeren. High-fidelity switch models for software-defined network emulation. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, pages 43–48. ACM, 2013.

[12] A. Vahdat, M. Ku¨ zniar, P. Pere´ sini, and D. Kosti´c. What you need to know about SDN flow tables. In Passive and Active Measurement, pages 347–359. Springer, 2015.

[13] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: rapid prototyping for software-defined networks. In Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, pages 19. ACM, 2010.

[14] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, G. Papen, A. C. Snoeren, and G. Porter. Circuit switching under the radar with REACToR. In 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), pages 1–15, 2014.

[15] H. Liu, M. K. Mukerji, D. A. Maltz. Network traffic characteristics of data centers in the wild. In Proc. ACM SIGCOMM conference on Internet measurement, 2010.

[16] J. Bowers, A. Raza, D. Tardent, and J. Miglani. Advantages and control of hybrid packet optical-circuit-switched data center networks. In Photonics in Switching, pages PM2C–4. Optical Society of America, 2014.

[17] K. Chen, A. Singla, A. Singh, K. Ramachandran, et al. QSA: An Optical Switching Architecture for Data Center Networks With Unprecedented Flexibility. Networking, IEEE/ACM Transactions on, 2014.

[18] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, G. Prabhakar, S. Sengupta, and M. Sritharan. Data Center TCP (DCTCP). In ACM SIGCOMM computer communication review, volume 40, pages 63–74. ACM, 2010.

[19] T. Benson, J. Kim, W. J. Dally, and D. Abts. Polatis 6000n Protection Services Switch Data Sheet.

[20] G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper. Passive and Active Measurement of data centers in the wild. In the second ACM SIGCOMM workshop on Unprecedented Flexibility, pages 216–229. ACM, 2016.

[21] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee. DevoFlow: scaling flow management for high-radix networks. In ACM SIGCOMM Computer Communication Review, 2008.

[22] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker. Extending networking into the virtualization layer. In Hotnets, 2009.

[23] G. Porter, R. Strong, N. Farrington, A. Forencich, et al. Integrating microsecond circuit switching into the data center. In Proc. ACM SIGCOMM., 2013.

[24] Z. Qiu, C. Stein, and Y. Zhong. Minimizing the total weighted completion time of coflows in datacenter networks. In Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, pages 294–303. ACM, 2015.

[25] E. Roberts and L. Parashis. The role of optical interconnections in data-center architecture evolution. In Optical Fiber Communication Conference. Optical Society of America, 2013.

[26] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore. OFLOPS: An open framework for OpenFlow switch evaluation. In Passive and Active Measurement, pages 85–95. Springer, 2012.

[27] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armstead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, et al. Jupiter rising: A decade of colo topologies and centralized control in google’s datacenter network. ACM SIGCOMM Computer Communication Review, 45(4):183–197, 2015.

[28] M. Wang, B. Li, and Z. Li. sFlow: towards resource-efficient and agile service federation in service overlay networks. In Distributed Computing Systems, 2004. Proceedings. 24th International Conference on, pages 628–635. IEEE, 2004.

[29] Y. Zhao, K. Chen, W. Bai, M. Yu, C. Tian, Y. Geng, Y. Zhang, D. Li, and S. Wang. Rapier: Integrating routing and scheduling for flow-aware data center networks. In Computer Communications (INFOCOM), 2015 IEEE Conference on, pages 424–432. IEEE, 2015.