Public Review for

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

S. Vargaftik, C. Caba, L. Schour, Y. Ben-Itzhak

C-Share tackles the challenge of designing a data-center network that integrates optical circuit switches with packet switch designs. In this solution, the authors propose identifying large Elephant network flows and coalescing them onto a common optical circuit. This simplifies rerouting of these flows and reduces the amount of SDN rules required to manage these flows. Through their evaluation, the authors are able to show that their solution effectively separates elephant and smaller mice flows and improves throughput and completion time. The reviewers appreciated the application of host-based elephant flow detection and hybrid optical/electric network designs in the context of flatter network topologies and the gains observed by their design.

Public review written by

Phillipa Gill

University of Massachusetts – Amherst
C-Share: Optical Circuits Sharing for Software-Defined Data-Centers

Shay Vargaftik
VMware Research
shayv@vmware.com

Liran Schour
IBM Research Lab
lirans@il.ibm.com

Cosmin Caba
DTU Fotonik
cosmincaba@gmail.com

Yaniv Ben-Itzhak
VMware Research
ybenitzhak@vmware.com

ABSTRACT
Integrating optical circuit switches in data-centers is an on-going 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 re-routing methods. These solutions are based on separated packet and circuit planes that cannot 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 re-routing methods require a forwarding rule for each flow, which raises scalability issues.

In this paper, we present C-Share – a scalable SDN-based circuit sharing solution for data center networks. C-Share inherently enables elephant flows to share optical circuits by exploiting a flat top-of-rack tier network topology. C-Share is based on a scalable and decoupled SDN-based elephant flow re-routing 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 rule rate. 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.

CCS CONCEPTS
• Networks → Data center networks; • Hardware → Photonic and optical interconnect;

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, 28, 36], 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 re-routing 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 circuits using a relatively slow reconfigurable cross-board. OCS reconfiguration penalty, which is the time to establish a circuit, is tens of $\mu$s for 2D MEMS wavelength selective switches, e.g., [28, 31], and tens of ms for 3D MEMS optical circuit switches, e.g., [1, 4, 11, 17, 36]. 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 queueing delay. Therefore, delay-sensitive mice flows, and especially coflows [12, 13, 32, 38], are adversely affected.

Second, state-of-the-art solutions, e.g., [1, 17], introduce a coupled architecture in which both the detection and re-routing 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 re-route 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 [22], and the OF rule installation requires outbound latency to take effect in the data-plane (i.e., the time from the OF rule generation by the control-plane till the data-plane is configured accordingly).

In this paper, we present C-Share – 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 efficiently, with better manageability, and offer more deterministic network performance, particularly in latency [33]. C-Share takes this trend one step further, and

1 In this paper we use circuit and optical circuit interchangeably.
2 Collection of flows with a shared completion time that depends on the completion time of the last-flow.
Figure 1: The C-Share topology concept. Private optical circuit result in inefficient mice/elephant flow separation over $S_2 \rightarrow S_3$ link. Contrarily, a shared optical circuit reduces the load over $S_6 \rightarrow S_5$ and $S_5 \rightarrow S_4$ links by better mice/elephant flow separation.

C-Share introduces SDN-based scalable elephant flow re-routing 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 ToR tier switches connected to the optical circuit endpoints, e.g., [1, 17]. Private circuits are used throughout the paper for comparison and evaluation purposes by representing the current EPS/OCS solutions.

Shared Circuit is inherently supported by C-Share topology, and can also be utilized 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 that 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 ToR tier links, leading to better network performance.
3 C-SHARE ARCHITECTURE

Figure 2 depicts a block diagram of C-Share architecture, which decouples the elephant flow detection and re-routing phases to elephant flow detection and tagging, observation, and re-routing 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 field4, notated by DSCP_e (steps 2 and 3, over the Server and the Packet Network, respectively). The ToR 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 duration5 (step 4). Studies on live DCN traffic [23] 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 only matching flows ingress from ports connected to the servers. Shared circuit is configured by matching also ports connected to the ToR switches (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 differ by setting which of the switch’s input ports are matched by the re-routing rule of elephant flows through the optical circuit. Hence, different metadata values are assigned to packets from input ports connected to the servers and the ToR tier. Then, by mask matching on the metadata value of an ingress packet, one can configure the switch to either use the optical circuit as private by serving only packets from the servers, or shared by also serving packets from the ToR tier.

Figure 3 demonstrates Open vSwitch [30] configuration for private and shared circuits. At initialization, metadata values of 0b01 and 0b11 are assigned to packets arriving from the ToR tier and the servers, respectively. For a private circuit, a single OF rule is set to match packets with metadata values of 0b1* by using 0b10 mask. Therefore, only packets from all input ports connected to the servers are matched and transmitted through the circuit. Similarly, for a shared circuit, packets with metadata of 0b1 are matched by using a 0b01 mask. Hence, packets arriving from all input ports connected to both the ToR tier and the servers are matched and transmitted through the circuit. As described above, the OF rule is also set to capture the DSCP_e value (nw_dst), and the servers subnet destination (nw_dst) of the switch connected to the other end of the optical circuit.

3.2 Scalable Elephant Flow Re-Routing

By 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 end-point. Hence, C-Share significantly reduces the OF footprint, as compared to previous works which requires an OF rule for each rerouted flow – per-flow setup, e.g. [1, 7, 14].
OpenFlow entries [14]; HP ProCurve J9451A supports 1.5k OF. It results in a significantly smaller OF footprint, as we demonstrate.

Topologies: Ring and Flatted Butterfly [24]. Ring topology offers better network performance. Assuming that there are on average 1k simultaneous elephant flows [7, 9, 23] between two packet switches at the ToR tier, means that existing approaches require 1k OF rules for each of the packet switches, which might consume most of current OF switches flow table size. For instance, HP ProCurve 5400zl switches support up to 1.7K OpenFlow entries [14]. HP ProCurve J9451A supports 1.5k OF entries [22]; HP ProCurve 5406zl, Pica8 P-3290, and Dell PowerConnect 8132F support up to 1.5k, 2k and 750 rules, respectively [26]. Hence, the currently used per-flow setup approach results in an average flow table consumption of 50%-67% for elephant flows re-routing. Since C-Share requires only a 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 a limited OF rule setup rate. For instance, [22] indicates that flow rule setup rate of OF switches is limited to approximately 40 flow/sec. C-Share significantly reduces the required OF setup rate; hence, it proposes a feasible solution for current OF switches.

Once an optical circuit is configured, subsequent ingress elephant flows arriving at the packet switches are matched by the OF rule and, in turn, transmitted through the optical circuit. Consequently, C-Share mitigates the OF outbound latency for such subsequent flows. Previous works have measured the OF outbound latency: [21] reports that the outbound latency can be as high as 30ms; [34] measures the outbound latency of two switches by using OFLOPS, and reports ranges of 50-1000ms and 8-2000ms depending on the number of inserted flow entries; [26] measures outbound latency of up to 400ms. These measured outbound latency values are 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

Flat ToR Topologies: we evaluate C-Share for two flat ToR tier topologies: Ring and Flatted Butterfly [24]. Ring topology offers simple connectivity and is used by industrial DCNs. Facebook [16] presents a DCN architecture that uses Ring topology to connect the cluster and aggregation switches, and Google [35] uses Ring topology to connect cluster routers. Flatted butterfly (FBFly) creates a scalable, yet low-diameter network. Google [5] shows that FBFly is a power-efficient topology for high-performance data center networks. For both topologies, the bandwidth of the packet and circuit links are set for 1/10 ratio, as used by [28].

Traces: We use two DCN topologies to simulate aggregated traffic to the ToR tier, with skewed and uniform traffic patterns.

University of Wisconsin (UNI1) Traces are presented in [9], which contain recorded traffic among approximately 2900 servers for a one-hour duration. Analysis of this trace by [29] shows mostly sparse and skewed traffic. We analyze UNII pcap traces and extract the TCP session properties and their start time. Then, in order to simulate DCNs with a different number of hosts, we consolidate the hosts by subnets and merge the traffic for each to represent a node in our modified trace. The subnet sizes are chosen to meet the required number of hosts. We also reduce the time intervals between the sessions to obtain a moderate network load.

Synthetic Data Center Trace (Uniform) is created based on traffic characteristics from [7, 8, 23, 28], such that elephant flows are 10% of the number of flows and account for 90% of the demand. We generate traffic with random distribution of sessions between mice (2KB to 32KB) and elephant (up to 100MB) flows [8, 19], with a uniform traffic distribution [7].

Heuristic Scheduling: We develop a scheduling heuristic for C-Share, in which we perform demand estimation and OCS reconfigurations every 200ms (recall that the OCS reconfiguration penalty is set to 20ms to emulate 3DMEMS OCS typical reconfiguration penalty, e.g. [4]). For such a time windows, the scheduling rate is one order of magnitude slower than the OCS reconfiguration penalty. Hence, the demand estimation and OCS reconfigurations result in reasonably low overhead.

Specifically, at the beginning of each 200ms window, we perform demand estimation and use this information to decide which circuits to reconfigure. We first transform the demand matrix among the ToR switches into a new one according to the following rules which, by definition, are different for private ans shared circuits:

Private Circuits: the weight of entry (i, j) equal the product of (a) the demand originated at sender i and destined to receiver j and (b) the length of the shortest path between i and j.

Shared Circuits: the weight of entry (i, j) equal the product of (a) the total demand at sender i that is destined to receiver j (b) the length of the shortest path between i and j.

We then interpret this new transformed matrix as a bipartite graph and choose the maximal weighted matching that is translated to the new circuitry to be configured for this time window. The reasoning behind using the product of the demand and the length of the shortest path for the matrix entries is to greedily offload as much traffic as possible from the EPS (assuming OSPF or ECMP routing by the EPS, a packet traverses x links where x is the length of the shortest path among the source and the destination).

Remark. The heuristic scheduling we employ utilizes the shared circuits only for last-hop routing. That is, only flows that are destined to the receiver side of a circuit are considered (and weighted).
when computing the transformed demand matrix. C-Share does not have this limitation. Therefore, as we further convey in the future work section, finding advanced scheduling techniques for C-Share is of interest and is a future research challenge.

4.1 Emulation

We develop an emulated environment of C-Share by using Mininet [27] 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 [37] 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 (emulating a 3D MEMS OCS). We evaluate a ToR 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 UNI trace and uniform traces configured under moderate network load. Before turning to the focus of our evaluation, which demonstrates the advantage of using shared circuits as compared to private circuits, we depict in Table 1 a comparison of the average network throughput per flow, with and without an OCS plane and when employing shared circuits. As can be seen, having an OCS plane indeed improves the network throughput substantially. The improvement for the Ring topology is more significant since it inherently offers more limited connectivity than FBFly.

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 that 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 2 presents the OF rule footprint of UNI1 and uniform traces, under moderate network load.

**Table 2: OF rule footprint for elephant flow re-routing during one minute of trace, under moderate network load.**

C-Share significantly reduces the OF footprint.

Before turning to the focus of our evaluation, which demonstrates the advantage of using shared circuits as compared to private circuits, we depict in Table 1 a comparison of the average network throughput per flow, with and without an OCS plane and when employing shared circuits. As can be seen, having an OCS plane indeed improves the network throughput substantially. The improvement for the Ring topology is more significant since it inherently offers more limited connectivity than FBFly.

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 that 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 2 presents the OF rule footprint of UNI1 and uniform traces, under moderate network load.

**Table 2: OF rule footprint for elephant flow re-routing during one minute of trace, under moderate network load.**

C-Share significantly reduces the OF footprint.

Before turning to the focus of our evaluation, which demonstrates the advantage of using shared circuits as compared to private circuits, we depict in Table 1 a comparison of the average network throughput per flow, with and without an OCS plane and when employing shared circuits. As can be seen, having an OCS plane indeed improves the network throughput substantially. The improvement for the Ring topology is more significant since it inherently offers more limited connectivity than FBFly.

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 that 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 2 presents the OF rule footprint of UNI1 and uniform traces, under moderate network load.

**Table 2: OF rule footprint for elephant flow re-routing during one minute of trace, under moderate network load.**

C-Share significantly reduces the OF footprint.

Before turning to the focus of our evaluation, which demonstrates the advantage of using shared circuits as compared to private circuits, we depict in Table 1 a comparison of the average network throughput per flow, with and without an OCS plane and when employing shared circuits. As can be seen, having an OCS plane indeed improves the network throughput substantially. The improvement for the Ring topology is more significant since it inherently offers more limited connectivity than FBFly.

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 that 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 2 presents the OF rule footprint of UNI1 and uniform traces, under moderate network load.

**Table 2: OF rule footprint for elephant flow re-routing during one minute of trace, under moderate network load.**

C-Share significantly reduces the OF footprint.

Before turning to the focus of our evaluation, which demonstrates the advantage of using shared circuits as compared to private circuits, we depict in Table 1 a comparison of the average network throughput per flow, with and without an OCS plane and when employing shared circuits. As can be seen, having an OCS plane indeed improves the network throughput substantially. The improvement for the Ring topology is more significant since it inherently offers more limited connectivity than FBFly.

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 that 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 2 presents the OF rule footprint of UNI1 and uniform traces, under moderate network load.

**Table 2: OF rule footprint for elephant flow re-routing during one minute of trace, under moderate network load.**

C-Share significantly reduces the OF footprint.

Before turning to the focus of our evaluation, which demonstrates the advantage of using shared circuits as compared to private circuits, we depict in Table 1 a comparison of the average network throughput per flow, with and without an OCS plane and when employing shared circuits. As can be seen, having an OCS plane indeed improves the network throughput substantially. The improvement for the Ring topology is more significant since it inherently offers more limited connectivity than FBFly.

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 that 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 2 presents the OF rule footprint of UNI1 and uniform traces, under moderate network load.

**Table 2: OF rule footprint for elephant flow re-routing during one minute of trace, under moderate network load.**

C-Share significantly reduces the OF footprint.
Figure 5: Average completion time of mice coflows and elephant flows under intensive network load, over two ToR topologies: (a) Ring (10 to 16 packet switches) and (b) FBFly (9 to 36 packet switches).

Figure 6: OpenFlow rule footprint per switch during one minute of network trace under intensive network load, over two ToR tier topologies: (a) Ring (10 to 16 packet switches) and (b) FBFly (9 to 36 packet switches). The horizontal line indicates 1.7k OF rule entries. Any greater OF entry count might be an unfeasible scenario.

Ring and Flattened Butterfly ToR topologies, with a varied number of packet switches. Each packet switch serves 40 hosts. The packet and circuit links are set to 10Gbps and 100Gbps, respectively.

Figure 5 presents the average completion 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 poor in terms of connectivity. Therefore, the shared circuits can significantly increase the connectivity and mice/elephant flow separation, which results in increased improvement 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 to 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, 22, 26]). On the other hand, due to the scalability of FBFly, as the size of FBFly increases, less OF rules are required for re-routing 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, 28, 36], 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 [36] 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. It 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 [28] presents a state-of-the-art FPGA-based solution, which offers up to 100Gbps connectivity between any two hosts (hence, without oversubscribing the network). Furthermore, REACToR does not use an elephant tagging, since the mice and elephants flows are en-queued on different NICs by the end-hosts. C-Share, on the other end, targets a different network solution in which the network is oversubscribed and shared between mice and elephant flows; therefore, elephant flows tagging is required. 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 among the dedicated optical links. SDN-based works present elephant tagging, since the mice and elephants flows are en-queued on different NICs by the end-hosts. C-Share can be employed over such solution, and offer optical circuit sharing among the dedicated optical links. SDN-based works present elephant tagging, since the mice and elephants flows are en-queued on different NICs by the end-hosts. C-Share can be employed over such solution, and offer optical circuit sharing among the dedicated optical links. SDN-based works present elephant tagging, since the mice and elephants flows are en-queued on different NICs by the end-hosts. C-Share can be employed over such solution, and offer optical circuit sharing among the dedicated optical links. SDN-based works present elephant tagging, since the mice and elephants flows are en-queued on different NICs by the end-hosts. C-Share can be employed over such solution, and offer optical circuit sharing among the dedicated optical links. SDN-based works present elephant tagging, since the mice and elephants flows are en-queued on different NICs by the end-hosts. C-Share can be employed over such solution, and offer optical circuit sharing among the dedicated optical links. SDN-based works present elephant tagging, since the mice and elephants flows are en-queued on different NICs by the end-hosts. C-Share can be employed over such solution, and offer optical circuit sharing among the dedicated optical links. SDN-based works present elephant tagging, since the mice and elephants flows are en-queued on different NICs by the end-hosts. C-Share can be employed over such solution, and offer optical circuit sharing among the dedicated optical links. SDN-based works present elephant tagging, since the mice and elephants flows are en-queued on different NICs by the end-hosts. C-Share can be employed over such solution, and offer optical circuit sharing among the dedicated optical links. SDN-based works present elephant tagging, since the mice and elephants flows are en-queued on different NICs by the end-hosts. C-Share can be employed over such solution, and offer optical circuit sharing among the dedicated optical links. SDN-based works present elephant tagging, since the mice and elephants flows are en-queued on different NICs by the end-hosts. C-Share can be employed over such solution, and offer optical circuit sharing among the dedicated optical links. SDN-based works present elephant tagging, since the mice and elephants flows are en-queued on different NICs by the end-hosts. C-Share can be employed over such solution, and offer optical circuit sharing among the dedicated optical links. SDN-based works present elephant tagging, since the mice and elephants flows are en-queued on different NICs by the end-hosts. C-Share can be employed over such solution, and offer optical circuit sharing among the dedicated optical links.