Circuit Switching Under the Radar with REACToR

He Liu, Feng Lu, Alex Forencich, Rishi Kapoor, Malveeka Tewari
Geoffrey M. Voelker, George Papen, Alex C. Snoeren, and George Porter

University of California, San Diego

Abstract

The potential advantages of optics at high link speeds have led to significant interest in deploying optical switching technology in data-center networks. Initial efforts have focused on hybrid approaches that rely on millisecond-scale circuit switching in the core of the network, while maintaining the flexibility of electrical packet switching at the edge. Recent demonstrations of microsecond-scale optical circuit switches motivate considering circuit switching for more dynamic traffic such as that generated from a top-of-rack (ToR) switch. Based on these technology trends, we propose a prototype hybrid ToR, called REACToR, which utilizes a combination of packet switching and circuit switching to appear to end-hosts as a packet-switched ToR.

In this paper, we describe a prototype REACToR control plane which synchronizes end host transmissions with end-to-end circuit assignments. This control plane can react to rapid, bursty changes in the traffic from end hosts on a time scale of 100s of microseconds, several orders of magnitude faster than previous hybrid approaches. Using the experimental data from a system of eight end hosts, we calibrate a hybrid network simulator and use this simulator to predict the performance of larger-scale hybrid networks.

1 Introduction

Designing scalable, cost-effective, packet-switched interconnects that can support the traffic demands found in modern data centers is already an extremely challenging problem that is only getting harder as per-server link rates move from 10 to 40 to 100 Gb/s and beyond. In this paper, we focus particularly on the challenge of upgrading an existing network fabric that supports 10-Gb/s end hosts to a network that can deliver 100 Gb/s to each end host. We argue that this transition is inevitable because the PCIe Gen3 bus found in many current servers can support 128 Gb/s, making emerging 100-Gb/s NICs a drop-in upgrade for existing hardware.

Unlike previous generational upgrades, moving from 10- to 100-Gb/s link rates requires a fundamental transition in the way a data center is wired. At 100 Gb/s, inexpensive copper cabling can no longer be used at distances greater than a few meters: virtually all cables other than those internal to an individual rack must be optical. If these cables interconnect electronic packet switches, they further require optoelectronic transceivers at both ends. Many popular packet-switched data-center topologies like multi-rooted trees [25] require large numbers of connections between racks. Hence, the cost of these designs begins to be dominated not by the constituent packet switches, but instead by the transceivers mandated by the optical interconnects necessary to support the increased link speed [10].

In contrast, if the switches internal to the network fabric are themselves optical, the need for transceivers can be significantly reduced. Researchers have previously proposed hybrid architectures consisting of a combination of packet switches and optical circuit switches managed by a common logical control plane [7, 10, 26]. Traditionally, however, their applicability has been limited by the delay incurred when reconfiguring the circuit switches, as traffic has to be buffered while waiting for a circuit assignment. Architectures based upon legacy optical circuit switches designed for wire-area applications are fundamentally dependent on stable, aggregated traffic to amortize their long reconfiguration delays. Therefore, their use has been restricted to either the core of the network [10] or to highly constrained workloads [26].

Researchers have recently demonstrated optical circuit switch prototypes with microsecond-scale reconfiguration delays [6, 17, 19]. In prior work, we showed that such a switch, when coupled with an appropriate control plane [23], has the potential to support more dynamic traffic patterns, potentially extending the applicability of circuit switches to cover the entire network fabric required to interconnect racks of servers within a data center. Circuit switching alone, however, incurs substantial delays in order to achieve efficiency (e.g., 61–300 μs to deliver 65–95% of the bandwidth of a comparable packet switch in the case of our switch [23]), rendering it inadequate to meet the demands of latency sensitive traffic within a data center. Moreover, the buffering required to tolerate such delays with large port counts at 100 Gb/s is substantial. By integrating a certain level of packet switching, hybrid fabrics have the potential to address

these shortcomings. Existing hybrid designs, however, are not capable of coping with the lack of traffic stability and aggregation present at the rack level of today's data centers [2, 4, 15, 16].

In this paper, we propose a hybrid network architecture in which optical circuit switching penetrates the data center network to the top-of-rack (ToR) switch. We leverage our recent work on the Mordia optical circuit switch to build and experimentally prototype the first hybrid network control plane that uses rapidly reconfigurable optical circuit switches to potentially provide packet-switch-like performance at substantially lower cost than an entirely packet-switched network. Our hybrid network design consists of a 100-Gb/s optical circuit-switched network deployed alongside a pre-existing 10-Gb/s electrical packet-switched network. In this model, ToR switches support 100-Gb/s downlinks to servers, and are "dual-homed" to a legacy 10-Gb/s electrical packet switched network (EPS) and a new 100-Gb/s optical circuit switched network (OCS).

REACToR's design is based on two key insights. The first is that it is impractical to buffer incoming traffic bursts from each end host within the ToR's switch memory. For a traditional in-switch time-division, multiple-access (TMDA) queueing discipline, this architecture would require a dedicated input buffer for each potential circuit destination. Given the unpredictable nature of the end-host network stack [16], these buffers would likely need to be quite large.

Instead, REACToR buffers bursts of packets in low-cost end-host DRAM memory until a circuit is provisioned, at which point the control plane explicitly requests the appropriate burst from each end host using a synchronous signaling protocol that ensures that the instantaneous offered load matches the current switch configuration. Because each REACToR is dual-homed to an EPS, the control plane can simultaneously schedule the latency-sensitive traffic over the packet switch. The packet switch can also service unexpected demand due to errors in demand estimation or circuit scheduling.

The second insight is that if circuit switching is sufficiently fast, then delays due using flow-level circuit-switched TDMA at the end-host network stack will not degrade the performance of higher-level packet-based protocols; in a sense the circuit switch will "fly under the radar" of these end-host transport protocols. As technology trends enable faster OCS reconfiguration times, this hybrid architecture blurs the distinction between packets and circuits. By combining the strengths of each switching technology, a hybrid network can deliver higher performance at lower cost than either technology alone, even at the level of a ToR switch.

We evaluate our design for a 100/10-Gb/s OCS/EPS hybrid network using a scaled-down 10/1-Gb/s hardware prototype that supports eight end hosts. The prototype consists of two FPGA-based REACToRs with four downward-facing 10-Gb/s ports each. Both REACToRs connect to the Mordia [23] microsecond OCS and a commodity electrical packet switch. The circuit switch supports a line rate of 10 Gb/s while the packet switch is rate limited to 1 Gb/s to enforce a 10:1 speed ratio. End hosts connect to our prototype using commodity Intel 10-Gb/s Ethernet NICs that we synchronize using standard 802.1Qbb PFC signaling.

Link rate	Full fat tree	Helios-like	REACToR
10 Gb/s	$2-4$	$1-3$	N/A
100 Gb/s	4	3	$1^{\dagger}$

Table 1: Number of transceivers required *per upward-facing ToR port* for different network architectures. (†Presuming a 10-Gb/s packet network is already in place.)

Our experiments show that our REACToR prototype can provide packet-switch-like performance by delivering efficient link utilization while reacting to changes in traffic demand, and that its control plane is sufficiently fast that changes in circuit assignment and schedule can be made without disrupting higher-level transport protocols like TCP. Using simulation of more hosts, we also illustrate the large benefits that a small underprovisioned packet switch provides to a hybrid ToR relative to a pure circuit ToR. We conclude that REACToR can service published data-center demands with available technology, and can easily scale up to make effective use of next-generation optical switches and 100-Gb/s hosts by reusing an existing 10-Gb/s electrical packet-switched network fabric.

2 Background

We start by motivating the benefits of a hybrid-ToR network design, describing the work that REACToR builds upon, and then discussing our design assumptions.

2.1 Motivation

Consider a data-center operator that wants to upgrade an existing 10-Gb/s data center network—i.e., the part of the network that connects the top-of-rack switches together—to 100 Gb/s.

Table 1 shows the number of optical transceivers required for each upward-looking port of the ToR for three different network architectures. The first architecture is a fully provisioned three-level fat-tree network [1]. If all of the links in the backbone network are optical, then this network requires four transceivers per upward port. In a Helios-like [10] architecture an optical circuit switch is placed at the uppermost layer of the network, saving one transceiver per port as compared to the number used in a fat-tree network.

At 10 Gb/s, if the links between aggregation switches are short enough to be electrical, then transceivers may only be required between aggregation and core switches, potentially reducing the number of transceivers by up to three per port. At 100 Gb/s, however, while electrical interconnects may still be viable from an end host to a ToR (i.e., distances less than 5 meters), all connections from the ToR to the rest of the network are likely to be optical. Hence, either architecture will require a full compliment of optical transceivers. Moreover, in order to upgrade the network the operator will have to replace the existing 10-Gb/s transceivers with new 100 Gb/s transceivers.

The REACToR architecture, in contrast, re-uses the existing 10-Gb/s packet-based network and deploys a parallel 100-Gb/s circuit-switched optical network under a common control plane. As compared to the other two architectures listed in Table 1, REACToR requires only one 100-Gb/s transceiver per upward-facing port of the ToR because the OCS does not use transceivers. This means that for a fully provisioned three-level fat-tree network, if the per-port cost of the OCS used in REACToR is less than three times the cost of a 100-Gb/s optical *transceiver*, then a REACToR hybrid network will cost less than an equivalent 100-Gb/s packet-based network—even if the 100 Gb/s *switches* themselves were free. Larger networks require even more transceivers per end host: a five-level network requires eight transceivers to support each upward facing port, making the economics of REACToR even more compelling. Over-subscribed networks will use fewer transceivers in the core network, but the scaling trends are still applicable.

While this example uses a 10-Gb/s EPS and a 100-Gb/s OCS, the actual link rates for which a REACToR architecture will be cost competitive with a fully provisioned or over-subscribed packet-switched network depends on market trends. Many OCS architectures are based on MEMs devices and can easily support link rates in excess of 100 Tb/s[1] per port. For this kind of device, the cost per optically switched bit is decreasing and is fundamentally inversely proportional to link rate. While the costs per switched bit of optical transceivers and packet switches are also decreasing, the rate of decrease is much slower. These trends imply the cost per switched bit will eventually become comparable at some link rate. What is less clear is the precise link rate when this crossover point will occur and the economic viability of a data-center network that supported such a link rate.

2.2 Related work

Hybrid data-center network architecture design is an active research area. Helios [10] and c-Through [26] both rely on slower 3D-MEMs based optical circuit switching, restricting their use to either highly aggregated traffic (i.e., in the core of the network), or highly stable traffic (e.g., long file transfers). OSA [7] combines an OCS-based reconfigurable topology with multi-hop overlay networking. The bandwidths of links in OSA can be varied through the use of wavelength-selective switching. In addition to optical switching, reconfigurable wireless links have also been proposed in data-center contexts [12, 14, 28]. In contrast to these previous approaches, which employ switching technologies with relatively long reconfiguration times, REACToR relies upon the Mordia [8, 23] OCS, which can be reconfigured in 10s of microseconds, in order to service a much larger portion of the offered demand through the circuit-switched portion of the hybrid network fabric.

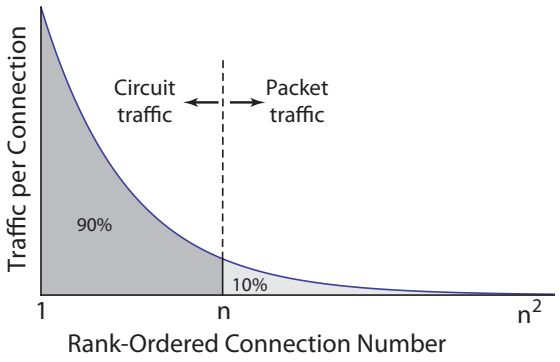


Figure 1: Rank-ordered traffic for each of the n^2 elements of a demand matrix, for which most of the traffic (e.g. 90%) is carried in a few ($O(n)$) flows.

The question of how much buffering should be deployed in a network has been considered under a wide variety of settings. In the Internet, a common rule of thumb has been that at least a delay-bandwidth product is necessary to support TCP effectively. Appenzeller *et al.* [3] challenged this assumption for core switches, and argue that for links carrying many TCP flows, less buffering is necessary. In the data center, Alizadeh *et al.* propose modifications to TCP that, along with appropriate switch support, can reduce the amount of buffering required down to a single packet per flow [2]. Other network technologies have also been created that reduce in-network buffering, including Myrinet [5] and ATM [20]. Numerous proposals for entirely bufferless "network-on-chip" (NoC) networks have been proposed [21], including hybrid NoC networks that also leverage packet switches [13].

2.3 Design assumptions

Studies of data-center traffic show that the traffic demand inside a data center is frequently concentrated, with a large fraction of the traffic at each switch port of a ToR

[1]The mirrors are typically reflective from approximately 1.3 μm to 1.6 μm which corresponds to a bandwidth of approximately 400 THz.

destined to a small number of output ports [15]. Such locality is not surprising, as application programmers and workload managers frequently use knowledge about the location of end hosts to coordinate workloads to minimize inter-rack traffic. Based on these empirical observations, a fundamental premise of all hybrid networks—including REACToR—is that a large fraction of the network traffic is carried by a small number of relatively long-lived flows. This observation can be expressed in terms of the n^2 rank-ordered elements of a demand matrix for a network that connects n ToRs. Figure 1 shows an example where 90% of the inter-ToR traffic is carried by only n flows.

In such settings, the demand matrix is frequently both sparse and stable [9, 12, 14]. This kind of traffic demand is generally suitable for a large port-count optical circuit switch, but these assumptions can be violated for specific workflows over any given time interval. Therefore, REACToR switches the few high-traffic flows using an OCS while forwarding the relatively small amount of traffic between most ToRs using an under-provisioned standard packet switch.

While rack-level coordination can lead to bursty traffic at the upward looking ports of a ToR, we carry this assumption one step further. Unlike previous hybrid designs that focus on the core of the network, REACToR critically depends upon *individual hosts* being able to fill circuits assigned to them with data, which in turn depends on hosts transmitting groups of packets to the same destination ToR at fine time scales.

To verify this assumption, in previous work we measured individual flows, at microsecond granularity, emanating from a single host under a variety of workloads [16]. We find that host mechanisms such as TCP segmentation offloading in the NIC and generalized segmentation offloading in the operating system, cause traffic to frequently leave the NIC in bursts of 10s to 100s of microseconds. In Section 5.1, we expand upon this analysis to show that circuit switching these flows can further enhance this behavior while not disturbing the transport protocol. For regimes in which circuit switching does not affect the transport performance of an end host, we say that its flows are "flying under the radar".

3 Design

A REACToR-enabled data center consists of N servers grouped into R racks, each consisting of n nodes. We assume that a preexisting 10-Gb/s packet-switched network is already deployed within the data center. Overlaid on top of this packet-switched network is an additional 100-Gb/s circuit-switched network. At each rack is a hybrid ToR called a REACToR, which is connected to the packet-switched network with $u_p \leq n$ uplinks and is connected to the circuit-switched network with a separate set of $u_c \leq n$ uplinks. This means that the packet-switched network supports $R \times u_p$ ports, and the circuit-switched network supports $R \times u_c$ ports. Each REACToR has n downward-facing 100-Gb/s ports to its n local servers. In this work, we consider the fully provisioned case where $u_p = u_c = n$; however, additional cost savings are possible when either or both of u_p and u_c are less than n. Our architecture is agnostic to the particular technology used to build the circuit-switched fabric, but, given technology trends, we presume it is optical.

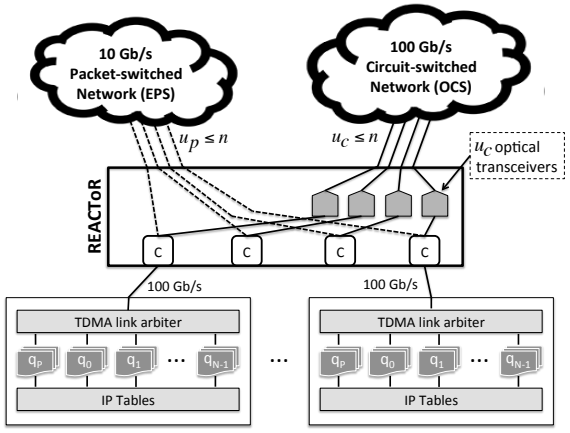


Figure 2: 100-Gb/s hosts connect to REACToRs, which are in turn dual-homed to a 10-Gb/s packet-switched network and a 100-Gb/s circuit-switched optical network.

Referring to Figure 2, an (n, u_p, u_c)-port REACToR consists of n downward-facing ports connected to servers at 100 Gb/s, $u_p = n$ uplinks connected to the packet-switched network at 10 Gb/s, and $u_c = n$ uplinks connected to the 100-Gb/s circuit-switched network. At each of the n server-facing input ports, there is a classifier (labeled 'C' in the figure) which directs incoming packets to one of three destinations: to packet uplinks, to circuit uplinks, or through an interconnect fabric to downward-facing ports to which the other rack-local servers are attached. There is no buffering on the path to the packet uplinks, as buffering is provided within the packet switches themselves. There is also no buffering on the path to the circuit uplinks; instead, packets are buffered in the end-host where they originate. When a circuit is established from the REACToR to a given destination, the REACToR explicitly pulls the appropriate packets from the attached end-host and forwards them to the destination.

REACToR relies upon a control protocol to interact with each of its n local end-hosts to: (1) direct the end host to start or stop draining traffic from its output queues (which we refer to as *unpausing* or *pausing* the queue, respectfully), (2) set per-queue rate limits, (3) provide circuit schedules to the end-host, and (4) retrieve demand

estimates for use in computing future circuit schedules. We first motivate the need for this functionality by describing the various other aspects of REACToR's design before detailing the host control protocol in Section 3.4.

3.1 End-host buffering

Each end-host buffers packets destined to the REACToR in its local memory, which is organized into traffic classes, one per destination ToR, with an additional class for packets specifically destined for the EPS (e.g., latency-sensitive requests). Each traffic class has its own dedicated output queue (i.e., $\{Q_0, Q_1, ..., Q_{N-1}\}$), with an additional queue for the EPS class, Q_P, as shown in Figure 2. At any moment in time, the REACToR can ask an end host to send packets from at most two classes: one forwarded at line rate to an OCS uplink (or local downlink port), and another forwarded to an EPS uplink. This latter class of traffic must be rate limited at the source NIC to conform to the link speed of the EPS to prevent overdriving the EPS. In the reverse direction, the EPS may emit packets into the REACToR at its full rate to a particular downward-facing port. Because that downward-facing port could potentially be shared by incoming line-rate circuit traffic heading to the same destination, REACToR must further ensure that the circuit traffic is sufficiently rate-limited so that there is enough excess capacity to multiplex both flows at the destination. Hence, end hosts will be directed by REACToR to similarly rate-limit traffic classes destined to the OCS at the source NIC, but at much higher rates. Further details on rate limiting are provided in Section 3.3.

Today, end-host NICs support modest amounts of buffering, on the order of a few megabytes. However, it is not organized in a way that can be directly used to support circuits. NICs partition their buffers into a small set of 8 to 64 transmit queues, which the OS uses to batch and store packets waiting to be sent. The scheduling policy for these queues is typically built into the NIC (e.g., round robin), so the actual transmit time of individual packets is outside the control of the OS.

To achieve high circuit utilization in REACToR, the NIC needs the ability to send data for a particular circuit destination to the ToR as soon as a circuit becomes established, and to fill that circuit continuously until it is torn down. At any one time, each circuit uplink within a REACToR is exclusive to a particular source port (attached end host), so efficiency degrades any time that source has no data to send. Thus, packets headed to the same circuit destination (i.e., remote host) should be grouped together within a host's memory, so that when a circuit to that destination becomes available, that group of packets can be sent from the NIC to the REACToR at line rate.

Within each host, we define a traffic class per destination host, and task the OS with classifying outgoing packets into the appropriate class based on, e.g., the destination IP address. REACToR then uses the host control protocol to pause and unpause end-host queues. In this model, the role of the OS and of the NIC changes somewhat: rather than the OS "pushing" packets to the NIC buffers based on queuing policies in the host, the NIC is responsible for "pulling" packets from the host memory into the NIC buffers according to the circuit schedule just in time to transmit them to the connected circuit. (We note that the NIC design advocated by Radhakrishnan et al. [24] would be especially well suited for this model.)

Demand estimation. Over a short time scale (i.e., 100s of μs, depending on the size of the NIC buffers), the occupancy of these traffic classes defines the imminent end-host demand because the packets in these queues have already been committed to the network by the OS. It is possible to query the OS, the application, or even a cluster-wide job scheduler to form longer-term demand estimates. For example, Wang et al. [26] use TCP send buffer sizes as estimates of future demand. Our prototype uses a demand oracle. In any case, the circuit scheduler uses these demand estimates to determine a set of future OCS circuit configurations.

3.2 Circuit scheduling

To make effective use of the capacity of the circuit switch, REACToR must determine an appropriate schedule of circuit switch configurations to service the estimated demand over an accumulation period W. This is the responsibility of a logically centralized, but potentially physically distributed, circuit scheduling service, which implements a hybrid circuit scheduling algorithm. This service collects estimates of network-wide demand, in the form of an $N \times N$ matrix D. The service computes a schedule, P_k, of m circuit switch configurations, which are permutation matrices[2], and corresponding durations, ϕ_k.

The number m of configurations that comprise the schedule is constrained because each circuit configuration requires a finite reconfiguration time δ, during which time no data can be forwarded over the circuit switch. When δ is large with respect to W, it is more efficient to use fewer configurations. When δ is small with respect to W, more configurations can be used. Including this reconfiguration delay, the duration of the schedule is constrained by the length of the accumulation period so that $\sum_{k=1}^{m} \phi_k + \delta m \leq W$. The goal of the scheduling algorithm is to maximize $\sum_{k=1}^{m} \phi_k$ subject to these constraints.

Obviously, if the switch introduces a reconfiguration delay, then it is impossible to service fully saturated demand at line rate. Existing research in constrained scheduling has focused on switches that run

[2] A permutation matrix is a matrix of 0s and 1s in which each row and column has and only has a single 1.

faster than the link rate, with the ratio of the switch rate to the link rate called the speedup factor. These algorithms [11, 18, 27] produce a variable-length schedule which is dependent on the actual reconfiguration delay.

Hybrid networks in general, and REACToR in particular, do not use a speedup factor. Instead, REACToR uses the lower-speed packet switch as a way to make up for the reconfiguration delay and any scheduling inefficiency. This "back channel" is a key distinction between REACToR and traditional blocking circuit scheduling because REACToR continues to service a subset of flows over the EPS when circuits are not available, thereby increasing support for latency sensitive workloads.

We leave the selection and evaluation of an circuit switch algorithm as future work; for now we compute the schedule offline using a variant of existing constrained switching algorithms based on a predetermined demand matrix D. Any schedule computed for use in REACToR, however, is subject to a number of constraints.

Class constraints. In order to ensure the offered load can be effectively serviced by the ToR, REACToR imposes a number of constraints on the set of queues that can be unpaused at any particular time. First, the dedicated EPS queue (Q_P) is always unpaused but rate-limited to at most 10 Gb/s, providing the host with the ability to send latency-sensitive traffic directly to the EPS at any point in time. Second, at most one additional queue can be unpaused at any one time for transmission at (near) link rate (i.e., 100 Gb/s). When such circuit-bound (or rack-local) traffic arrives at an input classifier in the REACToR, it is directly forwarded to the appropriate circuit uplink (or downward-facing port) without any intermediate buffering. The third constraint is that, if a queue is unpaused for link-rate transmission in the current scheduling period, then it should never be unpaused for transmission to the EPS. This constraint serves two purposes: it prevents the EPS from being burdened with high-bandwidth traffic better served by circuits, and it gives that traffic class additional time to accumulate demand so that the circuits are highly utilized.

Fourth, any traffic class which is not assigned to a circuit (or downward port) during a scheduling period is instead remapped to the EPS, meaning that any packets in that class's queue are routed to the EPS uplink. Finally, all of the queues corresponding to EPS-bound traffic (i.e., both the dedicated EPS queue and and any classes not scheduled for a circuit in this period) must be rate limited such that the sum of their limits is less than or equal to the EPS link rate (e.g., 10 Gb/s).

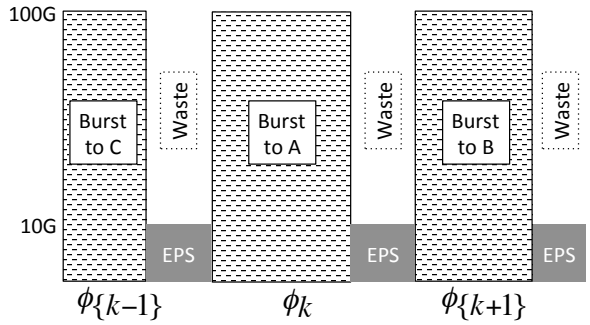
3.3 End-host rate limiting

At any given time, each of the REACToR's downward-facing server ports can transmit data from two sources: a circuit from a single source established through the OCS (or rack-local connection) fabric, and traffic from any number of sources forwarded through the EPS. At each downward-facing port there is a multiplexer which performs this mixing. When the sum of bandwidth from the EPS (B_{EPS}) and OCS (B_{OCS}) exceeds the rate of the REACToR port (B_{ToR}), then without intervention, $(B_{\text{EPS}} + B_{\text{OCS}}) - B_{\text{ToR}}$ traffic would be dropped. To prevent such drops, and to ensure high overall utilization, we rely on end-host rate limiting.

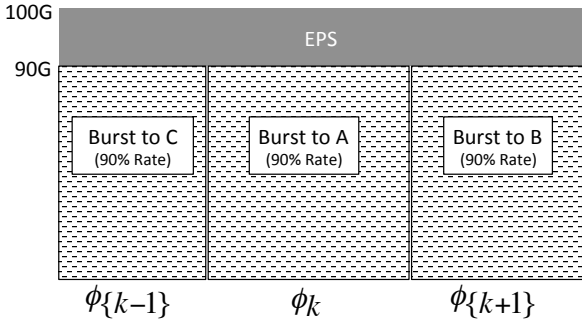
The first way that we use end-host rate limiting is to ensure that in steady state, $B_{\text{EPS}} + B_{\text{OCS}} \leq B_{\text{TOR}}$. Since the OCS is bufferless, the multiplexer gives priority to packets arriving from the OCS because otherwise they would have to be dropped. Assuming a REACToR with a 100-Gb/s OCS and 10-Gb/s EPS as an example, each end host would rate limit its circuit-bound traffic in the range of 90–100% of the link capacity to leave sufficient head room for the EPS traffic, based on the estimate of EPS demand in the current schedule. Each time that a set of configurations for a scheduling period is computed, a rate limit is also computed per configuration, reflecting the estimated load from the EPS. Note that this estimate need not be perfect, and in fact we expect the EPS to absorb inaccuracies in scheduling, demand estimation, and rate limiting. For each scheduling interval, the associated rate limits are computed and sent to each end-host via the host control protocol.

The circuit rate limit also serves a second purpose, which is providing statistical multiplexing at the downward-facing REACToR port. Underpinning the design of REACToR is the assumption that on short time scales, traffic emanating to a single destination is bursty. Each burst by definition consists of a number of packets sent back-to-back. From the point of view of the REACToR port multiplexer, this means that, absent other controls, during the first portion of a given circuit-switch configuration interval ϕ_k, the entire port's bandwidth would be dedicated to servicing a single burst of traffic from the OCS. Thus, any packets originating from the EPS would be delayed until the end of ϕ_k. Figure 3(a) shows a pictorial representation of this behavior. The challenge that arises is that the line rate of the EPS is presumed to be lower than the REACToR port speed and the OCS. Hence, the open region at the end of ϕ_k can only be filled with packets at the rate of the EPS (e.g., 10 Gb/s) instead of the OCS (100 Gb/s). Thus, for this example, the region at the end of ϕ_k only gets 10% utilized since the EPS can only drive 10% of the outgoing port bandwidth.

Instead, REACToR seeks to ensure that the circuit traffic is spread out across ϕ_k by limiting it to less than full line rate (e.g., 90 Gb/s of a 100-Gb/s link). Rate limiting over time allows the EPS-serviced traffic to be multiplexed on REACToR's downward-facing ports at a uni-



(a) When a circuit-switch configuration interval ϕ_k begins, queued traffic forms bursts which saturate the link during the first part of the configuration, leaving capacity for EPS traffic at the end of ϕ_k; since the EPS runs at a fraction of the line rate, it cannot efficiently use the remaining time.



(b) By rate limiting circuit traffic, the EPS can spread its traffic out over the entire configuration interval.

Figure 3: Rate limiting prevents bursts from the OCS from starving the EPS, which would otherwise be unable to make full use of each circuit-switch configuration interval ϕ_k. In both cases, the circuit-switched traffic achieves 90 Gb/s during each interval.

form rate across all configuration intervals ϕ_k, enabling the entire interval to be utilized by both circuit traffic and packet traffic. By setting circuit rate limits in the end host, as described above, the traffic headed to the circuit is paced to the appropriate rate. Figure 3(b) shows the resulting treatment of circuit and packet data within that same configuration interval ϕ_k.

3.4 REACToR host control protocol

An instance of the REACToR host control protocol runs between each end-host and its REACToR switch. REACToR uses the protocol to retrieve demand estimates collected by end-hosts, to set per-queue rate limits, as described above, and to convey impending schedules to the end host from the circuit scheduler. These functions are relatively straight forward. In this section, we examine the fourth use of the host control protocol: managing end-host traffic classes and buffering. The key to achieving efficient use of the hybrid network is being able to drain the appropriate classes with fine-grained precision at the right times. We now describe the host control protocol that achieves this precision.

Overview: To ensure reliable transmission, we cannot reconfigure the OCS until all incoming circuit traffic has ceased, since the OCS is unable to carry traffic during the time δ when it is being reconfigured. While classifiers on each REACToR input port can shunt all traffic to the EPS nearly instantaneously, in general we would like to ensure that almost all circuit-bound traffic has been paused before reconfiguring the OCS. Otherwise, a massive queue would build up at the EPS at the end of each schedule. To avoid this buildup, we leverage the 802.1Qbb Priority Flow Control (PFC) protocol to pause traffic at the end host. Each traffic class in the end host corresponds to a PFC class.[3] At the end of each schedule, for each attached host, the REACToR first sends a PFC frame to pause the traffic class destined for the current schedule's circuit (if any). Note that PFC frames are selective, so traffic destined to the EPS will continue to flow while the OCS is being reconfigured. Once inbound circuit traffic has ceased, the OCS can be reconfigured. After reconfiguration, the traffic class corresponding to the next schedule's circuit can be enabled by a PFC unpause frame.

Performance: The overall speed of the control plane is bounded by the speed at which REACToR can pause and unpause traffic classes buffered at the end hosts. Because the PFC frame must be both received and processed at the NIC before traffic stops, there will be some delay between when the controller wants to pause traffic and when the traffic finally stops arriving at the incoming ports at the REACToR. To quantify this delay, we extended the classifiers on our prototype to timestamp all incoming packets and mirror these timestamped packets to a collection host. We then measured the time from when the classifier sends a PFC frame to a host until it stops receiving packets from that host.

We measured the minimum (maximum) delay on an Intel 82599-based 10 Gb/s NIC as 1,014 (2,188) ns, with the actual delay varying as a function of PFC offset, meaning that if the PFC frame arrives more than 185.6 ns after the start of the current frame, the NIC will generate an additional frame before pausing, likely due to pipelining within the NIC implementation.

Once the OCS has established a circuit and is ready to receive traffic, the REACToR needs to restart traffic for the newly connected destination by sending another PFC frame. The measured 'on' delay (i.e., from when the configuration is started by the transmission of a PFC frame

[3] Although the current PFC specification is limited to eight frame priority levels, it is possible to reuse classes across schedule periods by recoloring.

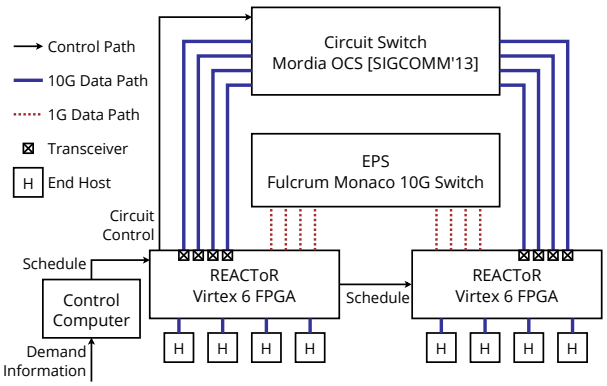


Figure 4: Our prototype REACToR network.

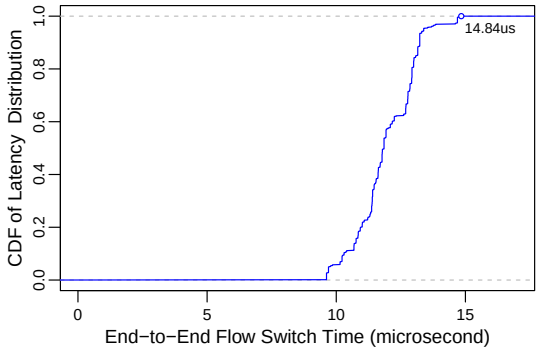


Figure 5: Observed end-to-end circuit switch reconfiguration delay δ.

unpausing the traffic) ranges between 1.2 μs and 1.3 μs. From the 'off' delay measurement, it is clear that we can hide the first microsecond of delay by sending the PFC frame before we actually want the traffic to stop, but it may take an additional 1.3 μs for all ports to cease sending. There is one additional source of delay: a port may be busy sending an outgoing packet at the moment the classifier wishes to send the PFC frame. This delay is bounded by the 1500-byte MTU in our prototype, leading to a worst-case combined delay of approximately 2.5 μs, which is the lower bound of the speed of the control plane achievable in REACToR with 10-Gb/s end hosts, a 1500-byte MTU size, and the 802.1Qbb implementation on our NIC.

4 Implementation

To evaluate our design, we have implemented two prototype four-port 10-Gb/s REACToRs (shown in Figure 4) using two FGPAs, a Fulcrum Monaco 10-Gb/s electrical packet switch, and the Mordia microsecond OCS [23]. Mordia is 24-port reconfigurable OCS built from six 4-port "binary MEMs" wavelength-selective switches, with a reconfiguration delay of $\delta = 12\mu$s, which includes the physical switching time of the MEMs devices and the time to reinitialize the attached 10-Gb/s transceivers. Thus, our REACToR prototype supports the same 10:1 bandwidth ratio described earlier, but at 10 Gb/s (OCS) and 1 Gb/s (EPS) rather than 100/10 Gb/s.

Each REACToR is implemented with a HiTech Global HTG-V6HXT-100GIG-565 FPGA development board, which supports 24 ports of 10-Gb/s I/O. The circuit scheduling service runs as a user-level process on a dedicated Linux-based control server, and transmits schedules to the FPGA via a dedicated 10-Gb/s Ethernet connection. In our implementation, the end hosts are servers equipped with Intel 82599-based NICs. The end hosts classify traffic according to the destination using the Linux `tc` facility. The classifier on the FPGA selectively enables or disables packets to a given destination using the IEEE 802.1Qbb priority-based flow control standard, which supports eight flow classes. We use seven of these classes to correspond to the n circuit destinations reachable from a REACToR, and the eighth is reserved for the EPS-dedicated class.

At each switch reconfiguration, the controller on the FPGA updates the OCS and enables the corresponding end-host traffic classes using 802.1Qbb PFC frames. The controller also configures the classifiers so that they forward the appropriate line-rate flow to the circuit uplink, and forward the remaining traffic to the EPS.

Circuit switch characterization: The average reconfiguration delay for the Mordia switch is approximately 12 μs, with a maximum observed delay of 14.84 μs (as shown in Figure 5). The transceivers we use vary in their "lock" time, necessitating setting a more conservative reconfiguration delay. This variance is an engineering artifact of our hardware and is not fundamental; the IEEE 802.3av (10G-EPON) specification, for instance, calls for a 400-ns lock time. Except as noted, in the experiments that follow, we configure REACToR to assume a 30-μs reconfiguration time which, contained within at least a 160-μs configuration period, delivers at least 81% link efficiency.

REACToR host control protocol: To tightly time synchronize the attached hosts, REACToR sends the schedule to each attached host using two UDP packets. The first packet contains the impending schedule for the upcoming 3-millisecond period, whereas the second packet indicates the start of the three-millisecond time period, serving as a precise periodic heartbeat. End hosts receive these packets in a kernel module via the `netpoll` kernel APIs, which reduces the delay in acting on them to less than 15 μs.

5 Evaluation

In this section, we evaluate the performance of our REACToR prototype implementation. We first show that, with buffering and scheduling packets at end-host NICs, circuit-switching does not negatively impact TCP throughput. Second, we show that the REACToR can dynamically update and switch schedules of many flows without impacting throughput. Third, we show that REACToR can serve a time-varying workload that consists of multiple high- and low-bandwidth flows, promoting flows as appropriate from the packet-switched fabric to the circuit-switched fabric. Finally, we use simulation to illustrate the large benefits that a small underprovisioned packet switch provides to a hybrid ToR.

To generate arbitrary traffic patterns, we implemented a Linux kernel module based on `pktget` [22] that can send MTU-sized UDP packets at arbitrary rates up to line rate. When the module is sending, it runs on a dedicated core and each packet it sends has a sequence number. At the same time, the module also serves as a traffic sink that receives UDP traffic via the `netpoll` kernel interface, and records the sequence number and source address of packets. For packet timing measurements, we configured the FPGA to generate a record for each packet that captures the source, destination, and a timestamp with 6.4-nanosecond precision. The prototype sends these records out-of-band to a collection host using one of the 10 Gb/s ports of the FPGA, which we then process offline.

5.1 TCP under TDMA scheduling

In Section 2.3, we described how application flows exhibit intrinsic short-term correlated bursts as a consequence of the NIC trying to efficiently use the link. We therefore consider how flows behave in a hybrid fabric where a circuit scheduler pauses flows at the host while they wait for an assigned circuit and unpauses them when the circuit is established. While its flow is paused, an application may generate additional packets, increasing the size of its burst when its flow is eventually unpaused and thereby more efficiently use its circuit. However, the increased latency and latency variation induced by pausing and unpausing flows may detrimentally impact the transport protocol (e.g., TCP) or the application itself.

To study the impact of circuit scheduling on TCP throughput, we generate stride workloads where a single host sends to another host, and at the same time sinks a TCP flow from a third host. First we consider the case where we pause and unpause a bi-directional circuit, i.e., pause both data and TCP ACKs at the same time. Next we consider the case where we pause the data in the flow, but allow ACKs to return unimpeded (e.g., via the EPS). Finally, we consider the case where we pause the ACKs, but enable data packets to transmit unimpeded.

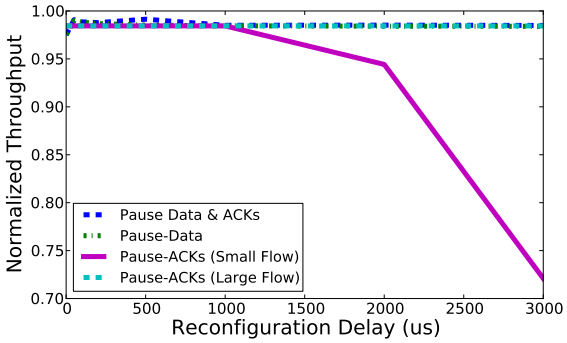


Figure 6: Effect of pausing/unpausing data/ACK packets on TCP throughput.

Figure 6 shows the resulting normalized throughput when varying the reconfiguration delay δ for a stride workload with eight hosts. In the first case, the normalized throughput of uni-directional and bi-directional circuits is close to ideal, showing that pausing data packets on the end hosts does not affect throughput for pause lengths considered by REACToR. When pausing only the ACKs, we find that there are two regimes to consider. During slow start ('Small Flow'), pausing ACKs decreases the overall throughput of the flow—up to 30% for 3-ms delays. For shorter delays (e.g., $\leq$1 ms) there is no detectable effect for pausing ACKs. Once the flow leaves slow start ('Large Flow'), there is no effect on throughput regardless of the reconfiguration delay.

These experiments consider the effect of circuit scheduling on TCP traffic in the absence of packet loss. In practice, packets may be lost for a variety of reasons. We repeated the experiments above where each end host drops packets uniformly at random with a configurable drop probability. While TCP throughput suffers as expected with increasing drop rates, the difference in performance with and without circuit scheduling (e.g., with and without issuing PFC pause frames) is insignificant for steady state loss rates up to 1%.

5.2 Switching "under the radar"

Next we evaluate the speed and flexibility with which REACToR can be reconfigured. We first run an all-to-all workload on eight hosts, where every host streams a TCP flow to each of the other seven hosts using all available bandwidth. To serve this workload, we load REACToR with a schedule of seven TDMA periods that fairly shares the links among all the flows. Each schedule period is 1.5 ms, within which each host sends and receives from each other host for 214.3 μs (including a 30-μs circuit reconfiguration delay) in each circuit configuration. We schedule all data packets via the circuit switch, and all TCP ACKs via the packet switch. We could use the same schedule for every period, but to further exercise

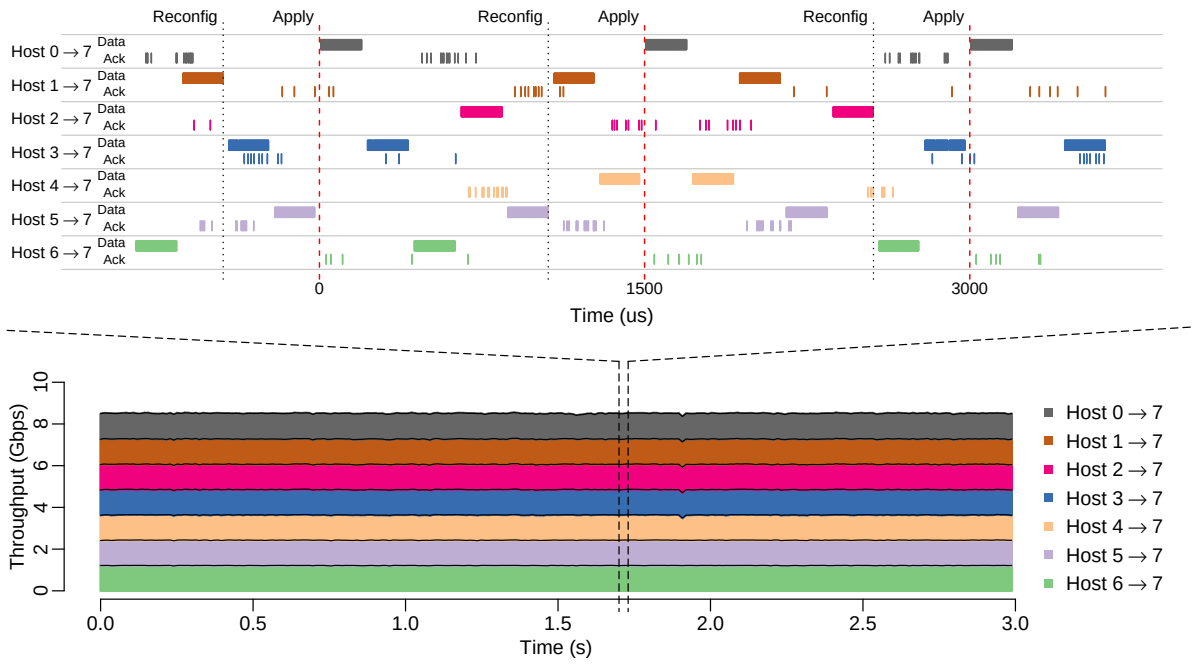


Figure 7: All-to-all workload with circuit configurations changing every scheduling period.

our prototype we change the schedule so that hosts receive circuits in different permutations in each period.

Figure 7 shows three seconds of an all-to-all workload where flows start at the same time on the hosts. The bottom part shows the achieved throughput as reported by one of the hosts: the flows from the other seven hosts evenly split the available bandwidth. Total TCP goodput received is 8.1 Gb/s, the maximum given the 86% duty cycle resulting from the 30-μs reconfiguration delay in a 214.3-μs circuit.

At the application level, the achieved TCP goodput maximizes network capacity and is stable over time. However, if we zoom in and look at the packet traces, as shown in the top part of the figure, we can see the fine-grained behavior of scheduling the flows on circuits. A control packet triggers a new schedule each period, which the controller sends to the REACToR during the previous period (at the time marked 'Reconfig') and the switch loads just before the new period starts ('Apply'). The schedule partitions each period into seven circuit configurations, one for each of the seven hosts sending to the host we are observing.

At time offset zero, for instance, host 0 has the first configuration in the schedule. Its data packets arrive over the circuit it receives, and no other host can send data packets through the circuit switch to host 7. The second configuration schedules host 3, and so on. ACKs received at host 7 use the packet switch, and hence can overlap circuits scheduled for other hosts. (The flow assignments are asymmetric; when host 0 is sending to host 7 at time zero, host 7 is sending to host 6 and receiving ACKs from it.)

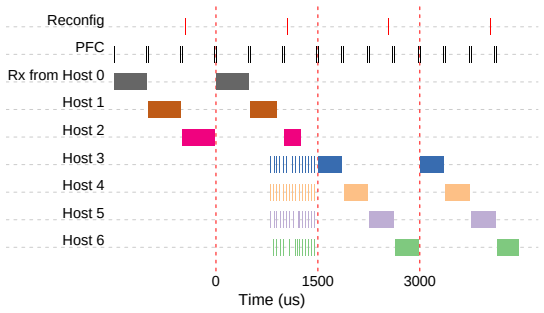


Figure 8: Changing the number and duration of configurations in scheduling periods.

This all-to-all workload does not vary demand over time. Given the frequency with which we can reconfigure the circuit switch, we can also serve time-varying workloads by serving different workload demands under different scheduling times with different numbers of configurations and circuit assignments.

We use another experiment to demonstrate this flexibility. We divide the eight hosts into two groups: G_A consists of hosts 0–3, and G_B hosts 4–7. We then generate traffic among the hosts using two workloads. The first is a group-internal all-to-all, where each host streams TCP packets to the other three hosts in its group at the maximum possible rate. To serve this workload, REACToR uses a schedule that has three configurations in a schedul-

ing period. The period lasts 1,500 μs, and each configuration lasts 500 μs (including a 30-μs reconfiguration delay). The second is a cross-talking all-to-all workload where each host in G_A streams to all the other four hosts in G_B, and vice versa. For this workload, REACToR uses schedules with four configurations. These scheduling periods also last 1,500 μs, but each configuration lasts 375 μs (again including a 30-μs reconfiguration delay).

In the experiment, we change from the group-internal to the cross-talk workloads midway through, loading the REACToR with correspondingly different schedules. Figure 8 shows the incoming packets to host 7 around the workload transition time. We controlled the experiment so that the workload changes at an inconvenient, but more realistic, time for REACToR: *during* a scheduling period, at time 750 μs on the graph. REACToR's schedules commit the switch based on predicted demand, and workloads are apt to change their demand independent of when REACToR can conveniently accommodate them. At this workload transition, REACToR is halfway through its scheduling period and packets already queued at the first three hosts continue to arrive via circuits. Overlapping these flows, the other four hosts start sending packets to host 7. These hosts do not have circuits, so the packets arrive via the EPS at a much lower rate.

At the end of its committed scheduling period (time 1,500 μs), REACToR can then react to the workload transition and schedule circuit configurations that match the workload. At this time, host 7 changes from receiving packets in 500-μs configurations, scheduled round-robin from hosts 0–2, to receiving packets in 375-μs configurations from hosts 3–6.

In summary, these experiments demonstrate the speed and flexibility with which REACToR can reconfigure its circuit schedules given a known demand. Applications achieve their expected goodput at a high level, while individual flows are paused and released at fine time scales when their circuits are scheduled. Further, REACToR can adjust the circuit schedule to adapt to changes in application behavior and demand.

5.3 Time-varying workloads

Next we show that REACToR can dynamically serve rapidly changing traffic demands and efficiently move flows from the EPS to the OCS.

In this experiment, we vary the number of high bandwidth and low bandwidth flows among hosts at small timescales. The workload pattern is again all-to-all among eight hosts, which we observe from the perspective of one of the hosts and its seven incoming flows. Initially one of the flows is a high-bandwidth flow sending at full demand and served on the circuit switch, and the other flows are lower bandwidth flows (each paced at 96 Mb/s) served on the packet switch. At time t_1, one of

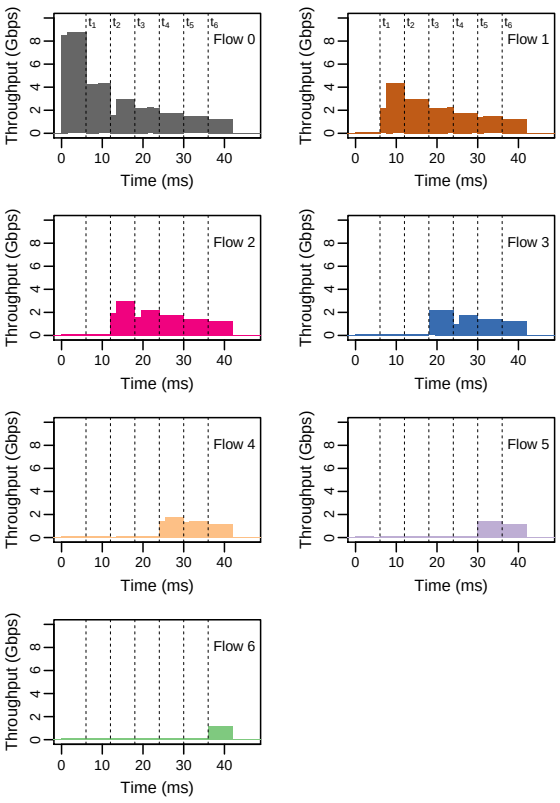


Figure 9: Goodput achieved for a time-varying workload of three flows to a single end host.

the low bandwidth flows changes to a second high bandwidth flow — representing a dynamic shift in application behavior — and needs to be served on the circuit switch. At each subsequent time step, another lower bandwidth flow becomes high bandwidth and transitions from the EPS to the OCS.

Figure 9 shows the throughput achieved by each of the flows. Initially, the high bandwidth flow has exclusive use of the circuit switch. At each time step t_i, another flow transitions from low to high bandwidth and REACToR promotes it from the EPS to the OCS. Each time, all high bandwidth flows then adjust to fairly share the link bandwidth to the receiving host. In each case, REACToR seamlessly handles the shift in traffic demands.

Note that a flow might send at a lower rate during the first 1.5 ms scheduling period. This happens when the schedule changes and a flow is served earlier in this period than the previous one. As a result, the queue buffer does not yet have enough enqueued packets to fully utilize the link. The queue buffer will be built up starting from the second scheduling period, and the flow will fully utilize the link again.

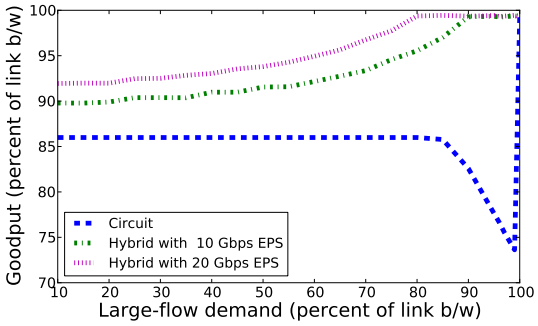


Figure 10: Performance of a circuit switch ToR and REACToR in different workload regimes.

5.4 Large benefits from a small EPS

As a final step, we illustrate how an underprovisioned packet switch in REACToR substantially relaxes the constraints of a pure circuit switch. In particular, we show how the ability to offload small flows to the packet switch enables REACToR (1) to maintain high circuit utilization and high workload goodput under our workload assumptions, and (2) to support full simultaneous endpoint connectivity for small flows.

We use simulation for these experiments to evaluate behavior beyond the constraints of our testbed. The simulator models a single REACToR switch, including the behavior of end hosts with NIC buffers, a circuit switch with switching overhead, a packet switch with buffers, and the circuit scheduler from Section 3.2. We validated the simulator output using our prototype: for workloads involving eight or fewer hosts, flow goodput calculated by the simulator always had errors less than 1% of flow goodput measured on the prototype implementation.

Maximizing circuit utilization Using the simulator, we explore the performance regimes of a single hybrid ToR-like REACToR at rack scale. For comparison, we simulate 64 end hosts connected first to a pure circuit switch and then to a REACToR switch via 100-Gb/s links. We compare with a circuit switch, not because we expect it to perform ideally well, but because it helps illustrate how a REACToR switch performs. In this experiment, each host j sends traffic to its neighboring twenty-one hosts $j+1$ through $j+21$, one flow per host. The total offered demand across all twenty-one flows is 100 Gb/s. The flow from j to $j+1$ is a "large" flow whose demand D we vary up to the full 100 Gb/s. The other twenty "small" flows have equal demands dividing the remaining (100-D)-Gb/s bandwidth equally. For scheduling circuits, each configuration has a duration of at least 40 μs (including reconfiguration delay), the scheduling period is 3000 μs (at most 75 configurations), and the reconfiguration delay is 20 μs (hence each configuration has at least 50% utilization). For REACToR, we simulate a 100-Gb/s circuit switch and a packet switch internally, where the packet switch is 10 Gb/s or 20 Gb/s.

Figure 10 shows the results for this experiment. The x-axis shows the demand of the large flow from each host as a percentage of link rate (100 Gb/s), and the y-axis shows the goodput of the ToR given the offered workload. We show three curves, one for a pure circuit switch and two for REACToR, with the curves overlapping at points. We note, of course, that a fully-provisioned packet switch as the ToR could switch this workload at full rate.

The lowest curve shows the results of using a pure circuit switch for the ToR, with the right-most point of the curve as the ideal case for a circuit switch. Hosts send all of their traffic in the large flow at 100 Gb/s (small flows have zero demand). In this case all of the flows can take full advantage of a circuit when the switch schedules one for them: each flow has data to transmit during their entire allocation in the circuit schedule. Once the small flows start to have a non-zero demand, though, there is a cliff in circuit switch performance. The demands to the other hosts, although small, are all non-zero; as a result, the switch schedules each small flow a circuit to carry its traffic. But the small flows do not have the traffic demand to fully utilize their circuit allocations, leaving them under-utilized. As the larger flow decreases in demand moving to the left, and the smaller flows correspondingly increase, the circuit switch performance improves as the small flows are better able to utilize their allocations. Once the small flows are able to fully use their circuits (when the large flow demand is at 87%), the pure circuit switch performance levels off. At this point, the lower goodput of the circuit switch is entirely due to reconfiguration delay overhead.

In comparison, the middle curve shows the performance of a hybrid ToR-like REACToR with a 100:10 capacity ratio. Between 90–100 Gb/s for the large flow ($<$ 10 Gb/s combined for the small flows), REACToR performs just like a packet switch because the combined demands of the small flows go through REACToR's packet switch while the large flows go through REACToR's circuit switch. This regime represents REACToR's ability to efficiently switch traffic that does not have good burst behavior. As long as the combination of those flows fits within the EPS "budget", REACToR has the performance of a 100-Gb/s packet switch using a combination of a 100-Gb/s circuit switch and a 10-Gb/s packet switch.

Below 90 Gb/s, REACToR performance gradually and gracefully degrades as the combined demands of the small flows exceed the 10-Gb/s per-host rate of REACToR's packet switch; notably, it avoids any discontinuities in performance. REACToR then needs to schedule

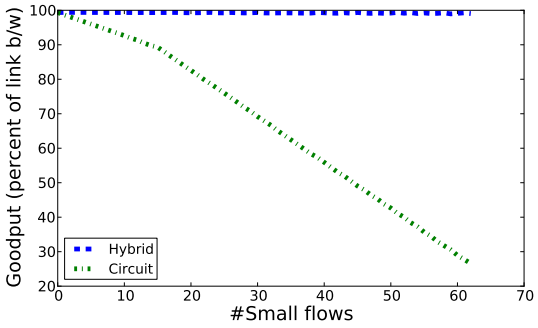


Figure 11: Performance of a circuit switch ToR and REACToR as a function of the number of small flows.

an increasingly larger portion of small flow demand on the circuit switch. REACToR goodput will decrease as a combination of imperfect utilization of circuits when assigned to small-flow demand, and additional reconfiguration delays for those circuit assignments.

Note that for this curve the circuit and packet switches had a 100:10 capacity ratio. There is nothing fundamental about this choice. A network using REACToR switches could tailor this ratio to balance cost and workloads: networks with more shorter-burst flows can deploy more EPS resources at higher cost, or vice versa. In terms of Figure 10, more EPS resources shift the point of 100% goodput for REACToR to the left, as shown by the top-most curve corresponding to an internal 20-Gb/s packet switch in REACToR.

Endpoint connectivity. In addition to maintaining high circuit utilization, the underprivisioned packet switch also enables REACToR to support many simultaneous flows between endpoints in the tail of the workload distribution. To illustrate this point, we perform one last experiment focusing on the number of simultaneous small flows between distinct endpoints in the network. In a network of 64 hosts, we represent the aspect of the workload well matched to circuits using one single large flow consuming 90% of the capacity: an ideal case for a pure circuit switched network. We then evenly split the remaining 10% among n small flows, where n varies between 1 and 64.

Figure 11 shows the goodput of the ToR (percentage of offered demand serviced by the ToR) as a function of the number of small flows for this experiment. At $n = 1$, both the hybrid and pure circuit ToRs perform the same on the trivial single large flow. The bottom curve shows the pure circuit ToR goodput in the presence of small flows. Goodput steadily decreases because the circuit switch has to assign circuits to every flow. As the number of small flows increases, the demand in each flow decreases; circuit durations decrease, but the rate of reconfigurations correspondingly increases. Hence the pure circuit ToR becomes increasingly less efficient.

The top curve shows the hybrid ToR performance. By construction, its internal packet switch can satisfy the bandwidth demands of the small flows and therefore efficiently handle the full endpoint connectivity of the workload. If the total demand of the small flows comprising the tail of the workload exceeds the capacity of the underprovisioned packet switch, then the performance of the hybrid ToR will trend towards the left-hand regime in Figure 10 (e.g., where the large flow demand drops below 90% with a 10G EPS).

6 Conclusion

Hybrid ToRs, such as REACToR, have the potential to enable scalable, high-speed networks by pairing the numerous advantages of optical circuit switching with comparatively underprovisioned packet switching. The key insight driving our work is that by moving the vast majority—but not all—of the buffering out of the switch and into end hosts, more scalable interconnect fabrics can be supported.

Practically speaking, this only works if 1) end hosts emit bursts of traffic to a given destination that are both predictable and of sufficient duration to fill OCS circuits, and 2) the hybrid scheduler operates at timescales that are invisible to the transport and applications running on the end hosts. In the first case, in-NIC buffering that historically has been used to drive line rate transmissions can be repurposed to stage impending data bursts, therefore fully using OCS circuits. In the second case, for a two-REACToR prototype, we have shown that we can schedule end hosts to make use of an OCS without negatively impacting TCP performance. A design challenge posed by interconnecting a large number of REACToRs is co-scheduling and synchronizing directly connected REACToRs to avoid the need for buffering on uplink ports. We leave this global scheduling problem for future work.

Acknowledgments

This research was supported in part by the NSF through grants EEC-0812072, MRI-0923523, and CNS-1314921. Additional funding was provided by a Google Focused Research Award and a gift from Cisco Systems. The authors thank Mindspeed for technical support and providing the custom switch board. The manuscript benefited from feedback and discussions with D. Andersen, J. Ford, D. Maltz, A. Vahdat, our shepherd, Changhoon Kim, and the anonymous NSDI reviewers.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity, data center network architecture. In *Proc. ACM SIGCOMM*, Aug. 2008.

[2] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center TCP (DCTCP). In *Proc. ACM SIGCOMM*, Aug. 2010.

[3] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *Proc. ACM SIGCOMM*, Oct. 2004.

[4] T. Benson, A. Akella, and D. A. Maltz. Network traffic characteristics of data centers in the wild. In *Proc. ACM IMC*, 2010.

[5] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. Su. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, 15(1):29–36, Feb. 1995.

[6] N. Calabretta, R. Centelles, S. Di Lucente, and H. Dorren. On the Performance of a Large-Scale Optical Packet Switch Under Realistic Data Center Traffic. *Journal of Optical Communications and Networking*, 5(6):565–573, June 2013.

[7] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, and X. Wen. OSA: An Optical Switching Architecture for Data Center Networks and Unprecedented Flexibility. In *Proc. USENIX NSDI*, Apr. 2012.

[8] N. Farrington, A. Forencich, G. Porter, P.-C. Sun, J. Ford, Y. Fainman, G. Papen, and A. Vahdat. A Multiport Microsecond Optical Circuit Switch for Data Center Networking. *IEEE Photonics Technology Letters*, 25(16):1589–1592, June 2013.

[9] N. Farrington, G. Porter, Y. Fainman, G. Papen, and A. Vahdat. Hunting mice with microsecond circuit switches. In *Proc. ACM HotNets*, Oct. 2012.

[10] N. Farrington, G. Porter, S. Radhakrishnan, H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat. Helios: A hybrid electrical/optical switch architecture for modular data centers. In *Proc. ACM SIGCOMM*, Aug. 2010.

[11] S. Fu, B. Wu, X. Jiang, A. Pattavina, L. Zhang, and S. Xu. Cost and delay tradeoff in three-stage switch architecture for data center networks. In *Proc. IEEE High Performance Switching and Routing*, July 2013.

[12] D. Halperin, S. Kandula, J. Padhye, P. Bahl, and D. Wetherall. Augmenting Data Center Networks with Multi-gigabit Wireless Links. In *Proc. ACM SIGCOMM*, Aug. 2011.

[13] N. Jerger, M. Lipasti, and L. Peh. Circuit-Switched Coherence. *Computer Architecture Letters*, 6(1):5–8, July 2007.

[14] S. Kandula, J. Padhye, and P. Bahl. Flyways To De-Congest Data Center Networks. In *Proc. ACM HotNets*, Oct. 2009.

[15] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The Nature of Data Center Traffic: Measurements & Analysis. In *Proc. ACM IMC*, Nov. 2009.

[16] R. Kapoor, A. C. Snoeren, G. M. Voelker, and G. Porter. Bullet Trains: A Study of NIC Burst Behavior at Microsecond Timescales. In *Proc. ACM CoNEXT*, Dec. 2013.

[17] B. Lee, A. Rylyakov, W. Green, S. Assefa, C. Baks, R. Rimolo-Donadio, D. Kuchta, M. Khater, T. Barwicz, C. Reinholm, E. Kiewra, S. Shank, C. Schow, and Y. Vlasov. Four- and Eight-Port Photonic Switches Monolithically Integrated with Digital CMOS Logic and Driver Circuits. In *Proc. OFC/NFOEC*, Mar. 2013.

[18] X. Li and M. Hamdi. On scheduling optical packet switches with reconfiguration delay. *IEEE Journal on Selected Areas in Communications*, 21(7), Sept. 2003.

[19] D. Marom. Switching Capacity of MEMS Tilting Micromirrors. In *Proc. IEEE Optical MEMS and Nanophotonics*, Aug. 2012.

[20] J. Martin, K. K. Chapman, and J. Leben. *Asynchronous Transfer Mode: ATM Architecture and Implementation*. Prentice-Hall, Inc., 1997.

[21] T. Moscibroda and O. Mutlu. A Case for Bufferless Routing in On-Chip Networks. In *Proc. ISCA*, June 2009.

[22] R. Olsson. pktgen the linux packet generator. *Proc. Linux Symposium*, July 2005.

[23] G. Porter, R. Strong, N. Farrington, A. Forencich, P.-C. Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat. Integrating microsecond circuit switching into the data center. In *Proc. ACM SIGCOMM*, Aug. 2013.

[24] S. Radhakrishnan, Y. Geng, V. Jeyakumar, A. Kabbani, G. Porter, and A. Vahdat. SENIC: A scalable NIC for end-host rate limiting. In *Proc. USENIX NSDI*, Apr. 2014.

[25] A. Vahdat, M. Al-Fares, N. Farrington, R. N. Mysore, G. Porter, and S. Radhakrishnan. Scale-out networking in the data center. *IEEE Micro*, 30(4):29–41, Aug. 2010.

[26] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. S. E. Ng, M. Kozuch, and M. Ryan. c-Through: Part-time Optics in Data Centers. In *Proc. ACM SIGCOMM*, Aug. 2010.

[27] B. Wu, K. L. Yeung, and X. Wang. Improving scheduling efficiency for high-speed routers with optical switch fabrics. In *Proc. IEEE Globecom*, Dec. 2006.

[28] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng. Mirror Mirror on the Ceiling: Flexible Wireless Links for Data Centers. In *Proc. ACM SIGCOMM*, Aug. 2012.