JOURNAL OF
Optical Communications
and Networking

Performance trade-offs in reconfigurable networks for HPC

Min Yee Teh,[1,*] Zhenguo Wu,[1] Madeleine Glick,[1] Sebastien Rumley,[2] Manya Ghobadi,[3] and Keren Bergman[1]

[1]Columbia University, New York, New York 10027, USA
[2]HES-SO—University of Applied Sciences and Arts Western Switzerland, Fribourg, Switzerland
[3]Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA
*Corresponding author: mt3126@columbia.edu

Received 24 December 2021; revised 10 April 2022; accepted 12 April 2022; published 11 May 2022

Designing efficient interconnects to support high-bandwidth and low-latency communication is critical toward realizing high performance computing (HPC) and data center (DC) systems in the exascale era. At extreme computing scales, providing the requisite bandwidth through overprovisioning becomes impractical. These challenges have motivated studies exploring reconfigurable network architectures that can adapt to traffic patterns at runtime using optical circuit switching. Despite the plethora of proposed architectures, surprisingly little is known about the relative performances and trade-offs among different reconfigurable network designs. We aim to bridge this gap by tackling two key issues in reconfigurable network design. First, we study how cost, power consumption, network performance, and scalability vary based on optical circuit switch (OCS) placement in the physical topology. Specifically, we consider two classes of reconfigurable architectures: one that places OCSs between top-of-rack (ToR) switches—ToR-reconfigurable networks (TRNs)—and one that places OCSs between pods of racks—pod-reconfigurable networks (PRNs). Second, we tackle the effects of reconfiguration frequency on network performance. Our results, based on network simulations driven by real HPC and DC workloads, show that while TRNs are optimized for low fan-out communication patterns, they are less suited for carrying high fan-out workloads. PRNs exhibit better overall trade-off, capable of performing comparably to a fully nonblocking fat tree for low fan-out workloads, and significantly outperform TRNs for high fan-out communication patterns.

https://doi.org/10.1364/JOCN.451760

1. INTRODUCTION

Over the past decade, massive increases in computation power among the top high performance computing (HPC) and data center (DC) systems have not been accompanied by equivalent increases in the inter-node communication bandwidths [1,2]. The resulting steady decline of the bytes-per-FLOP ratio will adversely affect the performance of next-generation communication-intensive workloads. Artificial intelligence (AI) and distributed machine learning (ML) workloads, which are becoming more dominant, will be especially impacted. These trends will drive the demand for low latency, high throughput interconnection networks that can efficiently scale to support the applications' performances. The practice of over-provisioning networks to provide the requisite capacity will be increasingly impractical as network costs become a growing fraction of overall system cost [3,4].

To address these challenges, researchers have made large strides in improving networks with better routing schemes [5–9], task placement [10], and topology designs [11–13]. The integration of optical circuit switching in the context of high performance systems has been one such effort, and is an intense area of recent research [14–16]. In addition to bringing significant benefits to bandwidth density and power, optical circuit switching enables topological reconfiguration, which is a key enabler for traffic-aware topology designs at runtime. These dynamic network architectures represent a paradigm shift from the predominantly static topologies deployed in today's high performance systems.

While past literature has proposed a variety of architectures that integrate optical circuit switching to improve system performance, none has provided, to the best of our knowledge, a comprehensive comparison across the different architectures. In this work, we tackle two main challenges in the reconfigurable network design space. First, we study how the placement of optical circuit switches (OCSs) in the physical network affects network performance, cost/power consumption, and scalability. We consider two reconfigurable network classes that are representative of a broad number of prior proposals: one places OCSs between top-of-rack (ToR) switches, which we call ToR-reconfigurable networks (TRNs),

and the other places OCSs between pods of switches, called pod-reconfigurable networks (PRNs). Second, we study the effects of various reconfiguration periods and latency on network performance, using realistic simulations driven by HPC and DC applications.

To this end, our primary contributions in this paper are noted below.

- **Effects of OCS placement on performance.** The choice of OCS placement has tremendous implications on system performance. We found that a TRN, which directly interconnects racks, is about 2× more cost and power efficient than the considered PRN variant. Most importantly, we found that OCS placement also affects the types of workload for which a reconfigurable network is optimized. By comparing a PRN and a TRN at equal cost, we found the TRN to perform better when communication patterns have small fan-outs. For such traffic patterns, a TRN is capable of supporting up to 55% more input load than a PRN before network saturation. For large fan-out communication patterns, our results show that a PRN can support 20% more input load on average than an equal-cost TRN.
- **Effects of the reconfiguration period on network performance.** We found that shorter reconfiguration periods generally lead to better performance. Interestingly, we found that reliance on the reconfiguration period is also dependent on the OCS placement. Specifically, for our simulated workloads, an increase in reconfiguration period from 1 μs to 1 s degrades throughput by ∼2.1× in a PRN; a similar increase in reconfiguration period results in a throughput drop ≥100× in a TRN. This is because the static topology between racks of the same pod in a PRN, or lack thereof in a TRN, provides a constant interconnect that greatly relaxes its reliance on frequent reconfigurations.
- **Demand-aware reconfigurations with longer periods *or* demand-oblivious reconfigurations with shorter periods?** Recognizing the challenges of sequencing demand-based reconfigurations at small time scales, recent proposals such as Sirius [14] instead rely on demand-oblivious round-robin reconfigurations to deliver equal bandwidth between all endpoints. We find a demand-aware reconfiguration with longer periods to be the better strategy for HPC communications, which generally show strong spatial locality and comprised small messages. Our simulations using real workloads show that a demand-aware TRN (TRN-DA) with a 100 ms reconfiguration period delivers 0.99–4.98× the throughput of a demand-oblivious TRN with a 1 μs round-robin reconfiguration period.

2. BACKGROUND AND RELATED WORK

High throughput and low latency have always been the core goal of any communication network. How that goal manifests in network topology, however, has varied over time with new system considerations and available technologies.

A. Static Networks

Fat trees and other regular topologies. Modern HPC systems have reached a scale of parallelism for which the topology connecting compute nodes is critical. As messages in a supercomputing cluster are often latency sensitive, as opposed to heavyweight Transmission Control Protocol (TCP) socket connections, HPC network designs have traditionally prioritized lossless and deadlock-free routing. As a result, their networks are built using regularly structured topologies such as hypercube, torus, and Dragonfly (DF) [12,17]. In commodity DC clusters where bandwidth-intensive cloud and big data applications dominate, fat trees (FTs) have been the *de facto* deployment standard, as they can deliver full throughput with small-radix commodity packet switches [18–21].

Expanders. Kassing *et al.* [22] showed that expander (EXP) networks [23–25], when paired with multi-path routing [5], deliver performance comparable to fully non-blocking FTs. EXP networks exhibit many desirable properties: (1) their flat (non-hierarchical) structure and high bisection bandwidth supports high throughput at low cost [23,24], and (2) they support fine-grained expansion, which is a useful property for DC operators who generally expand their systems over time [26,27].

B. Dynamic Reconfigurable Networks

Prior studies have shown that DC and HPC traffic patterns are highly skewed, where the majority of traffic is exchanged between a small subset of source–destination pairs. For instance, a study on Microsoft's data center network (DCN) reveals that 0.4% of rack pairs generate about 80% of total traffic [28,29], while Facebook (FB) similarly reported a highly skewed communication pattern in its frontend clusters [30]. Under such an imbalanced traffic load, static networks that are designed with uniformity in mind are suboptimal. This has motivated many researchers to explore reconfigurable network architectures based on OCSs.

Optical circuit switching technology. Devices that support OCS functionalities come in many forms, such as software-controlled optical patch panels [31], electrical circuit switches [32], 2D and 3D micro-electromechanical systems (MEMS) [33,34], silicon photonic switches [35–37], wireless transceivers based on free space optics [15,38] or 60 GHz waves [29,39], RotorSwitch [40,41], and tunable lasers [14]. Of these technologies, only optical patch panels and 3D MEMS OCSs have been commercialized.

Early reconfigurable network proposals, such as Hybrid Flexibly Assignable Switch Topology (HFAST) [42] and Helios [43], were primarily based on 3D MEMS technology that had high switching latency in the tens of milliseconds range, a problem which most off-the-shelf OCSs still face [33]. Subsequent works have aimed at minimizing switching delay to support more agile reconfigurations. These architectures demonstrate switching latency at μs [15,35,38,40,41] or even ns or sub-ns [14,44] levels, though these systems are based on non-commercialized prototypes.

Regardless of the technology of the underlying OCS, the means by which it performs switching can be largely treated as a black box from the perspective of the network controller. We refer to the physical wiring between electrical packet switches (EPSs) and OCSs as the *physical topology*. Reconfiguring OCSs stablishes a new set of circuit connections between the input

and output ports at the physical layer, effectively realizing a specific *logical topology* overlay on the physical topology.

SDN-enabled layer-2/3 reconfiguration. Actuating a topology reconfiguration requires applying a sequence of atomic updates on the link and network layers, all of which can be coordinated using software-defined networking (SDN). The reason reconfigurations must be applied sequentially in incremental steps—as opposed to updating everything in one fell swoop—is to ensure routing consistency as the network carries live traffic [45]. It is important to maintain routing consistency in a network with dynamic topology to prevent unnecessary packet drops or routing "blackholes" due to route changes as a result of circuit switching. To maintain routing consistency, packets must be routed to their destination based on either the pre- or post-configured topology. This means that a complete reconfiguration to the next logical topology must be split into the following atomic steps: (1) draining the ports about to be reconfigured to avoid packet blackholes, (2) switching the OCS, (3) locking and synchronizing the transceivers to establish communication mode, and (4) undraining the ports. Given how SDN has become commonplace in today's commodity clusters for traffic engineering and monitoring [21,46], building a SDN-controlled reconfigurable network with off-the-shelf MEMS OCSs can be very practical.

Reconfiguration latency. Many past proposals have measured an end-to-end topology reconfiguration. A notable example is in [47], where the author broke down the end-to-end reconfiguration delay based on an experimental test bed built with commodity OpenFlow EPSs is 204.3 ms. Here, the transceiver locking and synchronization process itself took ∼204 ms; this suggests that lowering end-to-end reconfiguration delay may require a joint-optimization of both the switching hardware and the higher level protocols. Several proposals in the past decade have demonstrated lower reconfiguration delays at tens of μs [15,40] or tens of ns [14], though all these systems of OCSs are still research prototypes and not commercially available. More recent research has demonstrated sub-ns reconfiguration delays and clock synchronization to realize optical packet switched (OPS) architectures [44,48], though evaluating these architectures is future work.

Reconfiguration strategy. There are two main topology-reconfiguration strategies that have been explored in past works: (1) demand-aware and (2) demand-oblivious (i.e., round-robin) reconfigurations. The majority of the proposed reconfigurable networks in prior works have been demand-aware, which optimize logical topology periodically in response to traffic demands of the underlying workload. These network designs, however, face substantial deployment challenges due to the need for network-wide demand estimation and a centralized network controller that optimizes logical topology based on the estimated demands. All of these result in a highly complex control plane. In recent years, several networks architectures based on a demand-oblivious reconfiguration, such as RotorNet [40] and Sirius [14], have also been proposed. These systems are compelling due to their simplicity, as the OCS *rotates* across a fixed set of circuit patterns in a round-robin schedule, such that all source–destination pairs receive a uniform bandwidth. This strategy, in theory, removes a significant source of complexity associated with collecting a network-wide traffic matrix to optimize topology, allowing for topology reconfiguration on a much smaller time scale. That said, to make rotation-based reconfigurable network performance competitive would require custom-designed host congestion control, which adds some deployment complexity that is sometimes overlooked in an otherwise simple scheduler-less reconfigurable network. [In rotation-based reconfigurable networks, a valiant load balancing (VLB)-like indirect routing strategy is used to improve load balancing under skewed workloads. To ensure that packets are routed correctly for the current OCS configuration, each host maintains a separate first-in-first-out (FIFO) queue for every other destination node. As such, the number of queues each host must maintain scales up very quickly as the number of servers increases.] We compare the performance of both demand-aware and demand-oblivious reconfigurable networks at different reconfiguration periods in Section 6.

In summary, although the literature on reconfigurable networks has grown considerably over the past two decades [14–16,32,35,38,41–44,49–55], there is very little understanding of the trade-offs of different reconfigurable network architectures. This is due to (1) prior proposals having all been point designs and (2) the lack of commercial adoption, with no recorded system having deployed a reconfigurable network. In this work, we study the fundamental design problems in reconfigurable networks at a system level.

3. RECONFIGURABLE NETWORKS

We formally define two abstract models based on OCS placement: PRN and TRN. Figure 1 shows their physical topologies. Figure 2 formalizes the PRN and TRN models by (1) describing their physical and logical topology and (2) discussing the relevant optimization algorithms most suited to each model.

A. ToR-Reconfigurable Network

In an effort to build more energy-efficient networking solutions, recent proposals in reconfigurable networks tend to "flatten" the hierarchical Clos designs (two-level spine-leaf or three-level spine-aggregation-leaf architectures) that have become the *de facto* standard HPC and DC topology. These efforts have led to developments in TRNs that consider topology reconfiguration on a rack-to-rack basis directly from ToR switches.

1. Sparse Topology Optimization

For most reasonably sized systems, there are generally significantly more ToR switches in the entire network than there are uplinks available for network connection per ToR. This is because commercially available leaf switches have fairly small port counts, typically between 16 and 32. As a result of the small number of uplinks per ToR, k', with respect to the number of reconfigurable ToR switch nodes, $|V|$, (i.e., $|V| > k'$), the graph is sparse. A graph is *sparse* if the number of nodes, n, is greater than the node degree, k; otherwise, the graph is *dense*.

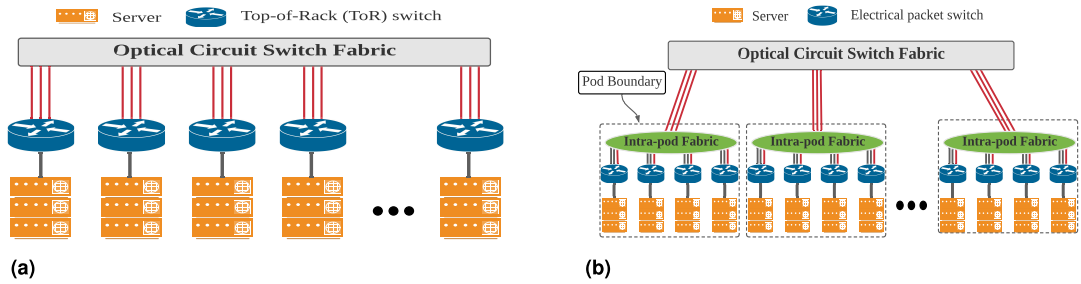


Fig. 1. Abstract models (a) for the TRN model and (b) for the PRN model.

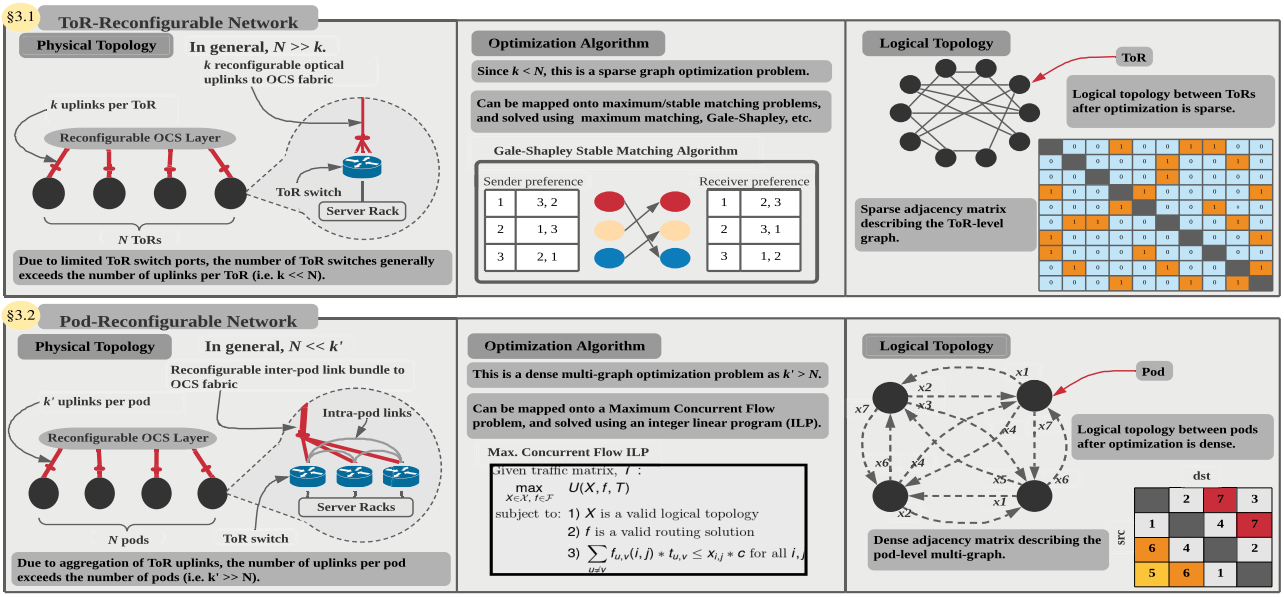


Fig. 2. Overview of pod- and ToR-reconfigurable network topologies. We summarize the optimization problems and the employed algorithms for both reconfigurable network models. k and k' denote the number of uplinks per ToR switch and per pod, respectively. N denotes the number of nodes in the topology optimization problem.

A full-mesh network that links every node pair in an all-to-all topology is a specific instance of a dense graph.

When optimizing topology in sparse graphs, as shown in Fig. 2, one approach is using maximum weighted bipartite matching problems, which can be solved in polynomial time using algorithms such as Edmonds–Karp [56]. The main issue with algorithms such as Edmonds–Karp, however, is that they are centralized algorithms that scale poorly with network size (e.g., Edmonds–Karp has an $O(n^4)$ runtime, where n is the number of nodes in the matching graph). An alternative approach is to cast the problem as a stable matching problem that can be solved in a distributed fashion using algorithms such as Gale–Shapley [15] with a runtime of $O(n^2)$, where n is the number of nodes.

B. Pod-Reconfigurable Network

In contrast to a TRN, which places the OCS layer directly between ToR switches, a PRN places the OCS layer between pod units. A pod consists of several racks that are *aggregated* into a logical unit of deployment and are interconnected by a static switch fabric using either electrical wires or optical fibers.

In large-scale systems beyond thousands of racks, aggregating multiple racks into pods helps improve the modularity, manageability, and cabling complexity of PRNs compared to TRNs [57]. In enterprise DC networks with FT topologies, pods typically interconnect several racks that are in close proximity to a two-level leaf-spine sub-network [26]. The pod unit also arises naturally in topologies that contain notions of groups, such as DF [12,13]. In this work, we consider PRNs with two variants of intra-pod topology: (1) two-layer Clos (PRN-2L) and (2) a full-mesh similar to a DF group (PRN-M).

1. Dense Multi-Graph Optimization

Unlike a TRN model where topology optimization is done at the ToR-switch level, the optimization is done at the pod level in a PRN model. This changes the nature of the graph optimization from that of a TRN in two main ways: (1) aggregating racks into a pod means that there are far fewer pods than the number of racks (e.g., aggregating two racks into a pod means the number of pods is a factor of two fewer than the number of racks), and (2) the total number of reconfigurable uplinks per pod generally exceeds the total number of pods.

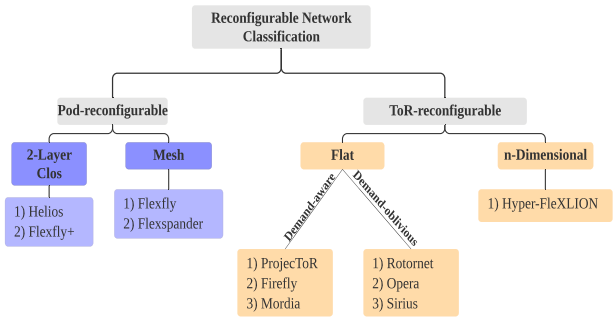


Fig. 3. Classification of reconfigurable networks.

Thus, aggregating racks into pod units makes the PRN optimization problem smaller in size than an equal-sized TRN. In addition, the optimization problem in a PRN also changes from a sparse graph matching problem in the case of a TRN, to a dense multi-graph optimization problem. As outlined in Fig. 2, optimizing PRNs generally involves solving a dense multi-graph optimization problem based on variants of the multi-commodity flow problem. Of course, it is still possible to render the PRN optimization graph sparse by over-deploying pods to the point where the number of pods exceeds the number of uplinks per pod, though we leave such designs for future analysis.

Classifying reconfigurable networks. The proposed PRN and TRN abstractions allow us to effectively classify many prior reconfigurable network proposals, as shown in Fig. 3. PRNs reconfigure topology between pods; pods comprise multiple EPSs that are interconnected statically via an intra-pod fabric. Conversely, TRNs alter the topology between racks directly by reconfiguring the ToR switch uplinks.

C. Flow Control

Congestion control protocols such as TCP and InfiniBand (IB) that are running on today's high performance clusters were designed assuming the network topology is static. When network topology becomes dynamic, the host transport protocol may need to be redesigned. At present, the literature focusing on transport protocol design for reconfigurable networks remains underdeveloped. Past work has relied on standard TCP/IB protocols [16,35,49], modified TCP stacks [58], or novel transport protocols specific to the underlying network architecture [14,40].

To ensure that our comparison of reconfigurable and static networks is fair, we use the same flow control mechanism for all simulated networks to manage congestion. The flow control needs to generalize well across a broad class of networks (both static and dynamic), and also works well under different reconfiguration periods. Given this, we consider a simplified flow control as follows. First, we assume a *lossless* network by implementing link-layer credit flow control. Hosts rely on back-pressure for congestion control by sending packets only when downstream EPSs have sufficient buffer space. Second, we assume each host has sufficient buffer to support the reassembling of out-of-order packets. This design choice is necessary for our purposes for two reasons: (1) it allows us to implement per-packet load balancing to improve bandwidth utilization on every evaluated topology, and (2) it allows the transport layer to work well not only under different static topologies, but also under dynamic topologies reconfigured at different time scales.

4. SCALABILITY ANALYSIS

We now analyze the network scalability of PRNs and TRNs in terms of (1) the maximum number of server hosts for a given set of EPS and OCS radices and (2) the processing overhead of demand-aware reconfigurations.

A. Network Scalability

We first look at the scalability of dynamic and static networks in terms of number of hosts (i.e., servers) as a function of EPS radix. For this analysis, we utilize an important concept from graph theory known as the Moore bound [59,60].

Moore bound. The Moore bound is the upper bound on the number of switches (and thus the number of server hosts) a k-regular network with a target diameter of d can support. (All switches have k identical network-facing ports.) The expression for the Moore bound, $M_{k,d}$, is

$$M_{k,d} = 1 + k\frac{(k-1)^d - 1}{k-2}, \qquad \text{for } k > 2. \tag{1}$$

For flat networks (i.e., networks that directly interconnect ToRs), Eq. (1) can be used to estimate the maximum network size attainable given fixed-radix EPSs as basic building blocks. Table 1 shows the maximum network size as a function of EPS radix, k, for every studied topology. For the PRN and TRN models, we consider two variants for each class. The two TRN variants we consider are a flat TRN (TRN-Flat), which is representative of architectures such as ProjecToR [15], RotorNet [40], and Sirius [14], and a TRN that partitions racks into three dimensions (TRN-3D), which is representative of architectures such as Hyper-FleX-LION [16]. The two PRN variants considered are a PRN with two-level Clos intra-pod fabric (PRN-2L), such as Helios [43], and a PRN with mesh intra-pod fabric (PRN-M), such as Flexfly [35] and Flexspander [49]. All considered PRN designs preserve a dense pod-level topology (i.e., the number of pods $\leq$ number of uplinks per pod); thus, the pod-level network graph has a diameter of one. For static networks, we consider FTs with three- and four-level FTs (FT3 and FT4, respectively), and canonical DFs with one inter-group port per switch (i.e., $h = 1$ according to the notation in Kim *et al.* [12]). Figure 4 shows the network size of different topologies as a function of EPS radix, and Table 1 contains the expressions of the various topology parameters as a function of EPS radix. Given an EPS radix, k, we allocate $\frac{k}{2}$ ports for connection to hosts. To ensure a fair comparison between different topologies, the diameter of each evaluated topology is fixed according to Table 1.

PRN-M, TRN-3D, and DF show weaker scalability compared to the other topologies. For DF, this is due to its full-mesh intra-group fabric; a DF group with m switches requires $m - 1$ EPS links to be allocated for intra-pod connections. This, as a result, leaves fewer ports available for global connection, thereby limiting the maximum number of groups

Table 1. Expression of the Largest-Attainable Scale (in Terms of the Number of Servers) That Different Classes of Topologies Can Support Given an EPS Radix of k[a]

	PRN		TRN					
Quantity	**2L**	**Mesh**	**Flat**	**3D**	**FTn**	**DF**	**DF+**	**EXP**
OCS radix	$(\frac{k}{2})^2+1$	$\frac{k(k+4)}{16}+1$	$(\frac{k}{2})^2(\frac{k}{2}+1)$	$\frac{k}{6}+1$	0	0	0	0
Total EPS	$\frac{k^3}{4}+k$	$(\frac{k}{4}+1)^3-\frac{k^2}{16}-\frac{k}{4}$	$(\frac{k}{2})^2(\frac{k}{2}+1)$	$(\frac{k}{6}+1)^3$	$(2n-1)(\frac{k}{2})^{n-1}$	$(\frac{k}{2})^2+\frac{k}{2}$	$\frac{k^3}{4}+k$	$(\frac{k}{2})^3+(\frac{k}{2})^2+\frac{k}{2}+1$
Num. endpoints	$(\frac{k}{2})^4+(\frac{k}{2})^2$	$\frac{k}{2}[(\frac{k}{4}+1)^3-\frac{k^2}{16}-\frac{k}{4}]$	$(\frac{k}{2})^3(\frac{k}{2}+1)$	$\frac{k}{2}(\frac{k}{6}+1)^3$	$2(\frac{k}{2})^n$	$(\frac{k}{2})^3+(\frac{k}{2})^2$	$(\frac{k}{2})^4+(\frac{k}{2})^2$	$(\frac{k}{2})^4+(\frac{k}{2})^3+(\frac{k}{2})^2+\frac{k}{2}$
Diameter	3	3	3	3	$2(n-1)$	3	3	3
Examples	Helios [43], spatially switched modular DCN [57]	Flexfly [35], Flexspander [49]	Sirius [14], RotorNet [40], ProjecToR [15]	Hyper-FleX-LION [16]	VL2 [19]	Dragonfly [12], Cray Aries [61]	Dfly+ [13], Megafly [62]	Xpander [24], Jellyfish [23], Slimfly [25]

[a]The concentration (i.e., server nodes per switch) is set to be $\frac{k}{2}$ to prevent a network bottleneck due to server oversubscription. FTn denotes an n-level fat tree.

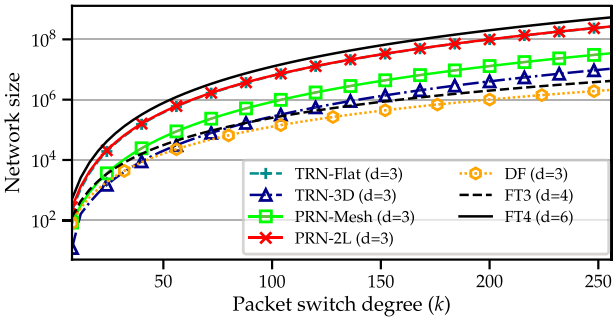


Fig. 4. Maximum network size (number of servers) as a function of EPS radix for various network topologies.

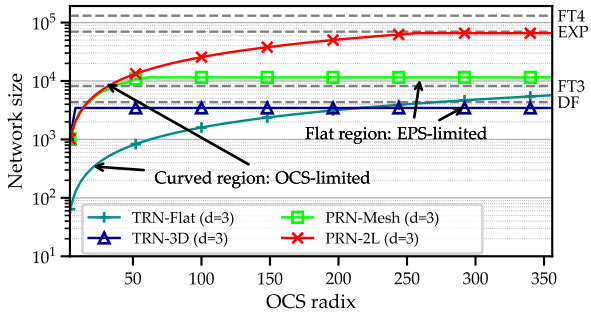


Fig. 5. Maximum network size (number of servers) as a function of OCS radix for various networks built using radix 32 EPSs. Note that static networks do not require OCSs, so we show their sizes as horizontal dotted lines for comparison. The DF+ line overlaps with the EXP and is thus not shown.

that can be built. Compared to PRN-M, PRN-2L's two-tiered pod fabric gives it much higher scalability. In fact, PRN-2L exhibits higher scalability than all other topologies in Table 1, with the exception of FT4. Note that since DF+ also possesses a two-level spine-leaf Clos intra-pod topology identical to that of PRN-2L, both topologies exhibit similar scalability as a function of EPS port count; this is not shown in Fig. 1 to avoid clogging up the figure.

B. Minimum OCS Radix Requirements

The OCS port count is a key factor when determining the maximum attainable scale of a reconfigurable network. Figure 5 shows the maximum number of hosts a network topology can support as a function of the OCS radix, given an EPS radix of 32. As we assume all network topologies are built using EPSs' identical port counts, changing the EPS radix does not affect the relative scalability of different network topologies. The maximum network size of static networks (e.g., FT, DF, and EXP) are shown in Fig. 5 as straight lines as reference. Note that each curve shows two regions: a growing region that indicates the network size is OCS-radix limited, and a flat region that indicates the network size is EPS-radix limited.

Figure 5 shows the TRN-Flat curve flattening out at the slowest rate, which indicates that it requires a large OCS port count to scale. This is because the minimum required OCS radix is proportional to the number of racks. Considering that medium and large networks could contain $\Theta(1000)$'s of racks, some past TRN-Flat designs such as Firefly [38] and ProjecToR [15] have opted for wireless transceivers on ToR switches over OCS-based solutions to achieve better scalability. This is because the wireless transceivers essentially act as an OCS with an infinite radix. Partitioning the racks in a TRN into multiple dimensions similar to Hyper-FleX-LION [16] and HyperX [11] can lower the minimum OCS radix requirement, but also causes scalability to be more EPS-radix limited. Overall, PRN has a lower requirement on the OCS radix, as the minimum OCS radix scales with the number of pods, rather than the number of racks in the case of a TRN. This suggest that PRNs are suitable for large-scale deployments with many racks and/or when the OCS radix is low.

Figure 6 shows the minimum requisite OCS radix as a function of the EPS radix on the left, and the minimum OCS radix per unit rack as a function of the EPS radix on the right. We similarly find TRN-Flat exhibiting the worst scaling property out of all the compared network topologies in terms of minimum required OCS radix. By partitioning the racks into three dimensions in TRN-3D, the minimum required OCS radix is lowered substantially compared to TRN-Flat. Although the TRN-3D outperforms both PRN-2L and PRN-Mesh

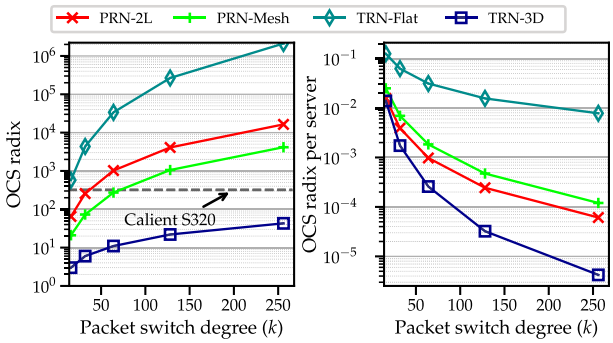


Fig. 6. The left figure shows the minimum required OCS radix as a function of the EPS radix. The horizontal dotted line marks the port count of a commercial Calient S320 with 320 ports. The right figure shows the minimum required OCS radix per server as a function of the EPS radix.

given the same EPS radix, its maximum network size (in number of servers) is actually smaller than both PRN variants, as TRN-3D's network size is more EPS-radix limited.

C. Demand-Aware Processing Overhead

We estimate the demand-based optimization overhead in terms of average compute time for a PRN and a TRN as the network scales, as shown in Fig. 7. These experiments are implemented in Python 2.7, and run on a desktop with 2.6 GHz dual Intel i7 cores with 16 GB of RAM. For the PRN, we use Gurobi [63] to solve a relaxed linear programming (LP) that maximizes throughput [64]. For the TRN, we implement two variants of matching algorithms: (1) a centralized max-weight matching and (2) a distributed Gale–Shapley matching based on [15]. The compute time is averaged over 50 runs per network size; each run takes a randomly generated traffic matrix as input.

In general, the processing overhead is a function of the problem size (i.e., the number of pods for a PRN and the number of ToR switches for a TRN). To make this analysis concrete, we express the average compute time in terms of network size, measured in terms of server numbers, assuming an EPS radix of 32. Each ToR switch connects to 16 servers, and each pod in a PRN contains 16 racks. We see that computing TRN matchings with a centralized max-weight matching scales poorly, with compute time quickly exceeding 1 s for systems beyond 1600 servers (100 racks). Although the PRN's relaxed-LP algorithm is also centralized, PRNs can benefit from solving a substantially smaller optimization problem that scales with the number of pods, as opposed to the TRNs' matching problems that scale with the number of racks. However, as the network scales beyond 32k servers, the PRN's processing overhead exceeds that of TRN with distributed Gale–Shapley implementation. For larger-scale systems (25k servers and above) where the processing overhead is of the order of 100 ms, a "set-and-forget" paradigm, in which the topology reconfiguration is triggered by the arrival of new jobs, may be the preferred approach.

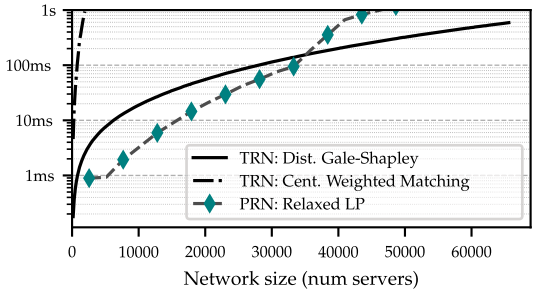


Fig. 7. Demand-aware processing overhead as a function of network size.

We believe that these overheads can be improved in production by employing a variety of relaxation techniques, such as partitioning the network into smaller logical slices [65], exploiting parallelization/GPU-aided acceleration, or implementing the core sub-routines in "close-to-metal" programming languages such as C. We leave these explorations for the future.

5. COST AND POWER CONSUMPTION ANALYSIS

In this section, we compare the cost and power consumption of four different network topologies: (1) FT, (2) EXP, (3) PRN, and (4) TRN at small, medium, and large system scales. The small, medium, and large networks support ∼8k, ∼50k, and 1M endpoints, respectively. (A million server nodes is the expected scale of exascale computing.) We assume PRNs have a two-layer Clos leaf-spine intra-group topology similar to that of a DF+ group, while TRNs use a flat ToR-level physical topology (i.e., TRN-Flat). Further, we assume that all networks are built using 100 Gb/s per-port EPSs with a homogeneous switch radix of 32. This means that the relative cost and power consumption of the studied network topologies are insensitive to a different choice of EPS radix. The minimum requisite OCS radices to support small-, medium-, and large-scale systems are shown in Table 2. (Note that for all reconfigurable networks, the minimum requisite OCS radix decreases when the EPS radix increases. For example, when the EPS radix is 256, the requisite OCS radix for a TRN-Flat to support a large-scale system with 1 million servers is 7813. This is because a larger EPS radix means more servers can be aggregated under each ToR switch, which helps reduce the number of pods and ToR switches in the network.)

A. Power Consumption

We estimate the network power consumption by summing up the expected power consumption from the following components: (1) EPS, (2) OCS, and (3) optical transceivers. Based on vendor-published data [66], each 32-port EPS switch has a

Table 2. Minimum Requisite OCS Radix for Each Network Built Using 32-port EPSs at Three Different Scales[a]

Scale	PRN-2L	PRN-M	TRN-Flat	TRN-3D
Small	32	32	500	8
Medium	391	391	6250	19
Large	3907	3907	62,500	40

[a]Small-, medium-, and large-scale systems carry ∼8k, ∼50k, and ∼1 million servers, respectively.

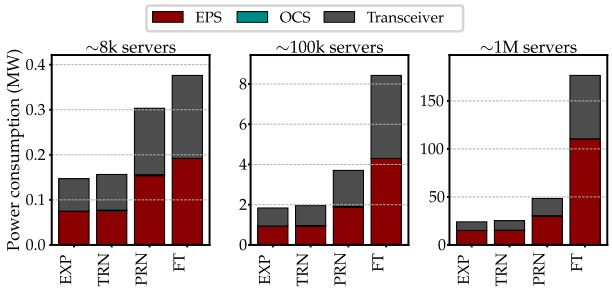


Fig. 8. Network power consumption breakdown at three different scales: small scale with ∼8k hosts, medium scale with ∼50k hosts, and large scale with ∼1M hosts. Note that OCSs add negligible power consumption to the network, and thus their power breakdown is barely visible.

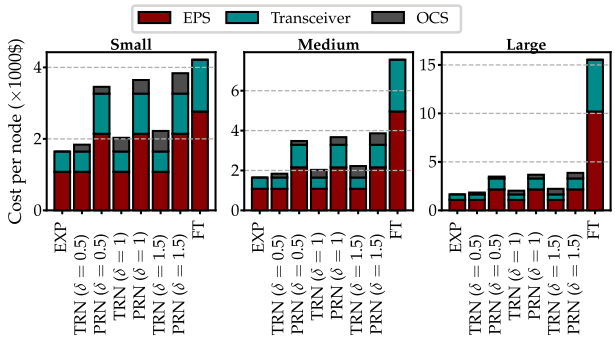


Fig. 9. Network cost breakdown of network topologies at three different scales: small scale with ∼8k hosts, medium scale with ∼50k hosts, and large scale with ∼1M hosts.

power consumption of 136 W. The OCS power consumption is assumed to be 90 W for a 320-port MEMS OCS, based on the power consumption of a single 320-port Calient MEMS OCS [33]. Without packet-decoding and buffering, OCSs consume significantly less power than EPSs on a per-port basis. For optical transceivers, we assume each component has 2.2 W power consumption for 100 Gb/s per-port bandwidth [67,68].

Figure 8 shows the estimated total network power consumption at different scales. As expected, FTs are the most power-consuming designs at all scales, as their hierarchical topology require many transceivers and EPSs to build. Conversely, the lack of a hierarchical structure allows TRNs and EXPs to scale with fewer switches and transceivers. This makes them more power efficient than a FT. This also explains why PRN-M is less power consuming than PRN-2L at small and medium scales. At large network scale, however, the gap in power consumption between PRN-M and PRN-2L shrinks. This is because PRN-2L can be built using 32-port EPSs, while flatter networks such as PRN-M, TRNs, and EXPs require higher radix switches that are individually more power consuming.

B. Cost Model

We now describe the cost model used to estimate the cost of different network topologies at small (8k hosts), medium (50k hosts), and large (1M hosts) scales. Our cost model breaks down the network cost based on three components: (1) EPSs, (2) optical transceivers, and (3) OCSs, similar to past works [12,22,25,27,69]. For EPS and transceiver costs, we use the reported prices from vendors for the Mellanox SN2700 Open Ethernet 100 GbE switch [66] and the Enhanced Data Rate (EDR) 100 Gb/s QSFP28 optical transceiver [67], which have reported per-unit prices of $17,285 and $555, respectively. In contrast to EPSs and optical transceivers, the commercial price of OCSs is considerably more obscure. This is because OCSs are at present an emerging technology manufactured in low volume. To estimate the cost of OCSs, we introduce a multiplicative factor δ on top of the average cost of an electrical port. Past works [22] have utilized $\delta = 1.5$, but this is overly pessimistic, as OCS pricing is based on low volume OCS prototypes. We believe that commercialization will help drive the cost of OCSs down significantly. In this experiment, however, we estimate the cost using $\delta = 0.5$, 1, and 1.5.

Figure 9 depicts the network cost breakdown, which shows that FTs are the most costly designs. This is because multi-rooted FTs require more switches and transceivers to support for the same sized network as other topologies. By comparison, flat networks such as EXPs and TRNs that directly connect ToR switches are more cost efficient than FTs for the same scale. While the cost of a small-scale PRN is close to that of a three-level FT when $\delta = 1.5$, the cost of FTs quickly dominates at medium and large scales.

6. SYSTEM-SCALE SIMULATIONS

In this section, we evaluate the performance of both reconfigurable and static networks using simulations. At a high level, our evaluation is structured as follows:

- Section 6.B compares various static and reconfigurable network topologies.
- Section 6.C compares reconfigurable networks under varying reconfiguration periods with no reconfiguration latency.
- Section 6.D discusses the difference between rotation-based reconfigurable networks and static meshes.
- Section 6.E studies the effects of different duty cycles on performance.

A. Methodology

We use the event-driven packet-level simulator called NetBench [70] to evaluate network performance. Compared to the static evaluation used in prior works [16,35,50], our simulated networks incorporate live topology reconfiguration during simulation runtime and can therefore more realistically capture the effects of online topology reconfiguration on performance. We have extended NetBench to support *in situ* topological reconfigurations, which is a feature most network simulators do not natively support. The code is made public to support the reproducibility of this work [71]. Since NetBench does not model the compute phases, we emulate barrier/synchronous communications to a limited degree by aligning co-flow arrivals in the trace. We recognize that this

may not fully capture the inter-dependencies between synchronous flows, which can introduce some biases into our simulations. We wish to explore the extent of these biases in future works. We assume EPSs have 100 Gb/s links and implement virtual output queueing (VOQ) with 20 KB buffer space per port. Network interface controllers similarly have 20 KB per-port buffer space. Packets have a maximum transmission unit (MTU) of 1500 bytes.

Topology. We assume all network topologies are constructed using EPSs with radix 32. Since all compared networks assume the same EPS radix, our analyses based on EPS radix of 64 (omitted for brevity) show little change in the relative performance of the evaluated network topologies. PRN uses a two-level Clos (leaf-spine) interconnect for its intra-pod topology, similar to a DF+ group. The simulated TRN interconnects all ToR switches with the OCS layer. The requisite OCS radices to build the simulated TRN and PRN are 232 and 16, respectively.

We use three static network topologies for baseline comparison: (1) a fully non-blocking three-layer FT (FT3), (2) a DF+/Megafly (DF+), and (3) an EXP/low-diameter (EXP). DF+ topology [13,62] has been shown to outperform DF networks. (This is because DF+'s two-level leaf-spine intra-group topology provides a non-blocking property for all intra-pod communications. This choice of intra-pod topology is also advantageous, as it requires only two virtual channels (VCs) to guarantee deadlock freedom, while a DF clique group would require three VCs.) The EXP baseline is representative of various low-diameter and EXP networks such as Slimfly [25], Jellyfish [23], and Xpander [24]. The DF+/Megafly and EXP networks are static, non-reconfigurable baselines to the PRN and TRN, respectively. Any differences in performance between the DF+ and EXP topologies and their dynamic PRN and TRN counterparts can be attributed to topology reconfiguration.

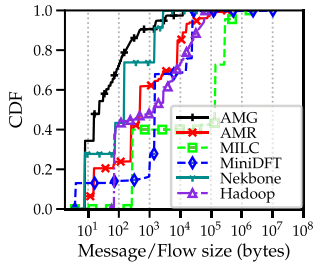
We equalize the EXP, DF+, TRN, and PRN in terms of cost following the cost model to ensure a fair performance comparison. (We do not include OCS port cost in the equalization process, as adding OCSs only introduces topological reconfigurability, but does not add capacity to the network.) This is done by *overprovisioning* the EXP and TRN by a factor of two, as shown in Table 3. The exception to this is the non-blocking FT, which is more expensive and acts as a reference.

Routing. The routing used on a PRN is based on the traffic-aware, globally-direct oblivious (TAGO) routing scheme in [64], which sends intra-pod traffic using equal cost multi-path

Table 3. Simulation Parameters of Different Topologies

Topology	Routing Policy	# VCs	Overprovision Factor
PRN	TAGO [64]	2	1
TRN	VLB [72]	2	2
DF+	TAGO [64]	2	1
EXP	HYB [22]	4	2
FT3[a]	ECMP	1	–

[a]With the exception of fat tree, which is more expensive, all other simulated topologies are equalized in terms of cost.



App	Type	Number of Ranks
MILC	HPC	512
Nekbone	HPC	1024
AMG	HPC	1728
AMR	HPC	1728
MiniDFT	HPC	4525
Hadoop	DC	6690

Fig. 10. Flow size distribution of the simulated workloads (left) and a table of the application parameters (right).

(ECMP) and inter-pod traffic using direct and indirect pod-to-pod routes. Packets that traverse indirect pod-to-pod routes will first be routed to a randomly chosen intermediate pod, before being forwarded to their destination pod. For TRNs, we employ a variant of topology-aware VLB similar to RotorLB [40], which sends traffic via both direct and indirect paths. Packets traversing indirect paths are first sent to an intermediate ToR before being routed to the destination rack. For the EXP, we implement a hybrid valiant-ECMP (HYB) multi-path routing as described in Kassing *et al.* [22]. Finally, the FT topology uses ECMP routing. All simulated topologies employ per-packet load balancing.

Workloads. Our simulations are driven by a mix of both synthetically generated loads and traces collected from real HPC and DC workloads. The HPC workloads are identified by the Exascale Computing Project (ECP) [73]. We collected the network traces for each of the simulated HPC mini-apps using DUMPI [74], which translates the MPI collectives into network flow arrivals. The DC workload is based on Hadoop, a common big data processing workload in production DCs [30]. Figure 10 shows the workload parameters and flow size distributions. For the scales of the simulated workloads, the processing overhead for a demand-aware reconfiguration based on Fig. 7 is under 10 ms.

B. Comparing Network Topologies

Our first set of experiments uses synthetic loads to stress test the networks under a variety of traffic patterns:

- **Permutation.** Each rack sends all its traffic to one randomly chosen destination rack.
- **Group permutation.** Each rack sends all its traffic evenly to racks in the adjacent group/pod.
- **3D stencil.** Racks are logically arranged in a 3D lattice; racks communicate with their neighbors in each dimension.
- **Disaggregated storage.** Racks are split evenly into two groups: compute racks and storage racks. Storage racks communicate solely with the compute racks. Compute racks have an equal probability of communicating with both compute and storage racks.
- **Bisection.** Racks are evenly binned into two groups. Racks in group 1 communicate with equal probability with racks in group 2, and vice versa.
- **Uniform.** Uniform probability of communication between every rack pair.

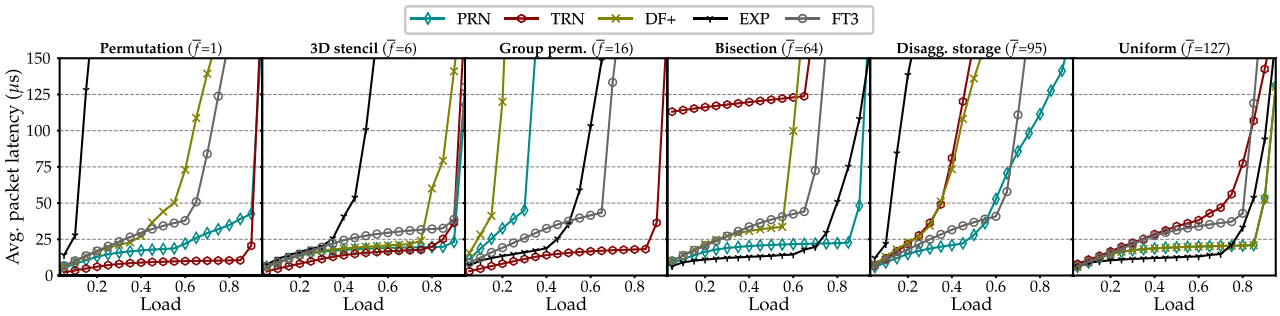


Fig. 11. Average packet latency for different topologies under varying loads. The average number of communication partners per rack (i.e., fan-out), $\bar{f}$, is annotated alongside each workload.

We characterize the *fan-out*, $\bar{f}$, of the traffic pattern as the average number of racks a source rack sends traffic to. In the context of a traffic matrix, fan-out/-in refers to the average number of non-zero entries in a row/column. For instance, in a network with N racks, a uniform traffic pattern would have $\bar{f} = N - 1$, while a permutation traffic pattern would have $\bar{f} = 1$. The associated fan-out of each synthetic traffic pattern is annotated in Fig. 11. All network topologies carry 128 racks. For PRNs and TRNs, the logical topology is reconfigured based on the expected traffic pattern between pods (for PRN) or between racks (for TRN) at the start of the simulation and fixed throughout the simulation.

The average packet latency of network topologies under varying injection rates is depicted in Fig. 11. TRN outperforms all other topologies when traffic patterns have small fan-outs (e.g., permutation, stencil, and group permutation), capable of supporting as much as 90% injection rate before saturation. Owing to its ability to steer the ToR uplinks to establish direct high bandwidth circuits between the "hot" rack pairs, a TRN can effectively bypass the need for multi-hop routing and still retain high throughput. However, a TRN's performance degrades quickly for workloads with fan-outs that exceed the total number of uplinks per ToR EPS, such as bisection, disaggregated storage, and uniform. In our experiment setup, each ToR EPS has 32 uplinks with which to form direct channels, so communication between rack pairs that are not directly connected will have to traverse indirect two-hop routes via an intermediate rack. This leads to larger hop counts and creates congestion for other packets.

PRNs showcase excellent performance in all workloads except group permutation, which is adversarial to topologies such as DF+ with notions of group/pods. As expected, PRNs still outperform DF+ due to their topological reconfigurability, allowing PRNs to place their inter-pod links between traffic hotspots. PRNs show better performance when fan-out is large, as their dense pod-level topology can provide the path diversity needed to reach "faraway" racks, while the static intra-pod topology can be used to handle local traffic. We see that DF+ (green cross line) and PRNs (blue diamond line) perform equally under uniform traffic, as PRNs' inter-pod topology, like that of DF+, is also uniform.

Overall, TRNs are better optimized for workloads with small fan-outs, but suffer performance degradation as fan-out increases. Our evaluation shows PRNs to be more versatile under both small and large fan-out workloads, as each PRN rack can reach "faraway" racks using the inter-pod links and relies on the static intra-pod interconnect for local communication.

Table 4. Server-to-Server Fan-Out Statistics of Simulated HPC and DC Traces

	Fan-Out Statistics		
Application	**Min.**	**Avg.**	**Max.**
AMG	18	79.2	293
AMR	15	36.2	168
Nekbone	16	29.6	36
MiniDFT	292	296	316
MILC	20	30.5	40
Hadoop	1	136	586

C. Effects of Reconfiguration Period on Performance

Using HPC and DC workloads, we evaluate the effects of reconfiguration period on performance of a PRN and a TRN under idealized conditions (i.e., no reconfiguration latency and no demand-aware processing overhead), starting from 100 ns and increasing in factors of 10 (e.g., 100 ns, 1 μs, 10 μs, 100 μs, 1 ms, 10 ms, 100 ms, 1 s). We measure performance using average flow throughput, where the throughput of each flow is measured by the ratio of total bits transferred to the time taken for bits to be transferred. The server-level fan-out of the evaluated HPC and DC traces are shown in Table 4. We assume that reconfiguration is instantaneous (i.e., zero latency) in this experiment to isolate effects of reconfiguration latency on performance. For the PRN, we employ a demand-aware reconfiguration by switching the inter-pod logical topology based on the current traffic demand between pods. For the TRN, we simulate both demand-aware and demand-oblivious (rotation-based) reconfigurations. TRN-DA represents networks such as ProjecToR [15] and Firefly [38], and the demand-oblivious rotation-based TRN (TRN-R) represents networks such as RotorNet [40] and Sirius [14]. Figure 12 shows the average flow throughput for all simulated workloads.

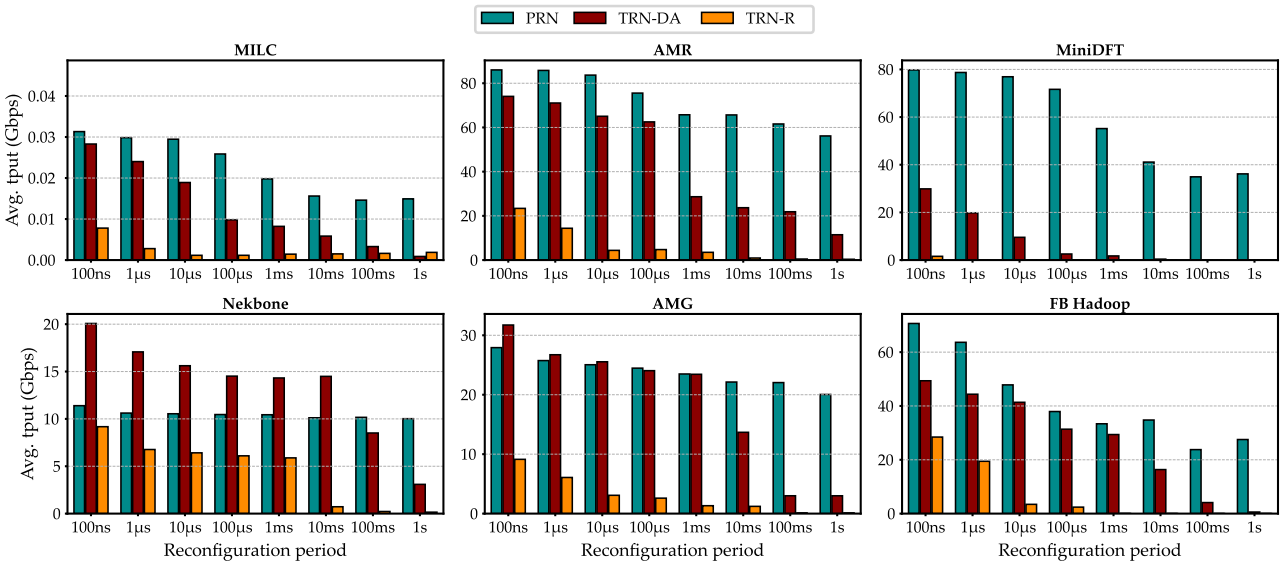


Fig. 12. Cost-equivalent evaluation of a PRN and a TRN. Average flow throughput of HPC and DC workloads as a result of various reconfiguration periods.

1. *Shorter Reconfiguration Periods Lead to Higher Throughput*

Intuitively, networks with shorter reconfiguration periods can better adapt to demand variations over time and would yield higher throughput as a result. As expected, this trend can also be observed in Fig. 12 in both the PRN and TRN-DA.

2. *PRNs Generally Outperform TRNs*

Even though PRNs have a static intra-pod topology, we observe a higher average throughput than TRNs for most workload cases other than Nekbone. This is because although HPC and DC workloads generate non-uniform communication patterns, they are rarely as skewed as a rack-to-rack permutation (see analysis in Section 6.B) to warrant direct rack-to-rack links. Even without reconfiguring intra-pod topology, PRNs can attain high throughput for local communication using multi-path routing schemes (e.g., ECMP or k-shortest path) to load-balance local links.

The results in Fig. 12 agree with our findings in Section 6.B that PRNs outperform TRNs under large fan-out workloads. For instance, in FB Hadoop and mini-density function theory (Mini-DFT), two applications that have the highest average fan-out, a PRN attains $\sim 1.5-4\times$ higher average throughput than TRN-DA, given a 1 µs reconfiguration period.

3. *PRN Is Less Dependent on Reconfiguration than TRN-DA*

A PRN exhibits less throughput degradation than TRN-DA and TRN-R as the reconfiguration period increases. Across all simulated workloads, when the reconfiguration period changes from 1 µs to 1 s, the PRN throughput degrades by approximately $2.1\times$, while TRN-DA experiences a throughput degradation as much as $\geq 100\times$ in the case of FB Hadoop. The robustness of PRNs to different reconfiguration periods is due to two main reasons. First, the non-reconfigurable intra-pod fabric in PRNs means that regardless of the rate of topology reconfiguration, local communications between racks of the same pod are not affected. So, workloads that communicate more heavily with local racks (e.g., FB Hadoop) [30] are less affected by the reconfiguration period.

Second, the pod-level connectivity graph of a PRN is dense (i.e., number of uplinks per pod > number of pods), which allows us to allocate at least one link between pod pairs that do not expect heavy traffic to handle potential demand bursts. As a result, the throughput performance is more robust to longer reconfiguration periods, as the dense pod-level topology is more resilient to demand changes. This is not possible in a TRN with a sparse rack-to-rack logical topology, which is dependent on frequent topology reconfigurations to optimize its rack-to-rack matchings to better serve current traffic demands.

4. *TRN-R Delivers Lower Throughput for Simulated HPC and DC Workloads*

As expected, TRN-R performs worse than TRN-DA in the same reconfiguration period, since it does not optimize topology based on demand. Our findings show that performance is higher with demand-aware reconfigurations with longer reconfiguration periods than with demand-oblivious reconfigurations with shorter reconfigurations. This suggests that for the identified workloads, a reconfiguration strategy that is infrequent but deliberate is better than one that is frequent but oblivious. In our simulations, we did not identify a workload for which TRN-R is conducive.

D. Rotating Expanders ≠ Static Full Mesh

Prior works such as Sirius [14] argued that demand-oblivious TRNs approximate a full-mesh interconnect by time-multiplexing across a sequence of circuit configurations

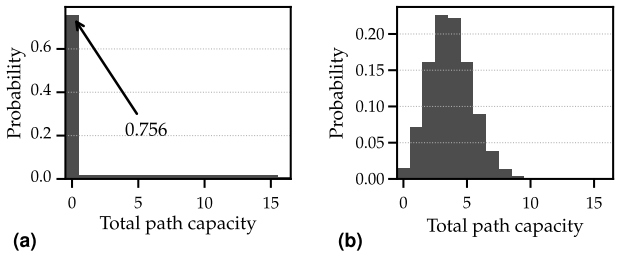


Fig. 13. Distribution of total path capacity between two racks across all OCS configurations for a TRN and a PRN. Both topologies contain 128 ToR switches; each ToR switch has 16 reconfigurable uplinks with unit capacity. (a) TRN-R 128 racks. (b) Eight-pod PRN, 16 racks/pod.

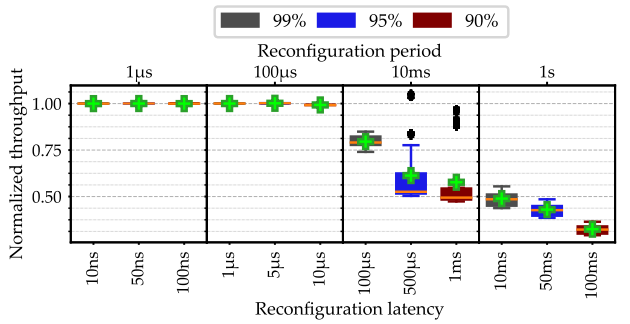


Fig. 14. Box plot of the normalized flow throughput of MILC in a TRN-Flat network under various combinations of reconfiguration period and reconfiguration latency. The duty cycle is the ratio of circuit uptime to the reconfiguration period. Green crosses denote the average normalized throughput.

such that over time, all endpoints receive an equal share of direct connections. However, in our simulations, we found that TRN-R performs poorly in terms of throughput (see Section 6.C) when compared to other network architectures. This suggests that while rotation-based reconfigurable networks may resemble a full mesh in a time-averaged sense, their performance is far from that of a uniform mesh. Here, we explore this contradiction.

Transient network paths. Maintaining a consistently high capacity between racks is key to a high performance network. Unlike static networks, the logical topology in reconfigurable networks is dynamic, and thus the number of routes between rack pairs also varies. Figure 13 shows the total number of paths available between randomly selected rack pairs across all possible OCS matchings. The figure shows that a large number of OCS configurations results in no paths between racks (i.e., "disconnected" racks) in TRNs. Disconnected hosts have to wait for the next reconfiguration epoch when a direct route is reestablished before sending traffic, leading to long-tailed packet latency. By comparison, ∼98% of OCS configurations result in non-zero path capacity between racks in a PRN, which is key to its demonstrated high throughput.

Coordinated host transport injection. Round-robin TRN-R architectures require customized host transport to maintain a separate FIFO queue for each destination and intermediate destination nodes. This is necessary to prevent hosts from congesting the network unnecessarily by injecting packets when the OCS configuration does not provide a path to the destination. However, this requires tight integration between the network controller and the host transport, which can introduce significant overhead to system design.

E. Performance under Varying Duty Cycles

Thus far, we assumed zero-latency reconfigurations. However, the process of reconfiguring an OCS requires a series of circuit setups and tear downs that incurs a nonnegligible delay. No packets can traverse the OCS during reconfiguration; this duration is referred to as the circuit *downtime*. The duration when optical circuits are online is referred to as the circuit *uptime*. We refer to the ratio of circuit uptime to the reconfiguration period as the *duty cycle*. To amortize the cost of circuit switching, topology reconfiguration events must be spaced sufficiently far apart. A common heuristic employed by prior works is to select a reconfiguration period based on the reconfiguration latency to realize a 90% duty cycle [15,38,43,54,58,75,76]. For example, if the reconfiguration latency is 10 μs, a corresponding reconfiguration period of 100 μs will realize a duty cycle of 90%.

We now study how duty cycles affect performance at various reconfiguration delays. Note that we consider an end-to-end reconfiguration delay that takes into account when a circuit is torn down until a new circuit is established and can carry live traffic again. To this end, we use an example demand-aware TRN and vary the reconfiguration periods (1 μs, 100 μs, 10 ms, 1 s) and delay combinations to realize duty cycles of 99%, 95%, and 90%. Figure 14 shows the distribution of normalized flow throughput for various combinations of reconfiguration periods and delay in a TRN-Flat network running MIMD Lattice Computation (MILC). The flow-normalized throughput is the ratio of actual flow throughput to the flow throughput when reconfiguration delay is zero.

The results show that even when duty cycles are identical, differences in reconfiguration delay can still affect performance. Specifically, while varying the duty cycle results in little difference in normalized throughput when reconfiguration delay is low (i.e., ≤ 10 μs), lower duty cycles can degrade throughput more noticeably when reconfiguration delay is high. For instance, while the difference in average normalized throughput of 99% and 90% duty cycles drops by ∼40% for reconfiguration delays above 100 μs, there is very little appreciable drop in performance when reconfiguration delays are below 10 μs. This is because a circuit downtime of 1 μs is fairly insignificant from the perspective of a packet, but a circuit downtime of 100 ms corresponds to a loss of 10 Gbits per link in a 100 Gbits/s network. Such an extended circuit downtime may cause congestion to build at the switch buffers, leading to a drastic increase in queueing delay and drop in throughput. The disproportionate throughput drop with a lower duty cycle when end-to-end reconfiguration delay is long suggests that network operators should operate at duty cycles much higher than 90% to ensure higher throughput.

7. DISCUSSION AND KEY TAKEAWAYS

In this section, we discuss the key takeaways from our evaluation on *cost-equivalent* TRN and PRN, and discuss the implications on network design.

• **Not all (*cost-comparable*) reconfigurable networks are built equal.** The placement of OCSs plays a role in the types of traffic patterns a reconfigurable network is optimized for, regardless of the technology of OCSs. For instance, TRNs have excellent performance when the communication pattern is highly skewed (i.e., low fan-out). Conversely, PRNs outperform TRNs when the workload exhibits a low-skew (i.e., high fan-out) communication pattern, where traffic is exchanged between a large number of source–destination pairs.

• **OCS placement affects dependence on the reconfiguration period.** Prior works such as [77] have mentioned that lower switching latency is needed for networks that place OCSs closer to the edge. This is partly because traffic near the edge (closer to the servers) tends to be burstier than that near the network's core [78] due to a lack of aggregation from switches to "smooth" out the traffic arrivals, which means the optical circuits need to be reconfigured more frequently to handle the burstier arrivals. Our results agree with these observations, as they show that a TRN, which places the OCSs closer to the edge than a PRN, requires a shorter reconfiguration period for good performance. This, in part, has to do with the TRN's sparse logical topology, which means that links have to be reconfigured more frequently when each source host communicates with many destination hosts simultaneously. The PRN's performance is less dependent on the reconfiguration period for three reasons: (1) a PRN has a static intra-pod fabric, so intra-pod communication is independent of reconfiguration; (2) traffic that traverses the reconfigurable optical network is less bursty in a PRN than in a TRN due to aggregation of traffic; and (3) a PRN's inter-pod topology is dense, which can provide high capacity between all source–destination pairs even without frequent reconfiguration.

• **A demand-aware network with a large reconfiguration period outperforms a demand-oblivious network with a short reconfiguration period.** Our simulations show that a TRN-DA with a 100 ms reconfiguration period delivers $0.99-4.98\times$ higher average throughput compared to TRN-R with a 1 μs reconfiguration period. This suggests that for the identified workloads, it may be better to reconfigure topology in a demand-aware but infrequent manner. This applies to HPC systems that run long-lasting (of the order of several seconds or longer) scientific applications [79] with highly skewed and stable communication patterns [35,53] or ML clusters with long-running training jobs that have predictable traffic patterns [80] over time. Note that our choice of (primarily HPC-focused) workloads does not cover the entire spectrum of possible traffic patterns. For systems that support predominantly short-lived jobs with highly variable traffic patterns over time, it may be beneficial to consider TRN-R-like reconfigurable networks that deliver uniform bandwidth to every pair of hosts.

• **No one-size-fits-all duty cycle; larger reconfiguration latency demands a higher duty cycle.** When reconfiguration latency is low. Current reconfigurations are limited by a sequence of events (physical switching, transceiver synchronization, port draining, routing updates, etc). Our study shows that even with the same duty cycle, the reconfiguration latency matters. When end-to-end reconfiguration delay is high, the network must operate at a much higher duty cycle (e.g., 99% or more) to prevent congestion buildup.

In summary, a TRN is suited for small-scale networks that run a well-behaved workload with low fan-out communication patterns, such as ML training [80]. For clusters that support large batch-processing jobs such as Hadoop, PRN is the better design, and it can better serve high fan-out workloads. In commodity clusters with high reconfiguration delay, the reconfiguration period must also be much larger to amortize the cost of circuit switching. Our findings suggest that PRNs, being less dependent on low reconfiguration periods to deliver high throughput, are more suited for reconfigurable networks built with commercial OCSs, whose higher switching latency may prohibit low reconfiguration periods.

8. CONCLUSION AND FUTURE WORK

While many reconfigurable networks based on optical circuit switching have been proposed in the past, very little is understood about the effects of physical OCS placement on system performance. In this work, we evaluate two classes of reconfigurable networks: (1) PRNs, which place OCSs between pods, and (2) TRNs, which place OCSs between ToR switches. We compare reconfigurable networks and state-of-the-art static networks in terms of cost/power consumption, scalability, and network performance. Using realistic simulations driven by real-world HPC and DC workloads, we assess reconfiguration periods and their effect on network performance.

Our findings raise many open problems for future work. First, further explorations in failure resilience and availability will better flesh out the trade-offs among different reconfigurable networks. Second, as prevalent host transport protocols such as TCP/IB today offer sub-optimal performance in dynamic networks, designing congestion control protocols that synergize with reconfigurable networks could further enhance performance and incentivize the adoption of reconfigurable networks in high performance systems.

Funding. Advanced Research Projects Agency - Energy (ARPA-E) ENLITENED Program (Project Award No. DE-AR00000843); Laboratory for Physical Sciences (LPS) Advanced Computing Systems (ACS) Research, under the National Security Agency IAC MAC 19-1979 Subcontract.

REFERENCES

1. "The Top500 HPC list," 2020, https://www.top500.org/green500/lists/2020/11/.
2. K. Bergman, "Empowering flexible and scalable high performance architectures with embedded photonics," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (2018), p. 378.
3. G. Georgakoudis, N. Jain, T. Ono, K. Inoue, S. Miwa, and A. Bhatele, "Evaluating the impact of energy efficient networks on HPC workloads," in *IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC)* (2019), pp. 301–310.
4. J. Shalf, S. Dosanjh, and J. Morrison, "Exascale computing technology challenges," in *Proceedings of the 9th International*

Conference on High Performance Computing for Computational Science—VECPAR (Springer-Verlag, 2011), pp. 1–25.
5. M. Besta, M. Schneider, M. Konieczny, K. Cynk, E. Henriksson, S. Di Girolamo, A. Singla, and T. Hoefler, "FatPaths: routing in supercomputers and data centers when shortest paths fall short," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis* (IEEE, 2020).
6. J. Y. Yen, "Finding the *k* shortest loopless paths in a network," Manage. Sci. **17**, 712–716 (1971).
7. H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "COPE: traffic engineering in dynamic networks," in *SIGCOMM* (2006), pp. 99–110.
8. H. Racke, "Minimizing congestion in general networks," in *43rd Annual IEEE Symposium on Foundations of Computer Science, Proceedings* (IEEE, 2002), pp. 43–52.
9. M. S. Rahman, S. Bhowmik, Y. Ryasnianskiy, X. Yuan, and M. Lang, "Topology-custom UGAL routing on Dragonfly," in *International Conference for High Performance Computing Networking, Storage, and Analysis (SC)* (2019), paper 17.
10. G. Michelogiannakis, K. Z. Ibrahim, J. Shalf, J. J. Wilke, S. Knight, and J. P. Kenny, "APHiD: hierarchical task placement to enable a tapered fat tree topology for lower power and cost in HPC networks," in *17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)* (2017), pp. 228–237.
11. J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "Hyperx: topology, routing, and packaging of efficient large-scale networks," in *International Conference for High Performance Computing Networking, Storage, and Analysis (SC)* (2009).
12. J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable Dragonfly topology," in *Proceedings of the 35th International Symposium on Computer Architecture (ISCA)* (2008).
13. A. Shpiner, Z. Haramaty, S. Eliad, V. Zdornov, B. Gafni, and E. Zahavi, "Dragonfly+: low cost topology for scaling datacenters," in *IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)* (IEEE, 2017), pp. 1–8.
14. H. Ballani, P. Costa, R. Behrendt, D. Cletheroe, I. Haller, K. Jozwik, F. Karinou, S. Lange, K. Shi, B. Thomsen, and H. Williams, "Sirius: a flat datacenter network with nanosecond optical switching," in *SIGCOMM* (2020).
15. M. Ghobadi, R. Mahajan, A. Phanishayee, N. Devanur, J. Kulkarni, G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper, "ProjecToR: agile reconfigurable data center interconnect," in *SIGCOMM* (2016).
16. G. Liu, R. Proietti, M. Fariborz, P. Fotouhi, X. Xiao, and S. J. B. Yoo, "Architecture and performance studies of 3D-Hyper-FleX-LION for reconfigurable all-to-all HPC networks," in *SC: International Conference for High Performance Computing, Networking, Storage and Analysis* (2020).
17. M. Y. Teh, J. J. Wilke, K. Bergman, and S. Rumley, "Design space exploration of the Dragonfly topology," in *International Conference on High Performance Computing* (2017).
18. C. E. Leiserson, "Fat-trees: universal networks for hardware-efficient supercomputing," IEEE Trans. Comput. **C-34**, 892–901 (1985).
19. A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," in *SIGCOMM* (2009).
20. M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *SIGCOMM* (2008).
21. A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat, "Jupiter rising: a decade of Clos topologies and centralized control in Google's datacenter network," in *SIGCOMM* (2015).
22. S. Kassing, A. Valadarsky, G. Shahaf, M. Schapira, and A. Singla, "Beyond fat-trees without antennae, mirrors, and disco-balls," in *SIGCOMM* (ACM, 2017), pp. 281–294.
23. A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: networking data centers, randomly," in *Networked Systems Design and Implementation (NSDI)* (2012).
24. A. Valadarsky, G. Shahaf, M. Dinitz, and M. Schapira, "Xpander: towards optimal-performance datacenters," in *International Conference on Emerging Networking Experiments and Technologies (CoNEXT)* (2016).
25. M. Besta and T. Hoefler, "Slim Fly: a cost effective low-diameter network topology," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2014).
26. S. Zhao, R. Wang, J. Zhou, J. Ong, J. C. Mogul, and A. Vahdat, "Minimal rewiring: efficient live expansion for Clos data center networks," in *Networked Systems Design and Implementation (NSDI)* (2019).
27. M. Zhang, R. N. Mysore, S. Supittayapornpong, and R. Govindan, "Understanding lifecycle management complexity of datacenter topologies," in *Networked Systems Design and Implementation (NSDI)* (2019).
28. T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," in *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking* (2009).
29. S. Kandula, J. Padhye, and P. Bahl, "Flyways to de-congest data center networks," in *Proceedings of HotNets* (2009).
30. A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *SIGCOMM* (2015).
31. "Telescent G4 network topology manager," 2021, https://www.telescent.com/products.
32. A. Chatzieleftheriou, S. Legtchenko, H. Williams, and A. Rowstron, "Larry: practical network reconfigurability in the data center," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)* (2018), pp. 141–156.
33. "Calient S-Series S320," 2021, https://www.calient.net/resources/datasheets/.
34. "Polatis optical circuit switch," 2021, https://www.polatis.com/series-7000-384x384-port-software-controlled-optical-circuit-switch-sdn-enabled.asp.
35. K. Wen, P. Samadi, S. Rumley, C. P. Chen, Y. Shen, M. Bahadori, K. Bergman, and J. Wilke, "Flexfly: enabling a reconfigurable Dragonfly through silicon photonics," in *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis* (2016).
36. T. J. Seok, N. Quack, S. Han, R. S. Muller, and M. C. Wu, "Large-scale broadband digital silicon photonic switches with vertical adiabatic couplers," Optica **3**, 64–70 (2016).
37. T. Chu, L. Qiao, W. Tang, D. Guo, and W. Wu, "Fast, high-radix silicon photonic switches," in *Optical Fiber Communication Conference (OFC)* (Optical Society of America, 2018), paper Th1J.4.
38. N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer, "Firefly: a reconfigurable wireless data center fabric using free-space optics," in *SIGCOMM* (2014), pp. 319–330.
39. 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 *SIGCOMM* (2012).
40. W. M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A. C. Snoeren, and G. Porter, "RotorNet: a scalable, low-complexity, optical datacenter network," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (2017).
41. W. M. Mellette, R. Das, Y. Guo, R. McGuinness, A. C. Snoeren, and G. Porter, "Expanding across time to deliver bandwidth efficiency and low latency," in *Networked Systems Design and Implementation (NSDI)* (2020).
42. S. Kamil, J. Shalf, L. Oliker, and D. Skinner, "Understanding ultra-scale application communication requirements," in *Proceedings of the Workload Characterization Symposium* (2005).
43. N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers," in *SIGCOMM* (2011).
44. J. L. Benjamin, T. Gerard, D. Lavery, P. Bayvel, and G. Zervas, "PULSE: optical circuit switched data center architecture operating at nanosecond timescales," J. Lightwave Technol. **38**, 4906–4921 (2020).

45. W. Wang, S. Das, and T. E. Ng, "Abstractions for reconfigurable hybrid network update and a consistent update approach," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Optical Systems* (2021), pp. 6–11.
46. Facebook, "Facebook Open Switching System ("FBOSS") and Wedge in the open," https://engineering.fb.com/2015/03/10/data-center-engineering/facebook-open-switching-system-fboss-and-wedge-in-the-open/.
47. Y. Shen, *Reconfigurable Optically Interconnected Systems* (Columbia University, 2020).
48. K. A. Clark, D. Cletheroe, T. Gerard, I. Haller, K. Jozwik, K. Shi, B. Thomsen, H. Williams, G. Zervas, H. Ballani, P. Bayvel, P. Costa, and Z. Liu, "Synchronous subnanosecond clock and data recovery for optically switched data centres using clock phase caching," Nat. Electron. **3**, 426–433 (2020).
49. M. Y. Teh, Z. Wu, and K. Bergman, "Flexspander: augmenting expander networks in high performance computing and data centers with optical bandwidth steering," J. Opt. Commun. Netw. **12**, B44–B54 (2020).
50. G. Michelogiannakis, Y. Shen, M. Y. Teh, X. Meng, B. Aivazi, T. Groves, J. Shalf, M. Glick, M. Ghobadi, L. Dennison, and K. Bergman, "Bandwidth steering in HPC using silicon nanophotonics," in *International Conference for High Performance Computing Networking, Storage, and Analysis (SC)* (2019), paper 41.
51. Y. Xia, X. S. Sun, S. Dzinamarira, D. Wu, X. S. Huang, and T. E. Ng, "A tale of two topologies: exploring convertible data center network architectures with flat-tree," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (2017), pp. 295–308.
52. G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan, "C-through: part-time optics in data centers," in *SIGCOMM* (2011).
53. K. J. Barker, A. Benner, R. Hoare, A. Hoisie, A. K. Jones, D. K. Kerbyson, D. Li, R. Melhem, R. Rajamony, E. Schenfeld, S. Shao, C. Stunkel, and P. Walker, "On the feasibility of optical circuit switching for high performance computing systems," in *SC: Proceedings of the ACM/IEEE Conference on Supercomputing* (2005), paper 16.
54. G. Porter, R. Strong, N. Farrington, A. Forencich, C.-S. Pang, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating microsecond circuit switching into the data center," in *SIGCOMM* (2013).
55. W. Miao, J. Luo, S. Di Lucente, H. Dorren, and N. Calabretta, "Novel flat datacenter network architecture based on scalable and flow-controlled optical switch system," Opt. Express **22**, 2465–2472 (2014).
56. J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," J. ACM **19**, 248–264 (1972).
57. M. Fiorani, M. Tornatore, J. Chen, L. Wosinska, and B. Mukherjee, "Spatial division multiplexing for high capacity optical interconnects in modular data centers," J. Opt. Commun. Netw. **9**, A143–A153 (2017).
58. M. K. Mukerjee, C. Canel, W. Wang, D. Kim, S. Seshan, and A. C. Snoeren, "Adapting TCP for reconfigurable datacenter networks," in *Networked Systems Design and Implementation (NSDI)* (2020).
59. M. Miller and J. Sirán, "Moore graphs and beyond: a survey of the degree/diameter problem," Electron. J. Comb. **14**, 1–61 (2012).
60. W. G. Bridges and S. Toueg, "On the impossibility of directed Moore graphs," J. Comb. Theory B **29**, 339–341 (1980).
61. G. Faanes, A. Bataineh, D. Roweth, T. Court, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins, and J. Reinhard, "Cray Cascade: a scalable HPC system based on a Dragonfly network," in *International Conference on High Performance Computing, Networking, Storage and Analysis (SC)* (2012).
62. M. Flajslik, E. Borch, and M. A. Parker, "Megafly: a topology for exascale systems," in *ISC High Performance* (2018).
63. Gurobi Optimization, "Gurobi optimizer reference manual," 2019, http://www.gurobi.com.
64. M. Y. Teh, Y.-H. Hung, G. Michelogiannakis, S. Yan, M. Glick, J. Shalf, and K. Bergman, "TAGO: rethinking routing design in high performance reconfigurable networks," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2020), paper 25.
65. P. Bakopoulos, K. Christodoulopoulos, G. Landi, *et al.*, "NEPHELE: an end-to-end scalable and dynamically reconfigurable optical architecture for application-aware SDN cloud data centers," IEEE Commun. Mag. **56**(2), 178–188 (2018).
66. "Mellanox Spectrum SN2700 32-Port 100GbE Open Ethernet Switch with Mellanox Onyx - Part ID: MSN2700-CS2R," 2021, https://www.colfaxdirect.com/store/pc/viewPrd.asp?idproduct=2993&idcategory=0.
67. "Mellanox EDR 100Gb/s QSFP28 Optical Transceiver - Part ID: MMA1B00-E100," 2021, https://www.colfaxdirect.com/store/pc/viewPrd.asp?idproduct=3187&idcategory=25.
68. H. Isono, "Latest standardization status and its future directions for high speed optical transceivers," Proc. SPIE **10946**, 1094604 (2019).
69. M. R. Jokar, J. Qiu, F. T. Chong, L. L. Goddard, J. M. Dallesasse, M. Feng, and Y. Li, "Baldur: a power-efficient and scalable network using all-optical switches," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)* (2020).
70. NetBench, https://github.com/ndal-eth/netbench.
71. "minyee/reconf_network_eval: first release of reconfigurable network topology evaluation framework," 2021, https://zenodo.org/record/4897956.
72. L. G. Valiant and G. J. Brebner, "Universal schemes for parallel communication," in *Proceedings of the 13th Annual ACM Symposium on Theory of Computing (STOC)* (Association for Computing Machinery, 1981), pp. 263–277.
73. "Characterization of the DOE mini-apps" [Accessed 16 February 2019], https://portal.nersc.gov/project/CAL/doe-miniapps.htm.
74. H. Adalsteinsson, S. Cranford, D. A. Evensky, J. P. Kenny, J. Mayo, A. Pinar, and C. L. Janssen, "A simulator for large-scale parallel computer architectures," Int. J. Distrib. Syst. Technol. **1**, 57–73 (2010).
75. H. Liu, M. K. Mukerjee, C. Li, N. Feltman, G. Papen, S. Savage, S. Seshan, G. M. Voelker, D. G. Andersen, M. Kaminsky, G. Porter, and A. C. Snoeren, "Scheduling techniques for hybrid circuit/packet networks," in *CoNEXT: Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies* (2015), paper 41.
76. H. Liu, F. Lu, A. Forencich, R. Kapoor, M. Tewari, G. M. Voelker, G. Papen, A. C. Snoeren, and G. Porter, "Circuit switching under the radar with reactor," in *Networked Systems Design and Implementation (NSDI)* (2014).
77. N. Farrington, A. Forencich, G. Porter, P.-C. Sun, J. E. Ford, Y. Fainman, G. C. Papen, and A. Vahdat, "A multiport microsecond optical circuit switch for data center networking," IEEE Photon. Technol. Lett. **25**, 1589–1592 (2013).
78. T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement* (ACM, 2010), pp. 267–280.
79. G. Michelogiannakis, B. Klenk, B. Cook, M. Y. Teh, M. Glick, L. Dennison, K. Bergman, and J. Shalf, "A case for intra-rack resource disaggregation in HPC," ACM Trans. Archit. Code Optim. **19**, 29 (2022).
80. M. Khani, M. Ghobadi, M. Alizadeh, Z. Zhu, M. Glick, K. Bergman, A. Vahdat, B. Klenk, and E. Ebrahimi, "SiP-ML: high-bandwidth optical network interconnects for machine learning training," in *SIGCOMM* (2021), pp. 657–675.