Flattened Clos: Designing High-performance Deadlock-free Expander Data Center Networks Using Graph Contraction

Shizhen Zhao[1,*], Qizhou Zhang[1,*], Peirui Cao[1], Xiao Zhang[1], Xinbing Wang[1], Chenghu Zhou[1,2]
[1]*Shanghai Jiao Tong University,* [2]*Chinese Academy of Sciences*

Abstract

Clos networks have witnessed the successful deployment of RoCE in production data centers. However, as DCN bandwidth keeps increasing, building Clos networks is becoming cost-prohibitive and thus the more cost-efficient expander graph has received much attention in recent literature. Unfortunately, the existing expander graphs' topology and routing designs may contain Cyclic Buffer Dependency (CBD) and incur deadlocks in PFC-enabled RoCE networks.

We propose Flattened Clos (FC), a topology/routing co-designed approach, to eliminate the PFC-induced deadlocks in expander networks. FC's topology and routing are designed in three steps: 1) logically divide each ToR switch into k virtual layers and establish connections only between adjacent virtual layers; 2) generate *virtual up-down* paths for routing; 3) flatten the virtual multi-layered network and the virtual up-down paths using graph contraction. We rigorously prove that FC's design is deadlock-free and validate this property using a real testbed and packet-level simulation. Compared to expander graphs with the edge-disjoint-spanning-tree (EDST) based routing (a state-of-art CBD-free routing algorithm for expander graphs), FC reduces the average hop count by at least 50% and improves network throughput by $2-10\times$ or more. Compared to Clos networks with up-down routing, FC increases network throughput by $1.1-2\times$ under all-to-all and uniform random traffic patterns.

1 Introduction

Driven by the need of low latency, high throughput and low CPU overhead, large Internet service providers such as Microsoft and Alibaba have deployed RDMA over Commodity Ethernet (RoCE) [14, 20] in their Clos data centers. RoCE requires a lossless network for optimal performance. To avoid packet loss in Ethernet, Priority-based Flow Control (PFC) is usually enabled to perform a hop-by-hop flow control to avoid exhausting switch buffers by upstreaming flows. However, enabling PFC introduces the risk of deadlocks, especially for the large-scale deployment of RoCEv2. Thanks to the layered structure of Clos data centers, the up-down routing in Clos networks can prevent deadlocks with proper safety mechanisms under normal operations [20] and failure scenarios [24].

However, as the data center traffic and the network bandwidth keep increasing, building Clos topologies is becoming cost-prohibitive [4]. In order to reduce the network cost, flatter expander graphs, such as Jellyfish [46], SlimFly [5], Xpander [50], FatClique [54], etc., have been proposed to build data centers. A recent study [36] shows that a full throughput expander uses 25% fewer switches than a full throughput Clos. Note that the throughput values of expander graphs are attained using a multi-commodity flow formulation based on the K-Shortest Path (KSP) routing [53]. Unfortunately, the KSP routing in expander graphs may contain Cyclic Buffer Dependency (CBD), and thus could incur severe PFC deadlocks. Therefore, the performance-gain or cost-reduction of expander graphs over Clos becomes questionable for RoCEv2 traffic.

The key to supporting RoCE in expander graphs is to eliminate CBD. Approaches to eliminate CBD can be generally grouped into three classes. The first approach is to assign different lossless priorities for packets at different hops [12, 15, 27]. This approach has been widely adopted in HPCs, in which the underlying Infiniband network supports 15 lossless priorities (*a.k.a.* Virtual Channel). However, due to the limited switch buffer space, data center switches can support at most two or three lossless priorities [20]. The second approach is to disable PFC and redesign RoCE to work with lossy networks, e.g., NDP [21], IRN [35], FatPaths [6], etc. However, lossy RoCE requires hardware support. For example, Mellanox ConnectX-4 onwards NICs support lossy RoCE, but Mellanox ConnectX-3 NICs do not. In addition, lossy RoCE may incur higher latency for mice flows, especially when a sender has to rely on a timeout to retransmit a lost packet. iWarp [41] is another RDMA technology that runs on lossy networks. However, its performance is poor because it relies on TCP to guarantee lossless delivery.

[*]These authors contribute equally to this work.

The third approach is to design a routing solution that is fundamentally free of CBD, just as the up-down routing in Clos. Along this direction, TCP Bolt [48] and DF-EDST [47] were proposed, and the key idea is to find as many edge disjoint spanning trees (EDST) in an expander graph as possible and then route each packet in one of the spanning tree from its source to its destination. The EDST-based routing is CBD-free, but its throughput performance is poor. The key reason is that the EDST-based routing cannot effectively utilize all the network resources: 1) the average path length is usually large and increases quickly with network size; 2) some network links could remain idle as they do not belong to any EDST.

Our work called **Flattened Clos (FC)** offers a novel topology-routing co-design to eliminate CBDs in RoCE networks. FC's topology is essentially a random regular graph that is mappable to a multi-layered topology. We construct FC's topology in two steps: 1) virtually split each ToR switch into k virtual switches, each of which belongs to a virtual layer, and 2) randomly interconnect the layer-i ($i = 2, ..., k-1$) virtual switches to the layer-$(i-1)$ and the layer-$(i+1)$ virtual switches. The multi-layered virtual structure of FC allows performing up-down routing based on virtual layers. To this end, we propose the Edge-Disjoint Virtual Up-Down Routing for FC. For every source-destination pair, FC's routing transforms the path-finding problem into a min-cost-flow problem and then finds the maximum number of edge-disjoint paths. We analyze FC's design as follows to demonstrate its feasibility:

1. We offer a theoretical guidance for choosing the right number k of virtual layers when constructing FC's topology (see the strategy (*) in Section 3.2.3), and validate the strategy via numerical analysis.

2. We prove that FC's routing is CBD-free, and thus is deadlock-free. In fact, FC's topology and routing paths can be viewed as contracted graphs of a virtual multi-layered network and virtual up-down paths. This *graph contraction* operation preserves the CBD-free property.

3. We show that FC's cabling complexity can be dramatically reduced by introducing a layer of Patch Panels (PP) or Optical Circuit Switches (OCS) to interconnect all the ToR switches. Admittedly, having this PP/OCS layer increases cable length and cable cost. As network size becomes large, the overall network cost of FC is still lower than that of Clos under similar bisection bandwidth.

4. We demonstrate that FC outperforms expander graphs with EDST routing [47, 48] (the state-of-art CBD-free routing for expanders). Specifically, FC reduces the average hop count (AHC) by at least 50% and increases network throughput by $2-10\times$ or more.

5. We compare the throughput performance between FC and Clos networks with up-down routing, built using the same number of hosts and electrical switches. FC achieves $1.1-2\times$ throughput for all-to-all and uniform random traffic patterns, but its near-worst throughput is lower. We argue that when OCSs are used to construct FC, vendors do not have to worry much about FC's near-worst throughput. By simply generating a different FC's topology, one can avoid matching an FC's topology with its near-worst traffic patterns.

6. We validate that FC is deadlock free using a small test bed and a packet-level simulator, even under extreme (but practical) cases where congestion control is disabled and switches are misconfigured with a very small PFC PAUSE threshold. In contrast, we see deadlocks triggered under shortest-path routing and thus ECMP&KSP routing is not safe.

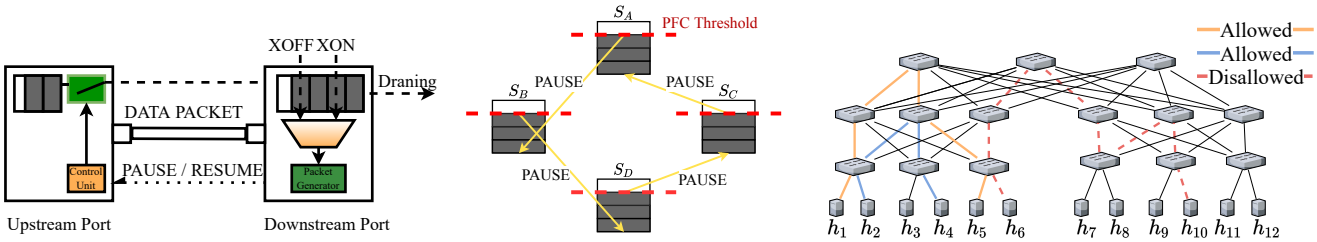
2 Background & Motivation

2.1 Deploy RDMA over Ethernet in Clos

Clos network, a.k.a. fat-tree, was proposed for data center network (DCN) architecture in [3], and has become the de-facto standard for large service providers, such as Google [45], Microsoft [19], Facebook [44], etc. TCP/IP is the dominant transport/network stack in today's data centers. However, the traditional TCP/IP stack cannot offer high throughput ($>$ 40Gbps or more) and ultra-low latency ($<$ 10us per hop) for modern data center applications such as cloud storage, deep learning framework and database [20, 30, 57]. Therefore, data center operators, e.g., Microsoft [20], Alibaba [30], etc., have started large-scale deployment of RDMA in Clos data centers to attain better network performance.

RDMA is a hardware offloading technology that offers several benefits such as high throughput, low latency and low CPU overhead by bypassing the host networking stack. HPC community has long used RDMA in special-purpose clusters, and deployed RDMA using Infiniband (IB) technology. However, modern data centers are built with IP and Ethernet technologies. For technical and economical reasons, RoCE was proposed for RDMA deployment in data centers.

The commonly used RoCE protocol is RoCEv2. RoCEv2 encapsulates an RDMA transport packet within a UDP packet to be compatible with the existing networking infrastructure of data centers. RoCEv2 was initially designed to run on a lossless network, which can be guaranteed by enabling the Priority-based Flow Control (PFC) [25]. Admittedly, there have been advanced RoCE designs, e.g., Resilient RoCE, IRN [35], etc., that could work with a lossy network. However, supporting RoCE in lossy networks requires handling packet retransmission using time out, selective ack, etc., which may not only complicate the NIC design, but also hurt network latency and throughput performance. As a result, lossy RDMA may not be able to substitue lossless RDMA in all cases. In this paper, we focus on lossless networks to support RoCEv2.



(a) Priority-based flow control. (b) PFC-induced deadlock. (c) Up-Down routing in a Clos network.

Figure 1: Key technologies of supporting RoCE in a lossless Clos network.

2.1.1 Priority-based Flow Control (PFC)

PFC is a hop-by-hop flow control approach to prevent switch buffer overflow, which is the primary cause of packet loss in data centers. As shown in Fig. 1(a), the downstream switch sends a PAUSE frame to its upstream switch when its ingress queue length exceeds a certain threshold (**XOFF**). The upstream switch stops transmission after receiving the PAUSE frame. A RESUME frame is sent when the downstream queue drains below another threshold (**XON**). It takes some time for the upstream switch to react to the PAUSE frame and stop transmission. So the downstream switch needs to reserve some buffer space to accommodate the packets sent by the upstream switch during this time. The buffer space is called *headroom*. PFC can guarantee zero packet loss when the headroom size is configured correctly. Typically, data center switches can support at most two or three lossless priorities [20] due to the buffer size limit. Although the switch buffers keep increasing, the data center link bandwidth has been increasing much faster and the buffer/bandwidth ratio is actually decreasing over time [18]. Hence, we believe that supporting more lossless priorities can be even more difficult for the foreseeable future.

2.1.2 PFC-induced Deadlocks

PFC can raise some performance issues such as unfairness, PFC storms and deadlocks [14, 20, 30, 57]. Specifically, the PFC-induced deadlocks may hinder the large-scale deployment of RoCEv2. When cyclic buffer dependency (CBD) exists, deadlocks can be triggered by PFC PAUSEs [23], causing packets to wait indefinitely for buffer resources [48]. As shown in Fig. 1(b), four switches S_A, S_B, S_C, S_D have reached the PFC threshold and start to send PAUSE frames; then the network is trapped into a deadlock and no switch can make any progress. Note that, the PFC-induced deadlock cannot go away once it occurs even if we restart all the servers.

Deadlock recovery is a common approach to combat deadlocks. It contains two steps: deadlock detection and deadlock resolution. Traditional approaches detect deadlocks in the control plane [34]. However, these solutions cannot react to deadlocks quickly enough due to the large communication latency between data planes and control planes. A recent work, ITSY [51], could detect deadlocks in the data plane and achieve at least 3.2× faster detection speed. However, ITSY requires programmable switch hardware (e.g., P4) support. As for deadlock resolution, temporary rerouting [34] is a common approach, but may create new congestion and deadlocks. ITSY [51] tried to resolve deadlocks completely in the data plane without rerouting, but the proposed solutions either incur packet loss or cannot efficiently handle concurrent deadlocks. To sum up, existing deadlock recovery mechanisms are not ideal. As a result, deadlock prevention has received much attention in the recent literature.

2.1.3 Avoiding Deadlocks in Clos Networks

Large vendors have gained years of experience in deploying RDMA in Clos data centers [20]. The following strategies are adopted to avoid deadlocks:

1. Perform up-down routing, which is CBD and deadlock-free under normal network conditions in Clos networks. (Note that containing a CBD is a necessary condition to have deadlocks.) As shown in Fig. 1(c), the paths of $h_1 \rightarrow h_5$ and $h_2 \rightarrow h_4$ obey the "up-down" rule and are allowed; but the path of $h_6 \rightarrow h_{10}$ contains a "down-up" segment and thus is not allowed.

2. Do not put multicast and broadcast packets into lossless classes. It was reported in [20] that ARP broadcasts+up-down routing can cause PFC deadlocks.

3. Use a different lossless class for rerouted packets upon network failures. [24] shows that packet rerouting may break the "up-down" rule and trigger PFC deadlocks.

2.2 From Clos to Expander

Despite of the success of deploying Clos data centers, however, a Clos network is inherently suboptimal in terms of bandwidth provision. As the Ethernet speed keeps increasing, the network cost, especially the power consumption of Clos networks, is becoming prohibitively high [4]. To reduce

the network cost, researchers have started seeking for more cost-effective network architectures.

One of the promising alternative for DCNs is expander graph. As shown in Fig. 2, expander graphs adopt a flat topology design: servers connect to ToR switches and these ToRs are directly interconnected without a layered structure. Examples of expander graphs include Jellyfish [46], SlimFly [5], Xpander [50], FatClique [54], etc. Expander graphs is more cost-effective for bandwidth provision than Clos networks. Using KSP routing, a full throughput expander uses 25% fewer switches than a full throughput Clos [36]. The network performance of expander graphs was also studied under other routing protocols, including ECMP, Valient Load Balancing (VLB) and a hybrid of the two [28]. However, none of these widely studied routing strategies is CBD-free.

2.2.1 ECMP/KSP are not CBD-free in Expanders

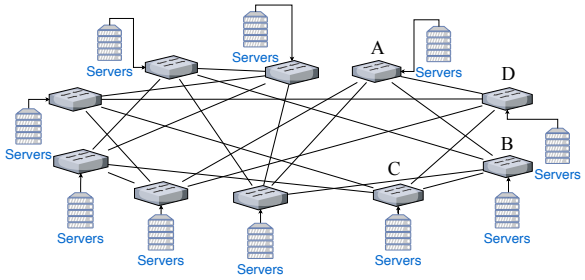


Figure 2: An expander graph.

Consider the expander graph in Fig. 2. This expander is a random regular graph with 4 inter-ToR links per ToR. Consider the four ToRs A, B, C, D and the shortest paths for $A \to C$, $B \to D$, $C \to A$, $D \to B$. Assume that there is a flow routed along the shortest path $A \to B \to C$. Then, if the egress port at link (B,C) is paused, the egress port at link (A,B) will be paused. If there is another flow routed along the shortest path $D \to A \to B$, since the egress port at link (A,B) is paused, the egress port at link (D,A) will also be paused. Similarly, if we have another two flows routed along the shortest paths $C \to D \to A$ and $B \to C \to D$, then the egress ports at link (C,D) and link (B,C) will be paused. Now, we find a CBD in this expander graph under shortest-path routing. To sum up, when there are 4 flows routed along the paths $A \to B \to C, D \to A \to B, C \to D \to A$ and $B \to C \to D$, if one of the egress ports $(A,B),(B,C),(C,D),(D,A)$ is paused for a sufficiently long time, a deadlock will be triggered.

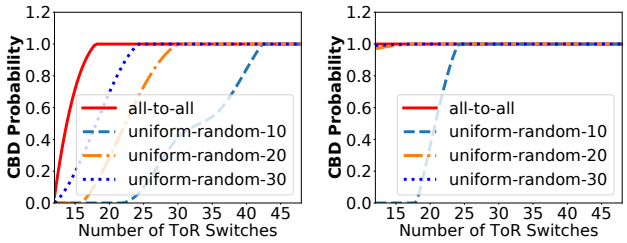
The above analysis indicates that shortest-path routing is not CBD-free. Now we consider ECMP and KSP routings. ECMP uniformly split traffic among all the shortest paths, while KSP split traffic among the first K shortest paths. (To improve network performance under ECMP/KSP, one can also optimize the path weights using a multi-commodity flow formulation.) Under ECMP or KSP routing, it is still possible to have four flows taking the paths $A \to B \to C, D \to A \to B, C \to D \to A$ and $B \to C \to D$ in the above example. Therefore, both ECMP and KSP routings are not CBD-free. Using the same approach, we can prove that the VLB routing and the hybrid of ECMP&VLB in [28] are not CBD-free, either.

2.2.1.1 Probability of Containing CBDs

We further analyze the probability of an expander graph containing CBDs under different traffic patterns. We generate two classes of expander graphs, Jellyfish [46] and Xpander [50]. In each expander graph, each ToR switch has 5 ports connected to other ToRs. For each expander graph, we evaluate two classes of traffic patterns under shortest-path routing (the algorithm that determines if a set of paths is CBD free in a given topology is offered in Appendix A.2):

All to All: Every source-destination pair has an on-going flow. This represents the most-likely case of having CBDs.

Uniform Random-p: Every ToR randomly picks p fraction of ToRs to communicate. This represents practical DCN traffic patterns in which the majority of traffic of a server is often destined to a few racks [44].



(a) Jellyfish CBD probability. (b) Xpander CBD probability.

Figure 3: Jellyfish and Xpander CBD analysis.

The results are depicted in Fig. 3. We can see that as the number of ToRs increases, the CBD probability quickly increases to one and Xpander graphs are more likely to encounter CBDs than Jellyfish graphs. Note that even under shortest-path routing (ECMP), the CBD probability becomes one with only tens of ToR switches. Other routing algorithms, including KSP, VLB, etc., contain even higher CBDs.

Remark on the necessity of eliminating CBDs: Even if the probability that the ECMP/KSP/VLB routing policies lead to CBDs is close to 1, the possibility that these CBDs eventually turn into deadlocks may not be that high. Nevertheless, eliminating CBDs can be still important. Some network applications requires five-nines availability, which means that the maximum downtime in a month must be less than 26.3 seconds. As long as the deadlock probability is non-zero, when a data center runs for a long time, a deadlock may be triggered eventually and hurts the overall system availability.

Remark on Tagger's approach: Given any routing paths, Tagger [24] offered a generic approach to eliminate CBDs. The key idea is to break each path into several segments

and assign each segment a lossless priority. As long as the path segments belonging to the same lossless priority are CBD-free, the entire network is CBD-free. Unfortunately, this approach may require too many lossless priorities to eliminate CBDs in large expander networks.

3 Flattened Clos

We propose a new class of expander graphs, called **Flattened Clos (FC)**, for efficient deadlock prevention. Our design is motivated by the CBD-free up-down routing in Clos networks and the *flattened butterfly* topology [29]. FC is a topology built on top of ToR switches. The key idea of FC is to split each ToR switch into a few virtual switches, and assign each virtual switch a virtual layer id. By creating links only between virtual switches in adjacent virtual layers, FC can adopt virtual up-down routing to avoid deadlocks. The detailed topology and routing designs of FC are described below.

3.1 Topology

We study a data center network with N ToR (Top of Rack) switches $S = \{S_1, S_2, ..., S_N\}$. Each switch has $p = s + h$ ports, h of which connected to the hosts and s of which connected to other ToRs. We construct an FC topology in two steps:

Step 1: Splitting Virtual Switches. To create an FC with k virtual layers, we logically split each switch $S_i, i = 1, 2, ..., N$ into k virtual switches, and label these virtual switches as $S_i^1, S_i^2, ..., S_i^k$. The virtual switch S_i^j belongs to the j-th layer, and has l_j number of links to connect to other switches. The total link count of the virtual switches $S_i^1, S_i^2, ..., S_i^k$ is equal to the total link count of S_i that connect to other ToRs, i.e.,

$$\sum_{j=1}^{k} l_j = s. \quad (1)$$

Step 2: Random Wiring. For each $j = 1, 2, ..., k-1$, we randomly generate a bipartite graph between the virtual switches in layer j and the virtual switches in layer $j+1$. Let $a_j, j = 1, 2, ..., k-1$ be the degree of each virtual switch in the j-th random bipartite graph. We must have

$$l_1 = a_1, l_2 = a_1 + a_2, ..., l_{k-1} = a_{k-2} + a_{k-1}, l_k = a_{k-1}. \quad (2)$$

When we generate random bipartite graphs, we never create links between S_i^j and S_i^{j+1} for $i = 1, 2, ..., N, j = 1, 2, ..., k-1$. The reason is that S_i^j and S_i^{j+1} actually belong to the same switch, and there is no need to create a link in between.

3.1.1 Theoretical Topology Properties of FC

In an FC's topology, each ToR switch has s links connected to other ToRs. Thus, FC falls into the category of random regular graphs (RRG). Here, we restate some useful theoretical results for RRGs in literature, which also applies to FC.

We represent a network by $G = (V, E)$, where V is the vertex set and E is the edge set. The bisection bandwidth of G can be characterized by *Edge Expansion*, which is defined as $EE(G) = min_{|S| \leq \frac{N}{2}} \frac{|\partial S|}{|S|}$, where N is the number of vertices in V, S is a subset of V, $|S|$ is the size of S, ∂S is the set of edges leaving S. The Edge Expansion of an s-regular graph is upper bounded by $s/2$ [50]. The following theorem indicates that random regular graphs attain near-optimal edge expansion.

Theorem 1 *(Near-optimal Edge Expansion [7]) For every $s \geq 3$ and $0 < \eta < 1$ satisfying $2^{4/s} < (1-\eta)^{1-\eta}(1+\eta)^{1+\eta}$, almost every s-regular graph G has its edge expansion*

$$EE(G) \geq (1-\eta)s/2.$$

Given a traffic matrix $T = [t_{uv}]$, where t_{uv} is the amount of requested flows from ToR switch u to ToR switch v. The throughput $\alpha(G, T)$ of a network G under the traffic matrix T is defined as the maximum value $\theta(T)$ for which $T \cdot \theta(T)$ is feasible in G. The following two theorems guarantee that random regular graphs achieve good throughput under both uniform and adversarial patterns.

Theorem 2 *(High throughput under all-to-all pattern [50]): For the all-to-all traffic pattern $T_{all\text{-}to\text{-}all}$, almost every s-regular graph G achieves a throughput*

$$\alpha(G, T_{all\text{-}to\text{-}all}) \geq \frac{1}{O(\log s)} \alpha(G^*, T_{all\text{-}to\text{-}all}),$$

where G^* is the s-regular graph that attains the optimal throughput under $T_{all\text{-}to\text{-}all}$.

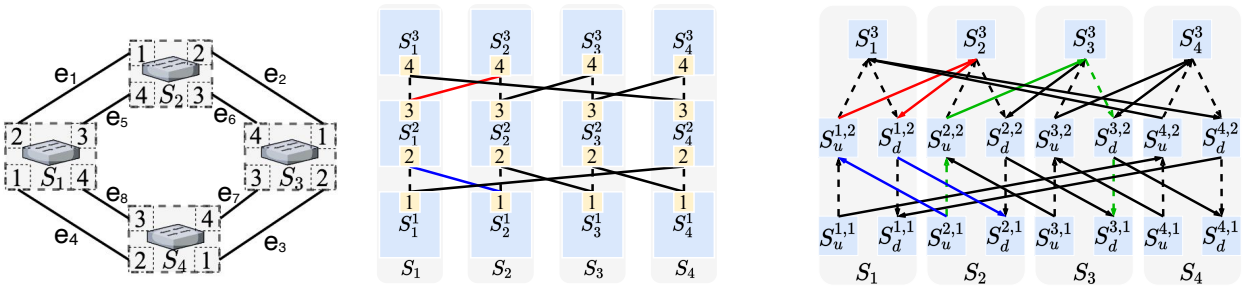
Theorem 3 *(Resilience to adversarial patterns [50]): For almost every s-regular graph G and every traffic pattern T, the throughput $\alpha(G, T) \geq \frac{1}{O(\log N)} \alpha(G^*, T)$, where G^* is the s-regular graph that attains the optimal throughput under T.*

3.2 Routing

3.2.1 Edge-disjoint Virtual Up-down Routing

Although FC's topology exhibits high network throughput in theory, such a throughput may not be achievable in PFC-enabled RoCE networks due to the potential risk of deadlocks. To completely eliminate the risk of deadlocks, we propose the CBD-free **Edge-disjoint Virtual Up-down Routing**. This routing strategy computes paths in three steps:

Step 1: Construct a Multi-layered Virtual Topology. According to the construction of FC's topology, each FC's topology is mappable to a multi-layered topology. Consider the toy example in Fig. 4(a). While we construct this topology, we have virtually divided each ToR switch S_i into $k = 3$ sub-switches S_i^1, S_i^2 and S_i^3. The first port of S_i belongs to S_i^1; the second and the third ports belong to S_i^2; the fourth port belongs to S_i^3. It is easy to check that each edge in Fig. 4(a) corresponds to a solid line in Fig. 4(b). Note that the k sub-switches



(a) Flattend Clos. (b) Virtualized topology. (c) Directed virtual up-down topology.

Figure 4: FC's topology and routing design.

$S_i^1, S_i^2, ..., S_i^k$ can communicate with each other, because they belong to the same physical switch. Hence, we also create an edge between S_i^j and S_i^{j+1} for every $j = 1, 2, ..., k-1$ (see the dashed lines) in Fig. 4(b).

Step 2: Construct a Directed Virtual Up-Down Topology. Our objective is to find the maximum number of virtual up-down paths in the multi-layered virtual topology. To enforce this "up-down" constraint, we further convert the undirected multi-layered graph in Fig. 4(b) to a directed virtual up-down graph in Fig. 4(c). Specifically, we first split each virtual node $S_i^j (j < k)$ into one "up node" $S_u^{i,j}$ and one "down node" $S_d^{i,j}$ in the directed up-down graph. (Note that we do not split the top layer virtual nodes S_i^k.) We then map each link in Fig. 4(b) to two directed links in Fig. 4(c): each undirected link $(S_{i_1}^{k-1}, S_{i_2}^k)$ in the top layer (see the red line in Fig. 4(b) as an example) is mapped to two directed links $(S_u^{i_1,k-1}, S_{i_2}^k)$ and $(S_{i_2}^k, S_d^{i_1,k-1})$; each undirected link $(S_{i_1}^{j-1}, S_{i_2}^j)$ $(j = 2, ..., k-1)$ (see the blue line in Fig. 4(b) as an example) is mapped to two directed links $(S_u^{i_1,j-1}, S_u^{i_2,j})$ and $(S_d^{i_2,j}, S_d^{i_1,j-1})$.

Step 3: Compute CBD-free Paths. For every source-destination pair (S_i, S_j), we first find a path set $\mathcal{P}_{ij}$ with the maximum number of virtual up-down paths from the node $S_u^{i,1}$ to the node $S_d^{j,1}$ in the directed virtual up-down topology using min-cost max-flow (see Appendix A.1 for more details). In this set $\mathcal{P}_{ij}$ of paths, each solid link is used at most once while each dashed link can be used multiple times. Then, for every path $P \in \mathcal{P}_{ij}$, we map it to a path in FC's topology (Fig. 4(a)). For example, as shown in Fig. 4(c), we find one up-down path $S_u^{2,1} \to S_u^{2,2} \to S_3^3 \to S_d^{3,2} \to S_d^{3,1}$ (marked with green) for the source-destination pair (S_1, S_2). Since $S_u^{2,1}$, $S_u^{2,2}$ are from the ToR switch S_2 and S_3^3, $S_d^{3,2}$, $S_d^{3,1}$ are from the ToR switch S_3, this path can be contracted to $S_2 \to S_3$ in the FC's topology. Since each solid link is used at most once in $\mathcal{P}_{ij}$, the resulting paths in the FC's topology must be edge-disjoint.

3.2.2 FC's Routing is CBD Free

In FC's edge-disjoint virtual up-down routing, we first compute an up-down path set $\mathcal{P}_{ij}$ based on the directed virtual up-down topology, and then contract all the paths in $\mathcal{P}_{ij}$ to obtain the final paths for FC's topology. Let $\mathcal{P} = \cup_{i,j}\mathcal{P}_{ij}$ be the set of virtual up-down paths obtained from the directed virtual up-down topology. According to Theorem 8 in Appendix A.2, $\mathcal{P}$ is CBD free. In order to prove that the final set of paths in FC's topology is CBD free, we need the following definition and lemma (see Appendix A.2.1 for the proof).

Definition 1 *Given a set of nodes V, $\{V_1, V_2, ..., V_m\}$ is called a partition of V, if the following conditions are met: 1) $V_{m_1} \cap V_{m_2} = \emptyset$ for every $m_1 \neq m_2$; 2) $V_1 \cup V_2 \cup \cdots \cup V_m = V$.*

Lemma 4 *Given a graph $G(V,E)$, a path set $\mathcal{P}$ and a partition $\{V_1, V_2, ..., V_m\}$ of V, a graph and path set pair $(\hat{G}(\hat{V}, \hat{E}), \hat{\mathcal{P}})$ is called a contraction of $(G(V,E), \mathcal{P})$ if*

1. *every node in $\hat{v}_i \in \hat{V}$ corresponds to the vertex set V_i;*
2. *the number of edges between $\hat{v}_i$ and $\hat{v}_j$ is the same as the total number of edges between V_i and V_j in $G(V,E)$;*
3. *each path $\hat{P} \in \hat{\mathcal{P}}$ is a contraction of a path $P \in \mathcal{P}$, i.e., $\hat{P}$ is obtained by first replacing each vertex in P by a vertex in $\hat{V}$ and then removing cycles and duplicated vertices.*

Then, if the path set $\mathcal{P}$ is CBD-free in $G(V,E)$, the path set $\hat{\mathcal{P}}$ must be CBD-free in $\hat{G}(\hat{V}, \hat{E})$.

Apparently, FC's topology and routing path set can be viewed as a contraction of the directed virtual up-down topology and the corresponding virtual up-down path set $\mathcal{P}$. Since the path set $\mathcal{P}$ is CBD free in the directed virtual up-down topology, then according to Lemma 4, we immediately know that FC's routing path set is CBD free.

3.2.3 How Routing Affects FC's Topology Design?

We have described FC's routing and topology designs. Note that there is a critical parameter k in the design. If k is not properly chosen, FC's routing policy may not be able to find a path for some switch pair, thus hurting the connectivity of FC. In this section, we offer a theoretical guideline to determine the number of virtual layers in FC.

Number of Switches	Number of Servers	k_{min}	k	Average Number of Edge Disjoint Up-down Paths	Average Path Length of Edge Disjoint Up-down Paths	Minimum Number of Edge Disjoint Up-down Paths
500	12000	3	3	10.05	4.29	4.00
			4	16.08	4.57	12.00
1000	24000	3	3	7.10	4.43	1.00
			4	14.01	4.86	9.00
2000	48000	4	4	11.85	5.13	7.00
			5	15.77	5.36	11.00
3000	72000	4	4	10.63	5.30	6.00
			5	14.66	5.54	10.00
5000	120000	4	4	9.17	5.52	3.00
			5	13.28	5.75	9.00

Table 1: Choosing the right k for FC.

Lemma 5 *Let x be the number of ancestors in the virtual layer k for each layer-1 virtual node. If $x > \sqrt{(2+\varepsilon)N\ln N}$, where $\varepsilon > 0$ is an infinitesimal value, then as $N \to +\infty$, with probability 1, every pair of layer-1 virtual nodes has a common ancestor in the virtual layer k.*

Proof 1 *We use A_{ij} to denote the event that the virtual nodes S_i^1 and S_j^1 have no common ancestor in the virtual layer k. Then, the probability that A_{ij} happens is*

$$\begin{aligned} P(A_{ij}) &= \frac{C_{N-x}^x}{C_N^x} \leq (1-\frac{x}{N})^x \\ &< \left(1-\frac{\sqrt{(2+\varepsilon)\ln N}}{\sqrt{N}}\right)^{\sqrt{(2+\varepsilon)N\ln N}} \\ &= \left(\left(1-\frac{\sqrt{(2+\varepsilon)\ln N}}{\sqrt{N}}\right)^{\frac{\sqrt{N}}{\sqrt{(2+\varepsilon)\ln N}}}\right)^{(2+\varepsilon)\ln N} \\ &< (1/e)^{(2+\varepsilon)\ln N} = N^{-(2+\varepsilon)}. \end{aligned}$$

Let A be the event that at least one pair of virtual nodes in layer 1 has no common ancestor in layer k. Then,

$$\begin{aligned} P(A) &= P(\cup_{i\neq j}A_{ij}) \leq \sum_{i\neq j} P(A_{ij}) \\ &= \frac{N(N-1)}{2} \times N^{-(2+\varepsilon)} < \frac{1}{2}N^{-\varepsilon}. \end{aligned}$$

Then, $\lim_{N\to+\infty} P(A) = 0$. This completes the proof.

Based on FC's routing, it is easy to calculate that the number of distinct up-paths from a virtual node in layer 1 is $(a_1+1)(a_2+1)\cdots(a_{k-1}+1)$, which is an upper bound of the number of ancestors in layer k. According to Lemma 5, we can choose a k such that

$$(a_1+1)(a_2+1)\cdots(a_{k-1}+1) > \sqrt{(2+\varepsilon)N\ln N}. \quad (3)$$

According to Equation (1) and (2), it is easy to obtain $\sum_{i=1}^{k-1} a_i = s/2$. We could choose $a_1, a_2, ..., a_{k-1}$ to maximize the left hand side of (3), and obtain

$$\left(1+\frac{s}{2(k-1)}\right)^{k-1} > \sqrt{(2+\varepsilon)N\ln N}. \quad (4)$$

Let $k_{\min}$ be the smallest integer solution of (4). We could choose a k value around $k_{\min}$. As shown in Fig. 5, $k_{\min}$ does not grow fast with respect to N.

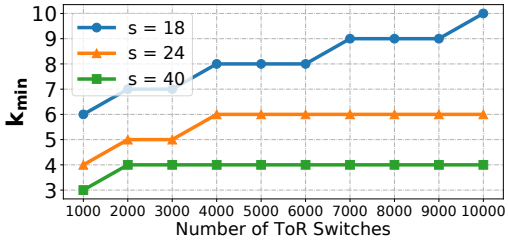


Figure 5: Relationship between $k_{\min}$ and network size N.

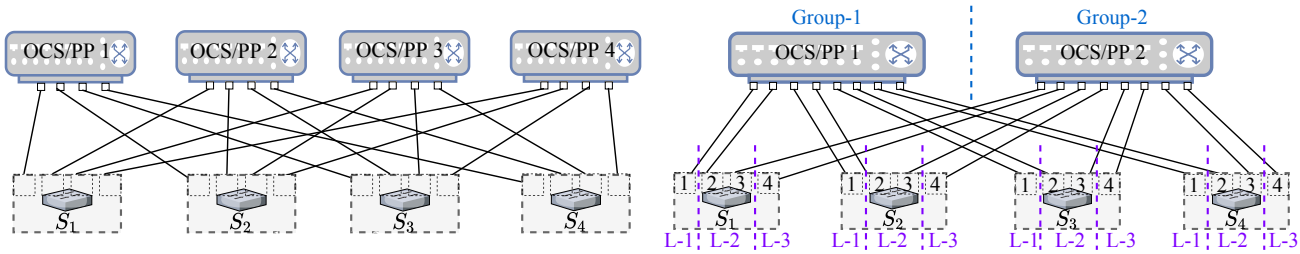
Numerical Verification: To verify the above theoretical result, we perform a numerical analysis using 64-port ToR switches. Each ToR switch has $h = 24$ ports connected to the hosts and $s = 40$ ports connected to other ToR switches. For different number of ToR switches ($N = 500/1000/2000/3000/5000$), we choose $k = k_{\min}, k_{\min}+1$, generate an FC's topology and count the number of distinct virtual up-down paths. As shown in Table 1, as we increase k, more paths can be found for every source-destination ToR switch pairs. Note that the average path length increases with respect to k. Hence, it is better to choose a smaller k. On the other hand, if we choose k to be too small, some ToR switch pairs may not have sufficient number of distinct paths. Here we suggest a simple strategy that works well for FC:

Strategy (*): Try $k_{\min}$ first; if not working, try $k_{\min}+1$.

For example, in the case where $N = 1000, k = k_{\min} = 3$, the minimum number of distinct paths between ToR pairs is 1. This creates a bottleneck in the network. Hence, $k = k_{\min} + 1 = 4$ will be chosen instead.

3.2.4 Computational Complexity of FC's Routing

The main complexity comes from using the min-cost max-flow solver to find edge-disjoint virtual up-down paths. Given a graph $G = (V, E)$ with n vertices and m edges, the computational complexity of the min-cost max-flow algorithm



(a) Uniform cabling through OCSs/PPs. Any inter-ToR topology is realizable by properly configuring the s OCSs/PPs one by one.

(b) Virtual-layered cabling through OCSs/PPs. There is 1 port for virtual layer-1 (L-1), 2 ports for L-2, and 1 port for L-3 in every switch. OCSs/PPs are divided into 2 groups.

Figure 6: Example of cabling with the help of optical circuit switches (OCS) / patch panels (PP).

implemented in Ortools is $O(m \cdot n^2 \cdot \log(n \cdot C))$, where C is the value of the largest link cost in the graph [1] (in our case, $C = 1$). If we choose the parameters k and $a_1, a_2, ..., a_{k-1}$ such that $(a_1+1)(a_2+1)\cdots(a_{k-1}+1) = \Theta(\sqrt{N \log N})$, each virtual node in the first virtual layer will be able to reach at most $\Theta(k\sqrt{N \log N})$ virtual nodes through at most $\Theta(k\sqrt{N \log N})$ edges. When we compute edge disjoint paths from S_i to S_j, we only need to focus on a subgraph of the directed virtual up-down topology, which contains all the nodes reachable from $S_u^{i,1}$ and $S_d^{j,1}$. This subgraph has $\Theta(k\sqrt{N \log N})$ nodes and edges. Thus, the overall computational complexity is $\Theta((k\sqrt{N \log N})^3 \cdot \log(k\sqrt{N \log N})) = \Theta(k^3 N^{3/2} (\log N)^{5/2})$.

3.3 Cabling

FC adopts random wiring for its topology design. However, random wiring has long been criticized for its high cabling complexity [50, 54]. Indeed, if we directly connect different ToR switch pairs, the number of distinct fiber lengths would be in the order of $\Theta(N^2)$. Directly connecting ToR switches could also increase the management complexity when we perform data center expansion [56].

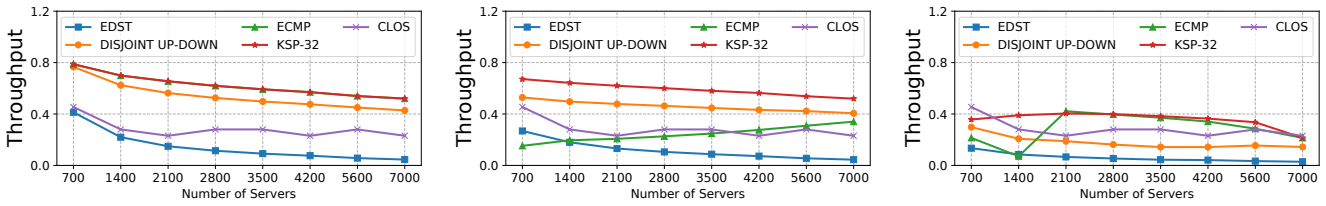
To reduce cabling complexity, motivated by TROD [9] and Google's Jupiter data center [40], we propose to use a set of co-located optical circuit switches (OCS) or patch panels (PP) to interconnect different ToR pairs and form FC's topology. Since these PPs/OCSs are co-located, the number of distinct fiber lengths reduces to $\Theta(N)$. Next, we offer two strategies to interconnect PPs/OCSs with ToR switches.

Uniform Cabling (see Figure 6(a)): Note that each ToR switch has s ports to be connected to other ToR switches. We use s OCSs/PPs, and construct a uniform bipartite graph between ToR switches and OCSs/PPs. Under this cabling strategy, it was proven in [55] (see Lemma 4 and Theorem 5 therein) that any inter-ToR topology is realizable by properly configuring the s OCSs/PPs one by one. According to this fact, we could first generate an FC topology without considering the layer of OCSs/PPs, and then decomposite this topology into s sub-topologies that can be mapped to each OCS/PP. This approach reduces cabling complexity. However, it encounters scalability challenge. Specifically, the port count of the commercially available OCSs/PPs is on the order of a few hundred. For example, a Calient s320 OCS [8] can offer 320 TX/RX ports. Thus, the number of ToR switches can be at most a few hundreds. Since each ToR switch typically connects to tens of servers, this uniform cabling strategy can support at most a few thousands of servers.

Virtual-Layered Cabling (see Figure 6(b)): Note that FC's topology is designed based on the concept of virtual layers. Assume that there are a_1 ports for layer-1, $a_1 + a_2$ ports for layer-2, ..., $a_{k-1} + a_k$ ports for layer-$(k-1)$, and a_k ports for layer-k. We group all the OCSs/PPs into $k-1$ groups, and connect $2a_i$ ports of each ToR switch to the i-th OCS/PP group. In the i-th OCS/PP group, each OCS/PP have half of its ports connected to ToR switches' layer-i ports and half of its ports connected to ToR switches' layer-$(i+1)$ ports. If we enforce that every OCS/PP should connect to all the ToR switches, we will encounter the same scalability challenge as the uniform cabling strategy. In the virtual-layered cabling strategy, $2\eta a_i$ number of OCSs/PPs are used in the i-th group, and each ToR switch will randomly choose $2a_i$ OCSs/PPs in group-i to connect its layer-i and layer-$(i+1)$ ports. Under this cabling strategy, the total number of ToR switches that can be supported becomes "$\eta \times$ port count of an OCS/PP". This strategy scales well. For example, if we use 320-port OCSs, 64-port ToR switches (assume that each ToR connects to 24 servers), and choose $\eta = 20$, then the maximum number of servers would be $20 \times 320 \times 24 = 153600$, which can definitely support a large-scale data center.

Remark on the parameter η: When $\eta > 1$, a cabling constraint is imposed to FC when we generate the topology between adjacent virtual layers, i.e., not all topologies are realizable because the interconnection between ToRs and each group of OCSs/PPs is not uniform. Fortunately, Appendix A.5.1 shows that enabling this cabling constraint when generating FC's topology has little impact on FC's routing statistics.

Remark on the network cost: Compared to traditional expander graphs, having a layer of OCSs in FC reduces the



(a) Throughput of the all to all traffic matrix. (The ECMP and KSP curves overlap together.)

(b) Throughput of uniform random traffic matrices (averaged over 10 runs).

(c) Throughput of the near-worst permutation traffic matrix.

Figure 7: Throughput simulation results under ECMP, EDST, Edge-disjoint Virutal Up-down and 32-way KSP routing.

number of distinct cable lengths, but unfortuanately increases the total cable length. We compare the total network cost between FC and Clos in Appendix A.4. To achieve similar bisection bandwidth, FC and Clos have similar network cost when the network size is small and FC's cost becomes lower as the network size increases to a point where 4 switch layers are required to build a Clos. In addition, FC uses fewer number of electrical switches and thus its network power consumption is lower.

A potential future direction: Using OCSs introduces another interesting problem: how to design deadlock-free and traffic-aware topology & routing policies. As reconfiguring OCSs incurs non-negligible delay, it may not be possible to reconfigure OCSs for every traffic-pattern change. Google's Jupiter data center [40] and our prior work [9, 49] both showed that low-frequency reconfiguration might be sufficent, because the traffic patterns in real data centers do not change arbitrarily. Low-frequency reconfiguration may also work for lossless RDMA networks, but requires further investigation.

4 Numerical Throughput Analysis

We numerically evaluate the throughput for FC in this section. We evaluate two scenarios. Due to space constraints, we only present one here and put the other one in Appendix A.5.2.

We generate FC's topologies of different sizes using up to 500 32-port ToR switches. Each ToR switch has 18 ports connected to other switches and 14 ports connected to servers. The number of virtual layers k is chosen based on the strategy (*) in Section 3.2.3. For each FC's topology, we evaluate four routing strategies: 1) FC's edge-disjoint virtual up-down routing, 2) EDST routing, 3) ECMP or Shortest-Path routing, and 4) KSP routing. Given a traffic matrix T, we use a multi-commodity flow formulation to calculate the maximum throughput value $\theta(T)$ such that $T \cdot \theta(T)$ is feasible under the given topology and routing paths. (For ECMP, the throughput is also calculated based on the multi-commodity flow formulation. Evenly spreading traffic among all the shortest paths may yield very poor throughput.) In addition, for each FC's topology, we also compare it with a Clos network generated using roughly the same number of switches with throughput optimized (see Appendix A.3).

We compute throughput values under all-to-all traffic patterns, uniform random traffic patterns and near-worst traffic patterns. In an all-to-all pattern, each server sends an equal amount of traffic to all other servers. In a uniform random pattern, each ToR randomly picks 10% of ToRs to communicate. To generate near-worst patterns, we 1) first construct a complete bipartite graph B with N source nodes and N destination nodes, where the weight of the edge (s,d) is the length of the shortest path from ToR s to ToR d; 2) and then find the permutation matrix with the maximum weight. This approach was also adopted in [26, 36] to generate near-worst patterns. We believe that the above three classes of traffic patterns offer an adequate coverage of real data center traffic patterns. The uniform random pattern is highly representative in real data centers. Indeed, Google's data center traffic patterns are approximately uniform random [40]. The all-to-all pattern is widely used in MPI communication. The near-worst pattern allows us to understand network's performance lower bound.

4.1 FC's Routing vs EDST Routing

The EDST routing is CBD-free for expander graphs. A random s-regular graph has $s/2$ edge-disjoint spanning trees with high probability [38]. Thus, EDST is a direct competitor of the Edge-disjoint Virtual Up-down Routing for FC's topology.

As shown in Fig. 7, FC's edge-disjoint virtual up-down routing (denoted by "DISJOINT UP-DOWN") performs consistently better than EDST for all the traffic patterns. When the network is small ($N = 50$), FC's routing achieves $2\times$ throughput of the EDST routing. As the network size increases, the performance of the EDST routing deteriorates quickly. When $N = 500$, the performance gain of FC's routing becomes $10\times$ and the gain keeps increasing with the network scale.

There are two reasons that lead to the poor performance of the EDST routing. First, existing edge-disjoint spanning tree (EDST) algorithms [42, 43] can find the maximum number of spanning trees, but there is no guarantee that the height of each spanning tree found is small. When we perform routing in a *tall* spanning tree, the average hop count would be large. This is also justified in the following routing-path analysis.

Number of Switches	Number of servers	k	Port Count of Virtual Switches	Routing	Average Number of Paths	Average Path Length	Average Shortest Path Length
50	700	4	[3, 6, 6, 3]	Edge Disjoint Up-down	8.02	3.86	2.68
				EDST	9.00	7.69	3.12
100	1400	4	[3, 6, 6, 3]	Edge Disjoint Up-down	6.43	4.22	3.04
				EDST	9.00	10.01	4.04
200	2800	4	[3, 6, 6, 3]	Edge Disjoint Up-down	4.99	4.56	3.44
				EDST	9.00	14.01	5.50
300	4200	4	[3, 6, 6, 3]	Edge Disjoint Up-down	4.24	4.75	3.70
				EDST	9.00	16.70	6.52
500	7000	5	[2, 4, 4, 5, 3]	Edge Disjoint Up-down	4.55	5.22	4.00
				EDST	9.00	21.96	8.14

Table 2: Edge-Disjoint Virtual Up-down Routing vs. the EDST Routing (32-port Switches are Used).

Second, some links remain unused in the EDST routing. In an expander graph with N ToR switches, each spanning tree contains $N-1$ links. Note that the total number of ToR-to-ToR links is $Ns/2$. When $Ns/2$ is not divisible by $N-1$, there must be links not used by any spanning tree.

Routing-Path Analysis: For several FC's topologies of different sizes ($N = 50/100/200/300/500$), we analyze the routing paths under FC's routing and the EDST routing. We calculate three metrics, including average number of paths, average length of paths and average length of the shortest paths. As shown in Table 2, although the EDST routing could find more paths than FC's routing, its average path length is much higher. When $N = 50$, the average path length under FC's routing is $1-3.86/7.69 \approx 50\%$ lower than that under the EDST routing. As N increases to 500, the reduction of average path length becomes $1-5.22/21.96 \approx 76\%$. We expect that this number will continue to increase for larger networks. The EDST routing cannot guarantee a small routing path length. In contrast, the parameter k restricts that FC's routing path length cannot exceed $2k$ and k increases slowly with N.

4.2 FC's Routing vs ECMP/KSP Routing

ECMP/KSP are widely-used routing protocols for expander graphs. In FC's topology, ECMP's throughput fluctuates significantly because ECMP cannot provide enough path diversity; KSP's throughput is more stable under different traffic patterns. This coincides with the findings in Jellyfish [46].

Fig. 7 shows that KSP's throughput is consistently higher than that of the FC's edge-disjoint virtual up-down routing. However, deploying KSP routing in expander networks poses a deadlock risk. We have shown in Section 2.2.1.1 that the probability that ECMP/KSP routing contains CBDs is close to 1. Although containing CBDs is not sufficient to trigger deadlocks, we will show in Section 6.1 that ECMP/KSP could indeed trigger deadlocks in certain cases in a real testbed.

How to close the throughput gap: FC uses only one lossless queue, and its throughput performance is lower than that of the KSP routing. The reason is that FC's routing has lower path diversity than the KSP routing. To improve path diversity, we could let FC use more than one lossless queues. We will explore this further in our future work.

4.3 FC vs Clos

Clos is the de facto standard topology for data centers and has witnessed the successful deployment of RDMA in production [20]. To ensure fair comparisons, given an FC's topology with N ToR switches and H servers, we choose a Clos network that offers the maximum throughput to the H servers using roughly the same number of switches (Appendix A.3).

As shown in Fig. 7(a) and 7(b), FC attains $1.1-2\times$ the throughput of Clos networks. Note that there is a decrease in throughput when the network size changes from 700 to 1400. The reason is that when the switch port count is 32, we can build a two-layered Clos to support 700 servers, but at least 3 layers are required in order for a Clos to support 1400 servers.

However, under near-worst traffic patterns, Fig. 7(c) shows that FC's throughput can be $15\%-50\%$ lower than that of the Clos networks. We argue that this issue can be resolved when a layer of OCSs is used to interconnect different ToRs. If the real traffic pattern is close to a near-worst pattern of the current topology, we can reconfigure the OCSs to generate a topology that matches this traffic pattern. Then, a natural question arises. How frequent should we reconfigure the topology? Certainly, the answer to this question depends on the traffic patterns. If the traffic patterns exbihit some long-term stability [40], occasional reconfigration might be sufficient. We will study this problem further in our future work.

5 Packet-Level Simulation

We cross-validate our throughput analysis using a packet-level simulator [22]. We generate an FC's topology using 144 32-port switches. Each switch has 8 ports connected to hosts and 24 ports connected to other switches. In total, there are 1152 hosts. On top of this topology, we run FC's routing or the EDST routing. We also generate a Clos network using 148 32-port switches with throughput optimized. This Clos network has 64 ToR switches, 56 aggregation switches and 28 spine switches. Each ToR has 18 ports connected to hosts and 14 ports connected to the aggregation switches. The toal number of hosts is still 1152. For this Clos topology, we use up-down routing. The port speed is set as 25Gbps.

We generate three sets of flows based on the all-to-all traffic pattern, a uniform random traffic pattern (each ToR choose 12.5% of ToRs to communicate) and the near-worst traffic pattern. In Facebook's data centers, the median link utilization varies between 10% to 20% and the busiest 5% utilization of links is between 23% to 46% [44]. Here, we set the network load as 0.3, meaning that the maximum ingress/egress traffic of each ToR is 0.3 × Number of Hosts per ToR × 25Gbps. For all the flows, we enable DCQCN for congestion control. We adopt dynamic PFC threshold such that the PFC is triggered when an ingress queue consumes more than 11% of the free switch buffer as suggested by HPCC [30]. We evaluate performance based on the flow completion time (FCT).

We summarize the FCT results in Table 3. FC attains higher throughput under the all-to-all pattern and the uniform random patterns. Correspondingly, FC achieves lower FCT in the packet-level simulation. FC's near-worst-case throughput is lower than that of Clos. But Fortunately, FC has lower average hop count and thus its FCT performance is not much worse. More detailed results are available in Appendix A.5.4.

6 Formation and Impact of Deadlocks

We study how to trigger deadlocks and understand the impact of deadlocks via both testbed experiments and simulations. We will show that under extreme but practical cases, FC's edge-disjoint virtual up-down routing is still deadlock-free; but ECMP/KSP could trigger deadlocks.

6.1 Trigger Deadlocks in a Real Testbed

We build a small testbed using four switches, each with 8 50Gbps ports. (The four switches are virtualized from a single CE12800 switch. The original port speed is 100Gbps and we limit the port speed as 50Gbps.) This testbed has 16 servers, each equipped with one Mellanox CX5 NIC with maximum rate configured as 50Gbps. (We use PCIE-3.0×8 to connect to the NICs, and thus these NICs cannot run at a rate higher than 64Gbps.) Each switch in this testbed has four ports connected to other switches and four ports connected to four servers. We virtually split each switch into 3 virtual switches, with $1,2,1$ number of ports respectively. The connections between FC's virtual switches are shown in Fig. 4(b), and the resulting topology is shown in Fig. 4(a). This topology can be also viewed as a subgraph of a large expander graph (see switches A,B,C,D in Fig. 2). If a deadlock occurs in this subgraph, a PFC storm will quickly propogate to the entire network.

We implement ECMP, edge-disjoint virtual up-down and EDST routings using ACL rules in our testbed. We enable PFC to guarantee that the network is loss-free. The PFC-pause threshold $XOFF$ is set to 50KB and the PFC-resume threshold XON is set to 47KB. Note that these PFC thresholds are lower than the recommended values. This allows the network to trigger more PFC pauses. As we will see shortly, the virtual up-down routing is deadlock-free even in this extreme situation. Note that this setup can be viewed as a misconfiguration of network switches. Microsoft reports that switch misconfiguration accounts for 38% of the high-impact failures in their data centers [52]. In a PFC-storm incident reported also by Microsoft [20], a switch parameter was misconfigured such that PFC PAUSE frames could be triggered more easily.

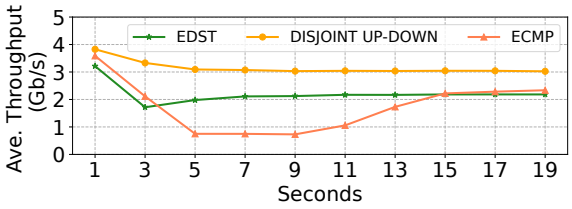


Figure 8: Average throughput of the testbed experiment.

We generate RoCEv2 traffic using the "ib_write_bw [31]" command. For every NIC, we establish an RDMA connection with every NIC under a different ToR switch. For example, NIC1 under the first ToR switch sends traffic to NIC5, NIC6, ..., NIC16. In total, we establish $16 \times 12 = 192$ RDMA connections. We configure the "- -run_infinitely" parameter at the client side of each connection to run the test indefinitely until interrupted by external.

Results: In the first experiment, we apply ECMP routing. ECMP is not CBD-free in this testbed. We see a deadlock after running our testbed for just a few seconds. (KSP typically generates more paths than ECMP, and thus KSP could also trigger deadlocks.) When deadlock happens, a large number of RDMA connections are broken. We deep dive into the source code of "ib_write_bw" to understand why many connections are tear down abnormally. We found that the PFC-deadlocks cause the verbs API "ibv_post_send" to fail and return an error code to the main program of "ib_write_bw". Once the main program catch the exception code, "ib_write_bw" will stop sending traffic and clean up the resources. Note that, if we use dynamic PFC thresholds or use the recommended values to set static PFC thresholds ($XOFF = 800KB, XON = 797KB$), we could not observe PFC deadlocks under ECMP routing. However, this does not eliminate the deadlock risk for ECMP.

In the second and third experiments, we set up the edge-disjoint virtual up-down and the EDST routing respectively to run the same test. In this case, we do not see any deadlock even under low PFC pause/resume thresholds and all RDMA connections can work continuously. This experiment demonstrate that both the virtual up-down routing and the EDST routing can avoid PFC-deadlock in lossless Ethernet.

Finally, we track the average throughput for all the 192 RDMA connections under different routing strategies over one minute, and plot the results in Fig. 8. The virtual up-down routing attains the highest average throughput, which is about 50% higher than that of the EDST routing. Under ECMP routing, the average throughput drops quickly at the

Network Setup	Num. of Hosts	Num. of Switches	Copper / Fiber Cable (km)	Num. of Transceivers	Network Cost (Million $)	All-to-All (load= 0.3)			Uniform Random (load= 0.3)			Near-Worst (load= 0.3)		
						Tput	P50 FCT	P99 FCT	Tput	P50 FCT	P99 FCT	Tput	P50 FCT	P99 FCT
FC	1152	144	2.3 / 55.5	3456	13.98	1.49	6.11	32.03	1.25	4.30	18.62	0.55	35.48	121.55
FC+EDST	1152	144	2.3 / 55.5	3456	13.98	0.48	12.42	99.88	0.39	196.46	509.73	0.16	7081.40	9889.53
Clos	1152	148	2.3 / 10.8	3584	10.77	0.78	11.65	44.68	0.78	6.95	39.55	0.78	42.67	118.74

Table 3: FCT Results vs. Throughput Analysis. ("Tput" is short for "Throughput".)

first 5 seconds due to PFC deadlocks. Although the average throughput of ECMP increases after deadlock recovery, 66% of the RDMA connections have already failed.

6.2 Understanding Deadlocks via Simulation

We perform packet-level simulation to understand how a deadlock is triggered. We use the same testbed topology (Fig. 4(a)) in our simulation. We generate 192 flows at time 0, and set all the flow sizes as 100MB. For different flows, we either disable congestion control or enable DCQCN for congestion control. When DCQCN is enabled, we set the ECN-marking related parameters as $Kmin = 5KB, Kmax = 200KB, P_{\max} = 0.01$ as suggested by the DCQCN paper [57]. To simulate the extreme cases where lots of PFC pauses are triggered, we set a small PFC-pause threshold and a small PFC-resume threshold ($XOFF = 50KB, XON = 47KB$).

We evaluate ECMP and FC's edge-disjoint virtual up-down routing. For adjacent switch pairs, there are two paths under both ECMP and FC's edge-disjoint virtual up-down routing. For non-adjacent switch pairs, there are 8 shortest paths (4 clock-wise paths and 4 counter-clock-wise paths) under ECMP routing and 2 edge-disjoint virtual up-down paths (1 clock-wise path and 1 counter-clock-wise path) under FC's routing. We assign a path to each flow using two strategies:

Balanced Allocation: There are 16 flows generated between every switch pair and we assign the same number of flows to each path under both routing strategies. In this case, every link between adjacent switch pair is shared by exactly 16 flows.

Imbalanced Allocation: Flows between adjacent switch pairs are still equally assigned to all the paths; but flows between non-adjacent switch pairs are only assigned to the clock-wise paths. This situation could happen due to hashing imbalance. In this case, every clock-wise link is shared by 24 flows, which becomes the bottleneck of the network. Incast can thus happen at the 4 switches. In addtion, under ECMP routing, the following paths $\{[e_1, e_2], [e_2, e_3], [e_3, e_4], [e_4, e_1]\}$ form a CBD (actually there are more CBDs), which makes ECMP prone to deadlocks.

Results: Under balanced allocation, we do not see deadlocks even if we use a small static PFC threshold and disable DCQCN. In Fig. 9, we compare the CDFs of the FCTs (Flow Completion Time) under both ECMP and FC's edge-disjoint virtual up-down routing with and without DCQCN. Both routing strategies yield similar FCT performancce.

Under imbalanced allocation, FC's routing can still finish all the flows and the FCT performance is shown in Fig. 9.

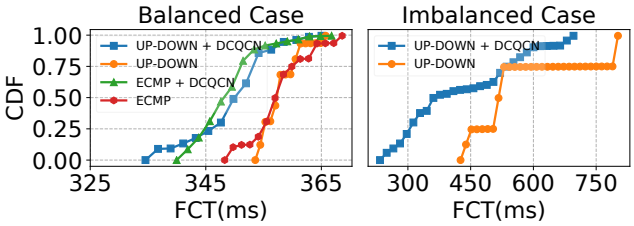


Figure 9: CDF of the FCTs of ECMP and FC's routing under balanced/imbalanced allocation in the testbed topology.

In contrast, the ECMP routing triggers a deadlock even if we enable DCQCN. To rootcause this issue, we record all the PFC pauses and PFC resumes. We find 4 critical PAUSE signals that lead to the deadlock: 1) at time 531 us, S_1 sends a PAUSE to the link e_4; 2) at time 537 us, S_4 sends a PAUSE to the link e_3; 3) at time 543 us, S_3 sends a PAUSE to the link e_2; 4) at time 552 us, S_2 sends a PAUSE to the link e_1. These events happen within just 21 us.

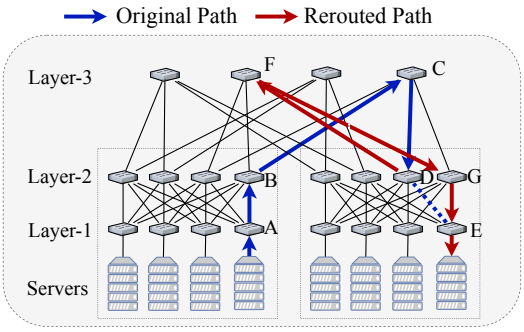
Takeaway: A DCN suffers from a high risk of deadlocks, when the following three conditions are met: 1) there exist CBDs in the network; 2) links in the CBDs are congested; 3) PFCs are triggered more frequently than usual. If we apply ECMP/KSP routing in an expander graph, we may have to constantly monitor the congested links and the abnormal switch behaviors. FC's design completely eliminates CBDs, and thus could significantly simplify the RoCEv2 deployment.

7 Discussion

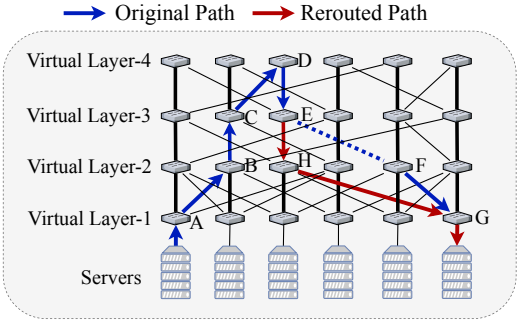
7.1 Handling Link/Node Failures

Link/node failures are common in practical data centers [16]. When a link/node fails, to avoid packet drop, local rerouting is performed to forward the affected packets along a different path to the destination [32]. Unfortunately, local rerouting may introduce CBDs and cause deadlocks even if the original network is CBD-free [24]. Consider the Clos network in Fig.10(a). Initially, packets from ToR A to ToR E follow an up-down path $A \to B \to C \to D \to E$. When the link DE fails, packets that arrived at the switch D cannot find an alternative downstream path to E and thus are bounced back to F. Then, the path from A to E becomes $A \to B \to C \to D \to F \to G \to E$. This path contains a down-up bounce, which could introduce CBDs into Clos networks.

To avoid deadlocks in Clos networks under link/node fail-



(a) Clos topology.



(b) FC's topology.

Figure 10: Rerouting under link failures.

ures, Tagger [24] adds a tag to all the packets, increases the tag on the bounce and puts packets with different tags into different lossless queues. This approach should also work for FC because we can treat FC as a virtual multi-layered network. Nevertheless, better approach may exist for FC. In Clos networks, every top-layer switch has a unique path to every ToR switch and thus every packet affected by a downstream link/node failure has to be bounced back to another top-layer switch. In contrast, every packet affected by a link/node failure in FC can freely choose any virtual layer as long as there is a link to forward this packet, because every virtual switch in the same column (see Fig. 10(b)) belongs to the same physical switch. For example, there is a flow from A to G in Fig. 10(b) and the original path is $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G$. When the link EF fails, the affected packets can be rerouted to $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow H \rightarrow G$. This new path is still an up-down path and thus tagging is not required. (Admittedly, if the rerouted path contains a down-up bounce, we still need to update the packet tags.) Based on the above analysis, we suspect that FC could be more efficient in handling link/node failures than a Clos network. We will explore this further in our future work.

7.2 Handling Route Reconfiguration

Route reconfiguration is common in data centers, which could happen when 1) new flows join/leave the network; 2) DCN topology changes; 3) Traffic Engineering is enabled; 4) an SDN controller reoptimizes routing paths after link/node failures. FC's design makes it easy to handle route reconfiguration. FC performs virtual up-down routing. As long as the virtual layers remain unchanged (i.e., which ToR port belongs to which virtual layer), the combined set of the original paths and the post-reconfiguration paths is CBD-free. This could dramatically simplify the workflow of route reconfiguration, because any transient state during route reconfiguration is guaranteed to be deadlock-free.

In rare cases, e.g., after data center expansion, we may need to change the virtual layers because the original number of layers may not be able to support a larger-scale network. In this case, there could be a CBD in a transient state during route reconfiguration. Existing solutions on deadlock-free route reconfiguration [11, 24, 33, 39] can be applied here.

7.3 The Scalability of Routing Tables

Expander graphs, including FC, Jellyfish [46], Xpander [50], FatClique [54], etc., face a common scalability challenge in the switch routing tables. Unlike Clos, expander graphs cannot easily aggregate IP addresses in the switch routing tables due to the increased routing complexity. To resolve this challenge, one potential solution is to design a hierachically routing strategy, e.g., divide ToRs into groups based on their IP prefixes and then perform intra-group and inter-group routing separately. This approach could increase the chance of IP aggregation in the switch routing tables, but may also hurt path diversity and load balancing efficiency. We will explore this tradeoff further in our future work.

8 Conclusion

We present FC, a topology-routing co-designed methodology to eliminate PFC-induced deadlocks, for cost-effective and safe deployment of RoCEv2 over expander networks. Motivated by the fact that the up-down routing paths of multi-layered Clos networks are CBD-free, we design FC's topology to exhibit a virtual layered structure, and propose an edge-disjoint virtual up-down routing for FC that is guaranteed to be CBD-free. We evaluate FC against several competitors using throughput analysis, testbed implementation and packet-level simulation. Our evaluation results demonstrate that 1) FC is deadlock-free while ECMP/KSP may trigger deadlocks; 2) FC significantly reduces average hop count and improves network throughput over the state-of-art EDST-based routing strategy; 3) FC attains higher throughput than Clos networks built using the same number of switches under all-to-all and uniform random patterns. These properties make FC a promising design for deadlock prevention in expander graphs.

Acknowledgement: This work was supported by the NSF China (No. 61902246, 62272292 and 61960206002). We also thank our shepherd Brent Stephens and the NSDI reviewers.

References

[1] Mincostflow solver of ortools. https://developers.google.com/optimization/reference/graph/min_cost_flow.

[2] 32×400Gbps Switch Price. https://www.fs.com/products/158704.html.

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

[4] 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.

[5] M. Besta and T. Hoefler. Slim fly: A cost effective low-diameter network topology. In *SC*, 2014.

[6] M. Besta, M. Schneider, M. Konieczny, K. Cynk, E. Henriksson, S. D. Girolamo, A. Singla, and T. Hoefler. Fatpaths: Routing in supercomputers and data centers when shortest paths fall short. In *SC*, 2020.

[7] B. Bollobás. The isoperimetric number of random regular graphs. *European Journal of combinatorics*, 9(3):241–244, 1988.

[8] CALIENT Technologies. https://www.calient.net/resources/#documents.

[9] P. Cao, S. Zhao, M. Y. The, Y. Liu, and X. Wang. Trod: Evolving from electrical data center to optical data center. In *ICNP*, 2021.

[10] Copper Cable Price. https://www.fs.com/products/149316.html.

[11] J.-J. Crespo, J. L. Sánchez, F. J. Alfaro-Cortés, J. Flich, and J. Duato. Upr: deadlock-free dynamic network reconfguration by exploiting channel dependency graph compatibility. *The Journal of Supercomputing*, 77:12826–12856, 2021.

[12] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. 1988.

[13] Fiber Price. https://www.fiber-mart.com/12-fibers-singlemode-smf-12-strands-flat-mtp-breakout-cable-lcscfcst-flat-fiber-cable-lszhriser-p-16935.html.

[14] Y. Gao, Q. Li, L. Tang, Y. Xi, P. Zhang, W. Peng, B. Li, Y. Wu, S. Liu, L. Yan, et al. When cloud storage meets rdma. In *NSDI*, 2021.

[15] M. Gerla and L. Kleinrock. Flow control: A comparative survey. *IEEE Transactions on Communications*, 28(4):553–574, 1980.

[16] P. Gill, N. Jain, and N. Nagappan. Understanding network failures in data centers: Measurement, analysis, and implications. In *SIGCOMM*, 2011.

[17] A. V. Goldberg and M. Kharitonov. On implementing scaling push-relabel algorithms. *Network Flows and Matching: First DIMACS Implementation Challenge*, 1993.

[18] P. Goyal, P. Shah, K. Zhao, G. Nikolaidis, M. Alizadeh, and T. E. Anderson. Backpressure flow control. In *NSDI*, 2022.

[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] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn. Rdma over commodity ethernet at scale. In *SIGCOMM*, 2016.

[21] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. W. Moore, G. Antichi, and M. Wójcik. Re-architecting datacenter networks and stacks for low latency and high performance. In *SIGCOMM*, 2017.

[22] HPCC. https://github.com/alibaba-edu/High-Precision-Congestion-Control.

[23] S. Hu, Y. Zhu, P. Cheng, C. Guo, K. Tan, J. Padhye, and K. Chen. Deadlocks in datacenter networks: Why do they form, and how to avoid them. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, 2016.

[24] S. Hu, Y. Zhu, P. Cheng, C. Guo, K. Tan, J. Padhye, and K. Chen. Tagger: Practical pfc deadlock prevention in data center networks. In *CoNEXT*, 2017.

[25] IEEE. https://1.ieee802.org/dcb/802-1qbb/.

[26] S. A. Jyothi, A. Singla, P. B. Godfrey, and A. Kolla. Measuring and understanding throughput of network topologies. In *SC*, 2016.

[27] M. Karol, S. J. Golestani, and D. Lee. Prevention of deadlocks and livelocks in lossless backpressured packet networks. *IEEE/ACM Transactions on Networking*, 11(6):923–934, 2003.

[28] S. Kassing, A. Valadarsky, G. Shahaf, M. Schapira, and A. Singla. Beyond fat-trees without antennae, mirrors, and disco-balls. In *SIGCOMM*, 2017.

[29] J. Kim, W. J. Dally, and D. Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. In *ISCA*, 2007.

[30] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh, et al. Hpcc: High precision congestion control. In *SIGCOMM*, 2019.

[31] Linux Rdma. https://github.com/linux-rdma/perftest.

[32] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson. F10: A fault-tolerant engineered network. In *NSDI*, 2013.

[33] O. Lysne, T. M. Pinkston, and J. Duato. A methodology for developing deadlock-free dynamic network reconfiguration processes. part ii. *IEEE Transactions on Parallel and Distributed Systems*, 16(5):428–443, 2005.

[34] P. López, J. M. Martínez, and J. Duato. A very efficient distributed deadlock detection mechanism for wormhole networks. In *HPCA*, 1998.

[35] R. Mittal, A. Shpiner, A. Panda, E. Zahavi, A. Krishnamurthy, S. Ratnasamy, and S. Shenker. Revisiting network support for rdma. In *SIGCOMM*, 2018.

[36] P. Namyar, S. Supittayapornpong, M. Zhang, M. Yu, and R. Govindan. A throughput-centric view of the performance of datacenter topologies. In *SIGCOMM*, 2021.

[37] Optical Transceiver Price. https://www.fs.com/products/128242.html.

[38] E. Palmer. On the spanning tree packing number of a graph: A survey. *Discrete Mathematics*, 2001.

[39] T. M. Pinkston, R. Pang, and J. Duato. Deadlock-free dynamic reconfiguration schemes for increased network dependability. *IEEE Transactions on Parallel and Distributed Systems*, 14(8):780–794, 2003.

[40] L. Poutievski, O. Mashayekhi, J. Ong, A. Singh, M. Tariq, R. Wang, J. Zhang, V. Beauregard, P. Conner, S. Gribble, et al. Jupiter evolving: Transforming google's datacenter network via optical circuit switches and software-defined networking. In *SIGCOMM*, 2022.

[41] R. Recio, B. Metzler, P. Culley, J. Hilland, and D. Garcia. A remote direct memory access protocol specification. Technical report, RFC 5040, October, 2007.

[42] J. Roskind. *Application of Edge Disjoint Trees to Failure Recovery in Data Communication Networks*. PhD thesis, PhD thesis, Department of Electrical Engineering and Computer Science, 1983.

[43] J. Roskind and R. E. Tarjan. A note on finding minimum-cost edge-disjoint spanning trees. *Mathematics of Operations Research*, 10(4):701–708, 1985.

[44] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren. Inside the social network's (datacenter) network. In *SIGCOMM*, 2015.

[45] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, et al. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. 2015.

[46] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey. Jellyfish: Networking data centers randomly. In *NSDI*, 2012.

[47] B. Stephens and A. L. Cox. Deadlock-free local fast failover for arbitrary data center networks. In *INFOCOM*, 2016.

[48] B. Stephens, A. L. Cox, A. Singla, J. Carter, C. Dixon, and W. Felter. Practical dcb for improved data center networks. In *INFOCOM*, 2014.

[49] M. Y. Teh, S. Zhao, P. Cao, and K. Bergman. Enabling quasi-static reconfigurable networks with robust topology engineering. *IEEE/ACM Transactions on Networking*, 2022.

[50] A. Valadarsky, G. Shahaf, M. Dinitz, and M. Schapira. Xpander: Towards optimal-performance datacenters. In *CoNEXT*, 2016.

[51] X. Wu and E. T. Ng. Detecting and resolving pfc deadlocks with itsy entirely in the data plane. In *INFOCOM*, 2022.

[52] X. Wu, D. Turner, C.-C. Chen, D. A. Maltz, X. Yang, L. Yuan, and M. Zhang. Netpilot: Automating datacenter network failure mitigation. In *SIGCOMM*, 2012.

[53] J. Y. Yen. Finding the k shortest loopless paths in a network. *management Science*, 17(11):712–716, 1971.

[54] M. Zhang, R. N. Mysore, S. Supittayapornpong, and R. Govindan. Understanding lifecycle management complexity of datacenter topologies. In *NSDI*, 2019.

[55] S. Zhao, P. Cao, and X. Wang. Understanding the performance guarantee of physical topology design for optical circuit switched data centers. In *SIGMETRICS*, 2021.

[56] 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 *NSDI*, 2019.

[57] Y. Zhu, Y. Zhu, H. Eran, D. Firestone, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, M. Zhang, and J. Padhye. Congestion control for large-scale rdma deployments. In *SIGCOMM*, 2015.

A Appendix

A.1 Finding Edge-Disjoint Paths Using Min-Cost Max-Flow

Definition 2 *(Min-Cost Max-Flow Problem) Given a flow network $G(V,E)$ with*

- *$u(v,w)$, upper bound on flow from node v to node w;*
- *$c(v,w)$, cost of a unit of flow on (v,w),*

and a source-destination pair (s,t), $[f(v,w)]_{(v,w)\in E}$ is called a flow assignment from s to t if the following constraints are met:

1. *Capacity constraints: $0 \le f(v,w) \le u(v,w)$;*
2. *Flow conservation constraints: $\sum_u f(u,v) = \sum_w f(v,w)$ for any node $v \neq s,t$ and $\sum_w f(s,w) = \sum_u f(u,t) = F$. Here F is called the total amount of flow from s to t.*

The objective of the min-cost max-flow problem is to find a flow assignment $[f(v,w)]_{(v,w)\in E}$ with the maximum flow that minimizes

$$\sum_{(v,w)} c(v,w)\cdot f(v,w).$$

Note that the constant parameters $u(v,w)$ are all positive and $c(v,w)$ can be either positive or negative. In addition, the min-cost max-flow problem has a very nice property that guarantees integer solutions:

Theorem 6 *(Integral Flow Theorem) Given a min-cost max-flow problem, if $u(v,w)$'s are all integers, then there exists an integer solution, i.e., $f(v,w)$'s are all integers, such that $[f(v,w)]_{(v,w)\in E}$ attains the maximum flow with minimum cost.*

In fact, when we solve a min-cost max-flow problem with integer bounds using the Scaling Push-Relabel algorithm [1, 17], the resulting optimal solution is guaranteed to be an integer solution.

Finding Edge-Disjoint Paths: As a consequence of Theorem 6, we can find the maximum number of edge-disjoint paths from s to t using min-cost max-flow. Specifically, let E_0 be the set of links that can be used at most once (see the solid links in Fig. 4(c)), and $E \setminus E_0$ be the set of links that can be used multiple times (see the dashed links in Fig. 4(c)). If we set the upper bound as $u(v,w) = 1$ for all the links $(v,w) \in E_0$ and set the upper bound as $u(v,w) = \infty$ for all the links $(v,w) \in E \setminus E_0$, then the resulting min-cost max-flow solution $[f(v,w)]_{(v,w)\in E}$ can be decomposed into F (F is the maximum flow) paths where links in E_0 can be used at most once. The F paths can be found by performing Depth First Search F times (see Algorithm 1). Note that when we perform DFS in line 5 of Algorithm 1, we will never encounter a cycle. Otherwise, by removing this cycle we could obtain another flow assignment with lower cost. Having this observation could slightly simplify the DFS implementation. We do not need to track the set of visited nodes during the DFS search.

Algorithm 1: Find Edge-Disjoint Paths in the Directed Virtual Up-Down Graph

Input : A directed virtual up-down graph (see Fig. 4(c)) and a source-destination pair (s,t).
Output : Maximum number of edge-disjoint up-down paths from s to t.

1 Let E_0 be the set of solid lines in the directed virtual up-down graph. Construct a flow graph by setting the link capacity and the link cost as 1 for all links in E_0, and setting the link capacity as ∞ and the link cost as ε (an infinitesimal value) for all links not in E_0.
2 Solve the min-cost max-flow problem. Let $[f(v,w)]_{(v,w)\in E}$ be the optimal solution and let F be the maximum flow from s to t.
3 Use $\mathcal{P}$ to store the set of paths, and initialize $\mathcal{P} = \emptyset$.
4 **for** *i in {1,2,...,F}* **do**
5 Use Depth First Search to find a path P from s to t such that $f(e) \ge 1$ for every edge e in P.
6 Store P in $\mathcal{P}$.
7 For every edge e in P, decrement $f(e)$ by one.
8 **end**
9 Return $\mathcal{P}$.

A.2 A Sufficient and Necessary Condition for CBD-Free Routing

We first introduce the concept of *link dependency graph*.

Definition 3 *Given a network $G(V,E)$ and a path set $\mathcal{P} = \{P_1, P_2, ..., P_K\}$, a link dependency graph $G'(V',E')$ can be constructed as follows:*

1. *V' is the set of directed links used by at least one path $P \in \mathcal{P}$;*
2. *For any $e_1, e_2 \in V'$, there is a directed link from e_1 to e_2 in E' if and only if e_1 is the next hop of e_2 in one path $P \in \mathcal{P}$.*

Then, the following theorem offers a sufficient and necessary condition for a set of paths to be CBD-free.

Theorem 7 *Given a network $G(V,E)$, a path set $\mathcal{P} = \{P_1, P_2, ..., P_K\}$ is CBD free if and only if the corresponding link dependency graph $G'(V',E')$ contains no loops.*

Proof 2 ***Necessity $\Rightarrow$:*** *If the path set $\mathcal{P}$ is CBD free, we prove that $G'(V',E')$ contains no loops. We prove this by contradiction. Suppose that $G'(V',E')$ contains a loop $v_1' \to v_2' \to ... \to v_s' \to v_1'$. Let e_i be the link in $G(V,E)$ that corresponds to v_i'. Since e_1 is the next hop of e_2 in a path, if e_1 is paused, e_2 will be paused. Based on the same argument, $e_3, ..., e_s$ will be paused. Since e_s is the next hop of e_1 in a path, the pause of e_s will in turn pause e_1. Then, a CBD is formed, which contradicts the assumption that the path set $\mathcal{P}$ is CBD-free.*

Sufficiency $\Leftarrow$***:*** *If $G'(V',E')$ contains no loops, we prove that the path set $\mathcal{P}$ is CBD free. We again prove this by contradiction. Suppose that $\mathcal{P}$ contains a CBD. Then, there must exist a sequence of links $e_1, e_2, ..., e_s$ such that e_i is the next hop of e_{i+1} in a path and e_s is the next hop of e_1 in a path. Then, the corresponding vertices of $e_1, e_2, ..., e_s$ in $G'(V',E')$ forms a loop, which contradicts to the assumption that $G'(V',E')$ contains no loop.*

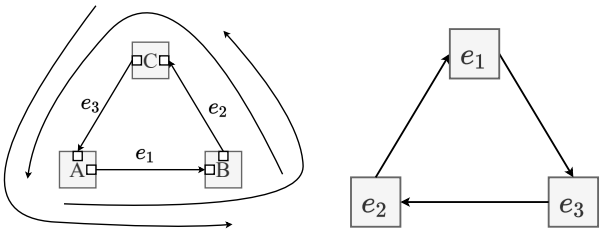
According to Theorem 7, we design Algorithm 2 to check if a set of paths is deadlock-free.

Algorithm 2: Check if a set of paths is deadlock-free

Input : A set of paths $\mathcal{P} = \{P_1, P_2, ..., P_K\}$ and a network $G(V,E)$.

Output : Whether $\mathcal{P}$ is deadlock-free.

1 Construct a link dependency graph $G'(V',E')$ based on Definition 3.
2 Use deep first search to check if $G'(V',E')$ has a loop.
3 Return true if G' has no loop; return false otherwise.



(a) An example network. (b) Link dependency graph.

Figure 11: Deadlock detection with a link dependency graph.

We use the example in Fig. 11 to illustrate the idea of the deadlock detection algorithm. Given a path set $\mathcal{P} = \{A \to B \to C, B \to C \to A, C \to A \to B\}$, we can construct a link dependency graph with three vertices: $e_1(A \to B), e_2(B \to C), e_3(C \to A)$. It is easy to see that this link dependency graph contains a loop. Thus, the path set $\mathcal{P}$ contains a CBD.

Using Theorem 7, we can prove that up-down routing is CBD-free in a multi-layered network.

Theorem 8 *In a multi-layered network, the path set generated by up-down routing is CBD-free.*

Proof 3 *For all the links in a multi-layered network, we can define a partial order as follows. A link e_1 is considered smaller than another link e_2 if either of the following three conditions is met:*

1. *e_1 is an up link while e_2 is a down link;*
2. *e_1, e_2 are down links and e_1 is at a higher layer than e_2;*
3. *e_1, e_2 are up links and e_1 is at a lower layer than e_2.*

Then, when we construct a link dependency graph based on up-down paths, it is easy to verify the following fact: if there is a directed link from e_1 to e_2, we must have $e_2 < e_1$. Therefore, the link dependency graph cannot contain a loop. As a result, the path set generated by up-down routing is CBD-free.

A.2.1 Proof of Lemma 4

Proof 4 *Since the path set $\mathcal{P}$ is CBD free in $G(V,E)$, the corresponding link dependency graph $G'(V',E')$ must contain no loop. In this case, we could construct a new graph $G''(V',E'')$ by adding a link from $v_1 \in V'$ to $v_k \in V'$ whenever there exists a sequence of node $v_2, v_3, ..., v_{k-1} \in V'$ such that $(v_i, v_{i+1}) \in E'$ for every $i = 1, 2, ..., k-1$. It is easy to check that $G''(V',E'')$ is also loop-free.*

Now we consider the contraction process. Let $\hat{G}'(\hat{V}',\hat{E}')$ be the link dependency graph of $(\hat{G}(\hat{V},\hat{E}),\hat{\mathcal{P}})$. We can prove that $\hat{G}'(\hat{V}',\hat{E}')$ is a subgraph of $G''(V',E'')$. First, in $\hat{G}(\hat{V},\hat{E})$, the edges within each vertex set V_i (i=1,2,...,m) are removed. Thus, $\hat{V}' \subseteq V'$. Second, for any edge $(e_1,e_2) \in \hat{E}'$, there must be a path $\hat{P} \in \hat{\mathcal{P}}$, such that e_1 is the next hop of e_2 in $\hat{P}$. Note that $\hat{P}$ is obtained by contracting a path $P \in \mathcal{P}$. We must have e_1 as a down-streaming hop (not necessarily next hop) of e_2 in P. Based on the construction of $G''(V',E'')$, we know that $(e_1,e_2) \in V''$. Therefore, $\hat{E}' \subseteq V''$. Based on the above analysis, we immediately know that $\hat{G}'(\hat{V}',\hat{E}')$ is a subgraph of $G''(V',E'')$. Since $G''(V',E'')$ is loop-free, $\hat{G}'(\hat{V}',\hat{E}')$ must also be loop-free. Then, according to Theorem 7, we must have that the path set $\hat{\mathcal{P}}$ is CBD free in the topology $\hat{G}(\hat{V},\hat{E})$.

A.3 Generating a Clos Network with H Hosts Using N p-Port Switches

Given N p-port switches and H hosts, we study how to construct a Clos network with the maximum throughput.

We first consider a 2-layered Clos Network. For each switch, let h be the number of ports connected to hosts. Then, the total number of switches in the first layer (i.e., the ToR layer) is $\lceil H/h \rceil$. As long as $\lceil H/h \rceil \le p$, we can put $p-h$ switches in the second layer and create a complete bipartite graph between the ToR switches and the switches in the second layer. In total, $\lceil H/h \rceil + p - h$ switches are used. To maximize throughput, we only need to find the smallest h by solving the following optimization problem:

$$\boxed{\min\ h \text{ such that } \lceil H/h \rceil \le p, \lceil H/h \rceil + p - h \le N.} \quad (5)$$

In many cases, it may not be feasible to construct a 2-layered Clos network or a 2-layered Clos network may not be throughput optimal. Hence, we also need to study how to construct a multi-layered Clos network.

We adopt a trial-and-error approach to find the throughput optimal L-layered Clos network ($L = 3, 4, ...$). Starting from $h = 1$, we try if it is possible to construct an L-layered Clos

Number of Servers	Servers per ToR		Number of Switches		Number of OCSs	Copper / Fiber Cable (km)		Number of Transceivers		Network Cost (Million $)		Bisection Bandwidth	
	FC	Clos	FC	Clos	FC	FC	Clos	FC	Clos	FC	Clos	FC	Clos
2400	8	16	300	406	24	4.8 / 158.3	4.8 / 40.6	7200	10560	27.58	29.76	1292	1280
	12	22	200	220	20	4.8 / 75.1	4.8 / 15.2	4000	4480	17.89	15.71	684	560
	16	25	150	166	16	4.8 / 40.2	4.8 / 8.7	2400	2688	12.93	11.62	396	336
4800	8	16	600	860	48	9.6 / 424.7	9.6 / 107.7	14400	19456	55.19	61.52	2526	2432
	12	21	400	482	40	9.6 / 202.4	9.6 / 48.7	8000	10560	35.80	34.70	1358	1320
	16	25	300	332	16	9.6 / 110.1	9.6 / 22.9	4800	5376	24.90	23.23	772	672
7200	8	16	900	1170	72	14.4 / 760.4	14.4 / 201.5	21600	29696	82.80	85.45	3786	3712
	12	21	600	761	40	14.4 / 361.2	14.4 / 88.4	12000	15488	52.50	54.13	2018	1936
	16	25	450	526	32	14.4 / 196.6	14.4 / 40.6	7200	8046	37.84	36.50	1156	1008
24000	8	16	3000	5500	240	48.0 / 4513.5	101.3 / 1393.5	72000	97008	276.29	383.53	12716	12288
	12	22	2000	2885	140	48.0 / 2047.1	71.6 / 493.8	40000	44298	175.53	199.58	6788	6400
	16	25	1500	2052	80	48.0 / 1102.0	62.3 / 268.8	24000	26880	124.60	140.70	3826	3584
48000	8	16	6000	12152	456	96.0 / 13571.0	227.0 / 5065.5	144000	192512	551.85	850.55	25614	24576
	12	21	4000	6691	260	96.0 / 5762.6	156.4 / 2071.1	80000	100958	350.11	465.72	13612	12672
	16	25	3000	4104	160	96.0 / 3002.8	124.7 / 940.8	48000	53760	249.32	282.75	7674	7168
72000	8	16	9000	21300	696	144.0 / 26774.9	406.1 / 11090.5	216000	288768	829.49	1471.83	38638	36864
	12	21	6000	10018	380	144.0 / 10830.7	234.1 / 4480.7	120000	151360	524.91	702.92	20626	19712
	16	25	4500	6828	240	144.0 / 5462.5	201.3 / 2021.8	72000	80640	374.11	468.66	11714	10752

Table 4: Cost Analysis: FC vs. Clos.

Switch [2]	OCS [8]	2m Copper Cable [10]	Fiber Cable [13]									Transceiver [37]		
			2m	5m	10m	15m	20m	30m	50m	100m	(100 + 50x)m	100m	500m	2000m
$ 59099	$ 60000	$ 189	$ 4.29	$ 4.71	$ 5.29	$ 5.71	$ 6.38	$ 7.38	$ 9.46	$ 16.54	$ (16.54 + 6.3x)	$ 499	$ 799	$ 1099

Table 5: Unit Price of Different Network Components.

network using at most N switches. If it is possible, we obtain the optimal h for the L-layered Clos network; otherwise, we increase h by one and retry the construction.

Starting from $L = 2$, we could use the above approach to find the best $h(L)$ for every L ($h(L) = \infty$ if it is not feasible to construct an L-layered Clos network). $h(L)$ may decrease at the beginning, but will eventually increase with respect to L. Whenever we see $h(L) < h(L+1)$, we can stop and return the minimum value of h, denoted by h^*. With h^*, the optimal throughput is $(p-h^*)/h^*$. When $h^* \leq \lfloor p/2 \rfloor$, the optimal throughput becomes larger than 1. In this case, the DCN offers abundant capacity while the access links between servers and ToRs become the bottleneck.

A.4 Network Cost Analysis

We offer a rough estimate about the total network cost for FC and Clos in this section. The network cost includes the electrical switch cost, the OCS cost and the cabling cost. Given an FC and a Clos with the same number of servers, we vary the number of servers per ToR switch and compute the number of required electrical switches and OCSs (only FC uses OCSs). To compute the cabling cost, we assume that intra-rack connections use direct attach copper cables, and inter-rack connections use optical fibers. An optical fiber requires an optical transceiver to connect to an electrical switch. The number of optical transceivers is easy to compute, which is equal to the total number of connected electrical switch ports (some switch ports may be unused) minus the total number of hosts. In contrast, the copper/fiber cable length depends on the detailed network layouts.

A.4.1 FC's Layout

In order to estimate the cable length, we make the following assumptions about FC's layout. On a data center floor, all the servers, switches and OCSs are hosted in racks. We use 2d coordinates (x,y) to represent a rack location.

- The i-th ToR switch is located at $((-1)^{\lfloor i/N_r \rfloor}(\lfloor i/(2N_r) \rfloor + 1), i\%N_r)$, where N_r is the number of racks per column;
- A rack can host four OCSs, and the i-th OCS is located at $(0, \lfloor i/4 \rfloor)$.

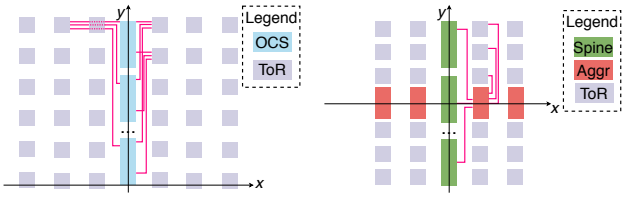
Fig. 12(a) shows FC's layout. For FC, the server-ToR connections use 2-meter copper cables and the ToR-OCS connections use fiber cables. We use Manhattan distance to compute the cable length between two racks. We vary N_r so that the total fiber cable length is minimized.

A.4.2 Clos's Layout

We focus on 3-layered and 4-layered Clos networks below. Both the 3-layered and 4-layered Clos networks follow the ToR-Aggregation-Spine architecture. In a 3-layered Clos,

Number of Switches	Number of Servers	k	Port Count of Virtual Switches	η	Cabling Constriant	Average Number of Paths	Average Path Length	Minimum Number of Paths
500	12000	4	[7, 13, 13, 7]	4	N	16.08	4.57	12.00
					Y	16.10	4.57	12.00
1000	24000	4	[7, 13, 13, 7]	7	N	14.01	4.86	9.00
					Y	14.01	4.86	9.00
2000	48000	5	[5, 10, 10, 10, 5]	13	N	15.77	5.36	11.00
					Y	15.77	5.36	11.00
3000	72000	5	[5, 10, 10, 10, 5]	19	N	14.66	5.54	10.00
					Y	14.66	5.53	10.00
5000	120000	5	[5, 10, 10, 10, 5]	32	N	13.28	5.75	9.00
					Y	13.28	5.75	9.00

Table 6: Using virtual-layered cabling has little impact on FC's routing statistics (64-port switches are used).



(a) FC cabling. (b) Clos cabling.

Figure 12: Layouts of FC and Clos. The red lines are the cable paths. To avoid visual clutter, most cable paths are omitted.

each spine is an electrical switch; in a 4-layered Clos, each spine is a 2-layered folded Clos built with electrical switches. We make the following assumptions about Clos' layout.

- The aggregation swtches in the i-th PoD are located at $((-1)^i(\lfloor i/2 \rfloor + 1), 0)$;
- The j-th ToR switch in the i-th PoD is located at $((-1)^i(\lfloor i/2 \rfloor + 1), (-1)^j(\lfloor j/2 \rfloor + 1))$;
- For a 3-layered Clos, a rack can host 24 spine switches, and the i-th spine is located at $(0, (-1)^{\lfloor i/24 \rfloor}(\lfloor (i + 24)/48 \rfloor))$. For a 4-layered Clos, we use 2 co-located racks to host a gigantic spine. Each gigantic spine is a folded Clos network, with 32 32-port switches in the first layer and 16 32-port switches in the second layer. The i-th gigantic spine is located at $(0, 2(-1)^i(\lfloor (i+1)/2 \rfloor))$ and $(0, 2(-1)^i(\lfloor (i+1)/2 \rfloor) + 1)$. Note that the intra-spine links use copper cables.

Fig. 12(b) shows Clos Network's layout. For Clos, the ToR-Aggregation and Aggregation-Spine connections use fiber cables. The Server-ToR connections use 2-meter copper cables. For 4-layered Clos, the intra-Spine connections also use 2-meter copper cables. Again, we use Manhattan distance to compute the cable length between two racks.

A.4.3 Comparison Results

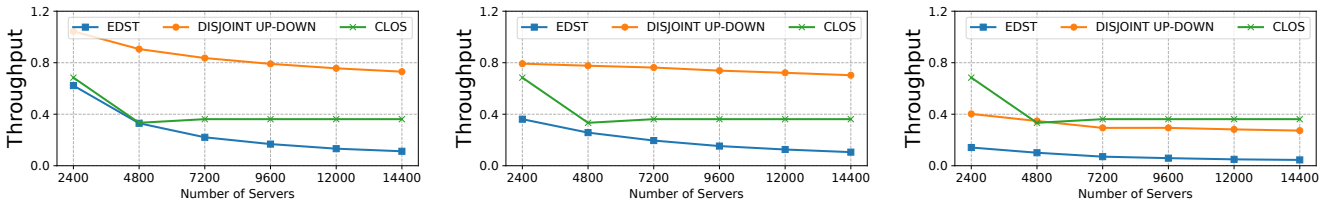
Table 4 compares the number of electrical switches/optical transceivers/OCSs and the copper/fiber cable length for FC and Clos networks. For each row, the number of servers per ToR in a Clos network has been carefully chosen such that its bisection bandwidth is equal to or slightly lower than that of FC. Generally speaking, given an FC and a Clos with the same number of hosts and similar bisection bandwidth, FC requires fewer number of electrical switches and optical transceivers, but it requires more fiber cables and additional OCSs.

Next, we offer a rough estimate on the total network cost for FC and Clos. The unit prices of different network components are summarized in Table 5. A 32×400Gbps electrical switch costs about $59000 [2] and a 320×320 optical circuit switch costs about $60000 [8]. Intra-rack connections, including server-ToR connections and intra-spine connections, use 2-meter 400Gbps copper cables, which cost about $189 [10]. The price of fiber cables increases sub-linearly with respect to the fiber length [13]. (The prices listed in [13] are the prices for 12-fiber bundles. We have divided the original price numbers by 12 in Table 5.) Hence, for each fiber cable used, we pick the shortest fiber in Table 5 that is longer than this fiber cable and use its price as the cost. The price of optical transceivers also varies depending on the transmission distance [37]. For FC, we use 2km optical transceivers because the adopted transceivers must have enough power budget to traverse an OCS. Note that traversing an OCS typically incurs about 1.5dB loss (at most 3dB) [8]. For Clos, we choose between 100m or 500m optical transceivers depending on the fiber cable length. We compare the total network cost in Table 4. When the network size is small (3-layered Clos is used), FC and Clos have similar network cost; when the network size is large (4-layered Clos is used), FC's network cost is smaller. In addition, under similar bisection bandwidth, FC uses fewer number of electrical switches and thus its network power consumption is lower (the power consumption of an OCS is only 50watts [8], which is negligible).

A.5 Additional Results

A.5.1 Impact of Cabling on FC's Routing

We generate FC's topology of different sizes & η values and evaluate if the virtual-layered cabling strategy has any impact on FC's routing. From Table 6, we cannot see clear difference



(a) Throughput of the all to all traffic matrix. (b) Throughput of uniform random traffic matrices. (c) Throughput of the near-worst traffic matrix.

Figure 13: Throughput simulation results using 64-port switches.

Number of Switches	Number of servers	k	Port Count of Virtual Switches	Routing	Average Number of Paths	Average Path Length	Average Shortest Path Length
500	12000	4	[7, 13, 13, 7]	Edge Disjoint Up-down	16.08	4.57	3.20
				EDST	20	18.34	5.26
1000	24000	4	[7, 13, 13, 7]	Edge Disjoint Up-down	14.01	4.86	3.50
				EDST	20	26.52	6.99
2000	48000	4	[7, 13, 13, 7]	Edge Disjoint Up-down	11.85	5.13	7.00
				EDST	20	33.49	9.12
3000	72000	4	[7, 13, 13, 7]	Edge Disjoint Up-down	10.63	5.30	6.00
				EDST	20	39.82	10.45

Table 7: Edge-Disjoint Virtual Up-down Routing vs. the EDST Routing (64-port Switches are Used).

when we enable/disable the cabling contraints.

A.5.2 Throughput Analysis

Clos, FC and Expander+EDST are three network architectures that are guaranteed to be deadlock-free. For networks built using 32-port switches, we demonstrate in Section 4 that

1. FC consistently outperforms Expander+EDST;
2. FC achieves higher throughput than Clos networks under all-to-all and uniform random traffic patterns.

In this section, we generate FC's topologies of different sizes using up to 600 64-port ToR switches. Each ToR switch has 40 ports connected to other switches and 24 ports connected to servers. The number of virtual layers k is chosen based on the strategy (*) in Section 3.2.3. We evaluate both FC's edge-disjoint virtual up-down routing and the EDST routing. For each FC's topology, we also compare it with a Clos network generated using roughly the same number of switches with throughput optimized. From Fig. 13, we can see that the above conclusions on FC's throughput benefits also hold for networks built using 64-port switches.

A.5.3 Routing-Path Analysis

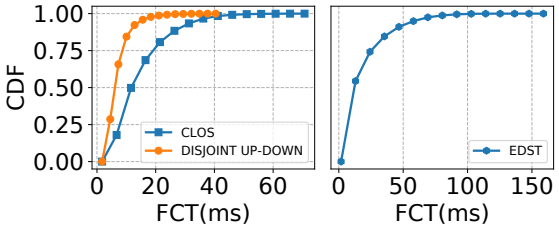
We then perform the same routing-path analysis for FC's edge-disjoint virtual up-down routing and the EDST routing. We generate FC's topologies of different sizes ($N = 500/1000/2000/3000$) using 64-port ToR switches with $s = 40$. As shown in Table 7, the average path length under FC's routing is still much shorter than that under the EDST routing.

A.5.4 More Packet-Level Simulation Results

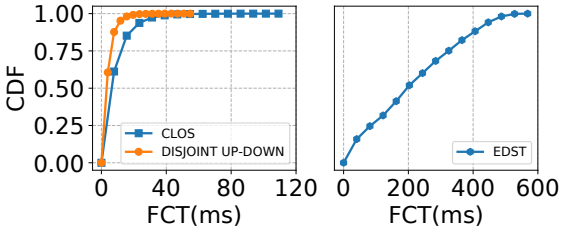
In Section 5, we perform packet-level simulation for three network setups: FC, FC+EDST, and Clos. Here, we present the detailed simulation results. We plot the CDFs for FCTs in Fig. 14. Apparently, the FCT performance under FC's routing is much better than that under the EDST routing. Hence, we mainly focus on the comparison between FC and Clos.

Under the all-to-all traffic pattern and the uniform random traffic pattern, FC achieves clearly better FCT performance than Clos because it has higher throughput. This coincides with our throughput analysis in Section 4.3.

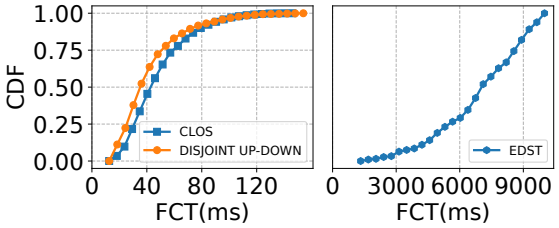
However, under the near-worst traffic pattern, we find that FC's FCT performance is just slightly worse than the Clos network's FCT performance. In contrast, our throughput analysis in Section 4.3 suggests that FC's throughput is lower than the corresponding Clos network's throughput. (In this case, FC's throughput under the near-worst pattern is about 0.5, while the Clos network's throughput is about 0.78.) The reason is that, the average hop count under FC is shorter than that under a Clos network; when the network is not congested, having a smaller average hop count compensates for the throughput gap between FC and Clos. Nevertheless, as network load increases, FC will encounter more severe congestion than Clos. We perform another simulation for the near-worst traffic pattern with network load increased from 0.3 to 0.7. (More specifically, we increase the size of each flow by $7/3$ times.) As shown in Fig. 14(d), we can see that FC's FCT performance is much worse than the Clos network's FCT performance. In fact, we see a large amount of PFC PAUSE frames in FC's network. But the good news is, there is no deadlock.



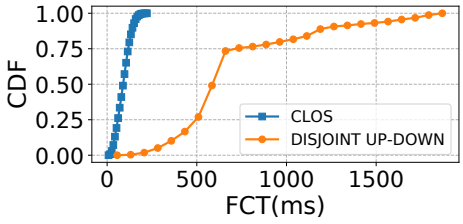
(a) All-to-all traffic pattern (Network Load= 0.3).



(b) Uniform random traffic pattern (Network Load= 0.3).



(c) Near-worst traffic pattern (Network Load= 0.3).



(d) Near-worst traffic pattern (Network Load= 0.7).

Figure 14: Compare FCTs for FC, Clos and Expander+EDST.