Enabling Quasi-Static Reconfigurable Networks With Robust Topology Engineering

Min Yee Teh, Shizhen Zhao, Peirui Cao, and Keren Bergman, *Fellow, IEEE*

***Abstract*—Many optical circuit switched data center networks (DCN) have been proposed in the last decade to attain higher capacity and topology reconfigurability, though commercial adoption of these architectures have been minimal. One major challenge these architectures face is the difficulty of handling uncertain traffic demands using commercial optical circuit switches (OCS) with high switching latency. Prior works have generally focused on developing fast-switching OCS prototypes to quickly react to traffic variations through frequent reconfigurations. This approach, however, adds tremendous complexity overhead to the control plane, and raises the barrier for commercial adoption of optical circuit switched data center networks. We propose COUDER, a robust topology and routing optimization framework for reconfigurable optical circuit switched data centers. COUDER co-optimizes topology and routing based on a convex set of traffic matrices, and offers strict throughput guarantees for any future traffic matrices bounded by the convex set. For the bursty traffic demands that are unbounded by the convex set, we employ a desensitization technique to reduce performance hit. This enables COUDER to generate topology and routing solutions capable of handling unexpected traffic changes without relying on frequent topology reconfigurations. Our extensive evaluations based on Facebook's production DCN traces show that, even with daily reconfigurations which could be realized by current commercial MEMS-based OCSs from Calient Technologies, COUDER achieves about 20% lower max link utilization, and about 32% lower average hop count compared to cost-equivalent static topologies. Our work shows that adoption of reconfigurable topologies in commercial DCNs is feasible even without fast OCSs.**

***Index Terms*—Network topology, next generation networking, software defined networking, optical switches.**

I. Introduction

WITH the explosive growth in data center traffic, building networks that meet the requisite bandwidth has also become more challenging. Modern data center networks (DCN) typically employ static uniform topologies, which have a regular structure and redundant paths to support high availability. However, static uniform topologies are inherently inefficient for carrying highly skewed and dynamic traffic that is common in DCNs [1], [2]. This has motivated several works on using optical circuit switches (OCS) to improve performance of data center networks [3], [4]. The cost of introducing OCSs to DCNs is low, because OCSs have low hardware cost and extremely low power consumption. Further, OCSs offers topological reconfigurability to DCNs, and introduces the possibility of Topology Engineering[1] (ToE) for dynamic link-allocation between "hotspots" to alleviate congestion.

In order to make the best use of the OCS reconfigurability to enhance DCN performance, the conventional wisdom is to perform on-demand reconfigurations based on DCN traffic. The main challenge lies in performing ToE under bursty traffic demands. Early works on dynamic network topologies [3], [4] use commercial OCSs to reconfigure DCN topology based on the currently observed traffic matrix (TM). However, these OCSs have large reconfiguration latency, so traffic demands could have changed after reconfiguration completes. To sidestep this issue, subsequent works focused on designing agile OCS prototypes capable of microsecond level reconfiguration to better react to traffic variations [6], [7], [8], [9]. However, these solutions require synchronizing many reconfiguration events at microsecond level, which introduces significant overhead to the network controller. The lack of experience in managing dynamic networks and the steep adoption barrier makes it hard for vendors to adopt dynamic networks topologies in commercial DCNs.

These challenges motivated us to explore the possibilities of reconfiguring topology at low frequencies (on the order of hours or more). To reduce control and management complexity, we avoid on-demand topology reconfiguration. Instead, we seek an alternative approach based on robust optimization to handle traffic variation. To this end, we introduce COUDER (**C**onvex hull **O**ptimized with **U**ncertainty **D**esensitization for **E**nhanced **R**obustness). COUDER extracts a convex set of traffic matrices which can bound a large number of historical traffic matrices, delivering strong performance guarantees for any bounded traffic matrices. For the unbounded traffic matrices, COUDER employs a desensitization technique to reduce the performance degradation caused by unexpected traffic bursts. COUDER eliminates the need for high frequency reconfiguration. Thus we can simply integrate commercial

Manuscript received 25 January 2021; revised 27 March 2022 and 8 August 2022; accepted 6 September 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Subramaniam. This work was supported in part by the NSF China under Grant 61902246 and Grant 62272292, in part by the Advanced Research Projects Agency—Energy (ARPA-E) Enlightened Program under Project DE-AR00000843, and in part by the Laboratory for Physical Sciences (LPS) Advanced Computing Systems (ACS) Research under the National Security Agency IAC MAC 19-1979 Subcontract. *(Corresponding author: Shizhen Zhao.)*

Min Yee Teh and Keren Bergman are with the Department of Electrical Engineering, Columbia University, New York, NY 10027 USA (e-mail: minyee.teh@columbia.com; keren.bergman@columbia.com).

Shizhen Zhao and Peirui Cao are with the John Hopcroft Center for Computer Science, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: shizhenzhao@sjtu.edu.cn; caopeirui3@sjtu.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2022.3210534, provided by the authors.

Digital Object Identifier 10.1109/TNET.2022.3210534

[1]Our usage of the term "topology engineering" is derived from the term "traffic engineering" in WAN and data center routing literature [5]. The key distinction is that traffic engineering (TE) optimizes *only* the routing solution under a set of expected traffic demands for a given topology, while topology engineering (ToE) optimizes *both* the topology and routing based on anticipated demands.

off-the-shelf OCSs, and ensure a gradual transition into optical data centers.

Contrary to prior ToE solutions based on non-commercial OCS prototypes that require sophisticated controls [6], [10], the source of COUDER's complexity is in its algorithm design. First, formulating COUDER using robust optimization is not straightforward. It is easy to guarantee performance for future TMs that are within the predicted TM set, while ensuring solution robustness for unbounded TMs is a big challenge. Second, topology optimization is generally an NP-hard combinatorial problem. Performing robust optimization incurs further algorithmic complexity. COUDER solves the above two challenges as follows. First, by optimizing a newly-defined "hidden" metric, *sensitivity*, COUDER is currently able to guarantee solution robustness for both bounded and unbounded TMs. Second, by properly arranging the OCS physical connections, COUDER reduces its NP-hard topology design problem to a sequence of network flow problems, which can be solved efficiently in polynomial time.

In §VI, we evaluate COUDER using both production DCN traces from Facebook [2] and synthetic traffic matrices. Performance is mainly measured with two metrics: maximum link utilization (MLU) and average hop count (AHC). Although there is a gap in performance between COUDER and an ideal dynamic network with *instantaneous* reconfiguration, COUDER's performance is attained with daily reconfiguration, a feat that can be readily achieved with current commercial OCSs and a minimal increase in management overhead to the SDN controller. Our evaluations also show that daily reconfiguration is sufficient for COUDER to outperform other static DCN topologies.[2] Compared to a static topology of comparable cost, COUDER reduces the MLU by about 20%, and the AHC by about 32%. Finally, we use packet level simulations to relate operator-centric performance metrics like MLU and AHC to user-centric application-level performance.

In short, the contributions of our work are:

- We explore a new dimension to dynamic networks that is based on infrequent topology reconfigurations. This greatly lowers the barrier to commercial adoption of ToE.
- We present a topology engineering framework that is robust to traffic variations called COUDER. COUDER co-optimizes topology and routing based on a convex traffic set to deliver strong throughput guarantees for traffic bounded by the convex set, and uses a desensitization technique for out-of-bounds traffic.
- We use extensive evaluations and simulations to analyze the performance of COUDER relative to other representative static and dynamic network topologies.
- We perform traffic analysis based on production traces, and validate the feasibility of predicting future TMs with a convex set. Specifically, we found that about 92% of traffic matrices can be bounded by under 30 minutes' worth of historical traffic.

II. Background and Motivation

Today's data center networks are static, with fat trees using small-radix commodity packet switches being the de-facto standard for commercial deployments (e.g., Google [18], Facebook [19], Cisco [20], Microsoft [11]). However, the continual

[2]We believe that more infrequent reconfigurations could be feasible, but the lack of sufficiently long traces from published sources precludes us from validating this claim.

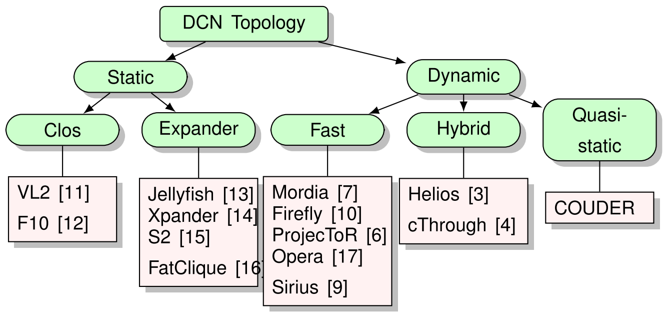


Fig. 1. Current landscape of data center network topology literature. COUDER offers a middle-ground approach between static topologies and aggressively-switching dynamic networks.

exponential growth of data center traffic would mean that future scaling would require ever larger fat trees with more layers, which can be cost-prohibitive. Broadly-speaking, most prior art in DCN topology (see Fig. 1) have been solving this issue mainly in two directions: static and dynamic (reconfigurable) topologies. On the static front, many recent works proposed eliminating the hierarchical Clos structure in favor of flatter topologies based on expander graphs (e.g. Xpander [14], Jellyfish [13], FatClique [16], S2 [15]). When used with non-minimal multipath routing, a lower cost expander can achieve a throughput comparable to that of a fully-provisioned fat tree [21]. However, static network topologies are generally designed with uniform connectivity, which makes them inefficient in carrying highly-skewed traffic [1], [22], [23].

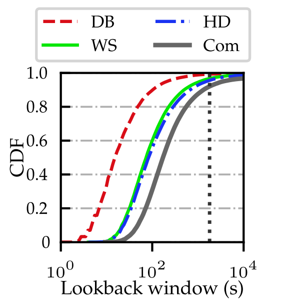
To deal with highly-skewed traffic, optical circuit switches (OCS) were proposed to build reconfigurable topologies. Unlike electrical packet switches (EPS), OCSs are transparent to in-flight packets as they neither process nor buffer packets. This makes them more power- and cost-efficient than EPSs, as OCSs do not require expensive transceivers to perform Optical-Electrical-Optical (OEO) conversion.

A. Divergence From Current Practices

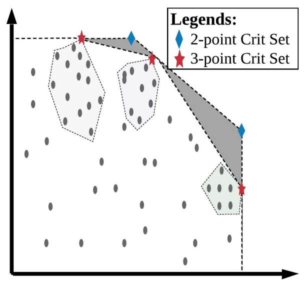
While reconfigurable networks present a futureproof solution to scaling future network performance, many prior works have proposed designs that are vastly divergent from today's static DCNs. The pioneering works like Helios [3] and c-Through [4] were largely limited by a high switching delay (30ms) of MEMS OCSs of the time, though it is still a problem that most current commercial OCSs still face [24].

The perceived need for handling rapid traffic variations with on-demand circuit switching has motivated subsequent works aimed at decreasing OCS switching latency [6], [8], [9], [10], [17], [25], [26]. This divergence from standard practices has become more pronounced in recent years, with the most recent Sirius [9] being capable of end-to-end reconfigurations every 100s of nanoseconds. We argue that the pursuit of aggressive reconfigurations actually increases the barrier to entry, and disincentivizes the widespread adoption of dynamic networks. Specifically, these architectures are challenging to adopt due reasons such as:

Massive Control & Management Complexity: Engineering a network controller capable of synchronizing thousands of topology-reconfigurations in seconds is inherently a challenging problem. Moreover, enabling microsecond-level reconfiguration may require a complete overhaul of standard congestion control protocols, resulting in architectures with tight vertical



(a) Percentage of bounded TM snapshots as lookback window changes. Dotted vertical line marks 30-minute lookback window.



(b) Topology engineering based on convex set of critical traffic matrices.

Fig. 2. a) Length of lookback window needed to bound inter-pod TMs derived from Facebook DCN clusters running database **(DB)**, web search **(WS)**, hadoop **(HD)**, combined **(Com)**. b) A convex traffic set that bounds all other TMs. Smaller convex sets typically form a larger bounds.

integration that have poor field maintainability and modular upgradability.

Limited Scalability: There is a trade off between switching latency and port-count scalability when building OCSs. Due to the limited switch radices of fast-switching OCSs, it may be difficult to interconnect 100,000s of servers commonly seen in many commercial data centers [27], while providing fast circuit switching between all end points.

Poor Failure Tolerance: In order to achieve low switching latency and good scalability at the same time, DCN reliability might be sacrificed. ProjecToR [6] introduces a potential for a single point failure through its "disco-ball" mirror switch. If the switch fails, the entire network will go down. Further, the "disco-ball" switch is based on free-space optics, and can thus be highly sensitive to environmental changes. Architectures like RotorNet [8] and Sirius [9] scale their networks by time-multiplexing across different topology settings. These designs may require even nanosecond-level topology reconfigurations. In this case, time synchronization might be the only choice for reconfiguration coordination, which can be risky due to unexpected delays in a large data center network.

We concede that, aside from the fundamental algorithmic challenges (e.g. synchronizing frequent switching at fine timescales), many of the aforementioned technical challenges may be resolved over time through operational experience. However, we argue that, at least in principle, infrequently-switched dynamic networks are a more natural next step in the evolution of today's (predominantly static) DCNs compared to hyper-agile dynamic networks proposed in many recent works. With infrequent reconfiguration, the switching latency of off-the-shelf OCSs, and the control plane complexity overhead become nonissue.

The tradeoff for pursuing infrequent reconfigurations is that it precludes on-demand switching, so the topology cannot "react" to traffic variations over time. Instead, the topology will have to be "pre-configured" for a broad range of traffic patterns. Throughout the rest of this paper, we show how COUDER achieves this with minimal topology reconfigurations.

B. Weak Temporal Stability in Pod-Level Traffic

Robust optimization could help deal with traffic variations to some extent. However, if DCN traffic is completely random, robust optimization would not help and frequent topology reconfiguration would be the only choice for reconfigurable topologies. Hence, we need to perform a traffic analysis to understand the feasibility of infrequent topology engineering. Our findings suggest that DCN traffic, especially at the pod level, exhibits a weaker form of temporal stability. Put more concretely, this means that it is possible to find a range (or "bound") that would contain most TMs in the near future, even if traffic patterns may vary significantly from one snapshot to the next.

To this end, we introduce how to find a reasonable traffic set for future TMs. Given a sequence of historical TMs, we first group all the TMs into K clusters using the k-means clustering algorithm; for every cluster, we then compute a component-wise max TM, which we refer to as a *critical TM*. These critical TMs $\{T_1, T_2, \ldots, T_K\}$ forms a convex set

$$\mathcal{T} = \left\{ T : \exists \lambda_k \geq 0, \sum_{k=1}^{K} \lambda_k = 1, \text{ s.t. } T \leq \sum_{k=1}^{K} \lambda_k T_k \right\}.$$

Clearly, all the considered historical TMs are contained in the above convex set.[3] In the rest of this paper, we also say a TM T is bounded by the critical TMs $\{T_1, T_2, \ldots, T_K\}$ if $T \in \mathcal{T}$.

We demonstrate the effectiveness of using $\mathcal{T}$ to predict future TMs with a case study based on published packet traces collected from Facebook's Altoona production data center. The traces contain up to one-day's worth of recorded packets [2]. These traces consist of packet information from three clusters of pods (a database cluster, a web search cluster and a hadoop cluster) [2]; there are a total of 21 pods. We aggregated the packet traces into a sequence of 1-second of pod-to-pod TM snapshots. For each TM in the sequence, we gradually increase the lookback window into the past until the current snapshot can be bounded by the convex set $\mathcal{T}$ formed by the historical TMs in the lookback window.

Fig. 2a shows the CDF of bounded TM snapshots as a function of the lookback window size. We generated three curves corresponding to the 3 clusters, and one curve from the combined clusters. Clearly, more TMs become bounded as lookback window sizes increase. Over 92% of TMs can be bounded with a 30-minute lookback window, and nearly all TMs are bounded with a 3-hour lookback window. We posit that inter-pod traffic satisfy the above form of weak temporal stability in time due to: 1) aggregation effects by the aggregation switches which reduces part of the burstiness, and 2) the stronger spatial structure due to the higher tendency for communication between pods assigned to the same service area/workloads/users [28]. These are common topological features that many other DCNs share. So if our posits are true, then many other DCNs could similarly exhibit, to varying degrees, weak stability.

The weak temporal stability property is meaningful because it suggests that by optimizing the topology based on the convex sets, we could derive a performance envelope that can bound performance for the common case traffic. However, some TMs will still inevitably fall out of bounds, so it is essential to

[3]Note that this convex set is not the convex hull (i.e. the smallest convex set) of the historical TMs. Convex hull is much harder to compute for high-dimensional TMs.

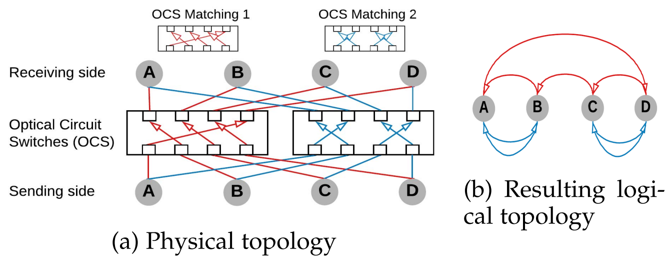


Fig. 3. Optical circuit switch (OCS)-enabled topology reconfiguration. (a) shows an example 4-node physical topology wired to two OCSs. Changing the matching state of the OCSs in (a) results in the logical overlay topology in (b).

employ additional techniques to improve tail performance. This is one of the technical challenges, which we detail in §V. It is worth noting that our approach would benefit from, though not entirely dependent on, the presence of weak traffic stability to work, as the evaluations in §VI-C show.

III. System Architecture and Design

We propose COUDER, an infrequent reconfigurable network for data centers. We first present COUDER's network architecture in §III-B, and then describe its topology design pipeline in §III-C and reconfiguration pipeline in §III-D.

A. Optical Circuit Switching (OCS)

Optical circuit switch comes in various forms, including, but not limited to, software-controlled optical patch panels [29], electrical circuit switch [30], 2D and 3D MEMS [31], [32], silicon photonics (SiP) [33], [34], [35], free space optics (FSO) [6], [10] or 60GHz wave [1], [36], RotorSwitch [8], [17], and tunable lasers [9]. Of these technologies, only optical patch-panels and 3D MEMS OCS have been commercialized and adopted by hyperscalers like Google [37]. In principle, COUDER is compatible with all of these OCS technologies. In this paper, however, we assume the network uses commercialized MEMS OCS.

While different OCS technologies may use different switching mechanisms, they are functionally-similar from network layer's perspective: each OCS "reflects" an input port signal to an output port *without* optical-electrical-optical (O-E-O) signal conversion or decoding/buffering in-flight packets. By remapping the input-to-output circuit connections, OCSs can effectively "reconfigure" the end-to-end connectivity of the network nodes to better serve expected workload demands, without needing human-recabling that is time-consuming and error-prone [38]. Further, OCS also offers additional benefits over EPSs in: 1) cost and power savings - OCSs do not require transceivers at the ingress and egress ports, 2) transparency - unlike EPSs, OCSs adds negligible latency to in-flight packets as they neither perform signal-conversion nor decode/buffer packets, and 3) seamless upgrade - OCSs are bandwidth-agnostic, therefore upgrading to higher link speeds over time does not require changing the core layer switches. More importantly, leveraging OCS topology reconfigurability. Fig. 3 presents an example of how OCS reconfigures network topology using different switch matchings.

We refer to the physical wiring between electrical packet switches (EPS) and the optical circuit switches (OCS) as the *physical topology*. Reconfiguring the OCSs establishes a new set of circuit connections between the input and output ports at the physical layer, effectively realizing a specific *logical topology* overlay on the physical topology.

B. Network Architecture

An example of the assumed DCN architecture is shown in Fig. 4, with a layer of OCSs interconnecting a number of pods. Each physical link between an OCS and the pods in Fig. 4 represents an optical fiber. An OCS sends incoming optical signals directly to a reconfigurable egress port without packet decoding and buffering. A pod is a typical deployment unit for data centers, whose fabric can be built from monolithic switches like CE12800 [39], from a collection of small-radix switches organized in a Clos-like [18], [19] or a clique-like [40], [41] structure. For instance, a pod in Facebook's Altoona fabrics is built using a two-layer Clos (a.k.a. leaf-spine) interconnect, with 48 top-of-rack (ToR) and four aggregation switches [42].

We refer to the (fixed) physical connections between the pods and the OCSs as the *physical topology*. Topology engineering reconfigures the OCSs to realize a specific *logical topology* as an overlay on the physical topology. As each pod can carry $O(100)$ uplinks, we assume that the number of uplinks per pod is much greater than the number of pods, so every pod-pair can share more than one logical link connection at any given time.

In contrast to many flexible network architectures with inter-ToR reconfigurability [4], [6], [7], [10], [36], we focus on inter-pod reconfigurability for the following reasons:

- *Scalability:* Using pods with hundreds of uplinks to the OCSs, our architecture could support up to about 100 pods. Since each pod could support $\Theta(1000)$ servers, our architecture can easily scale up to over 100k servers [43].
- *Traffic stability:* Compared to inter-ToR traffic, inter-pod traffic is more likely to exhibit *weak temporal stability* (see §II-B) due to averaging effects from the aggregation switches [22], [44]. The temporal stability makes infrequent topology reconfigurations more feasible.
- *Compatibility with current technology:* Interconnecting pod uplinks with spines using single-mode fibers and optical transceivers is already commonplace in current fat tree DCNs [45]. Single-mode transceivers typically have a link margin of $>$ 5dB, which is much higher than the insertion loss of $\leq$ 3dB common to commercial OCSs [24]. So, our network (see Fig. 4) can be easily realized by replacing the spine switches with OCSs.

1) Key Distinctions From Prior Works: COUDER assumed physical topology in Fig. 4 bears some similarities to that of pioneering works like Helios [3]. Both Helios and COUDER assume pods are physically-interconnected via a layer of OCSs at the core network. These classes of networks will henceforth be referred to as pod-reconfigurable networks. However, unlike Helios which uses both electrical packet switches and optical circuit switches at the core, we assume a fully-optical core. This assumption differentiates our approach to topology optimization from Helios, which opportunistically routes, at *small timescales*, the elastic flows using OCSs, and the latency-sensitive flows using the packet-switched core. However, distinguishing between elastic and latency-sensitive flows in practice is difficult, especially in cloud DCs where applications are not accessible to the network controller for security reasons. By contrast, COUDER is designed for

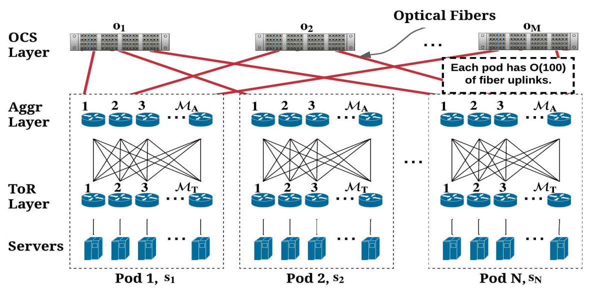


Fig. 4. An example of COUDER's physical topology. The red links represent physical *inter-pod* optical fibers, while the black links represent a physical *intra-pod* (either optical or copper) cable. A pod is the basic unit of deployment consisting of 1000's of servers. Each pod consists of $\mathcal{M}_A$ aggregation switches and $\mathcal{M}_T$ ToR-switches. In practice, operators can use large-radix monolithic switches (e.g., Huawei CE12800 with 576 x 100 GE ports [39]) or arranging several high-radix EPSs (e.g., Tomahawk 4s with 256 x 100 GE ports [46]) in parallel at the aggregation layer gives each pod hundreds of uplinks. All pods are fully-interconnected via OCSs core layer; reconfiguring the OCSs realizes a different pod-level *logical* topology.

reconfigurable networks that are switched infrequently at *large timescales* (a few minutes to several hours), and therefore does not require splitting the elephant from the mice flows at finer timescales, making it much easier to implement.

Many recently-proposed dynamic networks like Sirius [9], RotorNet [8], and ProjecToR [6] have instead favored ToR-reconfigurable designs. These topologies "flatten" out the traditional tiered structure by directly reconfiguring links between ToR switches. The flatter structure reduces the number of switches, hence reducing power consumption and capital costs. However, while ToR-reconfigurable networks may work well for small or medium-sized data centers, they scale poorly to the requisite size of hyper-scale data centers due to the limited switch radix of ToR switches. Consider, for example, a RotorNet built from 1024 ToRs of 16 uplinks each, with 2-hop path forwarding. In this setting, a pair of ToRs would need at least 4 cycles ($1024 \div 16 \div 16$) before they are connected by a 2-hop path through an intermediate ToR, and 64 cycles ($1024 \div 16$) before they are connected by a direct link. This can lead to significant flow completion time (FCT) deterioration as we scale up the number of server racks. Note that even though RotorNet and Opera are dynamic networks, they do not optimize topology based on expected traffic demands, and are therefore not traffic-aware.

C. Topology Design Pipeline

In order to realize infrequent reconfiguration, COUDER's logical topology must be robust to traffic variations that may occur during the long time window between topology updates. In §II-B, we showed using Facebook's production traces that by computing a set of critical TMs as illustrated in Fig. 2b, we can bound a large fraction of future traffic snapshots. As its first step, COUDER optimizes its topology based on the critical TMs (see Step 1 of Fig. 5) to optimize performance for the common-case bounded traffic patterns.

There are two main challenges to the design of COUDER. The first challenge is to deliver strong performance guarantees for outlier TMs that are not bound by the critical TMs. A naive robust optimization-based formulation of COUDER can only offer performance guarantees for the bounded TMs, and not for unbounded TMs. A similar issue was studied in COPE [5] in the context of wide area networking (WAN) traffic engineering (TE). Unfortunately, the proposed method there cannot be applied to topology engineering problems, because topology also becomes decision variables in COUDER. The second challenge lies in optimizing network topologies that typically involve solving integer linear programs (ILP) that are NP-Complete, so finding a solution for large networks is difficult. COUDER's topology design pipeline is designed to decouple the above two challenges (see step 2 and step 3 in Fig. 5); we detail these techniques in §V. Unlike circuit-scheduling systems like Solstice [47] that attempts to schedule circuit configurations to improve demand satisfaction time, COUDER is a topology optimization system that co-designs a *single* topology and routing solution that is resilient to long-term traffic variations. Therefore, COUDER can be considered an offline system that runs only when a topology reconfiguration event is triggered by the network controller irregularly (on the order of hours or more). We experimentally measure the average runtime of COUDER as a function of topology sizes; the results are available in Appendix E of the Supplementary Material.

D. Safe Reconfiguration Pipeline

COUDER is designed to support high performance in infrequently-reconfigured networks, which gives us room to prioritize "reconfiguring safely" over "reconfiguring quickly". There are two major safety considerations when reconfiguring topology. First, topology reconfiguration must be carefully sequenced to avoid routing packets into "black holes". The SDN controller must first "drain" links by informing packet switches not to route traffic through the optical links that are about to be switched. Only upon verifying that no traffic flows through these links can physical switching take place. After switching completes, the SDN controller can then "undrain" links and start sending traffic through them again. In general, this process is dominated by the software/control overhead rather than the physical switching of the OCSs.

Second, topology reconfiguration needs to be staged to ensure sufficient network capacity is maintained to carry live traffic at any given time. Our policy is that no more than $1 - \mu_{pred}$ fraction of links can be switched in a stage, where μ_{pred} is the worst-case maximum link utilization of all the critical TMs. If p fraction of links need to be reconfigured, $\left\lceil \frac{p}{1-\mu_{pred}} \right\rceil$ number of stages are required. This further prolongs COUDER's reconfiguration pipeline. The minimal rewiring optimization developed in [38] could help reduce the average number of reconfiguration stages, but may not work in the worst case.

Due to the above safety concerns, each reconfiguration event must be completed over a more gradual course, so the total reconfiguration latency can be much larger than those in other dynamic networks. Since DCN experiences reduced capacity during reconfiguration, reconfiguring topology at higher frequency may not necessarily improve performance. Our evaluation results in §VI-B.2 suggest that daily reconfiguration is sufficient for COUDER.

IV. Preliminaries

In this section, we introduce the recurring mathematical notations and definitions in this paper. All notations are tabulated in Table I.

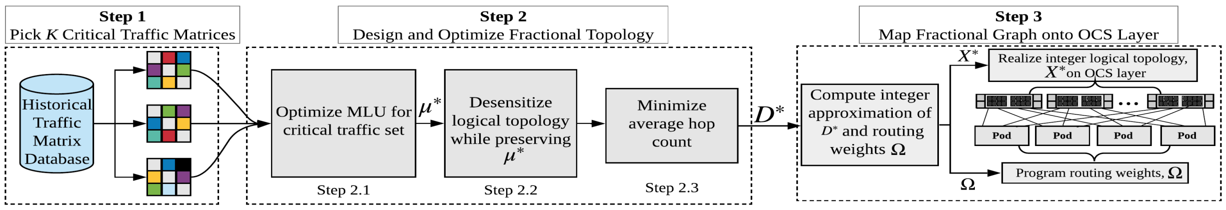

Step 1
Pick K Critical Traffic Matrices
Historical Traffic Matrix Database
Step 2
Design and Optimize Fractional Topology
Optimize MLU for critical traffic set
μ^*
Desensitize logical topology while preserving μ^*
Minimize average hop count
Step 2.1
Step 2.2
Step 2.3
D^*
Step 3
Map Fractional Graph onto OCS Layer
Compute integer approximation of D^* and routing weights Ω
X^*
Realize integer logical topology, X^* on OCS layer
Pod
Pod
Pod
Pod
Ω
Program routing weights, Ω

Fig. 5. Illustrating the topology design pipeline of COUDER.

A. Logical Topology

Let $\mathcal{S} = \{s_1, .., s_N\}$ be the set of pods with indices 1 through N, $\mathcal{O} = \{o_1, .., o_M\}$ be the set of OCSs with indices 1 through M, and x_{ij}^m be the number of directed links from pod s_i to pod s_j through OCS o_m. A logical topology is represented by $X = [x_{ij}], i, j = 1, \ldots, n$, where $x_{ij} = \sum_{m=1}^{M} x_{ij}^m$ denotes the number of links between pods s_i and s_j. Multiple unit links that between a pod-pair s_i and s_j (i.e. $x_{ij} > 1$) can be viewed as a single trunk link with $x_{ij} \times$ the bandwidth of a unit link. Logical topology X can be feasibly realized by a physical topology, if and only if (*i.f.f.*) it satisfies the following:

OCS Physical Constraints:

$$sum_{j=1}^{N} x_{ji}^m \leq h_{\text{ig}}^m(i), \quad \sum_{j=1}^{N} x_{ij}^m \leq h_{\text{eg}}^m(i),$$
$$\forall i = 1, .., N, \ m = 1, .., M; \qquad (1.1)$$
$$x_{ij} = \sum_{m=1}^{M} x_{ij}^m, \quad x_{ij} \text{ and } x_{ij}^m \text{ are all integers;} \qquad (1.2)$$

where $h_{\text{ig}}^m(i), h_{\text{eg}}^m(i)$ are the number of ingress/egress links of pod s_i through OCS o_m.

B. Path Selection and Routing Weights

Since OCSs are transparent to in-flight packets, the pod-level logical topologies of COUDER are actually mesh-like (direct connection between different pods). In this mesh-like topology, we allow inter-pod traffic being routed via either direct or indirect paths. For direct (source-destination) paths, a packet would traverse a single inter-pod link, directly from the source to the destination pod. For indirect (source-intermediate-destination) paths, we consider only indirect paths of length *two* where a packet would transit at an intermediate pod before being routed to its destination pod. Although indirect routing introduces path stretch and additional routing latency, we still consider them as they introduce path diversity and increase the overall routing capacity, without drastically increasing the routing complexity overhead. We found that considering indirect paths with path lengths greater than two offers little additional capacity despite creating a significant increase in routing complexity.

Let p denote a path, which is a sequence of *nodes* a packet traverses in the network. For each pod-pair (s_i, s_j), a direct path between the source takes the form of $[s_i, s_j]$, while an indirect path takes the form of $[s_i, s_v, s_j]$ such that $v \in \{1, 2, \ldots, N\}, v \neq i, j$. Then, $\mathcal{P}_{ij} = \{[s_i, s_j], [s_i, s_1, s_j], \ldots, [s_i, s_N, s_j]\}$ denotes the set of all direct and indirect paths between (s_i, s_j), and $\mathcal{P} = \cup_{(i,j)} \mathcal{P}_{ij}$ is the union of path sets of all source-destination pod pairs (i.e. set of all paths in the network). Let the fraction of traffic between pod-pair (s_i, s_j) that is routed along the path $p \in \mathcal{P}_{ij}$ be ω_p. We denote $\Omega = \{\omega_p, p \in \mathcal{P}\}$ such that:

$$\sum_{p \in \mathcal{P}_{ij}} \omega_p = 1, \quad \forall\, i, j = 1, \ldots, N. \qquad (2)$$

C. Optimization Objective

Our primary design objective is to minimize maximum link utilization (MLU). Given a logical topology X and a routing weight Ω, the MLU under a traffic matrix $T = [t_{ij}], i, j = 1, \ldots, n$, can be computed as follows:

$$\text{MLU} = \max_{(s_i, s_j) \in \Phi} \frac{\sum_{p \in \mathcal{P}(s_i \to s_j)} \left(\omega_p t_{\text{src}_p \text{dst}_p}\right)}{x_{ij} b_{ij}}, \qquad (3)$$

where $\Phi = \{(s_i, s_j) \in \mathcal{S} \times \mathcal{S} \mid i \neq j\}$ is the set of all pod pairs such that $i \neq j$, $\mathcal{P}(s_i \to s_j)$ is the set of paths that traverses links from s_i to s_j, b_{ij} is the link capacity between s_i and s_j, $\text{src}_p and \text{dst}_p$ are the of the source and destination pod indices of p, respectively. MLU is a good indicator of the congestion level at the most bottlenecked link; so a lower MLU is preferred. In practice, MLU cannot exceed 1, though we allow this here to capture how severe the congestion is.

Our secondary objective is to minimize average hop count (AHC). The AHC for routing a traffic demand is defined as the average path length weighted by the traffic proportion. Specifically, given a logical topology X and a routing weight Ω, the AHC for routing a traffic matrix $T = [t_{ij}], i, j = 1, \ldots, N$ is:

$$\text{AHC} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{p \in \mathcal{P}_{ij}} \left(\omega_p t_{ij} \ell_p\right)}{\sum_{i=1}^{N} \sum_{j=1}^{N} t_{ij}}, \qquad (4)$$

where $\ell_p = |p| - 1$ denotes the length of path p. Since the inter-pod routes have a path length of either one or two, the AHC has a range of $[1, 2]$. For example, if 40% of traffic is routed along direct paths of length one while the remaining 60% is routed indirectly along paths with length two, the AHC is then $0.4 * 1 + 0.6 * 2 = 1.6$. Just as lowering MLU is crucial to lowering congestion, lowering AHC is also key to lowering routing latency experienced by packets, and improving network efficiency by lowering bandwidth tax [17]. Our packet level simulations in §VII shows that when traffic demand is high, a lower AHC greatly lowers flow completion time (FCT).

V. Overall Methodology

Challenges & High-Level Approach: The core objective of COUDER is to enable high performance reconfigurable networks even with infrequent topology reconfigurations. To that end, we co-optimize the topology and routing configuration to attain:

TABLE I
NOTATIONS USED IN THIS PAPER

Notation	Description
$\mathcal{S} = \{s_1, .., s_N\}$	Set of all N pods
$\mathcal{O} = \{o_1, .., o_M\}$	Set of all M circuit switches
x_{ij}^m	Integer number of pod i's egress links connected to pod j's ingress links through o_m
$X = [x_{ij}] \in \mathbb{N}^{N\times N}$	Inter-pod topology; x_{ij} denotes the number (integer) of links connecting s_i to s_j
$T = [t_{ij}] \in \mathbb{R}^{N\times N}$	Traffic matrix, where t_{ij} denotes the traffic rate (Gbps) sent from s_i to s_j
$D = [d_{ij}] \in \mathbb{R}^{N\times N}$	Fractional topology; d_{ij} denotes the number (fractional) of links connecting s_i to s_j
$h_{eg}^m(s_i), h_{ig}^m(s_i)$	Number of physical egress and ingress links, connecting s_i to o_m
$r_{\text{eg}}^i, r_{\text{ig}}^i$	Number of egress and ingress links, of s_i
b_{ij}	Link capacity (Gbps) between s_i and s_j
p, ℓ_p	p is a sequence of pods representing a path, $\ell_p = \|p\| - 1$ is the length of p
$\text{src}_p, \text{dst}_p$	Source and destination pod index of path p
$\mathcal{P}_{ij}$	Set of paths between s_i and s_j
$\mathcal{P}(s_i \to s_j)$	Set of paths that traverses s_i to s_j links
$\Phi = \{(s_i, s_j) \in \mathcal{S} \times \mathcal{S} \mid i \neq j\}$	Set of all pod pairs such that $s_i \neq s_j$
ω_p	Fraction of $t_{\text{src}_p\text{dst}_p}$ traversing path p
μ	Maximum link utilization (MLU)
$\mathcal{M}_A$, $\mathcal{M}_T$	Number of aggregation and ToR switches in a pod, respectively

- High throughput and low latency / bandwidth tax for common-case traffic patterns.
- High robustness to traffic variations

By reconfiguring topologies infrequently at coarse timescales, COUDER sidesteps the complex control planes that would otherwise be needed to coordinate fine timescale OCS reconfigurations. This lowers the system and hardware complexity, which lowers the barrier-to-entry for adopting reconfigurable networks in production networks. However, the price for a lower system and hardware complexity is a higher algorithm complexity. There are two reasons for COUDER's algorithm complexity. First, optimizing for a topology that is robust to traffic variations over an extended period of time is challenging. Recall that our study in §II-B showed that some TMs cannot be bounded by a convex set of historical TMs. This means that a naive approach that optimizes network for a set of predicted traffic patterns cannot ensure that the topology has good performance as traffic demands change over time. Second, the topology optimization integer linear programming (ILP) problem is a hard combinatorial problem that scales exponentially with the number of pods and OCSs. Finding a solution that scales gracefully to large DCNs is challenging. We discuss how COUDER addresses these challenges in this section.

A. Computing Fractional Topology

We first introduce the following concept of *fractional* logical topology.

Definition 1: Given a set of pods $\mathcal{S} = \{s_1, \ldots, s_N\}$ and the number of ingress & egress links fr$_{ig}^i, r_{eg}^i$ of pod s_i, $D = [d_{ij}] \in \mathbb{R}^{N\times N}$ is a valid fractional topology i.f.f. *it satisfies*:

$$\sum_{j=1}^{N} d_{ij} \le r_{eg}^i, \sum_{i=1}^{N} d_{ij} \le r_{ig}^j \quad \forall\, i, j = 1, \ldots, N \tag{5}$$

Using a *fractional* logical topology allows us to decouple the topology design problem into two subproblems. In the first subproblem, we compute a fractional logical topology that is optimal for the critical TMs by ignoring the physical OCSs constraints. In the second subproblem, we realize the intent fractional logical topology on the OCS layer by computing a set of OCS matchings such that the overall integer logical topology best approximates the intent fractional topology.

1) Topology-Routing Co-Robust Optimization for MLU: Given K critical traffic matrices $\{T_1 = [t_{ij}^1], \ldots, T_K = [t_{ij}^K]\}$, a robust optimization formulation that minimizes MLU is:

Min-max MLU:

$$\min_{D,\Omega} \mu = \max\{\mu_1, .., \mu_K\}, \text{ s. t.} \tag{6.1}$$

$$\text{a) } D \text{ satisfies (5)} \tag{6.2}$$

$$\text{b) } \Omega \text{ satisfies (2)} \tag{6.3}$$

$$\text{c) } \max_{(s_i,s_j)\in\Phi} \frac{\sum_{p\in\mathcal{P}(s_i\to s_j)} \left(\omega_p t_{\text{src}_p\text{dst}_p}^k\right)}{d_{ij} b_{ij}} \le \mu_k, \quad \forall\, k = 1, .., K. \tag{6.4}$$

Note that (6) is non-linear due to the third constraint. We show in Appendix A of the Supplementary Material, how (6) can be linearized into a LP problem that can be easily solved using commercial solvers like Gurobi [48]. The resulting topology and routing solutions from solving (6) offer the following performance guarantee:

Lemma 2: Let D^{opt}, Ω^{opt} be the optimal solution of (6), and μ^* be the min-max MLU value for the critical TMs $\{T_1, \ldots, T_K\}$. Then, for any T bounded by these critical TMs, the MLU under the fractional topology D^{opt} and the routing weight Ω^{opt} is no higher than μ^*.

Note that (6) performs robust optimization on both the topology and routing solutions. We found that robust optimization for routing is necessary, although updating routing path splits can be done more frequently without changing OCS configurations. This is because our initial design did not performing robust optimization for routing, i.e., using different sets of routing weights for different traffic matrices T_k in (6), and then updating routing weights for every future TM. As traffic patterns may often be very different from prediction, the resulting MLU performance is generally unbounded. We point the readers to Appendix B of the Supplementary Material for details on why this is the case. Of course, topology-routing co-optimization in (6) does not preclude COUDER from updating routing more frequently than topology. One could fix topology and only perform robust optimization for routing based on (6).

2) Desensitization: (6) provides a strong performance guarantee for TMs that are bounded by the convex set, but not for outlier TMs. In practice, outlier TMs are inevitable due to unpredictable traffic bursts, and if not handled properly, could cause severe network congestion.

A similar issue in the context of traffic engineering was studied in [5], in which the authors' use a penalty envelope method to deliver bounded performance guarantees for outlier

TMs. The authors' in [5] employed duality to convert an infinite number of constraints into a dual problem with a finite set of linear constraints. Unfortunately, this approach fails when network topology also becomes decision variables.

Instead, we handle outlier TMs using a *desensitization* step, for which we introduce a sensitivity metric for every link (s_i, s_j). For any path p that traverses the link (s_i, s_j), a demand surge Δ in $t_{\text{src}_p\text{dst}_p}$ would increase the link (s_i, s_j)'s utilization by $\Delta\omega_p/(d_{ij}b_{ij})$. We define sensitivity for link (s_i, s_j) as

$$\text{SEN}_{ij} = \max_{p\in\mathcal{P}(s_i\rightarrow s_j)} \frac{\omega_p}{d_{ij}b_{ij}}.$$

Minimizing sensitivity helps prevent the routing solution from allocating too much weight on any single link. Specifically, having computed μ^* from (6), we fix μ^* and compute D, Ω based on the following formulation:

Desensitization:

$$\min_{D,\Omega} \max_{(s_i,s_j)\in\Phi} \text{SEN}_{ij} = \max_{p\in\mathcal{P}(s_i\rightarrow s_j)} \frac{\omega_p}{d_{ij}b_{ij}}, \text{ s. t.} \tag{7.1}$$

a) D satisfies (5) (7.2)

b) Ω satisfies (2) (7.3)

$$\text{c)} \max_{(s_i,s_j)\in\Phi} \frac{\sum_{p\in\mathcal{P}(s_i\rightarrow s_j)} \left(\omega_p t^k_{\text{src}_p\text{dst}_p}\right)}{d_{ij}b_{ij}} \leq \mu^*, \quad \forall\, k = 1, .., K. \tag{7.4}$$

Similar to (6), (7) is non-linear as its objective function takes the form of reciprocals of optimization variables. This means that we cannot rely on commercial LP solvers to solve (7) directly. Instead, we find the optimal $\text{SEN}^* = \max_{(s_i,s_j)\in\Phi} \text{SEN}_{ij}$ value in (7) using an iterative binary-search scheme. In each iteration, we fix SEN* and check if there exists D and Ω such that the maximum sensitivity is no greater than the SEN* value in the current iteration. If (7) is feasible, SEN* is reduced in the next iteration; otherwise, SEN* is increased. This binary-search scheme allows us to quickly converge to the SEN* value. The pseudocode for this algorithm is shown in Appendix C of the Supplementary Material.

3) Minimizing Average Hop Count (AHC): The formulation (7) guarantees good MLU performance for both bounded and unbounded TMs. Here, we optimize average hop count (AHC) to improve network efficiency. Let SEN* be the optimal value of (7), the final formulation is shown as follows:

Minimize Avg. Hop Count:

$$\min \max_{k=1,..,K} \frac{\sum_{i=1}^N \sum_{j=1}^N \sum_{p\in\mathcal{P}_{ij}} \left(\omega_p t^k_{ij} \ell_p\right)}{\sum_{i=1}^N \sum_{j=1}^N t^k_{ij}}, \text{ s. t.} \tag{8.1}$$

a) D satisfies (5) (8.2)

b) Ω satisfies (2) (8.3)

$$\text{c)} \max_{(s_i,s_j)\in\Phi} \frac{\sum_{p\in\mathcal{P}(s_i\rightarrow s_j)} \left(\omega_p t^k_{\text{src}_p\text{dst}_p}\right)}{d_{ij}b_{ij}} \leq \mu^*, \quad \forall\, k = 1, .., K. \tag{8.4}$$

$$\text{d)}\ \omega_p \leq \text{SEN}^* b_{ij} d_{ij},\ \forall\, i \neq j, p \in \mathcal{P}_{ij} \tag{8.5}$$

Remark: Note that the routing weight solution Ω^* is paired with the fractional topology D^*. Once we convert D^* to an integer topology, X^*, the routing weight set based on X^* needs to be recomputed using (6)-(8).

B. Realizing D^* on the OCS Layer

We now need to realize the integer logical topology, X on the OCS layer such that X best approximates D^*. The problem here is to decide the total number of links x^m_{ij} connecting pod s_i to pod s_j through OCS o_m, for every $i, j = 1, 2, \dots, N$ and $m = 1, 2, \dots, M$. Since there are M OCSs, we should split each d^*_{ij} entry of D^* into M integers, $x^m_{ij}, m = 1, \dots, M$, such that the following constraints are satisfied:

$$\lfloor d^*_{ij} \rfloor \leq \sum_{m=1}^{M} x^m_{ij} \leq \lceil d^*_{ij} \rceil, \quad \forall\, i, j = 1, \dots, N. \tag{9}$$

Then, a logical topology can be found by solving

Compute Integer Logical Topology:

Find $\{x^m_{ij}\}$ satisfying (1) and (9). (10)

In general, (10) is NP-Complete, as the proven NP-Complete 3-Dimensional Contingency Table problem [49] can be reduced to (10). Fortunately, data center vendors have the flexibility to design the DCN physical topology. If the physical topology has the following property:

Uniform Physical Striping Constraints:

$$h^1_{\text{ig}}(i) = \cdots = h^M_{\text{ig}}(i), \quad h^1_{\text{eg}}(i) = \cdots = h^M_{\text{eg}}(i), \quad \forall i = 1, \dots, N, \tag{11}$$

i.e., the ingress/egress links of each pod s_i are evenly distributed among all the OCSs, then (10) becomes polynomial-time solvable using Algorithm 1.

In Algorithm 1, x^m_{ij}'s are computed in two steps. First, we view the M OCSs as a single giant OCS and solve an integer logical topology X using (12) without accounting for the physical topology constraints. Next, we decompose X to fit the physical topologies of the M OCSs by solving (13). Unlike (10), (12) and (13) are of the form of a 2-dimensional contingency table problem. In Appendix D of the Supplementary Material, we prove that if (12) and (13) have fractional solutions, which can be easily verified, then their integer solutions can be found in polynomial time. It is easy to verify that steps 2-6 in Algorithm 1 guarantees that

$$\sum_{i=1}^{N} x^m_{ij} \leq \left\lceil \frac{\sum_{i=1}^N x_{ij}}{M} \right\rceil \leq \frac{r^j_{\text{ig}}}{M} = h^m_{\text{ig}}(j), \quad 1 \leq j \leq N$$

$$\sum_{j=1}^{N} x^m_{ij} \leq \left\lceil \frac{\sum_{j=1}^N x_{ij}}{M} \right\rceil \leq \frac{r^i_{\text{eg}}}{M} = h^m_{\text{eg}}(i), \quad 1 \leq i \leq N$$

Thus, the OCS configuration X^m fits its physical topology.

C. Complexity Analysis

The worst-case time complexity of COUDER is the sum of complexities for computing the fractional logical topology intent and the OCS matchings. To compute the logical topology intent, the complexity involves solving several LP problems sequentially. Let $O(f(n))$ be the complexity of an LP problem as a function of the number of decision variables, n. Given an N-pod network, each LP problem contains $O(N^2)$ commodities and $N - 1$ paths per commodity, resulting in

Algorithm 1: Compute an Integer Logical Topology That Best Approximates D^*

```
Data:
1) $D^*$ - fractional topology
2) $M$ - number of OCSs
Result:
1) $X^1 = [x^1_{ij}], \dots, X^M = [x^M_{ij}]$ - OCS configurations
// Step 1: Solve $X$ based on a giant OCS.
1 Solve an integer logical topology $X$ based on the following formulation:
```

$$\boxed{\begin{aligned} &\max_{X=[x_{ij}]} \sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij} \\ &\text{s.t. } \lfloor d^*_{ij} \rfloor \le x_{ij} \le \lceil d^*_{ij} \rceil, \quad \forall\, i,j = 1,..,N, \\ &\sum_{j=1}^{N} x_{ji} \le r^i_{\text{ig}}, \sum_{j=1}^{N} x_{ij} \le r^i_{\text{eg}}, \quad \forall\, i = 1,..,N. \end{aligned}} \tag{12}$$

```
// Step 2: Decompose $X$ to fit $M$ OCSs.
2 Initialize $X^{\text{rem}} = X$
3 for $m \in \{1, \dots, M\}$ do
4     Solve the OCS configuration $X^m = [x^m_{ij}]$ based on the following formulation:
```

$$\boxed{\begin{aligned} &\text{Find an integer solution for } X^m = [x^m_{ij}] \\ &\text{s.t. } \left\lfloor \frac{x^{\text{rem}}_{ij}}{M-m+1} \right\rfloor \le x^m_{ij} \le \left\lceil \frac{x^{\text{rem}}_{ij}}{M-m+1} \right\rceil, \quad \forall\, i,j \\ &\left\lfloor \frac{\sum_{i=1}^{N} x^{\text{rem}}_{ij}}{M-m+1} \right\rfloor \le \sum_{i=1}^{N} x^m_{ij} \le \left\lceil \frac{\sum_{i=1}^{N} x^{\text{rem}}_{ij}}{M-m+1} \right\rceil, \quad \forall\, j \\ &\left\lfloor \frac{\sum_{j=1}^{N} x^{\text{rem}}_{ij}}{M-m+1} \right\rfloor \le \sum_{j=1}^{N} x^m_{ij} \le \left\lceil \frac{\sum_{j=1}^{N} x^{\text{rem}}_{ij}}{M-m+1} \right\rceil, \quad \forall\, i \end{aligned}} \tag{13}$$

```
5     Update $X^{\text{rem}}$: $x^{\text{rem}}_{ij} = x^{\text{rem}}_{ij} - x^m_{ij}$
6 end
```

$O(N^3)$ decision variables. So, computing the logical topology has $O(f(N^3))$ complexity. To compute the OCS matchings, we propose Algorithm 1, which runs a min-cost flow algorithm with $O(N^4 \log(N))$ complexity [50] once, and max-flow algorithms, each with $O(N^3)$ complexity [51], on M OCSs. For most network instances we consider, $M \approx N$, which brings the complexity of Algorithm 1 to $O(N^4 \log(N))$. Overall, COUDER's complexity is $O(f(N^3) + N^4 \log(N))$.

We express the runtime complexity of the LP solver abstractly as a function of network size because the complexity of solver algorithms (e.g., interior points or simplex) varies. While most LP-solver algorithms have polynomial worst-case runtime complexities, most LP problem instances can be solved much more efficiently in practice than their worst-case runtime would suggest. So, in Appendix E of the Supplementary Material, we measured the runtime of COUDER empirically by varying the number of network pods in the range $[5, 100]$. Real world DCNs are limited in scale due to floor plan, cooling, wiring complexity, and power supply constraints [52]. So, we use a maximum network size of 100 pods in this analysis, which represents a realistic upper bound in DCN scale.

Empirical results in Appendix E of the Supplementary Material show that COUDER can easily support reconfigurations on the order of several hours, which is its intended use case. Specifically, COUDER has an average 45-second solve time for a large DCN with 50 pods, and an average $\sim$16-minute solve time for a megascale DCN with 100 pods. We can prune the forwarding routes to reduce the number of decision variables, or relax the optimality gap, to further minimize COUDER's runtime to support larger networks or higher rates of reconfiguration, though these strategies are left for future explorations.

VI. Performance Evaluation

We now analyze COUDER alongside other topology-routing solutions across long timescales. In order to scale our evaluation for extended time frames, yet still capturing the important macroscopic trends, we use a fluid network model here.

Traffic Matrices Our evaluations are driven by traffic matrices derived from production traces and synthetic generation. For production trace, we aggregate the 24-hour traces from [2] into one-second traffic matrix snapshots, which gives us slightly over 86000 snapshots.[4] We combined the three (Hadoop, websearch, and database) clusters into a single "data center-wide" trace, and show the evaluation results here. Due to space constraints, the remaining evaluation results based on individual clusters are available in Appendix F of the Supplementary Material. The aggregated traffic matrices and the software implementations are also made publicly available [53] to promote reproducibility.

Facebook's production trace shows a strong clustering effect. While these traffic patterns may be common in some production data centers, they may not be representative of all data center traffic workloads. For instance, DCNs with disaggregated storage generally have more dominant inter-cluster traffic, particularly between the compute and storage clusters. To simulate these traffic patterns, we synthesize a sequence of TMs by first splitting all the network pods into two evenly-sized clusters of pods: one for storage, the other for compute. For every TM snapshot, we then uniform-randomly generate a write/read request from each compute source pod to another compute/storage pod. Storage pods do not communicate amongst themselves.

Metrics The main metrics used for evaluation are max link utilization (MLU) to measure link congestion and average hop count (AHC) to measure latency and bandwidth-efficiency.

A. Comparison Between Topologies

We assess the performance of static and dynamic topologies at scale using traffic matrices derived from Facebook's production traces and synthetic-generation. All topologies are

[4] 1-second traffic snapshots is the finest-grained aggregation possible as the packet traces are recorded using second-level timestamps. The industry standard for traffic snapshots is typically on the order of minutes. By comparison, the 1-second snapshots used in our evaluations capture more temporal variations due to micro-bursts.

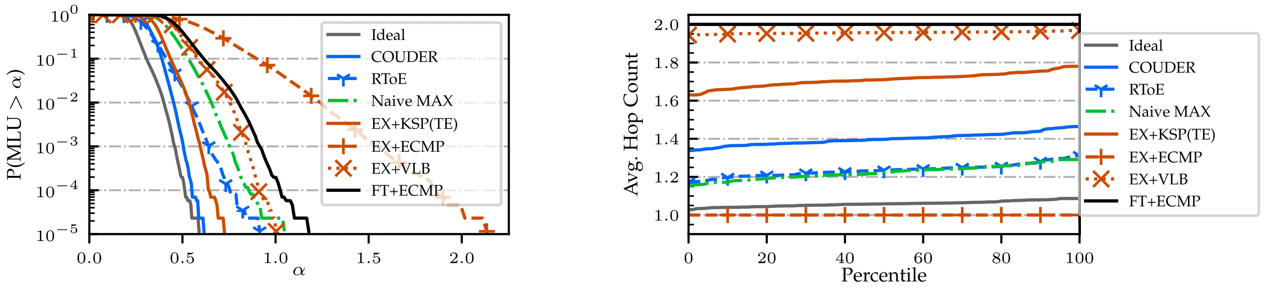


Fig. 6. Performance based on Facebook's inter-pod DCN traces, aggregated into 1-second traffic matrices.

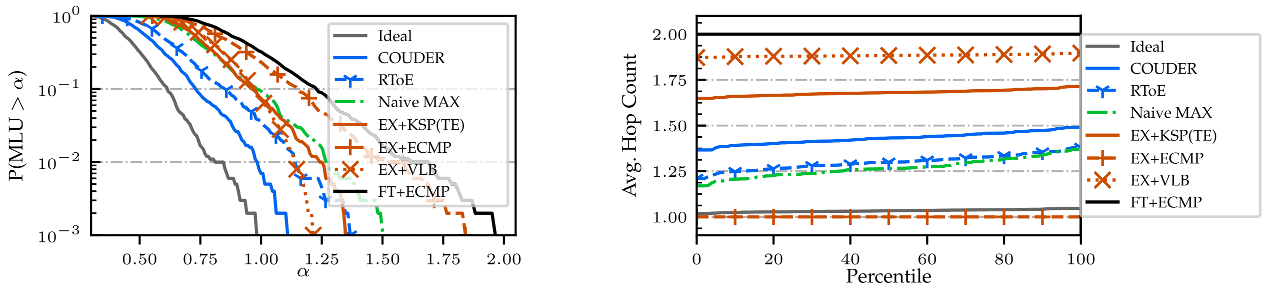


Fig. 7. Performance using synthetically-generated traffic matrices for data centers with disaggregated storage.

compared under cost-equivalent conditions (i.e. having an equal number of pods and total link capacity). Our evaluations assume a network with 21 pods, similar to the number of pods extracted from Facebook's traces. Each pod has 128 uplinks with 100 Gbps of bandwidth per link. We assume the OCS, like Calient's S320 [31], has 320 ports. To fully connect all the uplinks, a total of 9 320-port OCSs is needed. We use an unrealistic ideal, instantaneously-reconfigurable network with oracle knowledge of future traffic demands to outline the performance upper bound.

Quasi-Static Topology Engineering For TMs derived from Facebook's DCN traces, COUDER computes an inter-pod logical topology and a single set of routing weights based on 5 critical TMs extracted from the first hour's traffic matrices. For the synthetic desegregated storage TMs, we similarly extract 5 critical TMs from the first one-tenth of traffic snapshots. The computed topology and routing solution is fixed for all the remaining TMs. In the case of Facebook's workload, this is equivalent to a daily reconfiguration. In addition, we also evaluate a robust ToE approach (**RToE**), which similarly uses COUDER's multi critical TM optimization but without desensitization, and a naive (**Naive MAX**) approach that optimizes topology for the the historical-max TM.

Ideal Reconfigurable Network Instead of delving into every dynamic network in detail, we outline the optimal performance with an ideal (albeit highly unrealistic) dynamic network with instantaneous reconfigurability. The network computes an offline optimal topology and routing for each TM by solving a Multi-Commodity Flow (MCF) problem.

Fat Tree Fat tree with ECMP routing is currently the de facto standard for building commercial data centers, which we use here as a baseline. The simulated fat tree is cost-comparable to COUDER, carrying the same number of pods with a 3:1 oversubscription at the spine layer.

Mesh Expander Expanders have been shown to be highly-capable networks, able to achieve comparable throughputs to a non-subscribed fat tree at 70% the cost [21]. We also evaluate the performance of a cost-comparable expander network that directly connects pods without an OCS layer. The numbers of pod uplinks of the expander are identical to that of COUDER, and each pod's uplinks are wired uniformly to other pods. We use 3 different routing strategies on the mesh expander, namely equal cost multi path (**ECMP**), Valiant load balancing (**VLB**), and K-shortest paths with traffic engineering (**KSP(TE)**). For **ECMP**, traffic is routed directly from the source to destination pods. For **VLB** and **KSP**, traffic is routed along direct and indirect paths. **VLB** splits the traffic equally among the direct and indirect paths, **KSP(TE)** computes the path split ratios using COUDER.[5]

1) Discussion: Figs. 6 and 7 shows the MLU and AHC performances of different topologies for the Facebook and synthetically-generated desegregated storage workloads, respectively. Specifically, Fig. 6a and Fig. 7a show the MLU probability distribution function of MLU for the Facebook and synthetic traces in logarithmic scale. Each point $(\alpha, P(\text{MLU} > \alpha))$ denotes the probability of a traffic snapshot having an MLU greater than or equal to α. Overall, the fat tree exhibits the poorest performance. COUDER reduces MLU by about 50%, and AHC by about 60% over an oversubscribed fat tree. Due to the presence of an electrically-switched spine layer in the fat tree, packets must take two hops before reaching their destination pod.

Next, we can see the limitations of Naive MAX, which exhibits poorer MLU compared to not just RToE and COUDER, but also to the static expander. The core issue of Naive MAX is that it may overfit the topology to the expected demand, which greatly limits its robustness to other traffic patterns. The effects of desensitization are also highlighted, where the COUDER shows lower MLU but higher AHC compared to a standard robust ToE approach. This is because desensitization incurs a tradeoff between AHC and MLU: to stave off brittle solutions that are over-reliant on direct paths, we need to increase AHC. That said, we argue

[5]This can be done simply by setting the inter-pod logical topology to the mesh expander in COUDER's optimization pipeline.

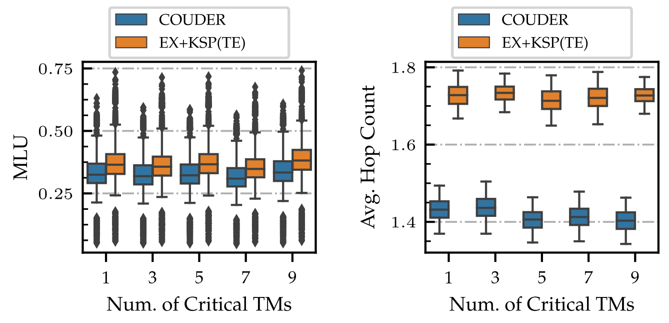


Fig. 8. The effects of choosing different numbers of critical TMs for COUDER on MLU and AHC performance.

that, within reason, minimizing MLU should take precedence over minimizing AHC. While packet latency does increase with AHC, its growth is bounded if MLU $\leq$ 1. Conversely, if MLU exceeds 1, the rate of traffic entering would exceed the throughput delivered by the network, leading to an unbounded buildup of packets and growth in packet latency.

We analyze the performance of a bandwidth-equivalent expander next. For both traffic workloads, expander with ECMP (EX+ECMP) exhibits the worst MLU performance. This is because in an expander with uniform capacity between pods, routing highly skewed traffic demands using exclusively direct paths can cause significant congestion. Using VLB on an uniform expander (EX+VLB) can alleviate this congestion by uniformly spraying traffic onto direct and indirect routes. However, VLB suffers from poor AHC due to its over dependence on indirect paths with higher "bandwidth tax" [17]. We also employ COUDER as a robust traffic engineering framework for an expander. On average, COUDER outperforms the expander by about 32% in AHC and 20% in MLU. These comparisons further showcase the benefits of COUDER: given just minimal topology reconfiguration, COUDER can attain a significant performance advantage over static networks.

Granted, there is still some performance gap between COUDER and an ideal reconfigurable network in terms of performance. However, as COUDER's performance is attained with daily reconfiguration and still comes within 10% of ideal in the case of Facebook's workload, it is realistically-attainable. An implicit assumption the ideal reconfigurable network operates under is the instantaneous knowledge of current traffic demands, but not only is traffic measurements at scale expensive, but the lack of traffic uncertainty allows the ideal reconfigurable network to aggressively optimize for MLU and AHC without regard for desensitization due to the lack of traffic uncertainty. Note that the AHC of the optimal solution is slightly greater than 1, due to some optimality-loss when rounding fractional topology to an integer one.

B. Impact of Parameters on Performance

Next, we study how picking different numbers of critical TMs (§VI-B.1) and different reconfiguration frequencies (§VI-B.2) affect network performance.

1) Effects of Critical Traffic Matrix Set Size: To evaluate the impact of different numbers of critical TMs, we repeat the experiments in §VI-A using different numbers of critical TMs.

The results are shown in Fig. 8. Recall from §II-B that the critical TMs chosen by our algorithm form an outer bound of the historical TMs. The outer bound becomes tighter as we increase the number of critical TMs (see Fig. 2b). Choosing a larger bound could cover more grounds to handle traffic bursts, but it weakens the performance guarantee for the bounded TMs. For instance, with a critical TM set size of one, the resulting convex hull formed by the entry-wise historical max TM would be the largest. In this case, the MLU performance with $K = 1$ turns out to be the worst, as shown in Fig. 8. Meanwhile, picking $K = 7$ critical TMs offers the best MLU performance, while $K = 5$ offers the best AHC performance. At any rate, COUDER's performance is not overly sensitive to variations of K. In this case, choosing any number of critical TMs between 5 and 7 should be fine.

2) Reconfiguration Frequency and Latency: Although our system is designed to enable infrequent topology reconfigurations, it is natural to wonder whether COUDER's performance can improve with more frequent reconfigurations.

To answer this question, we evaluate COUDER's performance with different reconfiguration frequencies, ranging between once every 30 seconds, 5 minutes, 1 hour, and 1 day. The initial convex set is computed based on the first hour's worth of traffic matrix snapshots. Each reconfiguration event will update the current convex TM set by considering the traffic snapshots from the previous reconfiguration window. This means that the bound set by the critical TMs increases monotonically over time. Note that a DCN experiences reduced capacity during reconfiguration, and different reconfiguration strategies affect both the duration and the capacity loss (see Fig. 9a). As described in §III-D, COUDER adopts a conservative reconfiguration strategy.

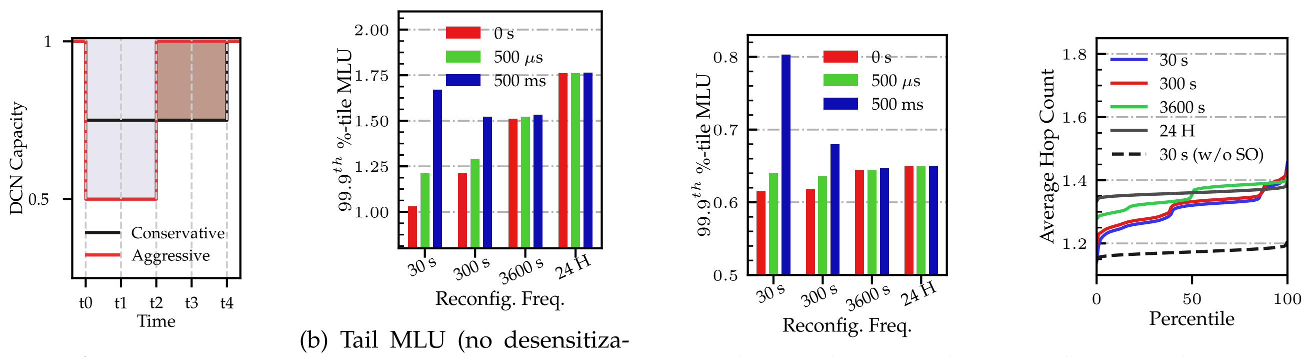
Fig. 9 shows COUDER's tail MLU[6] and AHC under different reconfiguration frequencies and latencies. Desensitization plays a decisive role here. Without desensitization, the tail MLU is higher, but it is much more dependent on the frequency of topology reconfigurations. Once desensitization is applied, COUDER retains an impressive MLU performance even with infrequent topology reconfiguration. Increasing the topology update frequency brings only minor improvements. In fact, when the per-stage reconfiguration latency is set to 500 ms, frequent topology updates can lead to a worse tail MLU. Because higher reconfiguration frequencies lead to lower duty cycles, the network would operate in a state of reduced capacity for longer periods of time. This greatly increases the risk of congestion due to a demand surge.

C. Robustness to Traffic Mispredictions

Although §II-B establishes that pod-level traffic is likely to exhibit weak temporal stability, outlier TMs can come at any time and may lead to severe congestion if not handled properly. Here, we test whether COUDER can maintain performance robustness when subjected to traffic with greater temporal variations. We do this by adding random noise to the Facebook evaluation traffic matrices.

First, $K = 5$ critical TMs are extracted from the entire sequence of 1-second inter-pod TMs derived from Facebook's packet traces, which are used for topology and routing optimization. Using the same sequence, we then compute a base component-wise max TM $T^b = [t^b_{ij}]$, where t^b_{ij} is the maximum traffic demand among all the t_{ij}'s, and a standard deviation matrix $\Sigma = [\sigma_{ij}]$, where σ_{ij} is the standard deviation of the all t_{ij}'s in the sequence. We then exhaustively enumerate all the possible burst sets, each of which contains one or two source-destination pod pairs. Then, for each burst set, denoted

[6]The non-tail performances for different reconfiguration frequencies are similar, and hence not shown.



(a) Leftover capacity (b) Tail MLU (no desensitization) (c) Tail MLU (desensitized) (d) Average hop count

Fig. 9. Impact of reconfiguration frequency and latency on performance.

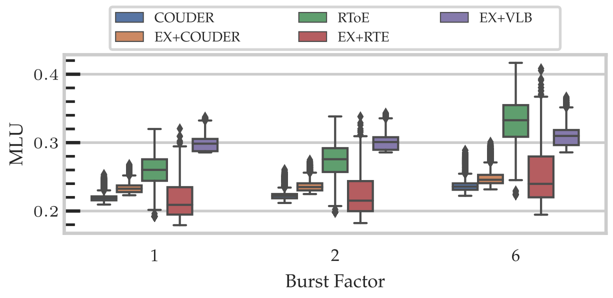


Fig. 10. MLU distribution under different burst levels. **EX+RTE** refers to an expander with robust traffic engineering, **EX+COUDER** refers to an expander with COUDER routing. **RToE** denotes robust ToE without desensitization. Each box captures the 75th, 25th percentiles and median values; the outliers beyond the upper and lower whiskers are dotted.

as B, the corresponding burst TM is $\tilde{T}(B) = [\tilde{t}_{ij}(B)]$, where

$$\tilde{t}_{ij}(B) = \begin{cases} t^b_{ij} + \text{burst\_factor} * \sigma_{ij}, & \text{for } (i,j) \in B, \\ t^b_{ij}, & \text{for } (i,j) \notin B. \end{cases} \tag{14}$$

The "burst_factor" parameter acts as a control knob for the level of burstiness.

Fig. 10 shows the MLU distribution at different burst levels. There are two features in MLU distribution that are indicators of robustness here: a low overall MLU distribution which means better performance for the evaluated TMs, and a small spread signals the solution's lower sensitivity to demand spikes in arbitrary pod pairs. COUDER with desensitization shows the best performance across all burst levels, given its lower MLU distribution with a small variance. This highlights COUDER's robustness to traffic variations, even compared to oblivious routing algorithms like VLB.

1) Robustness vs. Efficiency: Next, we stress test COUDER's worst-case performance under adversarial traffic. An adversarial TM is a *feasible* TM that maximizes the MLU for a given topology, X, and routing weights Ω. A TM is feasible if the total traffic exiting and entering every pod is no greater than its total egress capacity and ingress capacity, respectively. Formally, the worst-case MLU under adversarial traffic is defined as follows:

$$\max_{(s_i,s_j)\in\Phi} \sum_{p\in P(s_i\to s_j)} \frac{\omega_p t_{src_p,dst_p}}{b_{ij}}$$

$$\text{s.t.} \sum_{j=1}^{N} t_{ij} \le r^i_{eg} b_{ij}, \quad \sum_{j=1}^{N} t_{ji} \le r^i_{ig} b_{ji}, \quad \forall i = 1,..,N$$

In this experiment, we aim to evaluate the *robustness* vs. *efficiency* of various topology engineering solutions, where robustness measures the performance under adversarial workloads and efficiency measures how optimized a topology is for the expected traffic patterns. We generate a sequence of 500 random TMs in a 21-pod network with 256 100 Gbps uplinks per pod. These TMs form the set of expected traffic patterns, which are used to compute a topology and routing configurations. The x-axis value represents the maximum MLU across all the 500 TM snapshots. Here, ToE(Avg.) and ToE(Max.) denote topology engineering based on a *single* TM that is obtained from taking the entry-wise average and max, respectively, from the expected TM set. A similar notation for traffic engineering (e.g., TE(Avg.) and TE(Max.)) is used.

Fig. 11 shows the robustness measure (i.e., MLU under adversarial TM) on the y-axis vs. the efficiency measure (i.e., MLU under expected TMs) on the x-axis for various combinations of topology and traffic engineering solutions. We expect a tradeoff between efficiency and robustness, since optimizing for efficiency means that the topology must be overfitted to the expected TMs, at the cost of decreased robustness to adversarial TMs. This tradeoff relationship is observable from the trend of Fig. 11. Overall, COUDER exhibits a better tradeoff than the other approaches. Compared to naive ToE approaches such as RToE, ToE(Avg.)+TE(Avg.), ToE(Max.)+TE(Max.), COUDER exhibits much lower MLU under adversarial traffic patterns. This reaffirms the need for desensitization, which helps minimize MLU deterioration under adversarial workloads. As expected, COUDER shows higher efficiency for routing common-case traffic loads compared to static topologies like EX and FT, as it is capable of optimizing the logical topology for a higher performance for the common-case traffic patterns.

VII. Packet Level Simulations

The fluid model evaluations in §VI have been based on MLU (congestion) and AHC (network efficiency). In this section, we extend our evaluations using packet level simulations for two main reasons: 1) the fluid model uses aggregated TM snapshots, which smooths out the micro-bursts at packet timescales, and 2) MLU and AHC are operator-centric metrics that do not directly convey user-perceived application-level performance (e.g. flow completion time (FCT) [54]). This allows us to extend our evaluations by measuring packet latency, packet drops rate, flow completion time, etc., as a function of MLU and AHC. The simulator used is NetBench [55].

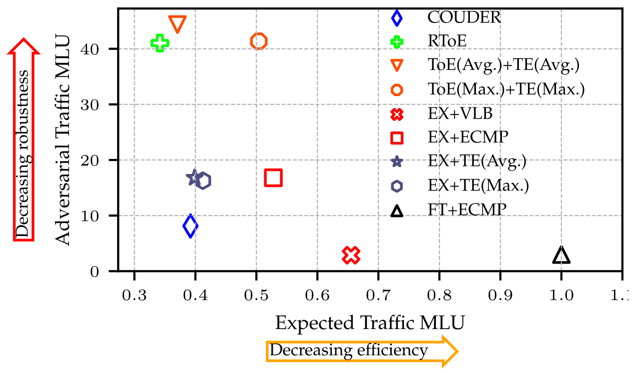


Fig. 11. MLU under worst-case adversarial traffic vs. expected traffic for various combinations of topology and traffic engineering approaches.

We assume DCTCP congestion control [56] in the simulations. Inter-pod links have 40 Gbps capacity and a propagation delay of 100 ns, which is roughly equivalent to the propagation delay of light through a 20 meter fiber. The simulator is given 0.5 second to warm up and down; flows that are initialized during these periods are not considered for analysis.

Next, we chose (at random) a 5-minute window to collect an aggregated inter-pod traffic matrix, $T = [t_{ij}]$, using the trace in [2]. Flows from pod s_i to pod s_j are generated following a Poisson arrival process with rate λt_{ij}, where the size of each flow follows uniform distribution. The inter-pod logical topology is computed based on the TM using COUDER, while a few routing weights are computed, each with a specific AHC value.[7] We vary the MLU by simply adjusting the flow arrival rates.

The impact of different MLU and AHC combinations to application-centric metrics is summarized in Figs. 12 and 13. As expected, higher MLUs lead to poorer performance (i.e. fewer flow completions, more packets are dropped, increased packet latency, etc). As higher MLU indicates more severe congestion, causing a drop in average flow throughput as more flows need to share the bottlenecked links.

Fig. 13, the packet drop rate increasing super-linearly with hop count when MLU is 0.6 and 0.8. This is because packets that traverse inflated paths leave a larger footprint, which can cause packet buildup in switch queues. This is fine when congestion is low, and packets simply experience only a slight increase in round trip latency. When congestion is high, however, the packet buildup eventually leads to buffer overflow. As packets get dropped, TCP will have to throttle its send rates, resulting in longer FCTs and fewer flow completions overall.

In summary, both MLU and AHC are intimately linked to application performance experienced by users. By optimizing for both MLU and AHC simultaneously, COUDER can improve application performance.

VIII. Related Work

A. Reconfigurable DCN Topology

Unlike COUDER, most reconfigurable DCN topologies rely on fast optical switching prototypes to handle traffic bursts. Using wavelength selective switches, an OCS prototype with microsecond-level switching time was proposed for building reconfigurable topologies in [7] and [25]. However,

[7]This can be easily done by adding an AHC constraint into COUDER's formulation (6)-(8).

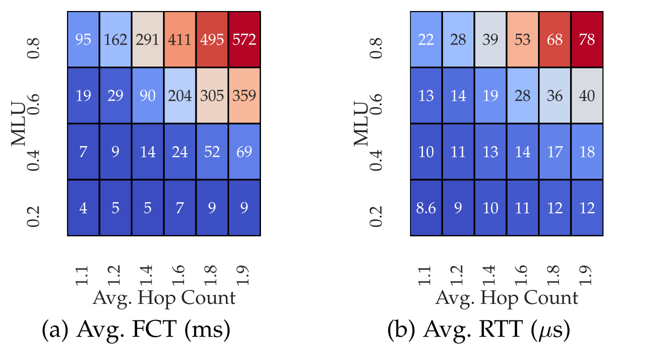


Fig. 12. Effects of MLU and average hop count on flow completion time and packet latency.

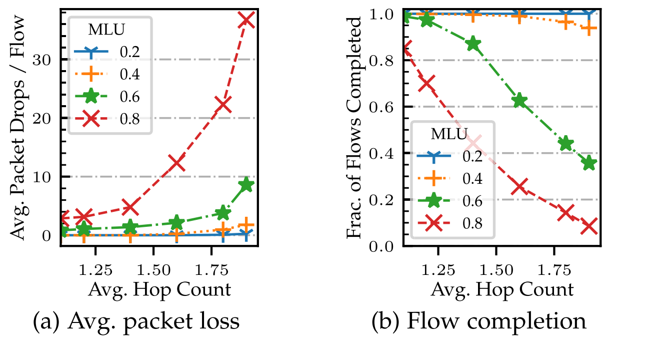


Fig. 13. Effects of operator-metrics like MLU and AHC on packet drops and flow completion based on Facebook's DCN traces.

this OCS prototype has limited port count that prevents reconfigurable topology from scaling to large data centers with over 100k servers. Some have proposed using steerable wireless transceivers [6], [10], [36] for scaling, while retaining a low switching latency simultaneously. However, wireless solutions typically face other deployment challenges related to environmental conditions in real DCNs, and to the need for sophisticated steering mechanisms. Architectures like Rotornet [8] and Sirius [9] improve DCN scalability by time-multiplexing across a set of preconfigured topologies. However, both approaches could easily overburden the SDN controller with microsecond-level reconfigurations. In [57], the authors explored the possibility of reducing reconfiguration without impacting performance using domain-sizing techniques; our work similarly aims to improve DCN performance with minimal reconfigurations, but exploits weak temporal stability in traffic patterns.

Online circuit-scheduling research is also relevant to reconfigurable networks, though the problem statement and approach fundamentally differs from those of ours. While circuit-scheduling is concerned with getting the optimal sequence of circuit configurations for a *single* traffic matrix [47], [58], [59], we are interested in optimizing a *single* topology for *many* TMs.

B. Traffic Engineering

COUDER relies on robust traffic engineering to deliver good performance. In contrast, many prior works on optical circuit-switched data centers performs traffic engineering based only on a single predicted traffic matrix [3], [4], [10], [60]. However, accurate traffic prediction is difficult, and an inaccurate prediction may lead to poor performance. More recent works on dynamic networks [8], [9], [17] have proposed using Valiant load-balancing (VLB) [61]. VLB is a traffic-oblivious routing algorithm that guarantees a throughput drop of no more than $2\times$ in the worst case, but suffers from poor performance and high bandwidth tax for common case traffic.

In terms of the core idea, COUDER shares many similarities with the robust traffic engineering (TE) works for wide area networks (WAN) space [62], [63], [64]. For instance, COPE [5] uses a dual-envelope approach to simultaneously optimize for both common- and worst- case traffic patterns. Unfortunately, these strategies that work in the context of TE cannot be generalized to ToE due to differences in problem structure. Specifically, ToE is a much harder combinatorics problem that involves optimizing *both* the topology *and* routing; conversely, in TE, only routing is optimized while the topology is fixed.

IX. Conclusion

We present COUDER, a robust topology engineering approach that does not rely on frequent reconfigurations to react to traffic changes. In contrast to prior ToE approaches that have generally relied on rapid OCS reconfiguration to handle traffic variations, COUDER designs inter-pod topologies based on multiple critical TMs extracted from historical traffic matrices, and adopts a desensitization technique to further enhance its topologies against unexpected bursts. Compared to static DCN topologies that do not use OCSs, COUDER shows clear performance benefits even with daily reconfiguration. Reconfiguring OCSs at such low frequencies greatly lowers the technological barrier to ToE deployment, thus paving a path towards the incremental adoption of optical circuit switched DCNs.

Acknowledgment

The authors would like to thank Calient Technologies for providing hardware support and sharing its insights on the stability and commercial availability of its MEMS OCS technology.

References

[1] S. Kandula, J. Padhye, and P. Bahl, "Flyways to de-congest data center networks," in *Proc. HotNets*, 2009.

[2] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social Network's (datacenter) network," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 123–137.

[3] N. Farrington et al., "Helios: A hybrid electrical/optical switch architecture for modular data centers," in *Proc. ACM SIGCOMM Conf.*, 2010, pp. 339–350.

[4] G. Wang et al., "C-through: Part-time optics in data centers," in *Proc. ACM SIGCOMM Conf.*, 2010, pp. 327–338.

[5] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "COPE: Traffic engineering in dynamic networks," in *Proc. SIGCOMM*, 2006, pp. 99–110.

[6] M. Ghobadi et al., "ProjecToR: Agile reconfigurable data center interconnect," in *Proc. ACM SIGCOMM Conf.*, Aug. 2016, pp. 216–229.

[7] G. Porter et al., "Integrating microsecond circuit switching into the data center," in *Proc. ACM SIGCOMM*, Aug. 2013, pp. 447–458.

[8] W. M. Mellette et al., "RotorNet: A scalable, low-complexity, optical datacenter network," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2017, pp. 267–280.

[9] H. Ballani et al., "Sirius: A flat datacenter network with nanosecond optical switching," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, Jul. 2020, pp. 782–797.

[10] N. Hamedazimi et al., "FireFly: A reconfigurable wireless data center fabric using free-space optics," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 319–330.

[11] A. Greenberg et al., "VL2: A scalable and flexible data center network," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2009, pp. 51–62.

[12] V. Liu, D. Halperin, A. Krishnamurthy, and T. E. Anderson, "F10: A fault-tolerant engineered network," in *Proc. NSDI*, 2013, pp. 399–412.

[13] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "JellyFish: Networking data centers, randomly," in *Proc. NSDI*, 2012, pp. 225–238.

[14] A. Valadarsky, M. Dinitz, and M. Schapira, "Xpander: Unveiling the secrets of high-performance datacenters," in *Proc. 14th ACM Workshop Hot Topics Netw.*, Nov. 2015, pp. 1–7.

[15] Y. Yu and C. Qian, "Space shuffle: A scalable, flexible, and high-performance data center network," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 11, pp. 3351–3365, Nov. 2016.

[16] M. Zhang, R. N. Mysore, S. Supittayapornpong, and R. Govindan, "Understanding lifecycle management complexity of datacenter topologies," in *Proc. NSDI*, 2019, pp. 235–254.

[17] W. M. Mellette, R. Das, Y. Guo, R. McGuinness, A. C. Snoeren, and G. Porter, "Expanding across time to deliver bandwidth efficiency and low latency," in *Proc. NSDI*, 2020, pp. 1–18.

[18] A. Singh et al., "Jupiter rising: A decade of clos topologies and centralized control in Google's datacenter network," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 183–197.

[19] N. Farrington and A. Andreyev, "Facebook's data center network architecture," in *Proc. Opt. Interconnects Conf.*, May 2013, pp. 49–50.

[20] *Cisco Data Center Spine-and-Leaf Architecture: Design Overview*, Cisco, San Jose, CA, USA, 2016.

[21] S. Kassing, A. Valadarsky, G. Shahaf, M. Schapira, and A. Singla, "Beyond fat-trees without antennae, mirrors, and disco-balls," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2017, pp. 281–294.

[22] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas. Conf.*, 2009, pp. 202–208.

[23] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th Annu. Conf. Internet Meas.*, 2010, pp. 267–280.

[24] *CALIENT Technologies*. Accessed: Jan. 2021. [Online]. Available: https://www.calient.net/

[25] H. Liu et al., "Circuit switching under the radar with REACToR," in *Proc. NSDI*, 2014, pp. 1–15.

[26] A. Singla, A. Singh, K. Ramachandran, L. Xu, and Y. Zhang, "Proteus: A topology malleable data center network," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw.*, 2010, pp. 1–6.

[27] Google. (2015). *Pulling Back the Curtain on Google's Network Infrastructure*. [Online]. Available: https://goo.gl/hx0vz3

[28] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, omega, and kubernetes," *Commun. ACM*, vol. 59, no. 5, pp. 50–57, Apr. 2016.

[29] (2021). *Telescent G4 Network Topology Manager*. [Online]. Available: https://www.telescent.com/products

[30] A. Chatzieleftheriou, S. Legtchenko, H. Williams, and A. Rowstron, "Larry: Practical network reconfigurability in the data center," in *Proc. NSDI*, 2018, pp. 141–156.

[31] CALIENT Technologies. (Mar. 2021). *Calient S-Series: S320*. [Online]. Available: https://www.calient.net/resources/datasheets/

[32] (2021). *Polatis Optical Circuit Switch*. [Online]. Available: https://www.polatis.com/series-7000-384x384-port-software-controlled-optical-circuitswitch-sdn-enabled.asp

[33] K. Wen et al., "Flexfly: Enabling a reconfigurable dragonfly through silicon photonics," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2016, pp. 166–177.

[34] T. J. Seok, N. Quack, S. Han, R. S. Ḿuller, and M. C. Wu, "Large-scale broadband digital silicon photonic switches with vertical adiabatic couplers," *Optica*, vol. 3, no. 1, pp. 64–70, Jan. 2016.

[35] T. Chu, L. Qiao, W. Tang, D. Guo, and W. Wu, "Fast, high-radix silicon photonic switches," in *Proc. Opt. Fiber Commun. Conf.*, 2018, pp. 1–3.

[36] X. Zhou et al., "Mirror mirror on the ceiling: Flexible wireless links for data centers," in *Proc. SIGCOMM*, 2012, pp. 443–454.

[37] L. Poutievski et al., "Jupiter evolving: Transforming Google's datacenter network via optical circuit switches and software-defined networking," in *Proc. ACM SIGCOMM Conf.*, Aug. 2022, pp. 66–85.

[38] 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 *Proc. NSDI*, 2019, pp. 221–234.

[39] Huawei. *CloudEngine 12800 Series Data Center Switches*. [Online]. Available: https://e.huawei.com/us/products/enterprise-networking/switches/data-center-switches/ce12800

[40] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *Proc. Int. Symp. Comput. Archit.*, Jun. 2008, pp. 77–88.
[41] M. Y. Teh, J. J. Wilke, K. Bergman, and S. Rumley, "Design space exploration of the Dragonfly topology," in *Proc. Int. Conf. High Perform. Comput.*, 2017, pp. 57–74.
[42] A. Andreyev, "Introducing data center fabric, the next-generation Facebook data center network," Facebook, Menlo Park, CA, USA, Tech. Rep., 2014. [Online]. Available: https://code.facebook.com/posts/360346274145943
[43] M. Y. Teh, Z. Wu, M. Glick, S. Rumley, M. Ghobadi, and K. Bergman, "Performance trade-offs in reconfigurable networks for HPC," *J. Opt. Commun. Netw.*, vol. 14, no. 6, p. 454, 2022.
[44] C. Delimitrou, S. Sankar, A. Kansal, and C. Kozyrakis, "ECHO: Recreating network traffic maps for datacenters with tens of thousands of servers," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Nov. 2012, pp. 14–24.
[45] G. Bernstein, "Navigating cabling options for enterprise and cloud data centers," Leviton Netw. Solutions, Bothell, WA, USA, Tech. Rep. 29, 2019.
[46] Broadcom. (2016). *Tomahawk 4 / BCM56990 Series*. [Online]. Available: https://docs.broadcom.com/doc/12398014
[47] H. Liu et al., "Scheduling techniques for hybrid circuit/packet networks," in *Proc. 11th ACM Conf. Emerg. Netw. Experiments Technol.*, Dec. 2015, pp. 1–13.
[48] Gurobi Optimization. (2019). *Gurobi Optimizer Reference Manual*. [Online]. Available: http://www.gurobi.com
[49] R. W. Irving and M. R. Jerrum, "Three-dimensional statistical data security problems," *SIAM J. Comput.*, vol. 23, no. 1, pp. 170–184, Feb. 1994.
[50] A. V. Goldberg and M. Kharitonov, "On implementing scaling push-relabel algorithms," in *Network Flows and Matching: First DIMACS Implementation Challenge*. Providence, RI, USA: American Mathematical Society, 1993.
[51] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum-flow problem," *J. ACM*, vol. 35, no. 4, pp. 921–940, Oct. 1988.
[52] P. Johnson. (2017). *With the Public Clouds of Amazon, Microsoft and Google, Big Data is the Proverbial Big Deal*. [Online]. Available: https://www.forbes.com/sites/johnsonpierr/2017/06/15/with-the-public-clouds-of-amazon-microsoft-and-google-big-data-is-the-proverbial-big-deal/?sh=80b1efb2ac3c
[53] M. Y. Teh. (2021). *First Release of Reconfigurable Network Topology Evaluation Framework*. [Online]. Available: https://doi.org/10.5281/zenodo.4897956
[54] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 59–62, 2006.
[55] (2017). *Netbench*. [Online]. Available: https://github.com/ndal-eth/netbench
[56] M. Alizadeh et al., "Data center TCP (DCTCP)," in *Proc. ACM SIGCOMM Conf.*, 2010, pp. 63–74.
[57] G. C. Sankaran and K. M. Sivalingam, "Domain sizing in optical traffic grooming based data center networks," in *Proc. IEEE 4th Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2015, pp. 94–99.
[58] S. Bojja Venkatakrishnan, M. Alizadeh, and P. Viswanath, "Costly circuits, submodular schedules and approximate Carathéodory theorems," in *Proc. SIGMETRICS*, 2016, pp. 75–88.
[59] M. Wang et al., "Neural network meets DCN: Traffic-driven topology adaptation with deep learning," in *Proc. Abstr. ACM Int. Conf. Meas. Modeling Comput. Syst.*, Jun. 2018, pp. 1–25.
[60] M. Y. Teh et al., "TAGO: Rethinking routing design in high performance reconfigurable networks," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2020, pp. 1–16.
[61] R. Zhang-Shen and N. McKeown, "Designing a fault-tolerant network using valiant load-balancing," in *Proc. 27th Conf. Comput. Commun.*, Apr. 2008, pp. 2360–2368.
[62] C. Zhanga, J. Kurosea, D. Towsley, Z. Ge, and Y. Liu, "Optimal routing with multiple traffic matrices tradeoff between average and worst case performance," in *Proc. 13TH IEEE Int. Conf. Netw. Protocols (ICNP)*, 2005, pp. 215–224.
[63] Y. Chang, S. Rao, and M. Tawarmalani, "Robust validation of network designs under uncertain demands and failures," in *Proc. NSDI*, 2017, pp. 347–362.
[64] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2003, pp. 313–324.
[65] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. ACM*, vol. 19, no. 2, pp. 248–264, Apr. 1972.

Min Yee Teh received the bachelor's degree in electrical engineering from Brown University in 2016 and the M.S. and Ph.D. degrees in electrical engineering from Columbia University in 2017 and 2021, respectively. His Ph.D. work was supported by the Wei Family Foundation Fellowship. His research interests lie in reconfigurable optical circuit-switched networks, network load balancing, and data center network topology design.

Shizhen Zhao received the bachelor's degree from Shanghai Jiao Tong University in 2010 and the Ph.D. degree from Purdue University in 2015. From September 2015 to January 2019, he worked at Google's Networking Team. He is currently a Tenure-Track Associate Professor with the John Hopcroft Center, Shanghai Jiao Tong University. He has published papers in top-tier conferences and journals, including NSDI, SIGMETRICS, MOBICOM, ICNP, INFOCOM, IEEE/ACM TRANSACTIONS ON NETWORKING, and IEEE TRANSACTIONS ON AUTOMATIC CONTROL. His current research interests include optimizing optical circuit-switched data center networks.

Peirui Cao received the B.S. degree from Southeast University and the M.S. degree from Beihang University. He is currently pursuing the Ph.D. degree with the John Hopcroft Center for Computer Science, Shanghai Jiao Tong University. His research interests include optimizing network performance, data center networks, and network measurement.

Keren Bergman (Fellow, IEEE) received the B.S. degree in electrical engineering from Bucknell University in 1988 and the M.S. and Ph.D. degrees in electrical engineering from M.I.T. in 1991 and 1994, respectively. She is currently the Charles Batchelor Professor in electrical engineering at Columbia University, where she also works as the Faculty Director of the Columbia Nano Initiative. At Columbia, she leads the Lightwave Research Laboratory, encompassing multiple cross-disciplinary programs at the intersection of computing and photonics. She serves on the Leadership Council of the American Institute of Manufacturing (AIM) photonics leading projects that support the institute's silicon photonics manufacturing capabilities and Datacom applications. She was a recipient of the 2016 IEEE Photonics Engineering Award. She is a fellow of the Optica.