

METTEOR: Robust Multi-Traffic Topology Engineering for Commercial Data Center Networks

Min Yee Teh
Columbia University

Shizhen Zhao
Shanghai Jiao Tong University

Keren Bergman
Columbia University

ABSTRACT

Numerous optical circuit switched data center networks have been proposed over the past decade for higher capacity, though commercial adoption of these architectures have been minimal so far. One major challenge commonly facing these architectures is the difficulty of handling bursty traffic with optical circuit switches (OCS) with high switching latency. Prior works generally rely on fast-switching OCS prototypes to better react to traffic changes via frequent reconfigurations. This approach, unfortunately, adds further complexity to the control plane.

We propose METTEOR, an easily deployable solution for optical circuit switched data centers, that is designed for the current capabilities of commercial OCSs. Using multiple predicted traffic matrices, METTEOR designs data center topologies that are less sensitive to traffic changes, thus eliminating the need of frequently reconfiguring OCSs upon traffic changes. Results based on extensive evaluations using production traces show that METTEOR increases the percentage of direct-hop traffic by about 80% over a fat tree at comparable cost, and by about 35% over a uniform mesh, at comparable maximum link utilizations. Compared to ideal solutions that reconfigure OCSs on every traffic matrix, METTEOR achieves close-to-optimal bandwidth utilization even with biweekly reconfiguration. This drastically lowers the controller and management complexity needed to perform METTEOR in commercial settings.

1 INTRODUCTION

Given the exponential growth in data center traffic, building networks that meet the requisite bandwidth has also become more challenging. Modern data center networks (DCN) typically employ multi-rooted tree topologies [34], which have a regular structure and redundant paths to support high availability. However, uniform multi-rooted trees are inherently suboptimal structures to carry highly skewed traffic common to DCNs [31, 48]. This has motivated several works on using optical circuit switches (OCS) to design more performant data center architectures [17, 55]. Compared to conventional electrical packet switches, OCSs offer much higher bandwidth and consumes less power. More importantly, OCSs introduces the possibility of Topology Engineering (ToE), which allows DCNs to dynamically allocate more capacity between “hot spots” to alleviate congestion.

Despite showing immense promise, optical circuit-switched data centers have not been widely deployed even

after a decade’s worth of research efforts. One of the most daunting challenges is to perform ToE under bursty traffic. Early works on ToE proposed reconfiguring topology preemptively using a single estimated traffic matrix (TM) [17, 55]. However, the bursty nature of DCN traffic makes forecasting TMs accurately very difficult [4, 32]. An inaccurate prediction may lead to further congestion. Even if predictions were accurate, the forecast could still turn stale if topology reconfiguration takes tens of milliseconds. Subsequent works have thus focused on designing OCSs with microsecond-level switching latency [20, 41, 42, 46], to enable faster reaction to traffic burst. However, these proposals require changing topology and routing frequently, an act that introduces significant complexity to the control plane, thus hindering the adoption by large vendors.

We tackle bursty DCN traffic from a different perspective, using a robust optimization-based ToE framework called METTEOR (Multiple Estimated Traffic Topology Engineering for Optimized Robustness). While prior works optimize topology for a *single* estimated traffic matrix [26, 55], our approach optimizes topology based on *multiple* traffic matrices (TM). Traffic uncertainty is captured by a set of multiple TMs. Optimizing topology using this set helps desensitize the topology to traffic uncertainties. To our knowledge, METTEOR is the first framework that tackles ToE from a robust optimization approach. The most compelling advantage of METTEOR is that it does not rely on frequent OCS reconfiguration to handle traffic changes, as long as the new traffic is captured by a traffic set, thus reducing the management complexity in commercial data centers. In fact, METTEOR shifts the major complexity of ToE from the system control aspect to the algorithm design aspect. Designing an optimal topology for multiple TMs is an immensely challenging problem [19, 66]. We first formalize the overall problem in §5, and discuss various techniques used for relaxing the algorithmic complexity in §6.

We apply METTEOR to the core layer of data centers. Based on traffic analysis of production data center traces, we found that while pod-level traffic do not exhibit strong temporal stability, they do exhibit a weaker form of temporal stability, which we refer to as traffic recurrence. This recurrent behavior in traffic leads to a slow-varying clustering effect, which is a novel observation in DCN traffic characteristics. By optimizing topology based on these slow-varying

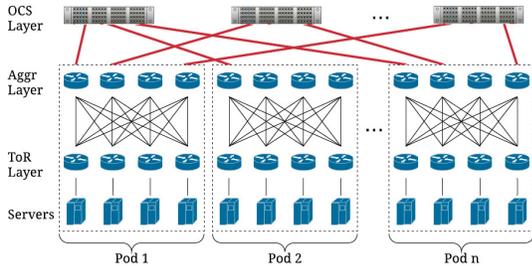


Figure 1: Physical topology model, with pods fully-interconnected via OCSs at the core layer.

clusters can achieve great performance without frequent re-configuration. Because of the low reconfiguration frequency, METTEOR requires minimal changes to the data center control plane, and thus can be viewed as a first step towards fully optical circuit switched data centers.

We evaluate METTEOR’s performance under different routing algorithms that minimize maximum link utilization (MLU). Based on production data centers traces, METTEOR increases the percentage of direct-hop traffic by about 80% over a fat tree at comparable cost, and by about 35% over a uniform mesh, at comparable maximum link utilizations (MLU). (However, the tail MLU of METTEOR may suffer if routing uncertainty exists.) Further, METTEOR with ideal routing performs close to an idealized ToE that requires instantaneous switching and frequent reconfigurations. Note that using METTEOR, we can obtain this level of performance with fortnightly OCS reconfiguration, making it deployable with the current off-the-shelf OCSs¹. Moreover, METTEOR is less dependent on the frequency of topology reconfigurations for good performance, when compared with the ToE solutions that optimize topology based on a single traffic matrix.

2 RELATED WORK

2.1 Traffic-Agnostic DCN Topology

DCN topologies have been traditionally designed to be static and traffic-agnostic, focusing on bisection bandwidth, scalability, failure resiliency, etc. They can be divided into either Clos-like and mesh-like topologies. Clos topology (e.g., Fat-Tree [1, 38]) is more widely-adopted in large-scale data centers (e.g., Google [50], Facebook [16], Cisco [12], and Microsoft [24]), as its regular hierarchical structure simplifies routing and congestion control. Mesh-like expander topology [51, 54, 61] also shows great promise, as its flatter hierarchy saves cost by eliminating the spine layer in Clos, while still offering rich capacity and path diversity.

However, DCN traffic is inherently skewed. A study from Microsoft [31] showed that only a few top-of-rack (ToR) switches are “hot” in a small (1500-server) production data center. Facebook [48] reported that the inter-pod traffic in

¹To approximate ideal routing does require frequent routing update. Fortunately, routing update can be much easier than OCS reconfiguration.

one of their data centers varies over more than seven orders of magnitude. As a result, traffic-agnostic networks can be inherently suboptimal under skewed DCN traffic.

2.2 Traffic-Aware DCN Topology

To handle fast-changing, high-skewed traffic patterns, some researchers have argued for reconfigurable DCN topologies based on optical circuit switches (OCS) [18, 35, 53, 67]. The pioneering work, Helios [17], proposed reconfiguring pod-to-pod topology using OCSs based on a *single* estimated traffic matrix. However, reconfiguring Helios incurs a significant delay (about 30ms), a problem that most commercial OCSs today still face [9]. Given that 50% of DCN flows lasting below 10ms [32], a 30ms reconfiguration latency could mean that the topology optimized for pre-switching traffic may no longer be a good fit for post-switching demands.

The need to cope with rapid traffic changes motivated subsequent works aimed at decreasing reconfiguration latency for OCSs. Some of these have focused on providing ToR-level reconfigurability [36, 52, 55], potentially reducing latency to microseconds level using sophisticated hardware. However, these approaches might not scale to data centers with thousands of ToRs, due to the low radix of ToRs and the finite size of OCSs. Others have proposed scaling up reconfigurable networks with steerable wireless transceivers [20, 27, 68], but these architectures face serious deployment challenges related to environmental conditions in real DCNs, and to the need for sophisticated steering mechanisms. The Opera architecture [40], built using rotor switches from [41], forms a mesh-like expander topology by multiplexing a set of pre-configured matchings in the time-domain. Unfortunately, frequently changing OCS connections may overload the SDN controller, and thus undermine data center availability.

Another line of work have looked into better algorithms that schedule circuits more optimally in the presence of reconfiguration delays [6, 37, 57]. However, the assumed problem setups of these works fundamentally differs from ours, as we are interested in designing a *single* topology optimized for many possible traffic demands.

2.3 Traffic Engineering

To fully realize the potential of reconfigurable topologies, traffic engineering (TE), is required. TE typically consists of two phases: 1) the path-selection phase, and 2) the load-balancing phase. The path-selection phase selects a set of candidate paths for carrying traffic. Given a selection of paths, the load-balancing phase then computes the relative weights for sending traffic along the candidate paths.

Path-selection in data centers typically employs the K-shortest-path algorithm [51, 54, 60]. As for load balancing,

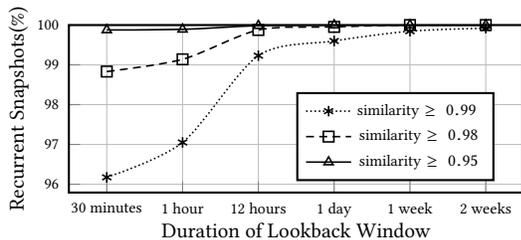


Figure 2: Percentage of TM snapshots with at least one historical “lookalike”. Two snapshots are considered “lookalikes” if their cosine similarity exceeds a given threshold.

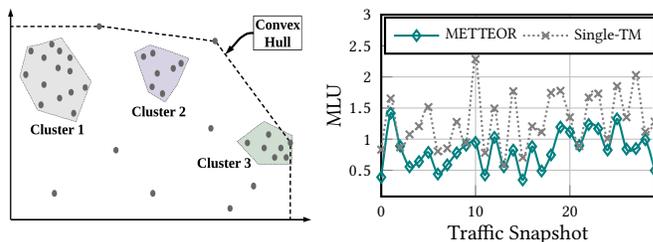
nearly all the related works [17, 27, 55] on optical circuit-switched data centers compute the relative weights by solving a multi-commodity flow (MCF) problem using a single predicted traffic matrix. However, predicting a traffic matrix accurately can be difficult, and an inaccurate traffic prediction may incur unexpected congestion. Rotornet [41] load-balances traffic using Valiant load-balancing (VLB) [65]. VLB has several desirable properties, such as being traffic-agnostic and robust under demand uncertainties by routing traffic via indirect paths, and having a worst-case throughput-reduction of $2\times$. However, DCN operators tend to have a strong sense of what traffic patterns may likely occur, based on a wealth of historical traffic data. This makes VLB overly conservative. Some TE literatures use robust optimization to strike a balance between network performance and robustness to traffic uncertainty [10, 56, 62]. Although these solutions are mainly designed for wide area networks (WAN), the core ideas are equally applicable to DCNs.

3 MOTIVATING METTEOR

3.1 Recurrence—A Weaker Form of Stability

The conventional wisdom in ToE is to switch topology as frequently as possible to handle demand changes. The belief that DCN traffic lacks stability has driven much work on designing faster OCSs and control planes. However, DCN traffic is not entirely random, especially at the pod level. In fact, while pod-level traffic matrices (TM) do not generally exhibit strong stability over time, they do exhibit a weaker form of temporal stability, which we refer to as *traffic recurrence*. This means that while most traffic snapshots may not be close to the snapshot preceding them, it is very likely that a similar TM has occurred in the recent past.

To quantify this phenomenon, we performed a simple case study on recurrence using 6 months’ worth of TM snapshots obtained from a data center. Each TM snapshot is a 5-minute-average of inter-pod traffic. We present results from 1 data center out of the 12 studied, though all other DCNs exhibit similar results. A TM snapshot is considered recurrent if it is close to at least one past TM within an observation period. The “closeness” between two TMs is measured with cosine similarity [58]. Fig. 2 plots the percentages of recurrent TM



(a) Multi-traffic ToE concept **(b) MLU timeseries**

Figure 3: METTEOR concept illustration.

snapshots as a function of the lookback window (i.e., how far back in the past we look). When closeness is loosely-defined (i.e. similarity ≥ 0.95), almost all snapshots are recurrent even with a 30-minute lookback window. When considering closeness as similarity ≥ 0.99 , over 96% of snapshots are recurrent within a 30-minute lookback window. Regardless of how closeness is defined, nearly all TMs are recurrent with a 2-week lookback window. This property of weak temporal stability may partially explain the slow-varying clustering effects in traffic patterns, which we explore in §7.

3.2 Toy Example - METTEOR

Fig. 3a shows a proof-of-concept for METTEOR. Clearly, no single TM can adequately represent all TMs properly in this case, so single-traffic-based ToE approaches, as in [17, 27, 55], may not work well. Our approach accounts for traffic uncertainty by optimizing topologies based on multiple TMs. When traffic is recurrent, many observed TMs will likely reappear in the future. With topologies optimized for a few representative TMs derived from historical snapshots, METTEOR could perform well for future recurring traffic.

Using a simple experiment, we motivate the use of METTEOR. In this example, we consider a network with 8 pods interconnected via an OCS layer in a manner similar to that in Fig. 1; each pod has 100 directed links of unit capacity. We generate 30 traffic matrices at random. For METTEOR, we find 3 traffic centroids using κ -means clustering algorithm, and optimize topology based on the 3 TMs. For comparison, we also optimize topology based on the average of all TMs. Then, for each TM, we compute the maximum link utilization (MLU) of routing each TM over the two topologies.

Fig. 3b shows the MLU performance. Clearly, METTEOR performs better, as it is able to design topology that is well-suited for most of the traffic snapshots. Under traffic uncertainties, a multi-traffic optimization approach may improve solution robustness by minimizing topology-overfitting to a single predicted demand.

4 METTEOR - SYSTEM LEVEL OVERVIEW

4.1 Network Architecture

The assumed DCN topology is shown in Fig. 1, with a layer of OCSs interconnecting all pods, each constructed

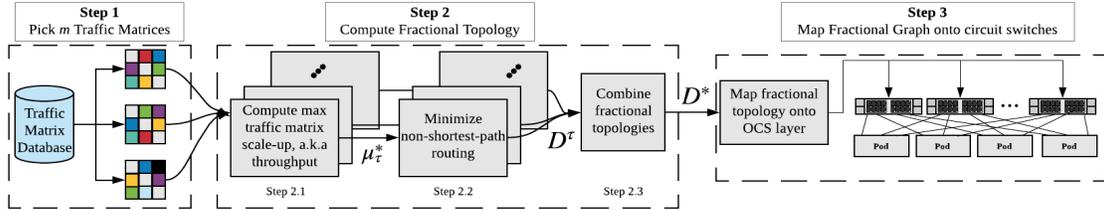


Figure 4: Illustrating the complete software workflow of METTEOR.

from packet switches. This topology resembles a Clos typically seen in large scale data centers, although we replace the core switches with OCSs. Like Helios [17], our work employs inter-pod reconfigurability, which deviates from some architectures that argue for inter-ToR reconfigurability [20, 27, 46, 55, 68]. We chose inter-pod reconfigurability over inter-ToR reconfigurability for the following reasons:

- *Scalability*: - Using pods with hundreds of uplinks to the OCSs, and downlinks to ToRs for $\Theta(1000)$ servers, our architecture can scale up to over 100k servers.
- *Traffic stability*: Inter-pod traffic shows more noticeable locality [48], and is more stable than inter-ToR traffic [13, 32] due to averaging effects from the aggregation switches.
- *High fan-out*²: Pods have much higher fan-out than ToRs. Combined with multi-hop routing, every pod is reachable within one or two hops, making it possible for *one* logical topology to serve several, possibly dense³, TMs.

In this paper, we refer to the (fixed) physical connections between the pod and OCSs as the *physical topology*. Topology engineering reconfigures the OCSs to realize a specific *logical topology* as an overlay on the physical topology.

4.2 Computing Logical Topology

Prior works have designed reconfigurable topology based on a *single* estimated traffic matrix, obtained either from switch measurements (e.g. Hedera [2]) or from end-host buffer occupancy [55]. However, due to the bursty nature of DCN traffic [32], even inter-pod traffic can be difficult to predict accurately, which fundamentally limits the robustness of such an approach.

Therefore, we compute logical topologies based on multiple TMs instead. The first step is to obtain multiple TMs that will be representative of future traffic (see Step 1 of Fig. 4), based on historical traffic snapshots. Traffic snapshots can be easily obtained from flow-monitoring tools like sFlow [45]. While we could get an accurate traffic estimation directly from applications, this would require application level modifications. Instead, we employ a simpler approach that exploits the spatial-temporal traffic behavior of production traffic to extract multiple representative TMs (see §7). The next step is to optimize topology for the extracted TMs

²Ability to form direct links with many destinations..

³While inter-ToR traffic matrices is quite sparse [20], inter-pod traffic matrices tend to be dense, with mostly non-zero entries.

(see §5 and §6), which is the biggest challenge of this paper. In fact, the topology optimization problem for even a single TM is already NP-complete; having multiple TMs further complicates this problem. Our goal is to design a polynomial-time heuristic to this problem. The algorithm design must be done carefully. Otherwise, a poorly-design topology can easily nullify the potential benefits of ToE.

4.3 Reconfiguring Logical Topology Safely

Despite having shown great promise on paper, ToE has not seen widespread commercial adoption. One key reason is that existing reconfigurable architectures do not consider high network availability. Network availability is generally defined as a high-level service level objective (SLO), measured as a number of “nines” in service uptime [23, 28]. Under the hood, however, availability is inextricably linked to factors like traffic volume, controller workload, hard/software failure rates, and packet loss [43].

Performing ToE frequently, if not done properly, could be detrimental to availability. For instance, high-frequency switching places a tremendous workload on the SDN controller. A poor-choice of switching configuration, or even a bug, risks failing entire DCN blocks; an admittedly rare risk, but one that increases with the rate of reconfiguration.

There are two major considerations when reconfiguring topology. First, reconfiguration must be carefully sequenced to avoid routing packets into “black holes.” For each reconfiguration event, 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.

Second, topology reconfiguration needs to be staged to maintain sufficient network capacity, especially when traffic demands are high. For instance, if 40% of links need to be reconfigured when network utilization is at 80%, the reconfiguration process must take at least 2 stages (switching 20% of links in each stage) to avoid congestion and packet loss due to over-utilization.

4.4 Bootstrapping Greenfield DCNs

METTEOR requires a sufficient history of TMs to find the right clusters. However, when a greenfield DCN is initially

deployed, or when new pods are added during DCN expansion, there are not sufficient traffic data to locate the correct traffic clusters. So, the initial configuration should aim for a uniform logical topology, and route traffic evenly along both direct and indirect paths. This reduces the risks of maximum congestion due to traffic bursts, at the cost of poor bandwidth tax performance as most traffic will traverse indirect paths. Once sufficient historical traffic measurement is available, then METTEOR can be triggered. Based on our experience, one week’s worth of traffic snapshots should suffice.

5 FORMALIZING METTEOR

We now formalize the mathematics of METTEOR. All notations are tabulated in Table 1.

5.1 Logical Topology

Let $\mathcal{S} = \{s_1, \dots, s_n\}$ be the set of pods, $\mathcal{O} = \{o_1, \dots, o_y\}$ be the set of OCSs, and x_{ij}^k be the number of links from pod s_i to pod s_j through OCS o_k . We represent a logical topology using $X = [x_{ij}]$, $i, j = 1, \dots, n$, where $x_{ij} = \sum_{k=1}^y x_{ij}^k$ is the number of links between pods s_i and s_j . The logical topology X **must** be feasible under a given physical topology, so it must satisfy the following group of constraints.

OCS-level (Hard) Physical Constraints:

$$\begin{cases} \sum_{j=1}^n x_{ji}^k \leq h_{\text{ig}}^k(i), \sum_{j=1}^n x_{ij}^k \leq h_{\text{eg}}^k(i), \forall i = 1, \dots, n, k = 1, \dots, y; \\ x_{ij} = \sum_{k=1}^y x_{ij}^k, \quad x_{ij} \text{ and } x_{ij}^k \text{ are all integers;} \end{cases} \quad (1)$$

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

5.2 Network Throughput

Let $T = [t_{ij}]$, $i, j = 1, \dots, n$ be a TM, where t_{ij} is the traffic rate (in Gbps) from pod s_i to pod s_j . Given a logical topology X , we measure its throughput μ w.r.t. T , such that μT is the maximum scaled TM that can be feasibly routed over the topology X . Routing feasibility is defined as follows.

Let $\mathcal{P} = \cup_{(i,j)} \mathcal{P}_{ij}$ be the candidate path set for all pod pairs (s_i, s_j) , where \mathcal{P}_{ij} is the set of candidate paths from s_i to s_j . We allow no more than two hops between pods, so $\mathcal{P}_{ij} = \{[s_i, s_j], [s_i, s_1, s_j], \dots, [s_i, s_n, s_j]\}$. The feasibility of routing T over X can be verified using:

$$\begin{aligned} &\text{Find } \Omega = \{\omega_p\}, p \in \mathcal{P} \text{ such that} & (2) \\ &1) \sum_{p \in \mathcal{P}_{ij}} \omega_p = t_{ij}, \forall i, j = 1, \dots, n \\ &2) \sum_{p \in \mathcal{P}, (s_i, s_j) \text{ is a link in } p} \omega_p \leq x_{ij} b_{ij}, \forall i, j = 1, \dots, n \end{aligned}$$

where b_{ij} is the link capacity between s_i and s_j , and ω_p is the traffic routed (in Gbps) via path p .

When computing throughput, we scale T until max link utilization (MLU) hits 1, where link utilization is the ratio of a link’s traffic flow rate to its capacity. As it turns out, this problem (2) is related to that of minimizing max link utilization (MLU) when routing an unscaled T over X . Thus, a lower MLU implies that there is more room for T to grow before MLU hits 1, which leads to higher throughput.

5.3 Design Objective

Given m traffic matrices, $\{T_1, \dots, T_m\}$, let $\{\mu_1, \dots, \mu_m\}$ be the throughputs of routing $\{T_1, \dots, T_m\}$ over X . We aim to design X such that $\min(\mu_1, \dots, \mu_m)$ is maximized:

$$\max_X \mu = \min\{\mu_1, \dots, \mu_m\}, \text{ s. t} \quad (3)$$

- 1) X is an integer matrix that satisfies (1)
- 2) $(X, \mu_\tau T_\tau)$ satisfies (2), $\forall \tau \in \{1, \dots, m\}$
- 3) The majority of traffic in T_τ is routed through direct paths in X , $\forall \tau \in \{1, \dots, m\}$

Note that (3)’s formulation ensures that the logical topology maximizes all TM throughputs as evenly as possible. Although we could maximize the total throughput of all TMs, we avoid this as it gives the logical topology freedom to selectively-optimize the throughputs of the “easier” TMs.

Solving (3) gives us the optimal logical topology. However, the runtime complexity scales exponentially with the number of pods and OCSs, which is too challenging for commercial solvers like Gurobi [25]. Some prior work has studied traffic engineering (TE) techniques based on multiple TMs [62, 63]. Unfortunately, those techniques cannot be applied here, as ToE, unlike TE, requires integer solutions.

The complexity of (3) is imposed by the structure of the physical topology. Since OCSs have limited radix, and the OCS layer may involve $\sim 10k$ links, this layer must use multiple OCSs. Using multiple OCSs, rather than one giant OCS, makes this optimization a strongly NP-complete combinatorics problem [19, 66]. Since tackling (3) head on is infeasible, we split the overall problem into smaller subproblems.

6 OVERALL METHODOLOGY

Next, we discuss the techniques employed to sidestep the complexity of (3). Specifically, we split the overall problem into steps 2 and 3 of Fig. 4.

First, we design a *fractional* logical topology (Step 2 in Fig. 4) that optimizes throughput for all TMs, instead of computing an integer solution directly. Without the integer requirement, this step can be solved using linear programming (LP). Next, we configure the OCSs such that the *integer* logical topology best approximates the fractional topology (Step 3 of Fig. 4). These steps are detailed in §6.1 and §6.2.

$\mathcal{S} = \{s_1, \dots, s_n\}$	Set of all n pods
$\mathcal{O} = \{o_1, \dots, o_y\}$	Set of all y circuit switches
x_{ij}^k	Integer number of pod i 's egress links connected to pod j 's ingress links through o_k
$X = [x_{ij}] \in \mathbb{N}^{n \times n}$	Inter-pod topology; x_{ij} denotes the number (integer) of s_i egress links connected to ingress links of 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 s_i egress links connected to ingress links of s_j
$h_{eg}^k(s_i), h_{ig}^k(s_i)$	Number of physical egress and ingress links, respectively, connecting s_i to o_k
r_{eg}^i, r_{ig}^i	Number of egress and ingress links, respectively, of s_i
b_{ij}	Link capacity (Gbps) between s_i and s_j
\mathcal{P}_{ij}	Set of all routing paths from s_i to s_j
ω_p	Traffic (Gbps) on path p
μ	Traffic scale-up factor

Table 1: Notations used in this paper

6.1 Computing Fractional Topology

Before proceeding, we need to define *fractional topology*.

Definition 6.1. Given a set of pods $\mathcal{S} = \{s_1, \dots, s_n\}$ and the number of ingress & egress links $r_{ig}^i, r_{eg}^i, D = [d_{ij}] \in \mathbb{R}^{n \times n}$ is a fractional topology iff it satisfies:

$$\sum_{j=1}^n d_{ij} \leq r_{eg}^i, \sum_{i=1}^n d_{ij} \leq r_{ig}^j \quad \forall i, j = 1, \dots, n \quad (4)$$

A fractional topology, D , simply describes the inter-pod (fractional) link count, where each pod's in/out-degree constraints are satisfied. This definition noticeably ignores the OCSs; since the OCS layer will be considered when rounding the fractional topology into an integer logical topology, accounting for them here unnecessarily increases the number of variables needed for representation⁴.

Our goal is to design a fractional topology, D , that leads to good throughput for all the input TMs. Initially, we formulated an LP that computes the optimal D for all TMs:

$$\max_{D \text{ satisfies (4)}} \mu, \quad \text{s.t. } (D, \mu T_\tau) \text{ satisfies (2)}, \forall \tau \in \{1, \dots, m\} \quad (5)$$

However, the above formulation scales badly due to the large number of constraints when considering multiple TMs in one LP. To achieve scalability, we use a two-step approach: 1) compute the optimal fractional topology for every TM, and 2) combine the fractional topologies into one.

6.1.1 Fractional Topology for One Traffic Matrix.

We first compute a fractional topology for a single TM based on two routing metrics: throughput and average hop count. However, there is a tradeoff between these two metrics under a given topology. For instance, throughput may be increased

⁴As the number of ports of an OCS and a pod is comparable, the total number of OCSs, y , must be in the same order as the total number of pods n . Factoring in the OCS layer increases the variable space from $O(n^2)$ to $O(n^2 y)$, causing our solver to run out of memory for large fabrics.

if we allow non-shortest-path routing, but this can increase hop count. We want to find a fractional topology that gives a Pareto-optimal tradeoff between these metrics.

Given a TM, T , and a set of candidate paths, \mathcal{P} , we compute a fractional topology D in two steps. First, we compute D that maximizes throughput μ for T as follows:

$$\max_{\mu, D} \mu \quad \text{s. t. } (D, \mu T) \text{ satisfies (2)}. \quad (6)$$

Let μ^* be the optimal value of (6). There could be many fractional topologies that maximize throughput μ^* for T . We select the one that minimizes the usage of the non-shortest paths. Let $\mathcal{P}'_{ij} \subset \mathcal{P}_{ij}$ be the set of non-shortest paths in \mathcal{P}_{ij} . The formulation is as follows:

$$\min_{\Omega, D} \sum_{p \in \mathcal{P}'} (\omega_p)^2 \quad \text{s. t. } (D, \mu^* T) \text{ satisfies (2)}. \quad (7)$$

Note that average hop count can be reduced implicitly by minimizing the routing weights of non-shortest paths, thus solving (7) helps D meet the third requirement in the formulation (3). We opted for a quadratic objective function in (7) over a linear one due to its ‘‘sharper’’ landscape, which helps desensitize solution to slight TM input variations. **In our evaluation, we found this step instrumental in increasing the amount of direct-hop traffic overall.**

6.1.2 Combining Fractional Topologies.

Having computed D^τ for every $T^\tau, \tau = 1, \dots, m$, we then linearly combine them into one, D^* , which is then used to map onto the OCS layer. This can be formulated as:

$$\max_{\alpha > 0, D^*} \alpha, \quad \text{s. t. } d_{ij}^* \geq \alpha d_{ij}^\tau, \forall i, j = 1, \dots, n \text{ and } \tau = 1, \dots, m. \quad (8)$$

The constraint in (8) guarantees that the throughput of routing T^τ in the combined fractional topology D^* is at least α times of that of routing T^τ in D^τ . By maximizing α , D^* achieves a good balance among different fractional topologies in terms of throughput.

6.2 Mapping D^* onto the OCS Layer

We now map D^* onto the OCSs such that the integer logical topology, X , best approximates D^* .

6.2.1 Problem Setup.

The goal here is to decide the total number of links x_{ij}^k from pod s_i to pod s_j through OCS o_k , for every $i, j = 1, 2, \dots, n$ and $k = 1, 2, \dots, y$. Since there are y OCSs, we split each d_{ij}^* entry in D^* into y integers $x_{ij}^k, k = 1, \dots, y$, such that $\sum_{k=1}^y x_{ij}^k \approx d_{ij}^*$, which can be formulated as

Soft / Matching Constraints

$$[d_{ij}^*] \leq \sum_{k=1}^y x_{ij}^k \leq \lceil d_{ij}^* \rceil, \quad \forall i, j = 1, \dots, n \quad (9)$$

Then, a logical topology can be found by solving

$$\text{Find } \{x_{ij}^k\} \text{ satisfying (1) and (9).} \quad (10)$$

Solving (10) strictly is NP-Complete, as 3-Dimensional Contingency Table problem, proven to be NP-Complete [29], can be reduced to our problem. Fortunately, unlike the physical OCS constraint (1), constraint (9) is “soft”, which can be relaxed to reduce algorithmic complexity.

Initially, we tried two natural ideas for solving (10). The first is to solve it directly using ILP. This naive approach has an extremely high runtime complexity; it also cannot gracefully relax the soft constraints when satisfying (9) is infeasible. The second idea is to employ a greedy maximum matching as in Helios [17], which maps each OCS to a maximum weight matching subproblem based on D^* , and then greedily solves these subproblems. However, this greedy approach could violate so many soft constraints, such that the resulting logical topology X may no longer be a good estimate of D^* , causing poor network performance.

We wanted an approach that (a) has low complexity, (b) is mathematically sound, and (c) can *gracefully* relax soft constraints when necessary. The ILP approach only achieves (b), while the greedy algorithm only achieves (a). Inspired by convex optimization theories, we developed two algorithms that achieve all three criteria.

6.2.2 Algorithm Intuition.

One standard approach for relaxing hard-to-satisfy constraints is the Barrier Penalty Method (BPM) [8]. The idea is to transfer all the soft constraints into an objective function $U(X)$ that penalizes soft constraint-violation:

$$\min_{X \text{ satisfies (1)}} U(X) = \sum_{i=1}^n \sum_{j=1}^n \left[\left(\sum_{k=1}^y x_{ij}^k - \lfloor d_{ij}^* \rfloor \right) \left(\sum_{k=1}^y x_{ij}^k - \lceil d_{ij}^* \rceil \right) \right] \quad (11)$$

Since x_{ij}^k are all integers, it is easy to verify that $U(X) \geq 0$ and $U(X) = 0$ iff all the soft constraints in (9) are satisfied. When (9) is not satisfiable, minimizing $U(X)$ provides a graceful relaxation of (9). Still, computing an integer X directly requires exponential runtime. We instead compute x_{ij}^k for each OCS k iteratively, while keeping x_{ij}^k for all other OCSs constant. Using first order approximation, computing x_{ij}^k for a given OCS k can be mapped to a min-cost flow problem, which can be solved in polynomial time [15]. Unfortunately, since BPM weighs every (i, j) soft constraint equally in its objective function, we found that BPM suffers from an increased soft constraints violation when D^* is skewed. This finding highlights the BPM’s limited adaptability to a wide range of fractional topologies.

To address the shortcomings of BPM, we employ another approach based on the Lagrangian dual method (LDM) [39]. LDM relaxes the soft constraints (9) with dual variables that

can adapt to the skewness of D^* over multiple iterations, leading to fewer soft constraint violations. Our METTEOR implementation uses LDM precisely for its adaptability. The full derivations of the LDM and the BPM are in B.1 and B.2, respectively, followed by their optimality evaluation in D.

7 PICKING REPRESENTATIVE TRAFFIC

The first step in METTEOR’s workflow is to extract multiple representative TMs. We show how to extract these TMs purely from historical traces, while assuming no knowledge of the underlying application mix.

Recall Fig. 2 in §3 showed that inter-pod traffic exhibits a weak form of temporal stability, which we call recurrence. This behavior causes TMs to form clusters that vary slowly over time. But how exactly does traffic recurrence lead to clustering behavior? We offer an informal reasoning as follows. Consider a traffic matrix snapshot as a point in high-dimensional space. Over time, recurrent snapshots will begin to “congregate” within the vicinity of one another to form clusters, rather than scatter around uniformly in space.

The appearance of clustering effects is predicated of traffic exhibiting both spatial and temporal locality. The spatial locality is inherent to data center job placement. Large DCNs tend to assign different groups of pods to specific production areas, so pods belonging to the same production areas are more likely to communicate with one another. Meanwhile, the temporal locality comes from traffic recurrence, and this property is partially determined by user behavior. The regularity of usage patterns from long-term customers in cloud data centers, or the routine running of batched jobs (e.g. integration tests) in private data centers may all cause traffic recurrence.

7.1 Traffic Clustering Effect

7.1.1 Visualization of Traffic Clusters.

Traffic matrices are high dimensional data points (each point has $\Theta(\text{pod num}^2)$ dimensions), so visualizing their temporal evolution is exceedingly difficult. To this end, we employ principal component analysis (PCA) to reduce the dimensionality of the TM snapshots, and project each snapshot onto the plane formed by the first and second principal components [21]. Since an optimal topology for a TM remains optimal regardless of scaling, we should not distinguish TMs differ only in their total traffic volume. Thus, we normalize all TMs to 1.

Fig. 5 shows an example of traffic from one of the production data centers, with this projection represented as a 2-D heatmap, where brighter colors indicate areas with a higher occurrence. The proportion of variance explained (PVE) by the first two components is 91% of the total variation. Each plot covers about 24 days’ worth of traffic snapshots. There are noticeable clustering effects, which manifest as “clouds”

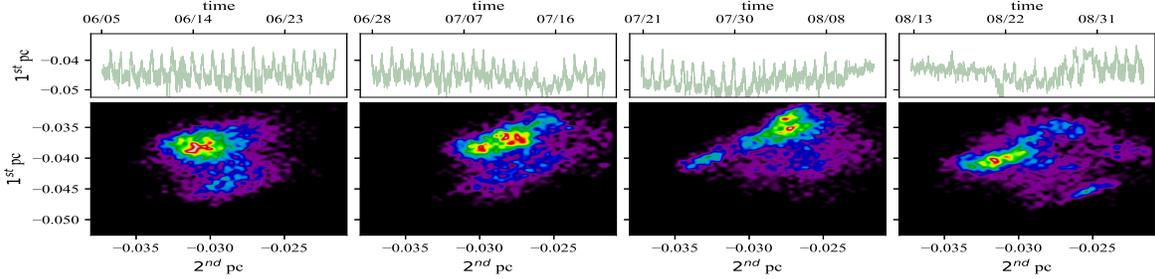


Figure 5: PCA on 3 months of production DCN traffic; each plot shows 24 days. Top: temporal variation along the first principal component. Bottom: PCAs as 2D-heatmaps; higher occurrence is represented by brighter colors.

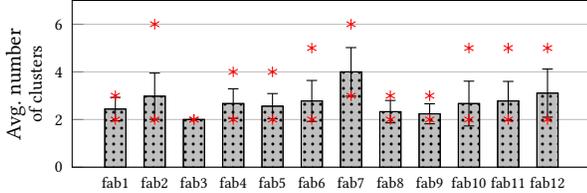


Figure 6: TM clustering statistics of 12 production DCNs. Six months of traffic snapshots are split into 10 segments of ~ 18 days each. The appropriate # of clusters of each segment is determined using silhouette method. Error bars denote the std dev, and asterisk marks denote max and min.

of bright patches. These clusters shift slowly over time, so topology reconfiguration to handle these shifts is necessary.

The top row in Fig. 5 shows the traffic temporal variation along the principal component. Note that while the clusters change slowly over the course of weeks, the variation along the principal component between snapshots is rather significant. The maximum variation from peak to trough accounts for about 5% of the total traffic, which is 85% of the largest inter-pod traffic, and 25 \times the average inter-pod traffic.

7.1.2 Clustering Statistics Across 12 Data Centers.

Next, we study the clustering effect across a fleet of 12 production DCNs. Using 6 months’ worth of historical traffic from each data center, we compute the optimal number of clusters for each 18-day period using the silhouette method [47]. This step repeats for 10 contiguous 18-day segments. Fig. 6 summarizes these statistics.

Across all 12 fabrics, the average “appropriate” number of clusters for each 18-day period is below 4. The appropriate number of clusters in each period, and how it evolves over time, are generally artifacts of the DCN’s underlying application mix and scheduler behavior. Therefore, the optimal number of clusters has to be determined individually for each fabric through traffic analysis.

The optimal number of clusters number changes from one 18-day segment to the next (see the error bar overlays in Fig. 6), though the deviations are small (within a ± 1 range of the average). This suggests that network operators can pick a consistent number of TMs used for METTEOR in all reconfiguration epochs for each fabric.

7.2 Finding Representative TMs

In theory, we could consider using the set of *all* historical TMs for optimization. Doing so guarantees coverage of any future traffic that is recurrent, but the runtime and memory complexity required to compute a topology for such a large set of TMs would also increase astronomically. Hence, to avoid adding significant computational complexity to METTEOR, we need to pick the smallest set of TMs that is still sufficiently representative of future traffic.

Though prior works have proposed effective methods for selecting traffic matrix estimators (e.g. CritMat [64]), we employ a simple, yet effective, κ -means clustering algorithm to find the centroids within the historical traffic snapshots. Computing these cluster centroids gives us a compact representation of historical traffic that retains much of the “features” of the historical traffic. As long as a future traffic snapshot is recurrent, there is a high probability it will be well-represented by at least one of these cluster centroids.

7.3 Accuracy of Cluster-based Prediction

Next, we test how well traffic clusters predict future traffic. First, we split 6 months’ worth of traffic into segments of two weeks. In cluster-based prediction, we extract κ cluster centroids from each 2-week segment, and use them to predict traffic in the next segment. We compare cluster-based prediction against two single-traffic-based predictions, namely *ave* and *max*, which pick the historical average and component-wise max values, respectively.

We use *cosine similarity* (defined in [58]) to evaluate how similar the predicted TM is to the actual TM. Given two TMs’ vector representations, \vec{T}_1, \vec{T}_2 , their cosine similarity $sim(\vec{T}_1, \vec{T}_2)$ measures how parallel (or similar) these two vectors are. If $sim(\vec{T}_1, \vec{T}_2)$ is close to 1, it follows that a \vec{T}_1 -optimized topology would also be close-to-optimal for \vec{T}_2 . For multiple representative TM cases, we pick the one that is most similar to the evaluated traffic snapshot.

Fig. 7 shows that cluster-based prediction yields higher accuracy than both *ave* and *max*, as they can more effectively capture long-term traffic behavior. The long tail of the $\kappa = 1$ curve indicates that fewer clusters may hurt worst-case accuracy, showing an inability to cover outlier TMs. Choosing

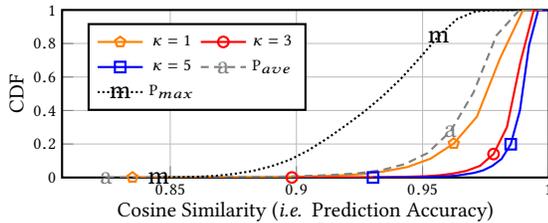


Figure 7: Accuracy of cluster-based v.s. single-traffic predictors in estimating future traffic.

$\kappa = 5$ over $\kappa = 3$ shows a diminishing improvement in prediction accuracy as κ increases. *max* has the lowest accuracy, as it captures the maximum element-wise demand that may not be representative in general.

8 PERFORMANCE EVALUATION

We now evaluate METTEOR’s performance over an extended timescale. The criteria we evaluate are: 1) performance comparison among different network topologies (§8.1), 2) performance robustness under different reconfiguration frequencies (§8.2), and 3) performance under routing uncertainties (§8.3). We assume a fluid traffic model to help us evaluate performance over extended periods, while still capturing the essential macroscopic properties.

Dataset: Our evaluations are driven by production DCN TM snapshots. Each snapshot captures inter-pod traffic over 5 minutes. The number of snapshots from each of the 12 simulated data centers totals up to 6 months’ worth of data (i.e. slightly over 50k snapshots per data center). We present a subset of our findings here; complete results are in A.

Metrics: The main metrics we look at are:

- *Link utilization* (LU) is a good indicator of link congestion, so a lower LU is preferred. However, MLU only reflects congestion at the busiest link, so we also look at the median LU to gauge the average case link congestion. Although LU cannot exceed 1 in practice because packets can be dropped, we allow LU to be greater than 1 in our evaluation, as it could reflect how severe the packet drop is.
- *Bandwidth tax* is the additional capacity on average needed to route traffic [40]. For instance, if 60% of traffic traverses indirect 2-hop paths and 40% of traffic traverses direct paths, then the bandwidth tax is $0.6 \times (2 - 1) + 0.4 \times (1 - 1) = 0.6$. Since we allow a maximum of 2 inter-pod hops for each packet in this paper, bandwidth tax is equal to the fraction of 2-hop traffic. Clearly, a lower bandwidth tax is preferred due to the following reasons. First, indirect paths increases packet latency. Second, lowering bandwidth tax directly lowers the number of concurrent flows going through each switch. As DCN switches typically have shallow buffers, lowering the number of concurrent flows through a switch helps reduce the probability of incast [11].

8.1 Topological Comparison

We first compare METTEOR with other DCN topologies.

Topology: Our main contender is METTEOR with the following settings. $\kappa = 4$ representative TMs are extracted from 2 weeks’ worth of historical traffic preceding each reconfiguration epoch, and the logical topology is reconfigured based on these representative TMs every two weeks.

Routing: We use traffic engineering (TE) for routing. As mentioned in §2.3, TE algorithms typically consists of a path-selection step, and a load balancing step. For path selection, we consider all paths between two pods that are within 2 hops. That is, in addition to a direct hop between the source and destination pods, traffic is allowed to transit at another intermediate pod before reaching its destination. For load balancing, we compute the optimal routing weights that minimizes MLU using an multi-commodity flow (MCF) formulation, as done in [27, 30].

Versus fat tree: We first compare against fat trees, which represent the industry standard in DCN topologies. Due to cost reasons, most network operators tend to oversubscribe to the aggregation or core layers [4, 16, 24]. Our evaluation includes a 1:3 oversubscribed fat tree, which has comparable cost to METTEOR, and a non-oversubscribed 1:1 fat tree. All fat tree topologies use ECMP routing.

Fat tree networks perform poorly in terms of bandwidth tax when compared against METTEOR’s topologies. A fat tree network has an additional spine layer of packet switches. Therefore, inter-pod traffic always consumes bandwidth of 2 hops (one between the source pod and the spine, and the other between the spine and the destination pod). Under a METTEOR topology, on average over 80% of traffic traverses single-hop paths (OCSs are transparent to DCN traffic in between adjacent OCS reconfigurations). In terms of MLU, a 1:1 fat tree, with its full bisection bandwidth, outperforms all other topologies, barring ideal ToE. When compared against a 1:3 fat tree of comparable cost, METTEOR reduces tail MLU by about 3 \times .

Versus uniform mesh: We also compare with a uniform mesh that directly connects pods without an OCS/spine layer. Uniform meshes are considered a class of expander networks, offering large bisection bandwidth at a lower cost than fat trees. Since METTEOR’s logical topology is also mesh-like with non-uniform interpod connectivity, a uniform mesh represents a natural baseline for comparison.

METTEOR consistently lowers bandwidth tax by ≈ 0.35 on average over a uniform mesh, due to its strategic link placement between hotspots. The lower bandwidth tax translates into an average median-LU improvement of 50%, due to a reduction in overall traffic load. In terms of MLU, METTEOR performs comparably to a uniform mesh. This comparison

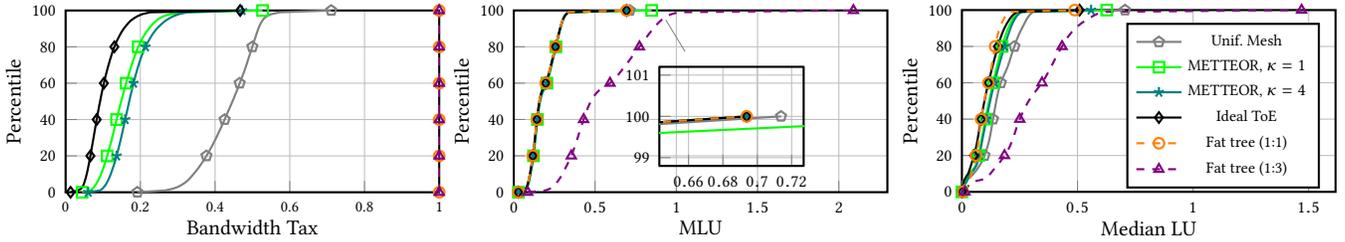


Figure 8: Percentile plot of network performance using 6-months’ worth of TM snapshots from a production DCN. METTEOR reconfigures topology every 2 weeks. METTEOR/Mesh/Ideal topologies use MCF routing, and fat tree uses ECMP.

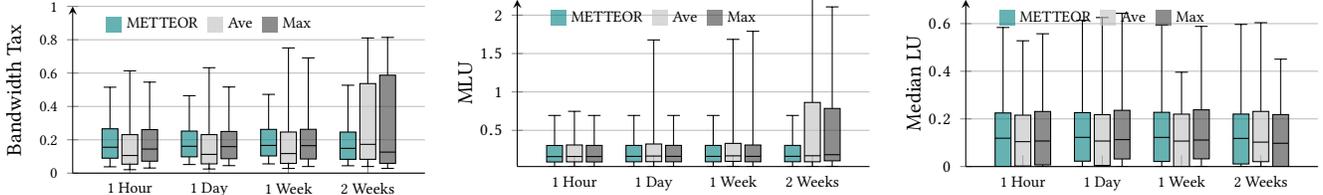


Figure 9: Performance sensitivity to reconfiguration frequency, comparing METTEOR to single-traffic approaches. Boxes represent the 95-th and 5-th percentile values; the whiskers represent the max and min values.

showcases the benefits of ToE, as we can reduce bandwidth tax without sacrificing MLU.

Versus ideal ToE: Ideal ToE computes an offline optimal topology that minimizes the MLU and bandwidth tax for each traffic matrix, and thus it represents the performance upper bound of topology engineering.

When paired with ideal load balancing, METTEOR’s MLU performance is very close to that of ideal ToE. As our evaluation traffic matrices are highly skewed, with a significant amount of traffic comes from a small subset of pods, MLU becomes limited by a pod’s total egress/ingress capacity. Therefore, even ideal ToE cannot do much to improve MLU. Still, ideal ToE lowers bandwidth tax over METTEOR by 0.08 on average, and 0.01 at the tail.

Versus single-traffic ToE: Finally, we compare METTEOR against single-traffic ToE to showcase the benefits of using multiple TMs. As an example of single-traffic ToE, METTEOR ($\kappa = 1$) has a longer tail than METTEOR ($\kappa = 4$) for all the metrics shown in Fig. 8. Indeed, it is generally very difficult to predict future TMs with a single TM due to traffic uncertainties. One may also propose using an element-wise average, or maximum traffic estimator in single-traffic ToE. We postpone the detailed comparison in §8.2.

8.2 Impact of Reconfiguration Frequency

Clearly, the frequency of topology-reconfiguration is a key factor in not just performance, but also the implementation and management complexity. The evaluations on METTEOR in §8.1 are based on biweekly reconfigurations. Here, we study the interplay between topology reconfiguration frequency and performance, by comparing METTEOR’s to other single-traffic based methods used in prior ToE

works [17, 20, 27, 55]. As in §7.3, we compare METTEOR against two other single-traffic ToE approaches: *Ave* and *Max*. *Ave* derives its traffic estimator by taking the average of its historical traffic snapshots, while *Max* derives its traffic estimator by taking the element-wise historical max.

Results in Fig. 9 show that in terms of tail MLU and bandwidth tax, METTEOR generally outperforms other single traffic-based ToE approaches given the same reconfiguration frequency. As METTEOR optimizes topology based on multiple estimated demands, it is more effective in covering future demands that resemble at least one of the predicted traffic used for topology-optimization. Furthermore, considering multiple traffic matrices when optimizing topology makes it less likely to overfit the logical topology to any single traffic demand, thereby reducing the performance penalty due to poor predictions. Note that METTEOR exhibits very little change in tail performance even at lower reconfiguration frequencies, further highlighting the topology’s robustness to traffic changes over time. This feature allows DCN operators gain much of the benefits of topology engineering even with sporadic reconfigurations.

8.3 A Discussion On Suboptimal Routing

Our evaluations so far have been based on ideal load balancing that can respond instantaneously to current traffic demands with a set of optimal routing weights that minimizes MLU. This allows us to analyze the merits of different topologies, irrespective of routing-induced suboptimality⁵. While close-to-optimal TEs have been demonstrated in the past (e.g. MicroTE [5]), they are all adaptive algorithms that operate

⁵Evaluations based on ideal MCF load balancing that minimizes MLU have been similarly done in [17, 27, 55].

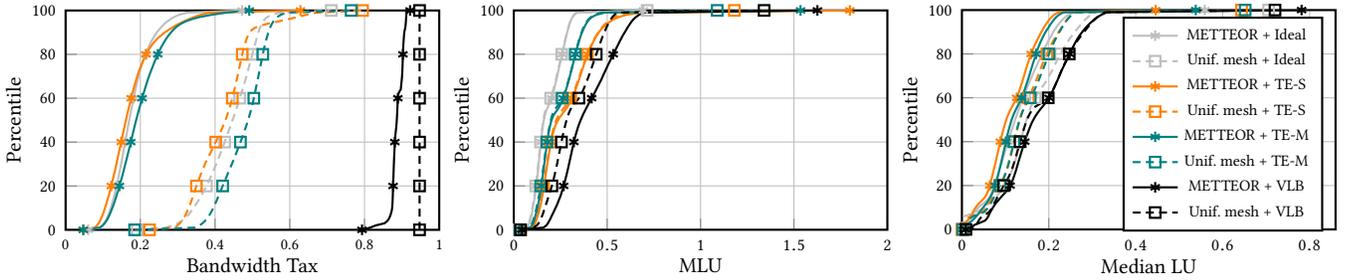


Figure 10: Percentile plot of network performance with different TE solutions. Solid lines denote METTEOR topology performance, while dashed lines represent uniform mesh.

at very fine timescales (e.g. sub-seconds), which may inflict huge management overheads to the SDN controller [33].

Since METTEOR operates on coarse timescales, this led us to question whether it can be paired with a coarse-grained TE to work well. As routing can no longer react to all TMs optimally, this brings routing-induced suboptimality into consideration. For this evaluation, we use 3 different load-balancing schemes that represent a large class of TE algorithms: 1) single-traffic TE, 2) Valiant load balancing (VLB), and 3) multi-traffic TE, and compare their performance when applied on uniform mesh and METTEOR ($\kappa = 4$) topologies. **Single-Traffic TE (TE-S):** TE-S computes routing weights using an MCF that minimizes MLU for a single predicted TM. The TE-S results in Fig. 10 updates routing weights every 5 minutes, based on the average traffic matrix over the past hour. We can see that TE-S performs well on average, but clearly suffers at the tail for both MLU and bandwidth tax.

Valiant load balancing (VLB): As a traffic-agnostic load-balancing algorithm, the canonical VLB derives its robustness by splitting traffic among many indirect paths at random. Our version of VLB splits traffic among direct and indirect paths, weighted by path capacity. However, because VLB sends a large portion of traffic via indirect paths, it exhibits very poor bandwidth tax performance in Fig. 10. Clearly, VLB’s indiscriminate traffic-splitting policy prevents it from favoring the tax-free direct paths, which makes VLB a poor choice of load-balancing for METTEOR.

Multi-Traffic TE (TE-M): TE-M is essentially a traffic engineering analog of METTEOR. On a high-level, TE-M picks multiple representative traffic matrices based on historical measurements, and computes a set of routing weights that minimizes MLU for all of the predicted traffic matrices. The full formulation is in E. Fig. 10 shows the performance of TE-M which updates routing weights every 5 minutes, based on 4 representative TMs chosen from the past hour. Clearly, TE-M retains an impressive bandwidth tax when used with METTEOR, while achieving better tail MLU than TE-S and VLB. This indicates that using multiple TMs can improve routing robustness under uncertainty.

Uniform Mesh vs. METTEOR: Recall from §8.1 that METTEOR improves bandwidth tax over static uniform mesh, without sacrificing tail MLU. Unfortunately, this is no longer true when TE is suboptimal. Based on Fig. 10, we can see that while METTEOR still outperforms a uniform mesh in bandwidth tax, it is more prone to a long MLU tail⁶. In fact, there is a tradeoff between average bandwidth tax and tail MLU. A “topology + routing” solution with better average bandwidth tax, tends to have a longer MLU tail.

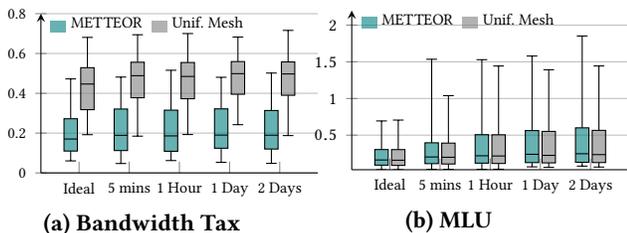
After analyzing the TM snapshots that caused long MLU tails, we found that the leading cause to be the sudden traffic bursts between pairs of pods thought to be “cold”, rather than an increase in traffic at the hotspots. To improve METTEOR’s tail MLU, we need to over-provision some capacity to the “cold” pod pairs. One interesting future work is to investigate the possibility of improving the above tradeoff with proper capacity-overprovisioning.

Impact of Routing Update Frequency: We found that, without ideal routing, METTEOR’s MLU exhibit long-tailed behavior, even if we use TE-M that updates routing weights every 5 minutes. Readers may wonder if the tail MLU can be improved with more frequent routing updates. However, we do not have data finer than 5 minutes. Instead, we evaluate TE-M under 4 different frequencies, ranging from once every 5 minutes to once every 2 days, and study the trend.

From Fig. 11, we can see that bandwidth tax is virtually unaffected by the routing frequency, with METTEOR still consistently outperforming a uniform mesh. Tail MLU does improve as routing-update frequency increases. We also plot Fig. 12 showing the percentage of TMs that can be supported by the underlying topology. Clearly, METTEOR works better with more frequent routing updates.

Summary: Under suboptimal TE, METTEOR still outperforms a uniform mesh in terms of bandwidth tax, though it is more susceptible to long-tailed MLUs. Updating routing weights more frequently helps the network respond better to traffic bursts, and improve tail MLU. Adaptive TE may be necessary to realize the full potential of METTEOR.

⁶Note that the MLUs up to 99.9 percentile values are roughly the same as those of the uniform mesh.



(a) Bandwidth Tax **(b) MLU**
Figure 11: TE-M performance on METTEOR ($\kappa = 4$) and uniform mesh. Whiskers denote 100-th and 0-th percentile; box represents 95-th and 5-th percentile.

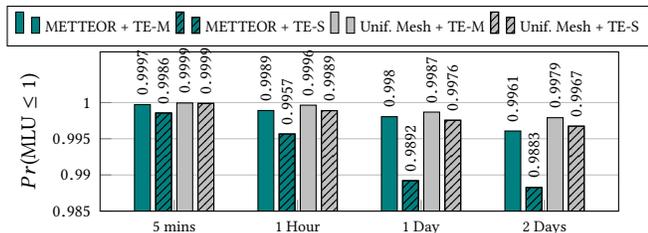


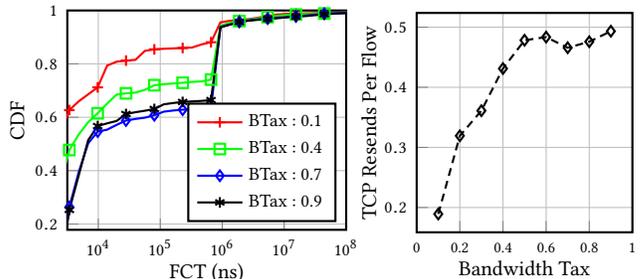
Figure 12: Probability of traffic demands met, as a function of TE update frequency.

Note: Due to restrictions on our access to the actual TM traces, the evaluations in §8.3 were done using approximate reconstructions of the TM traces used in §8.1 and §8.2. We reconstructed each TM from the first 5 principle-component projections from our PCA study, dropping higher-order terms. This by itself yields only a normalized approximation to the actual traffic matrices, so we inferred the correct scale factor based on MLU values from the ideal ToE results. This is a lossy reconstruction.

9 PACKET LEVEL SIMULATIONS

The evaluations thus far have focused on macroscopic metrics like link utilization and bandwidth tax. As important as these metrics are to DCN operators [14], their implications on application-level metrics such as flow completion time (FCT), are not immediately clear. To test how FCT at finer timescales relates to macroscopic metrics, we use the NetBench [44] packet-level simulator. The simulation emulates 2 seconds of real world time. The flow size distribution is based on a data mining workload from previous works [3]. Flows arrive following a Poisson process. We assume that the network links have capacity of 100Gbps, and the server-to-server latency is 600ns. We choose at random a TM to derive the communication probability between pods.

Our first set of simulations seeks to study the effects of different MLUs on FCT. First, a production TM snapshot is chosen at random. Next, 3 different logical topologies were generated via random sampling, such that routing the same traffic matrix over each logical topology with MCF results to 3 different MLUs. We enforce the routing weights obtained from MCF such that the bandwidth tax is 0.2 for all 3 logical



(a) Flow completion time (FCT) **(b) TCP resend per flow**
Figure 13: Flow-level performance for data-mining workload under different bandwidth taxes.

MLU	99th %tile FCT	99.9th % tile FCT	99.99-th % tile FCT
0.5	154ms	331ms	928ms
1.0	103ms	379ms	Incomplete
1.5	118ms	414ms	Incomplete

Table 2: Tail FCT performance of routing the same traffic matrix with different MLU, but fixed bandwidth tax.

topologies⁷ to remove confounding variables. Table 2 shows that larger MLU leads to longer-tailed FCT. Intuitively, a lower MLU means less link congestion, which ultimately helps more flows to complete as more traffic may traverse the network within a given amount of time.

We similarly study how differences in bandwidth tax may affect flow level performance, given the same MLU of 0.45. Fig. 13a shows the FCT as a function of bandwidth tax. Note that while there is little difference in FCT for larger flows, the small flows have shorter FCT when the bandwidth tax is low. Small flows are more latency-sensitive, and hence their FCTs are more likely to be affected by a high bandwidth tax. Fig. 13b shows that packet drop could happen in shallow-buffered data centers even before MLU reaches 1, and higher bandwidth tax leads to higher occurrences of TCP resends, which is detrimental to the throughput of small flows while waiting for packet timeout. Hence, bandwidth tax is equally important for DCN performance as MLU.

10 CONCLUSION

We present METTEOR, a robust topology engineering (ToE) approach that works for off-the-shelf OCSs. Unlike previous ToE solutions that react to every traffic change, METTEOR designs logical topologies based on multiple representative TMs extracted from the slow-varying traffic clusters. As a result, METTEOR can obtain most of the benefits of an ideal ToE, even with infrequent reconfiguration on the order of weeks. Reconfiguring topology at such low frequencies will lead to a lower technological barrier to ToE deployment, paving a path toward the incremental adoption of reconfigurable networks in commercial data centers.

⁷This can be easily done via MCF by constraining 20% of the total traffic to traverse indirect paths.

REFERENCES

- [1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *SIGCOMM*.
- [2] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, Amin Vahdat, et al. 2010. Hedera: dynamic flow scheduling for data center networks.. In *Nsdi*, Vol. 10.
- [3] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. 2013. pfabric: Minimal near-optimal datacenter transport. In *ACM SIGCOMM Computer Communication Review*, Vol. 43. ACM, 435–446.
- [4] Theophilus Benson, Aditya Akella, and David A Maltz. 2010. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 267–280.
- [5] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. 2011. MicroTE: Fine grained traffic engineering for data centers. In *Proceedings of the Seventh Conference on emerging Networking EXperiences and Technologies*.
- [6] Shaileshh Bojja Venkatakrisnan, Mohammad Alizadeh, and Pramod Viswanath. 2016. Costly circuits, submodular schedules and approximate carathéodory theorems. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*. 75–88.
- [7] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning* 3, 1 (2011), 1–122.
- [8] Stephen Boyd and Lieven Vandenbergh. 2004. *Convex Optimization*. Cambridge University Press.
- [9] CALIENT Technologies, Inc. [n. d.]. <https://www.calient.net/>.
- [10] Yiyang Chang, Sanjay Rao, and Mohit Tawarmalani. 2017. Robust validation of network designs under uncertain demands and failures. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*. 347–362.
- [11] Yanpei Chen, Rean Griffith, Junda Liu, Randy H Katz, and Anthony D Joseph. 2009. Understanding TCP incast throughput collapse in data-center networks. In *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*.
- [12] Cisco. 2016. Cisco Data Center Spine-and-Leaf Architecture: Design Overview. *Cisco White Paper* (2016).
- [13] Christina Delimitrou, Sriram Sankar, Aman Kansal, and Christos Kozyrakis. 2012. ECHO: Recreating network traffic maps for datacenters with tens of thousands of servers. In *Proc. 2012 IEEE International Symposium on Workload Characterization (IISWC)*.
- [14] Nandita Dukkkipati and Nick McKeown. 2006. Why Flow-completion Time is the Right Metric for Congestion Control. *SIGCOMM Comput. Commun. Rev.* 36, 1 (Jan. 2006), 59–62.
- [15] Jack Edmonds and Richard M. Karp. 1972. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *J. ACM* 19 (April 1972), 248–264.
- [16] Nathan Farrington and Alexey Andreyev. 2013. Facebook’s Data Center Network Architecture. *IEEE Optical Interconnects Conference* (2013).
- [17] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2011. Helios: a hybrid electrical/optical switch architecture for modular data centers. In *SIGCOMM*.
- [18] Mitchell H Fields, John Foley, Ron Kaneshiro, Larry McColloch, David Meadowcroft, Frederick W Miller, Sami Nassar, Michael Robinson, and Hui Xu. 2010. Transceivers and optical engines for computer and datacenter interconnects. In *Optical Fiber Communication Conference*.
- [19] Klaus-Tycho Foerster, Many Ghobadi, and Stefan Schmid. 2018. Characterizing the algorithmic complexity of reconfigurable data center architectures. In *Proc. ANCS*.
- [20] Monia Ghobadi, Ratul Mahajan, Amar Phanishayee, Nikhil Devanur, Janardhan Kulkarni, Gireeja Ranade, Pierre-Alexandre Blanche, Houman Rastegarfar, Madeleine Glick, and Daniel Kilper. 2016. Projector: Agile reconfigurable data center interconnect. In *SIGCOMM*.
- [21] Ali Ghodsi. 2006. Dimensionality reduction a short tutorial. *Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada* (2006).
- [22] Andrew V. Goldberg and Robert E. Tarjan. 1988. A New Approach to the Maximum-Flow Problem. *J. ACM* 35 (October 1988), 921–940.
- [23] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. 2016. Evolve or die: High-availability design principles drawn from googles network infrastructure. In *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 58–72.
- [24] Albert Greenberg, James R Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: a scalable and flexible data center network. In *SIGCOMM*.
- [25] LLC Gurobi Optimization. 2019. Gurobi Optimizer Reference Manual. In "<http://www.gurobi.com>".
- [26] Daniel Halperin, Srikanth Kandula, Jitendra Padhye, Paramvir Bahl, and David Wetherall. 2011. Augmenting Data Center Networks with Multi-gigabit Wireless Links. *SIGCOMM Comput. Commun. Rev.* 41, 4 (Aug. 2011), 38–49.
- [27] Navid Hamedazimi, Zafar Qazi, Himanshu Gupta, Vyas Sekar, Samir R Das, Jon P Longtin, Himanshu Shah, and Ashish Tanwer. 2014. Firefly: A reconfigurable wireless data center fabric using free-space optics. In *SIGCOMM*. 319–330.
- [28] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, Kirill Mendelev, et al. 2018. B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google’s software-defined WAN. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 74–87.
- [29] Robert W Irving and Mark R Jerrum. 1994. Three-Dimensional Statistical Data Security Problems. *SIAM J. Comput.* 23, 1 (1994), 170–184.
- [30] Sangeetha Abdu Jyothi, Ankit Singla, P Godfrey, and Alexandra Kolla. 2016. Measuring and understanding throughput of network topologies. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 65.
- [31] Srikanth Kandula, Jitendra Padhye, and Paramvir Bahl. 2009. Flyways to de-congest data center networks. In *Proc. HotNets*.
- [32] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken. 2009. The nature of data center traffic: measurements & analysis. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*.
- [33] Praveen Kumar, Yang Yuan, Chris Yu, Nate Foster, Robert Kleinberg, Petr Lapukhov, Chiun Lin Lim, and Robert Soulé. 2018. Semi-oblivious traffic engineering: The road not taken. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*. 157–170.
- [34] Charles E Leiserson. 1985. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE transactions on Computers* 100, 10 (1985), 892–901.
- [35] Hong Liu, Cedric F. Lam, and Chris Johnson. 2010. Scaling optical interconnects in datacenter networks – opportunities and challenges for WDM. In *IEEE 18th Annual Symposium on High Performance Interconnects (HOTI)*.
- [36] He Liu, Feng Lu, Alex Forencich, Rishi Kapoor, Malveeka Tewari, Geoffrey M Voelker, George Papen, Alex C Snoeren, and George Porter.

2014. Circuit Switching Under the Radar with REACToR. In *NSDI*.
- [37] He Liu, Matthew K Mukerjee, Conglong Li, Nicolas Feltman, George Papen, Stefan Savage, Srinivasan Seshan, Geoffrey M Voelker, David G Andersen, Michael Kaminsky, et al. 2015. Scheduling techniques for hybrid circuit/packet networks. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. ACM, 41.
- [38] Vincent Liu, Daniel Halperin, Arvind Krishnamurthy, and Thomas E Anderson. 2013. F10: A Fault-Tolerant Engineered Network. In *NSDI*. 399–412.
- [39] Steven H Low and David E Lapsley. 1999. Optimization flow control I: basic algorithm and convergence. *IEEE/ACM Transactions on Networking (TON)* 7, 6 (1999), 861–874.
- [40] William M Mellette, Rajdeep Das, Yibo Guo, Rob McGuinness, Alex C Snoeren, and George Porter. 2020. Expanding across time to deliver bandwidth efficiency and low latency. In *NSDI*.
- [41] William M Mellette, Rob McGuinness, Arjun Roy, Alex Forenych, George Papen, Alex C Snoeren, and George Porter. 2017. RotorNet: A scalable, low-complexity, optical datacenter network. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*.
- [42] Microsoft Research. [n. d.]. Project Sirius. In <https://www.microsoft.com/en-us/research/project/sirius/>.
- [43] Jeffrey C Mogul, Rebecca Isaacs, and Brent Welch. 2017. Thinking about availability in large service infrastructures. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*. ACM, 12–17.
- [44] Netbench. [n. d.]. <https://github.com/ndal-eth/netbench>.
- [45] Peter Phaal, Sonia Panchen, and Neil McKee. 2001. InMon corporation's sFlow: A method for monitoring traffic in switched and routed networks. (2001).
- [46] George Porter, Richard Strong, Nathan Farrington, Alex Forenych, Pang Chen-Sun, Tajana Rosing, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2013. *Integrating microsecond circuit switching into the data center*.
- [47] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.
- [48] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren. 2015. Inside the social network's (datacenter) network. In *SIGCOMM*.
- [49] Naum Z. Shor. 1985. *Minimization Methods for Non-differentiable Functions*. Springer-Verlag, New York.
- [50] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannan, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, et al. 2015. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. In *SIGCOMM*.
- [51] Ankit Singla, Chi-Yao Hong, Lucian Popa, and Philip Brighten Godfrey. 2012. Jellyfish: Networking Data Centers, Randomly. In *NSDI*.
- [52] Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, and Yueping Zhang. 2010. Proteus: a topology malleable data center network. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*.
- [53] Amin Vahdat, Hong Liu, Xiaoxue Zhao, and Chris Johnson. 2011. The emerging optical data center. In *Optical Fiber Communication Conference*.
- [54] Asaf Valadarsky, Michael Dinitz, and Michael Schapira. 2015. Xpander: Unveiling the Secrets of High-Performance Datacenters. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*.
- [55] Guohui Wang, David G Andersen, Michael Kaminsky, Konstantina Papagiannaki, TS Ng, Michael Kozuch, and Michael Ryan. 2011. c-Through: Part-time optics in data centers. In *SIGCOMM*.
- [56] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. 2006. COPE: traffic engineering in dynamic networks. In *Sigcomm*, Vol. 6. 194.
- [57] Mowei Wang, Yong Cui, Shihan Xiao, Xin Wang, Dan Yang, Kai Chen, and Jun Zhu. 2018. Neural network meets DCN: Traffic-driven topology adaptation with deep learning. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2, 2 (2018), 1–25.
- [58] Wikipedia. [n. d.]. Cosine Similarity. In https://en.wikipedia.org/wiki/Cosine_similarity.
- [59] Wikipedia. [n. d.]. Subgradient Method. In https://en.wikipedia.org/wiki/Subgradient_method.
- [60] Jin Y Yen. 1971. Finding the k shortest loopless paths in a network. *Management Science* 17, 11 (1971), 712–716.
- [61] Ye Yu and Chen Qian. 2016. Space shuffle: A scalable, flexible, and high-performance data center network. *IEEE Transactions on Parallel and Distributed Systems* 27, 11 (2016), 3351–3365.
- [62] Chun Zhang, Zihui Ge, Jim Kurose, Yong Liu, and Don Towsley. 2005. Optimal routing with multiple traffic matrices tradeoff between average and worst case performance. In *International Conference on Network Protocols (ICNP)*.
- [63] Junjie Zhang, Kang Xi, Min Luo, and H Jonathan Chao. 2014. Load balancing for multiple traffic matrices using SDN hybrid routing. In *15th International Conference on High Performance Switching and Routing (HPSR)*.
- [64] Yin Zhang and Zihui Ge. 2005. Finding critical traffic matrices. In *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*.
- [65] Rui Zhang-Shen and Nick McKeown. 2008. Designing a fault-tolerant network using valiant load-balancing. In *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE, 2360–2368.
- [66] Shizhen Zhao, Rui Wang, Junlan Zhou, Joon Ong, Jeffrey. C Mogul, and Amin Vahdat. 2019. Minimal Rewiring: Efficient Live Expansion for Clos Data Center Networks. In *NSDI*.
- [67] Xiang Zhou, Hong Liu, and Ryohei Urata. 2017. Datacenter optics: requirements, technologies, and trends. *Chinese Optics Letters* 15, 5 (2017), 120008.
- [68] Xia Zhou, Zengbin Zhang, Yibo Zhu, Yubo Li, Saipriya Kumar, Amin Vahdat, Ben Y Zhao, and Haitao Zheng. 2012. Mirror mirror on the ceiling: Flexible wireless links for data centers. In *SIGCOMM*.

A ADDITIONAL SIMULATION RESULTS

Here, we show the complete set of simulation results in Fig. 14, comparing METTEOR against a uniform mesh expanders and ideal ToE. Similar to the settings in §8.1, METTEOR re-configures logical topology on a biweekly-basis, using 4 traffic clusters ($\kappa = 4$) computed from 2-weeks' worth historical traffic matrix snapshots.

B ALGORITHMS FOR BPM AND LDM

Here, we fully flesh out the Barrier Penalty Method (BPM) and Lagrangian Dual Method (LDM), and provide the numerical algorithms needed for each method to work. Both methods are iterative, and will save the solutions that yields the lowest ratio of soft constraint violations encountered up till the current iteration. First, we introduce a goodness function for a feasible OCS switch configuration state, \mathbf{x} used to keep track of the best solution thus far:

$$\Psi(\mathbf{x}) = \sum_{i,j \in \mathcal{S}} \psi_{ij} \quad (12)$$

Where ψ_{ij} is an indicator variable that equals 1 when the (i, j) pod pair's soft constraints is satisfied, and 0 otherwise.

B.1 Detailed Walkthrough for BPM

Even though (11) has relaxed the soft constraints, solving it to optimality directly is still challenging due to its quadratic objective function. We want a low-complexity algorithm with good objective value, rather than the optimal solution. To achieve this, we use first-order approximation on the objective function $U(\mathbf{x})$:

$$\begin{aligned} & \min_{\mathbf{x}} U(\mathbf{x}) \quad (13) \\ & \approx \min_{\mathbf{x}} \left\{ U(\hat{\mathbf{x}}) + \sum_{k=1}^K \sum_{i=1}^n \sum_{j=1}^n \left(\frac{\partial U}{\partial x_{ij}^k} \Big|_{\mathbf{x}=\hat{\mathbf{x}}} \right) \times (x_{ij}^k - \hat{x}_{ij}^k) \right\} \\ & = C + \sum_{k=1}^K \left\{ \min_{\mathbf{x}^k} \sum_{i=1}^n \sum_{j=1}^n \left[\sum_{k'=1}^K 2\hat{x}_{ij}^{k'} - (\lceil d_{ij}^* \rceil + \lfloor d_{ij}^* \rfloor) \right] x_{ij}^k \right\} \end{aligned}$$

where $\hat{\mathbf{x}}$ is an initial value of \mathbf{x} , and C is a constant.

As the constraints in (1), and the approximation form of $U(\mathbf{x})$ in (13) are separable in k , we can solve for \mathbf{x} iteratively, one OCS at a time, as follows:

$$\begin{aligned} & \min_{\mathbf{x}^k} \sum_{i=1}^n \sum_{j=1}^n \left[\sum_{k'=1}^K 2\hat{x}_{ij}^{k'} - (\lceil d_{ij}^* \rceil + \lfloor d_{ij}^* \rfloor) \right] x_{ij}^k \quad (14) \\ & \text{s.t:} \quad \sum_{i=1}^n x_{ij}^k \leq h_{ig}^k(j), \sum_{j=1}^n x_{ij}^k \leq h_{eg}^k(i), \\ & \quad \max\{\hat{x}_{ij}^k - 1, 0\} \leq x_{ij}^k \leq \hat{x}_{ij}^k + 1 \quad \forall i, j = 1, \dots, n \end{aligned}$$

We add a range for every x_{ij}^k because the approximation in (13) only works in the neighborhood of $\hat{\mathbf{x}}$. (14) is easily solvable using min-cost circulation algorithms (see Appendix C). Further, since all the bounds (i.e., $h_{ig}^k(i)$, $h_{eg}^k(i)$) are integers, an integer solution of x_{ij}^k is guaranteed. Due to space limits, Appendix B.1 provides the BPM pseudocode.

We have gone through the intuition of BPM in §6.2.2. Here, we provide the detailed pseudocode in Algorithm 1.

<p>Data:</p> <ul style="list-style-type: none"> • $D^* = [d_{ij}^*] \in \mathbb{R}^{n \times n}$ - fractional topology • τ_{max} - number of iterations <p>Result: $\mathbf{x}^* = [x_{ij}^{k*}] \in \mathbb{Z}^{n^2 K}$ - OCS switch states.</p> <pre style="margin: 0;"> 1 Initialize: $\hat{\mathbf{x}} := 0, \mathbf{x}^* := 0;$ 2 for $\tau \in \{1, 2, \dots, \tau_{max}\}$ do 3 for $k \in \{1, 2, \dots, K\}$ do 4 Solve (14) based on Appendix C and let x^k be the integer solution; 5 Update $\hat{\mathbf{x}}$ in the k-th OCS by setting $\hat{x}^k = x^k;$ 6 if $\Psi(\mathbf{x}^*) < \Psi(\hat{\mathbf{x}})$ then 7 $\mathbf{x}^* := \hat{\mathbf{x}};$ 8 end 9 end 10 end</pre>

Algorithm 1: Barrier penalty method

Algorithm 1 is an iterative algorithm. Although only one OCS gets updated in each step, we obtain a new solution after combining other OCSs' old states. The goodness function $\Psi(\mathbf{x})$ is used to track the best solution obtained so far. In our implementation, we use (12) as our goodness function. Many alternative goodness functions exists, though their relative merits are subject for future work.

B.2 Detailed Walkthrough for LDM

Lagrangian Dual method was motivated by the dual ascent method in [7]. By introducing dual variables for soft constraints, LDM not only achieves graceful relaxation of soft constraints, but also relaxes the original NP-hard problem to a polynomial-time solvable problem. Nevertheless, LDM differs from the dual ascent method due to integer requirement. In this section, we detail the steps required for LDM to work.

B.2.1 Primal Problem.

Our goal is to find an integer solution of $\mathbf{x} = [x_{ij}^k]$ satisfying the soft constraint (9) and the hard constraints in (1). In theory, there is no need for an objective function of \mathbf{x} in our problem, since the problem itself is more concerned with satisfiability of the soft-constraints. However, this will lead

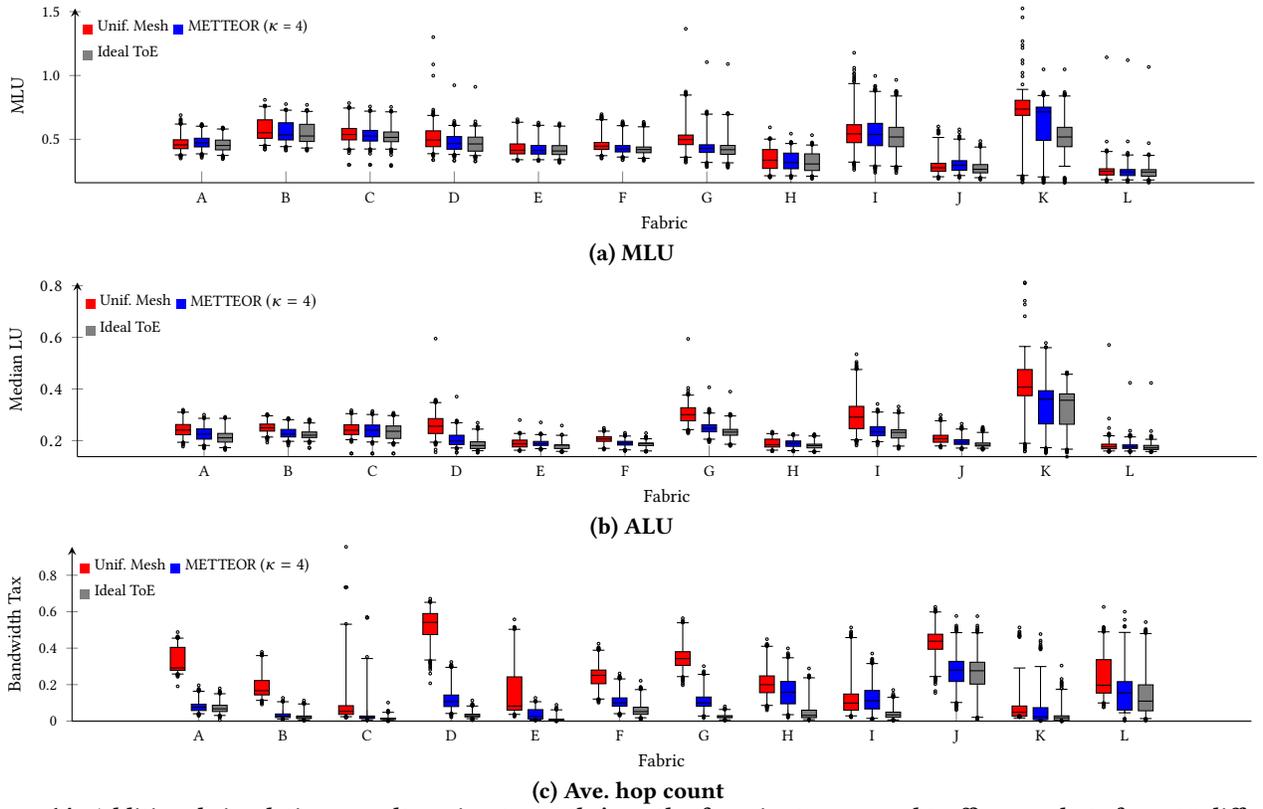


Figure 14: Additional simulations results, using 6 months' worth of 5-minute-averaged traffic snapshots from 12 different DCN fabrics.

to an algorithm with extremely poor convergence property. To speed up convergence, we introduce a strictly convex objective function for our primal problem, which is written as:

$$\begin{aligned} \mathbf{P} : \max_{\mathbf{x}} U(\mathbf{x}) &= \sum_{k=1}^K \sum_{i=1}^n \sum_{j=1}^n U_{ij}^k(x_{ij}^k) \\ \text{s.t.} : \quad &(1), (9), \text{ are satisfied} \end{aligned} \quad (15)$$

At first, we chose $U_{ij}^k(x_{ij}^k) = 0$, which is not strictly convex. As expected, the solution does not converge even after running large number of iterations. We then chose $U_{ij}^k(x_{ij}^k) = -(x_{ij}^k)^2$, which introduces a sharper objective function landscape that facilitated superior convergence. However, this objective function will result in a solution of \mathbf{x} that connects as fewer links as possible in each OCS, which not only wastes physical resources but also resulted in an overall decrease in network capacity. Finally, we went with:

$$U_{ij}^k(x_{ij}^k) = -\left(x_{ij}^k - h_{ij}^k\right)^2 \quad (16)$$

Where $h_{ij}^k = \min(h_{eg}^k(s_i), h_{in}^k(s_j))$, taking advantage of the fact that $h_{ij}^k \geq x_{ij}^k$ to ensure that the optimal solution maximizes the formation of logical links.

B.2.2 Dual Problem.

To relax the soft constraint (9), we introduce dual variables $\mathbf{p}^+ = [p_{ij}^+] \geq 0$, $\mathbf{p}^- = [p_{ij}^-] \geq 0$, and the following Lagrangian of the primal problem (15):

$$\begin{aligned} L(\mathbf{x}, \mathbf{p}^+, \mathbf{p}^-) &= \sum_{i=1}^n \sum_{j=1}^n \left[\sum_{k=1}^K U_{ij}^k(x_{ij}^k) - p_{ij}^+ \left(\sum_{k=1}^K x_{ij}^k - [d_{ij}^*] \right) \right. \\ &\quad \left. + p_{ij}^- \left(\sum_{k=1}^K x_{ij}^k - [d_{ij}^*] \right) \right]. \end{aligned}$$

Note that for every \mathbf{x} satisfying constraints (1), (9), and every $\mathbf{p}^+ \geq 0$ and $\mathbf{p}^- \geq 0$, the following inequality holds:

$$L(\mathbf{x}, \mathbf{p}^+, \mathbf{p}^-) \geq \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^K U_{ij}^k(x_{ij}^k).$$

Let

$$g(\mathbf{p}^+, \mathbf{p}^-) := \max_{\mathbf{x}} L(\mathbf{x}, \mathbf{p}^+, \mathbf{p}^-) \quad \text{s.t. (1) is satisfied}$$

We then have

$$\begin{aligned} g(\mathbf{p}^+, \mathbf{p}^-) &\geq \max_{\mathbf{x}} L(\mathbf{x}, \mathbf{p}^+, \mathbf{p}^-) \text{ satisfying (9), (1)} \\ &\geq \text{Optimal value of the primal problem (15)} \end{aligned} \quad (17)$$

Next, we introduce the dual problem:

$$\mathbf{D} : \min_{\mathbf{p}^+, \mathbf{p}^-} g(\mathbf{p}^+, \mathbf{p}^-) \quad \text{s.t. } \mathbf{p}^+ \geq 0, \mathbf{p}^- \geq 0. \quad (18)$$

Since the inequality (17) holds for all $\mathbf{p}^+ \geq 0$ and $\mathbf{p}^- \geq 0$, we must have

The minimum value of the dual problem (18)

≥ The maximum value of the primal problem (15).

Duality gap is then defined as the difference between the minimum value of the dual problem (18) and the maximum value of the primal problem (15).

If the primal decision variable \mathbf{x} were fractional numbers instead of integers, under mild constraints⁸, the duality gap would be 0. In that case, the optimal primal solution can be obtained by solving the dual problem instead. As we will see shortly, the dual problem (18) is much easier to solve. However, (15) is an integer problem with non-zero duality gap, hence solving the dual problem (18) cannot give us the optimal solution of the primal problem (15). Nevertheless, by optimizing the dual problem, we can still obtain a good sub-optimal solution to (15) that satisfies all the hard constraints and a vast majority of the soft constraints.

B.2.3 Subgradient Method.

The key aspect of LDM the optimization of the dual problem (18). Since the dual objective function is not differentiable, the typical gradient descent algorithm cannot be applied here. Hence, we use the subgradient method [49] instead, whose general form is given as follows:

Definition B.1. (Subgradient method [59]): Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a convex function with domain \mathbb{R}^n , a classical subgradient method iterates

$$\mathbf{y}^{(m+1)} = \mathbf{y}^{(m)} - \alpha_m \gamma^{(m)},$$

where $\gamma^{(m)}$ denotes a subgradient of f at $\mathbf{y}^{(m)}$, where $\mathbf{y}^{(m)}$ is the m -th iterate of \mathbf{y} . If f is differentiable, then the only subgradient is the gradient vector of f . It may happen that $\gamma^{(m)}$ is not a descent direction for f at $\mathbf{y}^{(m)}$. We therefore keep a list of f_{best} to keep track of the lowest objective function value found so far, i.e.,

$$f_{\text{best}} = \min\{f_{\text{best}}, f(\mathbf{y}^{(m)})\}.$$

Computing subgradient is the key step of the above subgradient method. The following lemma tells us how to compute a subgradient for the dual objective function $g(\mathbf{p}^+, \mathbf{p}^-)$.

LEMMA B.1. For a given $(\hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-)$, let $\hat{\mathbf{x}}$ be an integer solution that maximizes the Lagrangian $L(\mathbf{x}, \hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-)$, i.e.,

$$g(\hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-) = \max_{\mathbf{x}} L(\mathbf{x}, \hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-) = L(\hat{\mathbf{x}}, \hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-).$$

⁸For Slater's Condition: see §5.2.3 in [8].

Then, $\left[\lfloor d_{ij}^* \rfloor - \sum_{k=1}^K \hat{x}_{ij}^k, \sum_{k=1}^K \hat{x}_{ij}^k - \lfloor d_{ij}^* \rfloor \right], i, j = 1, \dots, n$ is a subgradient of $g(\mathbf{p}^+, \mathbf{p}^-)$ at $(\hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-)$, i.e.,

$$\begin{aligned} & g(\mathbf{p}^+, \mathbf{p}^-) - g(\hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-) \\ & \geq \sum_{i=1}^n \sum_{j=1}^n \left(\lfloor d_{ij}^* \rfloor - \sum_{k=1}^K \hat{x}_{ij}^k \right) (p_{ij}^+ - \hat{p}_{ij}^+) \\ & \quad + \sum_{i=1}^n \sum_{j=1}^n \left(\sum_{k=1}^K \hat{x}_{ij}^k - \lfloor d_{ij}^* \rfloor \right) (p_{ij}^- - \hat{p}_{ij}^-) \end{aligned}$$

for any $(\mathbf{p}^+, \mathbf{p}^-)$ in a neighbourhood of $(\hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-)$.

PROOF. Consider an arbitrary $(\mathbf{p}^+, \mathbf{p}^-)$. According to the definition of $g(\mathbf{p}^+, \mathbf{p}^-)$, we must have

$$g(\mathbf{p}^+, \mathbf{p}^-) = \max_{\mathbf{x}} L(\mathbf{x}, \mathbf{p}^+, \mathbf{p}^-) \geq L(\hat{\mathbf{x}}, \mathbf{p}^+, \mathbf{p}^-).$$

Then,

$$\begin{aligned} & g(\mathbf{p}^+, \mathbf{p}^-) - g(\hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-) \\ & \geq L(\hat{\mathbf{x}}, \mathbf{p}^+, \mathbf{p}^-) - L(\hat{\mathbf{x}}, \hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-) \\ & = \sum_{i=1}^n \sum_{j=1}^n \left(\lfloor d_{ij}^* \rfloor - \sum_{k=1}^K \hat{x}_{ij}^k \right) (p_{ij}^+ - \hat{p}_{ij}^+) \\ & \quad + \sum_{i=1}^n \sum_{j=1}^n \left(\sum_{k=1}^K \hat{x}_{ij}^k - \lfloor d_{ij}^* \rfloor \right) (p_{ij}^- - \hat{p}_{ij}^-), \end{aligned}$$

which completes the proof. \square

Remark 1. Note that for each $(\hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-)$, $\hat{\mathbf{x}}$ may not be the only solution that maximizes the Lagrangian $L(\mathbf{x}, \hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-)$, because $L(\mathbf{x}, \hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-)$ has integer variables \mathbf{x} . It is thus possible to have multiple subgradients for $g(\mathbf{p}^+, \mathbf{p}^-)$ at $(\hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-)$, in which case $g(\mathbf{p}^+, \mathbf{p}^-)$ is not differentiable at $(\hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-)$. If $g(\mathbf{p}^+, \mathbf{p}^-)$ were differentiable at $(\hat{\mathbf{p}}^+, \hat{\mathbf{p}}^-)$, there would be only one subgradient, which is the gradient of $g(\mathbf{p}^+, \mathbf{p}^-)$.

According to Lemma B.1, the most critical part of calculating subgradient is to find a maximizer for a given Lagrangian. By rearranging the dual objective function $g(\mathbf{p}^+, \mathbf{p}^-)$, we obtain the following:

$$\begin{aligned} & g(\mathbf{p}^+, \mathbf{p}^-) \\ & = \max_{\mathbf{x}} L(\mathbf{x}, \mathbf{p}^+, \mathbf{p}^-) \quad \text{s.t. (1) is satisfied} \\ & = \sum_{k=1}^K \max_{\mathbf{x}^k} \left[\sum_{i=1}^n \sum_{j=1}^n \left(U_{ij}^k(x_{ij}^k) + (p_{ij}^- - p_{ij}^+) x_{ij}^k \right) \right] \\ & \quad + \sum_{i=1}^n \sum_{j=1}^n (p_{ij}^+ \lfloor d_{ij}^* \rfloor - p_{ij}^- \lfloor d_{ij}^* \rfloor) \quad \text{s.t. (1) is satisfied} \end{aligned}$$

From the above equation, we can see that optimizing the Lagrangian can be decomposed into K subproblems:

$$\begin{aligned} \max_{\mathbf{x}^k} \quad & \sum_{i=1}^n \sum_{j=1}^n \left(U_{ij}^k(x_{ij}^k) + (p_{ij}^- - p_{ij}^+)x_{ij}^k \right) \quad (19) \\ \text{s.t:} \quad & \sum_{i=1}^n x_{ij}^k \leq h_{ig}^k(j), \sum_{j=1}^n x_{ij}^k \leq h_{eg}^k(i) \quad \forall i, j \end{aligned}$$

Although these subproblems have significantly fewer decision variables, they are still integer programming problem with quadratic objective function, which can be hard to solve. To further reduce complexity, we apply the same first-order approximation (see Eqn. (14)) again to the nonlinear terms in (19), and obtain

$$\begin{aligned} \max_{\mathbf{x}^k} \quad & \sum_{i=1}^n \sum_{j=1}^n \left(\frac{dU_{ij}^k}{dx_{ij}^k}(\hat{x}_{ij}^k) + (p_{ij}^- - p_{ij}^+)x_{ij}^k \right) \quad (20) \\ \text{s.t:} \quad & \sum_{i=1}^n x_{ij}^k \leq h_{ig}^k(j), \sum_{j=1}^n x_{ij}^k \leq h_{eg}^k(i), \\ & \max\{\hat{x}_{ij}^k - 1, 0\} \leq x_{ij}^k \leq \hat{x}_{ij}^k + 1 \quad \forall i, j \end{aligned}$$

where $\hat{\mathbf{x}}$ is the previous estimate of \mathbf{x} . The approximated problem (20) can be solved in polynomial time using the method in Appendix C.

B.2.4 Detailed Algorithm.

The detailed algorithm is shown in Algorithm 2. Note that we update dual variables right after computing a configuration for each OCS to hasten solution convergence. Another option is to update dual variables after iterating through all the OCSs for one round. The problem with this option is that OCSs with the same physical striping will be configured exactly the same way in the same iteration, causing the solution to oscillate and slows down convergence.

Notice that the harmonic step size function $\delta(\tau)$ is chosen because its sum approaches infinity as we take infinitely many step sizes. This way, we ensure that \mathbf{p}^+ , \mathbf{p}^- 's growth is not handicapped by the step size if their optimal values are large.

C MAPPING (14) TO A MIN-COST CIRCULATION PROBLEM

In this section, we study a general form of (14) as follows:

$$\begin{aligned} \min_{\mathbf{a}=[a_{ij}]} \quad & \sum_{i=1}^I \sum_{j=1}^J C_{ij} a_{ij} \quad (21) \\ \text{s.t:} \quad & \sum_{i=1}^I a_{ij} \leq P_j, \sum_{j=1}^J a_{ij} \leq Q_i, \\ & L_{ij} \leq a_{ij} \leq U_{ij} \quad \forall i = 1, \dots, I, j = 1, \dots, J \end{aligned}$$

Data:

- $D^* = [d_{ij}^*] \in \mathbb{R}^{n \times n}$ - fractional topology
- τ_{max} - number of iterations

Result: $\mathbf{x}^* = [x_{ij}^{k*}] \in \mathbb{Z}^{n^2 K}$ - OCS switch states

```

1 Initialize:  $\hat{\mathbf{x}} := 0, \mathbf{x}^* := 0, \mathbf{p}^+ := 0, \mathbf{p}^- := 0$ ;
2 Build network flow graphs  $G_1, \dots, G_k$  based on each
   OCS in  $\mathcal{O} = \{o_1, \dots, o_k\}$ ;
3 for  $\tau \in \{1, 2, \dots, \tau_{max}\}$  do
4   Set step size  $\delta := \frac{1}{\tau}$ ;
5   for  $k \in \{1, 2, \dots, K\}$  do
6     Solve (20) based on Appendix C, and let  $\mathbf{x}^k$  be
       the integer solution;
7     Update  $\hat{\mathbf{x}}$  in the  $k$ -th OCS by setting  $\hat{x}^k = x^k$ ;
8     if  $\Psi(\mathbf{x}^*) < \Psi(\hat{\mathbf{x}})$  then
9       |  $\mathbf{x}^* := \hat{\mathbf{x}}$ ;
10    end
11    Update dual variables using
        $p_{ij}^+ := \max\{p_{ij}^+ - \delta([d_{ij}^*] - \sum_{k'=1}^K \hat{x}_{ij}^{k'}), 0\}$  and
        $p_{ij}^- := \max\{p_{ij}^- - \delta(\sum_{k'=1}^K \hat{x}_{ij}^{k'} - [d_{ij}^*]), 0\}$ .
12  end
13 end

```

Algorithm 2: Lagrangian duality method

where $\mathbf{a} = [a_{ij}]$ is an $I \times J$ integer matrix to be solved, and $\mathbf{C} = [C_{ij}], \mathbf{P} = [P_j], \mathbf{Q} = [Q_i], \mathbf{L} = [L_{ij}], \mathbf{U} = [U_{ij}]$ are pre-defined constants. We would like to show that (21) can be easily mapped to a min-cost circulation problem, which is polynomial time-solvable with integer solution guarantees as long as $\mathbf{P} = [P_j], \mathbf{Q} = [Q_i], \mathbf{L} = [L_{ij}], \mathbf{U} = [U_{ij}]$ are all integers.

C.1 Min-Cost Circulation Problem

Definition C.1. (Min-Cost Circulation Problem) Given a flow network with

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

the goal of the min-cost circulation problem is to find a flow assignment $f(v, w)$ that minimizes

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

while satisfying the following two constraints:

- (1) Throughput constraints: $l(v, w) \leq f(v, w) \leq u(v, w)$;
- (2) Flow conservation constraints: $\sum_u f(u, v) = \sum_w f(v, w)$ for any node v .

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

LEMMA C.1. (*Integral Flow Theorem*) *Given a feasible circulation problem, if $l(v, w)$'s and $u(v, w)$'s are all integers, then there exists a feasible flow assignment such that all flows are integers.*

In fact, for feasible circulation problems with integer bounds, most max-flow algorithms, e.g., Edmonds-Karp algorithm [15] and Goldberg-Tarjan algorithm [22], are guaranteed to generate integer solutions.

C.2 Detailed Transformation Steps

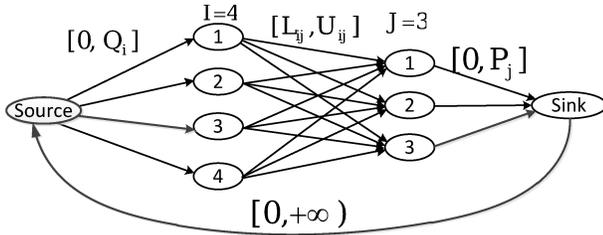


Figure 15: A flow graph example corresponding to equation (21).

We first construct a flow network based on equation (21) as follows (see Fig. 15 for graphical illustration):

- (1) Create a directed bipartite graph. Note that a is an $I \times J$ matrix. We create I nodes on the left hand side of the bipartite graph, and create J nodes on the right hand side of the bipartite graph. We add a directed link from i to j , and set the bounds of this link as $[L_{ij}, U_{ij}]$ and the cost of this link as C_{ij} .
- (2) Add a source node, and for each of the I left nodes, add a link that connects to this source node. The bounds of the i -th link is set as $[0, Q_i]$, and the cost is set to 0.
- (3) Add a sink node and J links from the J right nodes to this sink node. The bounds of the j -th link is set as $[0, P_j]$, and the cost is set to 0.
- (4) Add a feedback link from the sink node to the source node. The bounds of this feedback link is set as $[0, \infty)$, and the cost is set as a very small negative value $-\epsilon$, e.g., -10^{-6} .

We then assign flows to this flow network.

- (1) For the link from the i -th left node to the j -th right node, assign a_{ij} amount of flow.
- (2) For the link from the source node to the i -th left node, assign $\sum_{j=1}^J a_{ij}$ amount of flow.

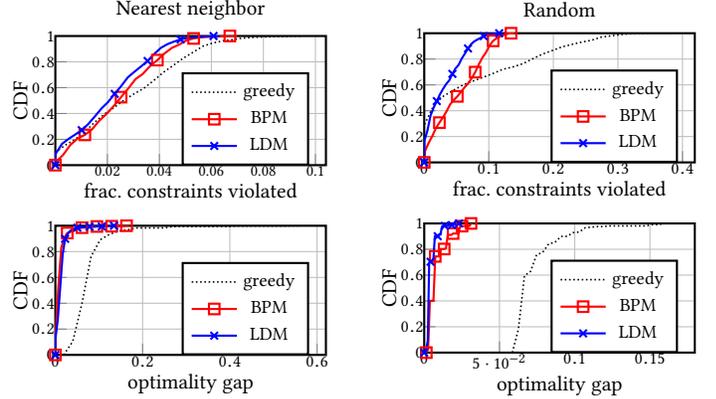


Figure 16: Optimality of OCS-mapping algorithms, using nearest neighbor (left col) and random (right col) traffic.

- (3) For the link from the j -th right node to the sink node, assign $\sum_{i=1}^I a_{ij}$ amount of flow.
- (4) For the feedback link from the sink node to the source node, assign $\sum_{i=1}^I \sum_{j=1}^J a_{ij}$ amount of flow.

It is easy to verify that the above flow assignment satisfies the flow conservation constraints in Definition C.1. Further, by enforcing the throughput constraints in Definition C.1, all the constraints in (21) are also satisfied. Further, the objective function of this min-cost flow problem is

$$\sum_{i=1}^I \sum_{j=1}^J C_{ij} a_{ij} + \epsilon \left(\sum_{i=1}^I \sum_{j=1}^J a_{ij} \right). \quad (22)$$

Since a_{ij} 's are all integers, $\sum_{i=1}^I \sum_{j=1}^J C_{ij} a_{ij}$ cannot be taken on a continuum of values. Then, as long as ϵ is small enough, minimizing (22) will also minimize the objective function in (21). The benefit of having a small negative cost ϵ is that more flows can be assigned if possible.

D OCS MAPPING - OPTIMALITY ANALYSIS

Although LDM and BPM are motivated by convex optimization theories, our problem requires integer solutions and is thus not convex. Therefore, neither LDM nor BPM can guarantee optimality. Nevertheless, we found via simulation that LDM and BPM show superior performance.

We generated 900 DCN instances with pod-counts between 12 and 66. Each DCN instance is heterogeneous, containing pods with a mixture of 256, 512, and 1024 ports, interconnected via 128-port OCSs. The greedy algorithm described in Helios [17] acts as a baseline. All 900 instances are tested using: 1) nearest-neighbor, and 2) random permutation TMs. For nearest-neighbor TM, each pod sends traffic only to pods within ρ -units of circular index distance, where

ρ is $\sim \frac{1}{8}th$ the fabric size; this imitates skewed, neighbor-intensive traffic. Random TM is generated by treating each off-diagonal entry as a uniform random variable.

Next, we compute a logical topology *w.r.t.* to its TM. We use two solution-optimality metrics: 1) soft-constraint violation ratio, and 2) optimality loss. Soft-constraint violations counts the number of (i, j) pairs where (9) is violated. Optimality loss measures the throughput loss/gap as we approximate the fractional topology with an integer one. This is measured as $1 - \frac{\mu_{int}^*}{\mu_{frac}^*}$; μ_{int}^* and μ_{frac}^* denote the throughputs under the integer and fractional logical topologies.

Fig. 16 shows LDM slightly outperforming BPM, due to its adaptability afforded by its dual variables, which help “coerce” the solution towards optimality. Both LDM and BPM clearly outperform the greedy method in the optimality gap and matching soft constraints.

E MULTI-TRAFFIC TRAFFIC ENGINEERING (TE-M) FORMULATION

Given m representative traffic matrices, $\{T_1, \dots, T_m\}$, and an integer logical topology, X , TE-M computes the optimal

routing weights, $\omega_p \forall p \in \mathcal{P}$, that minimizes MLU for all m input demands. Here, ω_p denotes the fraction of traffic sent via path p , such that $\sum_{p \in \mathcal{P}_{ij}} \omega_p = 1$. Rather than solving this

MLU directly, however, we scale up each input traffic matrix, T_τ , using μ_τ^* until MLU reaches 1. This additional step ensures that the computed routing weights will account for all traffic matrices. Once the scaling factor has been computed for each input traffic matrix, we then solve for the optimal routing weights that minimizes MLU, η for all the scaled traffic matrices with the following:

$$\begin{aligned}
 & \min_{\Omega} \eta \\
 \text{s.t. } & 1) \sum_{p \in \mathcal{P}_{ij}} \omega_p = 1 \quad \forall i, j = 1, \dots, n \\
 & 2) \sum_{p \in \mathcal{P}, (s_i, s_j) \text{ is a link in } p} \omega_p \mu_\tau^* t_{src_p, dst_p}^\tau \leq \eta x_{ij} b_{ij} \\
 & \quad \forall i, j = 1, \dots, n, \tau = 1, \dots, m,
 \end{aligned} \tag{23}$$

where t_{src_p, dst_p}^τ denotes the traffic demand (in Gbps) between the source and destination pods of path p .