To catch a moving ball by standing still : Designing static DCN topology for dynamic traffic

Min Yee Teh, Shizhen Zhao

ABSTRACT

Data center traffic patterns are highly skewed and dynamic, making them very difficult to predict with any high degree of confidence. Much of the established literature in data center network (DCN) topology are dedicated to traffic-agnostic topologies, rendering them suboptimal for highly-skewed traffic patterns. In order to deal with highly-skewed and changing traffic, many reconfigurable topologies have been proposed based on the idea of matching the topology to the traffic patterns. However, these implementations rely on sophisticated and immature switching hardware, not to mention introducing significant software challenges within the network control plane. In this paper, we offer a new perspective to designing traffic-aware topology robust to traffic dynamism using static topology. More specifically, we propose a traffic-aware topology design methodology with a static network that is optimized for highly-skewed traffic and also robust to dynamic traffic change. We simulate the network performance using real production traffic. Results show that our proposed methodology is robust under traffic changes, and can significantly decrease both network link congestion and average packet hop-count, thereby reducing overall network congestion while also decreasing the average packet latency.

1 INTRODUCTION

Based on established literature [3, 4], we know that inter-rack traffic patterns tend to be highly skewed. This implies a high degree of traffic locality, and that all-to-all uniform traffic rarely occurs. In addition to traffic skewness, studies [7] show that most data center fabrics demonstrate transient, bursty traffic behavior. These highly-skewed traffic naturally create network hotspots, and severely congesting network links while leaving other network links relatively underutilized. Furthermore, these transient bursts further complicates DCN design since predicting traffic patterns accurately is very difficult, if not close to impossible.

One way to "even" out highly-skewed traffic patterns is to load-balance using workload migration and clever job-placement [17]. However, as data center tends towards disaggregated hardware resources [25], relying solely on intelligent workload placement will for link load-balancing will become increasingly difficult. This is because disaggregating compute resources from storage resources will force same workloads to be mapped onto dedicated clusters that are physically-separated. Even with perfect knowledge on job arrival and generated network traffic behavior, workload-placement alone cannot fundamentally even out the strong pairwise traffic pattern inherent to DC resource desegregation. [20]

To account for traffic skewness and variability, many reconfigurable data center topologies [8, 11, 15, 20, 35] have been proposed using different optical circuit switching technologies. Optical circuit switching incurs non-negligible reconfig time, which could significantly increase data center's control plan complexity. As a consequence, a lot of effort has been put into driving down the switching time of these switching technologies. An implicit assumption here is that fast switching is a "must have" in order for a control plane to react to rapid and bursty traffic changes. In this work, we would like to challenge this assumption, and answer the question, "can slowly reconfigurable topology capture fast traffic dynamics?"

Another approach to even out out non-uniform traffic is through the use of congestion-aware routing algorithms. Most congestion-aware routing algorithms (i.e traffic engineering) balances link utilization across the network by deflecting traffic flow non-minimally to intermediate blocks first along less-congested paths. To first order, these congestion-aware routing schemes can introduce significant network delay to packets. More importantly, congestion-aware routing has the effect of creating network hotspots in other network areas, increasing overall traffic load within the network, and not to mention taking up precious bandwidth resources from other network resources. Therefore, it is clear that traffic engineering alone cannot sufficiently decongest and improve network throughput under many traffic permutations.

In order to handle highly-skewed and vastly dynamic traffic behavior within DCNs, there has been much work dedicated to designing reconfigurable network topologies using dedicated switching hardware with very fast switching speeds. The key idea is to allow the topology to be reconfigured dynamically to the traffic changes. Some prominent examples that come to mind are the Firefly [15] and ProjectTor [11], although many different topologies based on different hardware technologies have

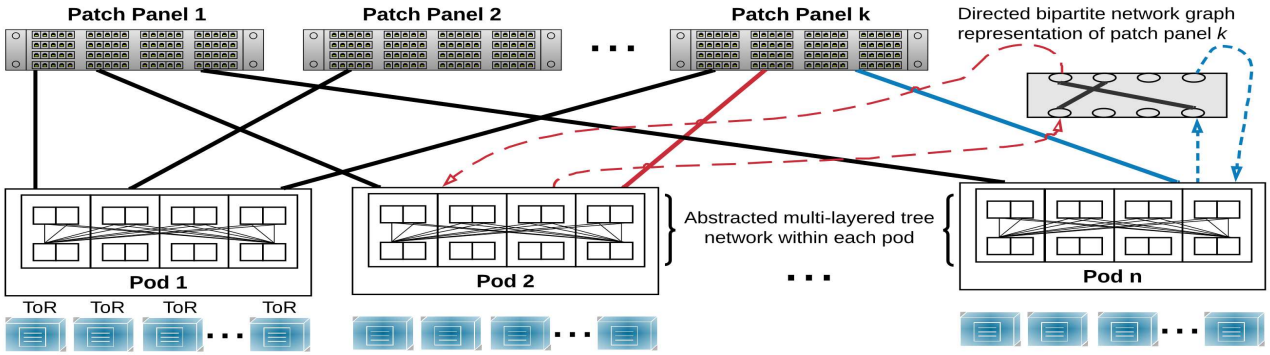

Figure 1: Helios topology fully interconnected via a patch-panel layer, alongside an illustration on the bipartite network graph representation of patch-panel k. All links in solid lines are bidirectional (undirected), and dotted links of the same color are the unidirectional (directed) representation of the same solid link.

been proposed in established literature. The key concept that motivates these reconfigurable DCN topologies are more or less identical: to match the topology with the traffic matrix in either a predictive or reactive manner. A lot of effort has been put into driving down the switching time of these switching technologies in hopes of driving down the cost of topology reconfiguration. Unfortunately, to fully interconnect even a moderately-sized DCN network to full bandwidth capacity using these specialized switching hardware can be overly cost-prohibitive, therefore making them impractical for deployment until the underlying technology has matured.

That being said, we set out by asking the question: is it possible to design a static topology that is optimized not only for highly-skewed traffic, but also one that is robust under dynamic traffic changes? Answering this question requires us to first understand DCN traffic temporal behavior. Thus, we've conducted some preliminary studies using historical traffic data collected from our production data centers. Based on the findings, we observed that all DCN fabrics exhibit remarkable multimodal traffic behavior in the time-domain, despite the fact traffic matrix show little to no temporal patterns. What this means is that there exists a small set of recurring traffic matrices that can be used to accurately represent all other traffic matrices, with only very few outliers.

These findings lead us to believe that, perhaps counterintuitively, by designing a static topology optimized for all of the traffic modes using only commercial patch panels can enhance network performance while still being sufficiently robust to dynamic traffic change. In other words, we believe that a sufficiently well-designed **static** topology can enhance network throughput under highly-skewed traffic patterns, while still being able to perform well under highly **dynamic traffic changes**.

Therefore, our contributions in this work is fourfold:

- We observed and surmise that DCN traffic can be represented using a small set of recurring traffic modes.
- We offer a traffic-aware topology design methodology on a DCN interconnected using off-the-shelve commodity patch panels to design a **static** topology that is highly robust to long term traffic change.
- We present a routing-topology co-optimization technique that considers multiple traffic modes (i.e traffic matrices) which also accounts for DCN heterogeneity.
- We propose 2 highly scalable, polynomial-time, and mathematically-guided algorithms for mapping fractional target topology onto the physical patch-panel with close-to-optimal performance.

Based on our proposed methodology, we present simulation results using production traffic data on 4 DCN fabrics, with the topology computed at the start of the month and remains static throughout the entirety of the month. For baseline comparison, we use a uniform topology load-balance using just congestion-aware traffic engineering with global traffic knowledge. Our month-long timeseries evaluation shows that using our proposed methodology consistently outperforms a uniform topology in maximum link utilization by 8% to 10%. Similarly, our results show that the average packet hop count can be significantly improved, with more than 90% of the traffic taking a direct-path towards its destination. These results suggest that subjected to the same traffic load, our proposed methodology can both decongest network links (by lowering max link utilization) while ensuring most traffics are able to stay within direct paths, thereby lowering average link congestion levels and average network-traversal latency.

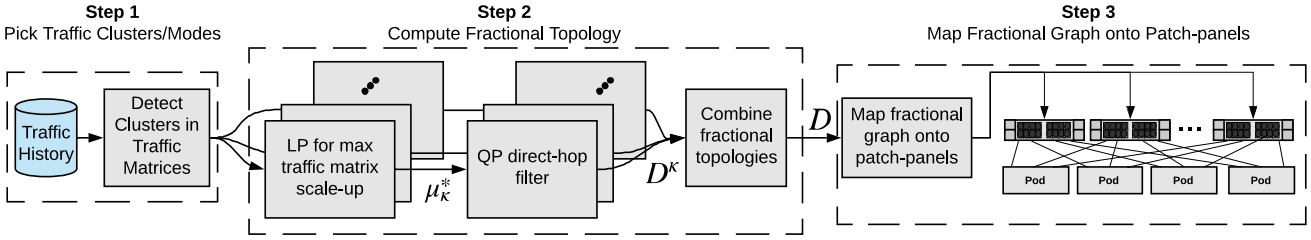

Figure 2: Overall traffic-aware topology engineering work flow

2 BACKGROUND & MOTIVATION

Over the course of the past decade, the increasing role of cloud computing has led to more emphasis on intelligent DCN design. As a result, the research community to study DCN topologies that are scalable, cost-effective and failure-tolerant using commodity static switches with much more rigor. Within the realm of established literature, the proposed topologies, and some defining examples, can be broadly listed as follows:

- **Clos topologies**: Fat-tree [2], Google's Jupiter [28], Microsoft's tapered fat-tree [12, 22]
- **High-radix, low-diameter topologies**: Dragonfly [18, 32], Slimfly [5], HyperX [1]
- **Direct connect topologies**: BCube [13], DCell [14], Dahu [27]
- **Random graph-based topologies**: Xpander [34], S2 [36], Jellyfish [29]

One major disadvantage these static topologies suffer from is that they are not designed with traffic patterns in mind, thus making them suboptimal to highly-skewed traffic patterns and dynamic traffic changes. In order to adapt to traffic dynamism, many reconfigurable topologies using sophisticated circuit-switches [9, 19, 33, 40] have been proposed. The central idea to these topologies is to utilize sophisticated switching hardware to dynamically reconfigure the topology based on either predicted or measured traffic changes. Prominent examples include free-space optics(FSO)-based technology like ProjectToR [11] and Firefly [15], and wireless beamforming based in [41]. Several hybrid topologies [8, 20, 35] have also been proposed by combining electrical packet switches and optical circuit switches within the DCN, and rely on complex scheduling techniques [21, 26] to optimize for network throughput. Meanwhile, increasing maturity in MEMS technology has also led to the debut of RotorNet [24] and Proteus [30]. Unfortunately, frequent topology reconfiguration topology introduces significant complexities within network control plane, including, but not limited to:

(1) raising software complexity since the routing tables will be forced to re-computed upon topology change.
(2) incurring non-trivial connection down time during circuit reconfiguration, causing disruption to unbuffered traffic flow.
(3) rendering network performance highly susceptible to degradation due to innacurate traffic prediction.

The aforementioned challenges can be avoided using a static topology, yet the common consensus is that static topologies do not possess sufficient flexibility to adapt to dynamic traffic changes. Conventional wisdom suggests that robustness to traffic dynamism and topology staticity are two topological properties that are fundamentally at odds. Our preliminary study on temporal traffic behavior using historical production traffic data, however, shows findings that seek to challenge this conventional wisdom. These findings are presented in Figure 3 and elaborated in Section 5. There, we will find that all DCN fabrics demonstrate κ-traffic modal behavior, even though predicting a traffic matrix at a given time accurately with high degrees of confidence is exceedingly difficult. In other words, all traffic matrices aside from a select few outliers can be represented by a small set of recurring traffic "modes". Based on these findings, we posit that a static topology optimized for all of the traffic modes can be robust under dynamic traffic change.

Therefore, we depart significantly from established approaches to traffic-aware DCN topologies by proposing a (perhaps counter-intuitively) **static** topology engineering methodology which is capable of handling dynamic traffic changes, and fully implementable on commodity patch panel that are both cheap and easy to operate. By designing a topology that is static for prolonged periods of time (on the timescale of DC-expansions), we can also nullify the need for a complex control plane or building dedicated hardware that is otherwise required in other reconfigurable topologies, thereby reducing potential control-plane failures, operational and construction cost.

The topology we are considering in this paper is heavily-inspired by Helios [8], with the core switches replaced entirely with patch panels. The merits of utilizing a patch

panel reconfigurable layer is discussed in detail in Zhao et al. [38]. Figure 1 provides an illustration of the topology. We treat the network as a directed network, hence each link can only carry traffic in one direction. Ingress and egress ports are dedicated to receiving and sending traffics, respectively. Each bidirectional link is formed by combining two directed links in opposite directions. We also assume patch panel-asymmetry, meaning that A's egress port being connected to B's ingress port does not imply that A's ingress port is connected to B's egress port.

3 CHALLENGES & OVERALL APPROACH

The first of the biggest challenges we face in the context of this work is to figure out how to deal with transient traffic changes using a static topology. This restriction is imposed due to the fact that building a data center of our scale that is fully-interconnected at full-capacity using dedicated switching hardware is cost-prohibitive. Having established that DCN fabrics exhibit multi-modal (henceforth denoted as κ-modal) traffic behavior in Section 2, we posit that designing a topology optimized for all traffic modes can render a static network robust to traffic changes. To identify these traffic modes, we treat all historical traffic matrices as points in high-dimensional space, as shown in Step 1 of Figure 2. This essentially reduces the problem problem of finding traffic modes to that of locating traffic clusters in high-dimensional space, a common machine learning problem which can be done efficiently using various available clustering algorithms. Although counter-intuitive, our simulation results done using production DCN traffic in Section 9 show that this idea works very well in-practise.

The next major challenge lies in computing an optimal logical topology for a given set of traffic demands. Unlike our previous challenge which stems from hardware restrictions and impracticalities of frequent reconfiguration using patch panels, this challenge is mathematical in nature and is a problem universal to all reconfigurable topologies. The main source of complexity is due to two reasons: 1) the logical topology needs to be in integer form, and 2) when the network is not fully-connected (i.e not all combinations of port connections are possible due to them being physically-wired to different patch panels). These reasons make the problem a strongly NP-hard combinatorial problem, as shown in Foerster et al.[10]. In this work, we take a routing-topology co-optimization approach. By considering both routing and topology within the same topology engineering problem, we can effectively optimize for a given network performance metric (such as link congestion, packet hop-count, etc) directly, ensuring that the computed logical topology is indeed optimized under a given traffic load when measured. However, computing the logical topology directly by formulating an integer linear program (ILP) is too computationally-expensive. So, we approach this problem is 2 steps, which form Step 2 and 3 of Figure 2.

In Step 2, we first co-optimize a topology alongside routing by assuming a fully-connected network where all port pairs can be interconnected with fractional link counts. This can allows to formulate step 2 as a simple linear program (LP) that can be solved quickly. Finally, we attempt to map the fractional optimal topology onto the patch panel layer with the objective of matching the logical topology to the fractional topology. Note this this portion is where the NP-hardness of the problem comes from, therefore we cannot guarantee optimality of solution.

4 DEFINITIONS & NOTATIONS

4.1 Definitions

We briefly define several recurring terminologies in the context of this work:

- **Physical Topology** Describes the physical connectivity between the pods and the patch panels, represented as a directed graph.
- **Logical Topology** Describes the connectivity (i.e number of integer links) between pods, abstracting away the layer of patch-panels.
- **Fractional Topology** Logical topology assuming that inter-pod link counts can occupy non-integer values.

4.2 Notations

Before giving a rigorous treatment to the topology engineering problem, we dedicate this section to introducing a series of recurring notations that will be used throughout this paper. These notations are tabulated in Table 1. For the sake of convenience, we shall abuse the notation $\mathbf{x} \in \mathbb{R}^{\|S\|\times\|S\|\times\|O\|}$ to denote the vector of $x_{ij}^k \ \forall \ i,j \in S, k \in O$, and $\mathbf{x}^k \in \|\mathbb{S}\|\times\|\mathbb{S}\|$ to denote the vector of $x_{ij}^k \ \forall \ i,j \in S$ associated with a particular k. Similarly, we shall abuse the vector notation $\mathbf{d} = \{d_{ij} \mid i,j \in S\}$, and $\boldsymbol{\omega_p} = \{w_p \mid p \in \underset{i,j\in S}{P_{ij}}\}$.

As shown in Figure 1, we assume each patch panel can be represented as a bipartite network flow graph. In

$S = \{s_1, ..., s_n\}$	Set of all n pods
$O = \{o_1, ..., o_k\}$	Set of all k patch-panels
x_{ij}^k	Number of pod i's egress links connected to pod j's ingress links through o_k
$\Psi\mathbf{x}$	Quality function of patch-panel configuration
$T = t_{ij} \in \mathbb{R}^{n \times n}$	Traffic matrix, where t_ij denotes the sent from s_i to s_j
$D = d_{ij} \in \mathbb{R}^{n \times n}$	Fractional topology, where d_ij denotes the number of s_i egress links connected to ingress links of s_j
$h_{eg}^k s_i, h_{ig}^k s_j$	Number of physical egress links connecting s_i to o_k, and number of physical ingress links of s_j to o_k, respectively
r_{eg}^i, r_{ig}^i	Number of egress and ingress links, respectively, owned by s_i
b_i	Link bandwidth of s_i
P_{ij}	Set of all viable routing paths from s_i to s_j within the topology
Pe_{ij}	Set of all paths that traverses logical links connecting s_i to s_j
l_p	Length/hop count of path p
ω_p	WCMP weight allocated along path p

Table 1: Commonly-used notations

our asymmetric topology model, $x_{ij}^k \neq x_{ji}^k$ for any $i \neq j$ in general.

5 STEP 1: DETECTING TRAFFIC MODES

Due to the switching speed of our direct-connect topology being limited by the patch-panel layer, choosing the right traffic matrix for a monthly redesign of the logical topology is extremely crucial. Based on our observations, the volume of traffic changes very frequently, but the traffic pattern doesn't change all that much for prolonged periods of time, with sporadic changes in pattern. Therefore, we do not need to reconfigure the topology so frequently since the traffic patterns do not change regularly. But over-fitting the topology to serve the long term traffic pattern might lead to significant performance degradation and congestion for the occasional drastic changes. A robust topology should be capable of serving multiple traffic patterns well. Therefore, the problem statement becomes: Given a set of traffic matrices that has been measured, we would like to pick

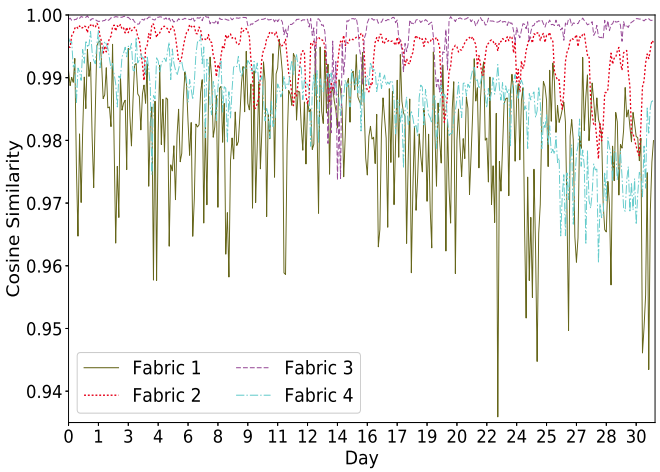

Figure 3: Cosine similarity between hourly traffic matrix using previous month's traffic modes

a set of representative traffic matrices. [37]. To reiterate, given a set of input traffic matrices, $T = \{T^1, T^2, ..., T^m\}$, identify a set of traffic matrices $Y = \{Y^1, Y^2, ..., Y^l\}$ that are "representative" of T such that $l \leq m$. Mathematically, we want to pick a set of κ matrices that is closest to all the points in T.

This problem is similar to that in [37]. Unlike the *CritMat*, however, we do not need to take the a set of traffic matrices that also dominates the others, since we only care about the pattern and not the sheer magnitude of the traffic matrices. In other words, we only care about normalized traffic matrices. To find the representative matrices, we employ κ-means-clustering technique to select κ representative traffic matrices. While a higher κ gives us more clusters, it becomes progressively harder to design a topology that can serve high κ values, since vastly different traffic modes may introduce many mutually-conflicting constraints. Also, as κ increases, there superposition of all modes may tend towards a uniform traffic matrix, hence rendering topology engineering completely unnecessary. It turns out that selecting $\kappa = 2, 3$ tends to be good enough choices.

6 STEP 2: COMPUTING FRACTIONAL TOPOLOGY

At this point, we've obtained κ representative traffic matrices, each representing a recurring traffic mode. Using these traffic modes, we detail a topology-routing co-optimization methodology to compute an optimal fractional topology. In this step, our goal is to compute a fractional topology that is optimized for all of the traffic modes found, as discussed in Section 5. The logic

employed here is that a fractional topology that is optimized for all of the traffic modes will allow our topology to cover sufficient grounds to account for traffic changes over time.

6.1 Definitions

We start off by giving a more rigorous definition of what a fractional topology is. As its namesake suggests, the number of links between two pods. In addition to non-integer inter-pod link counts, a fractional topology must satisfy additional constraints, as given by:

Definition 6.1. $D = d_{ij} \in \mathbb{R}^{n \times n}$ is a fractional topology i.f.f it satisfies

$$\sum_{j \in S} d_{ij} \leq r^i_{eg} \; \forall \; i \in S \tag{1}$$

$$\sum_{i \in S} d_{ij} \leq r^j_{ig} \; \forall \; j \in S \tag{2}$$

Note that (20) and (21) denote the egress degree constraints and ingress degree constraints, respectively. These are the most fundamental set of constraints that satisfy the out-degree and in-degree of each pod. In addition, there are two main properties of the fractional topology we wish to discuss. Firstly, the fractional topology noticeably ignores the patch panel layer. This is done because the accouting for the patch panel layer carries neither mathematical nor algorithmic significance. Not only does its accounting unnecessarily increases the number of variables needed to represent a fractional topology, the fact that the panel layer will be considered in Step 3 of Figure 2 as we map the fractional topology into an integer logical topology renders it redundant. Secondly, fractional number of links interconnect two pods are allowed because it permits us to compute a fractional topology by formulating a LP instead of an ILP, which has significantly higher complexity.

Next, we define $\Omega T, D = \{\omega_p \mid p \in \bigcup_{i,j \in S \times S} P_{ij}\}$ as the multi-hop routing solution of traffic matrix T on fractional topology D. For any pair of traffic matrix and fractional topology, we define the meaning for feasibility of T, D as:

Definition 6.2. T, D is feasible i.f.f there exists a routing solution $\Omega T, D$ s. t:

$$t_{ij} = \sum_{p \in P_{ij}} \omega_p \quad \forall \; i, j \in S \tag{3}$$

$$\sum_{p \in Pe_{ij}} \omega_p \leq d_{ij} b_{ij} \quad \forall \; i, j \in S \tag{4}$$

Constraint (3) enforces i, j traffic demands to be satisfied via all routable paths, while constraint (4) ensures that the traffic flowing through each link cannot exceed its bandwidth. Simply put, T, D is feasible if there exists a routing solution for T on D such that all traffic flow demands can be met without exceeding D's inter-pod link bandwidths.

6.2 Finding the Best-fitting Fractional Topology

For any given traffic mode matrix, T_κ, we want to find the corresponding fractional topology of best fit, namely D^*_κ. For T_κ, the fractional topology that has the best-fit is defined as:

1. D^*_κ should maximize μ_κ for all feasible $\mu_\kappa T_\kappa, D_\kappa$.
2. D^*_κ should be most similar to T_κ

Fulfilling the first property makes a fractional topology merely a "good" fractional topology. There may be many such fractional topologies that can maximize μ_κ, since fundamentally property 1 only optimizes for the hottest pod, and does not place much emphasis on the other pods that do not eject or inject as much traffic into the network. Therefore, we strengthen the best-fit definition by introducing property 2, which essentially selects the fractional topology that looks most similar to T_κ among the set of fractional topologies that maximizes μ_κ. Assuming that μ^*_κ denotes the maximum scale-up factor among the set of all feasible $\mu_\kappa T_\kappa, D_\kappa$, we find the best fitting fractional topology, D, for each traffic mode using:

$$\begin{aligned} \min_{\boldsymbol{\omega_p}, \mathbf{d}} \quad & \sum_{i,j \in S} \; \sum_{p \in P_{ij} l_p > 1} \left(\omega_p\right)^2 \\ \text{s. t: } 1 \quad & \sum_{p \in P_{ij}} \omega_p = \mu^*_\kappa t_{ij}, \quad \forall \; i, j \in S \\ 2 \quad & \sum_{p \in Pe_{ij}} \omega_p \geq \mu^*_\kappa \; d_{ij} b_{ij}, \quad \forall \; i, j \in S \\ 3 \quad & 20 \; , \; 21 \end{aligned} \tag{5}$$

At this point, we've obtained the best-fitting fractional topology to each of the traffic modes. We need to combine all these individual fractional topologies into one fractional topology with which we will map onto the patch panels to obtained the final integer logical topology. For computing the combined fractional topology, we linearly combine the individual best-fitting fractional topologies by formulating the following LP:

$$\begin{aligned} \max_{\alpha > 0, \mathbf{d}^*} \quad & \alpha \\ \text{s. t: } 1 \quad & d^*_{ij} \geq \; \alpha \; d^\zeta_{ij}, \; \forall \; i, j \in S, \; \zeta \in \{1, 2, ..., \kappa\} \\ 2 \quad & d^*_{ij} \text{ is feasible} \end{aligned} \tag{6}$$

Where α is the scaling factor of each individual fractional graph. Constraint 1 in (24) is imposed so that

each i, j fractional link in the combined fractional graph, d^*_{ij}, can dominate all other α-scaled $d^\zeta_{ij}\ \forall\ \zeta \in \{1, 2, ..., \kappa\}$.

7 STEP 3: MAPPING FRACTIONAL TOPOLOGY ONTO PATCH-PANELS

Having computed the optimal fractional topology, D^*, previously, we now attempt to map D^* onto the patch panels such that the final logical topology best matches the fractional topology. Unfortunately, we cannot ignore the patch panel layer here. Since the logical topology has to be in integer form, obtaining the perfect match to this problem is equivalent to an NP-hard three-dimensional matching problem [10, 16]. In this work, we present two different relaxation approaches inspired by techniques commonly employed in convex optimization that provide close to optimal solutions. Before diving into the details of each approach, we briefly discuss some common structures to the patch-panel problem:

Hard / Physical Constraints: This constraint states that the total number of integral flows that reach s_j through o_k cannot exceed the total physical links connecting the ingress ports of s_j from o_k:

$$\sum_{i \in S} x^k_{ij} \leq h^k_{ig} j \quad \forall\ k \in O,\ j \in S \tag{7}$$

By reciprocity, total number of integral flows through o_k originating from s_i cannot exceed the total number of physical links connecting the egress links of s_i to o_k .

$$\sum_{j \in S} x^k_{ij} \leq h^k_{eg} i \quad \forall\ k \in O,\ i \in S \tag{8}$$

Soft / Matching Constraints: We also want to match the logical topology with the fractional topology, therefore the following constraints are introduced:

$$c^-_{ij} \leq \sum_{k \in O} x^k_{ij} \leq c^+_{ij}, \quad \forall\ i, j \in S \tag{9}$$

Where $c^-_{ij} = \lfloor d_{ij} \rfloor$ and $c^+_{ij} = \lceil d_{ij} \rceil$. A logical topology is deemed to perfectly match the fractional topology if for every s_i, s_j pod pair, the inequality in (9) is satisfied. It is possible that certain permutations of D do not have a feasible solution.In those cases, we simply aim to minimize the number of soft constraints violated.

7.1 Lagrangian Dual Method

We draw significant inspiration from Low et al. [23] in our approach with the Lagrangian dual method. The act of transforming the primal problem into its Lagrangian dual problem introduces the dual variables that will be used to "score" the cost of forming each logical link. The dual problem is:

$$\begin{aligned} \min_{\mathbf{p}^\pm} \max_{\mathbf{x}} \{ U\mathbf{x} + \sum_{i,j \in S} \big[p^-_{ij} \big(\sum_{k \in O} x^k_{ij} - c^-_{ij} \big) - p^+_{ij} \big(\sum_{k \in O} x^k_{ij} - c^+_{ij} \big) \big] \} \\ \text{s.t } 1 \quad 7 \text{ and } 8 \\ 2 \quad \mathbf{p}^\pm \succcurlyeq 0 \end{aligned} \tag{10}$$

Where $p^\pm$ are the dual variables, and $U\mathbf{x}$ denotes the objective function of the primal problem. In our implementation, we chose the primal objective function to be:

$$U\mathbf{x} = \sum_{i,j \in S} \sum_{k \in O} -\big(x^k_{ij} - h^k_{ij} \big)^2 \tag{11}$$

Where $h^k_{ij} = \min\big(h^k_{eg} s_i, h^k_{ig} s_j \big)$. We take advantage of the fact that $x^k_{ij} \leq h^k_{ij}\ \forall i, j \in S,\ k \in O$ to "coerce" the resulting logical topology to form more logical links through higher x^k_{ij} values. This will force the logical topology solution to more fully-utilize the physical links available to them due to $U\mathbf{x}$ behaving like a least-squared curve-fitting problem. In addition, a quadratic objective function possesses a strong convexity which stabilizes solution convergence.

Observing (10) closely gives us a good intuition on how this method works. The dual variables $\mathbf{p}^\pm$ essentially "scores" the cost of forming logical links between pods. For instance, when a particular pod pair s_i, s_j is significantly under-provisioned (i.e $\sum_{k \in O} x^k_{ij} < c^-_{ij}$), p^-_{ij} will increase while p^+_{ij} will decrease such that future patch panels will attempt to form more logical links between s_i, s_j. The opposite is true for when s_i, s_j pair is significantly over-provisioned. In a sense, the regulatory behavior $\mathbf{p}^\pm$ exhibits a negative-feedback behavior by incentivizing or disincentivizing the formation of s_i, s_j logical links at each patch panel until the soft constraint is met. Rearranging (10) gives us:

$$\min_{\mathbf{p}^\pm} \{ \big[-\sum_{k \in O} \min_{\mathbf{x}^k} Q_k \mathbf{x}^k \big] + \sum_{i,j \in S} p^+_{ij} c^+_{ij} - \sum_{i,j \in S} p^-_{ij} c^-_{ij} \} \tag{12}$$

Where $Q_k \mathbf{x}^k \approx \sum_{i,j \in S} \big[p^+_{ij} - p^-_{ij} - 2h^k_{ij} + 2x^{k\,*}_{ij} \big] x^k_{ij}$. Note that $\min_{\mathbf{x}^k} Q_k$ is simply a Min-cost Flow (MCF) problem for o_k with x^k_{ij} being the arc flow and $p^+_{ij} - p^-_{ij} - 2h^k_{ij} + 2x^{k\,*}_{ij}$ the cost per unit flow, which is obtained by applying first order linear approximation on an otherwise quadratic $Q_k \mathbf{x}^k$ evaluated at $x^k_{ij} = x^{k\,*}_{ij}$. Interestingly, (12) decouples an otherwise difficult-to-solve convex optimization problem with integral constraints, (10) w.r.t x^k_{ij} into a series of MCF sub-problems, which can be solved efficiently with integer solutions using polynomial-time algorithms like Goldberg-Tarjan [6].

7.1.1 *Dual Gradient Descent.*

In order to solve the overall minimization problem w.r.t $\mathbf{p}^{\pm}$, we use an iterative gradient descent method with the step function $\delta\tau = \frac{1}{\tau+1}$, where τ is the current iteration. The equation that governs the dual variables for the next iteration is:

$$\mathbf{p}^{\pm}\tau + 1 = \mathbf{p}^{\pm}\tau \; - \; \frac{1}{\tau+1}\big(\pm\,\mathbf{c}^{\pm} \mp \sum_{k\in O}\mathbf{x}^{k*}\big) \qquad (13)$$

Note that $\sum_{\tau=0}^{\infty}\delta\tau$ forms a harmonic series, which sum approaches infinity as we take infinitely many iterations. This way, $\mathbf{p}^{\pm}$'s growth is not handicapped by the step size if their optimal values are large. The overall algorithm for computing (12) is included in Appendix **??** for interested readers.

7.2 Barrier Penalty Method

Here we introduce the barrier penalty method as a complementary implementation to the Lagrangian dual method. Like the previous method, this method is also inspired by the barrier function method commonly-used in convex optimization. In our implementation, the central idea is to introduce an objective that penalizes soft constraint violation, subject to the physical constraints (7) and (8):

$$\begin{aligned}\min_{\mathbf{x}} U\mathbf{x} = & \sum_{i,j\in S}\Big[\big(\sum_{k\in O}x_{ij}^{k} - c_{ij}^{-}\big)\big(\sum_{k\in O}x_{ij}^{k} - c_{ij}^{+}\big)\Big] \\ \text{s.t:} \quad & 7 \text{ and } 8\end{aligned} \qquad (14)$$

Note that the objective function in (14) is quadratic w.r.t $\sum_{k\in O}x_{ij}^{k}$, which is minimized when the soft-constraint for pod pair s_i, s_j is satisfied. Unlike the Lagrangian dual method, this method is somewhat reminiscent of a cooperative game in game theory in which every patch panel (the player) attempts to optimize for the same global objective function (the cost function). Optimizing for (14) with integral-constraints is likely NP-hard. Thus, we again apply first order linear approximation w.r.t x_{ij}^{k} to reduce complexity, arriving at:

$$\min_{\mathbf{x}} U\mathbf{x} \approx \sum_{k\in O}\min_{\mathbf{x}^{k}}\sum_{i,j\in S}\Big[\sum_{k\in O}2x_{ij}^{k\,*} - c_{ij}^{+} + c_{ij}^{-}\Big]x_{ij}^{k} \qquad (15)$$

Which is again a MCF problem with $\big[\sum_{k\in O}2x_{ij}^{k\,*} - c_{ij}^{+} + c_{ij}^{-}\big]$ as the cost per unit x_{ij}^{k} flow. Unlike (14), (15) can be solved using netflow algorithms in polynomial time with integer x_{ij}^{k} solutions.

8 ANALYSIS OF PATCH PANEL-MAPPING ALGORITHMS

8.1 Optimality Analysis

Here we examine the optimality of both the barrier penalty and Lagrangian dual methods. We've generated 900 different, heterogeneous network fabrics with topology size ranging from 12 to 100 pods in increments of 2. The properties of the network fabrics are briefly summarized in Table 2. All 900 topologies are tested using 4 types of traffic patterns, namely: 1) ρ-nearest neighbor traffic, 2) Bipartite traffic 3) Random traffic, and 4) Radix Ratio.

The ρ for ρ-nearest neighbor traffic is the traffic pattern in which a pod only sends traffic to peer pods that are within ρ-units of index in circular distance. Bipartite traffic splits the network fabric into roughly two-equal sized sets, in which each pod sends traffic uniformly to pods in the opposite set.

We evaluate the quality of the relaxed-optimization approaches of both the Lagrangian dual method and the barrier penalty method using two metrics: fraction of constraints satisfied and loss of optimality. Given a fractional demand graph, $d_{ij} \in \mathbb{R}^{N\times N}$, a constraint is satisfied if the integer connectivity satisfies: $\lfloor d_{ij} \rfloor \leq \sum_{k\in O}x_{ij}^{k} \leq \lceil d_{ij} \rceil$. Meanwhile, optimality loss is defined as the ratio of MLU obtained from routing over an integer logical topology to the MLU obtained from routing over a fractional topology, given the same traffic matrix. This is a useful metric, as it measures directly the increase in MLU compared to the theoretical lower bound achieved using a fractional topology. It is important to keep in mind that the fractional topology assumes full-connectivity, which is not the case in real DCN. Therefore, both optimality loss and constraint violation metrics are expected to improve when the fractional topology is derived when accounting for physical constraints.

Figure 4 shows that in both algorithms perform very well for most topology sizes, with the Lagrangian duality method consistently outperforming the barrier method aside for the bipartite traffic case, in which case they exhibit comparable optimality losses. Under the ρ-nearest neighbor traffic load, over 99% of the fabrics optimality loss of under 10%, while all the fabrics under bipartite and random traffic loads experience less than 10% optimality loss.

Evaluating the constraints violated also gives us a good idea of the quality of solutions obtained from the proposed methods. Results in Figure 5 also shows that in general, the Lagrangian duality algorithm slightly

Type	Hardware Description
Type 1	128-port patch-panels, One 100G 1024-port pod, $n - \lfloor \frac{n}{10} \rfloor m$ 40G 256-port pods, $\lfloor \frac{n}{10} \rfloor m$ 40G 512-port pods, where $m \in \{1, 2, ..., 10\}$.
Type 2	128-port patch-panels, Two 100G 1024-port pod, $n - \lfloor \frac{n}{10} \rfloor m$ 40G 256-port pods, $\lfloor \frac{n}{10} \rfloor m$ 40G 512-port pods, where $m \in \{1, 2, ..., 10\}$.

Table 2: Physical topology descriptions

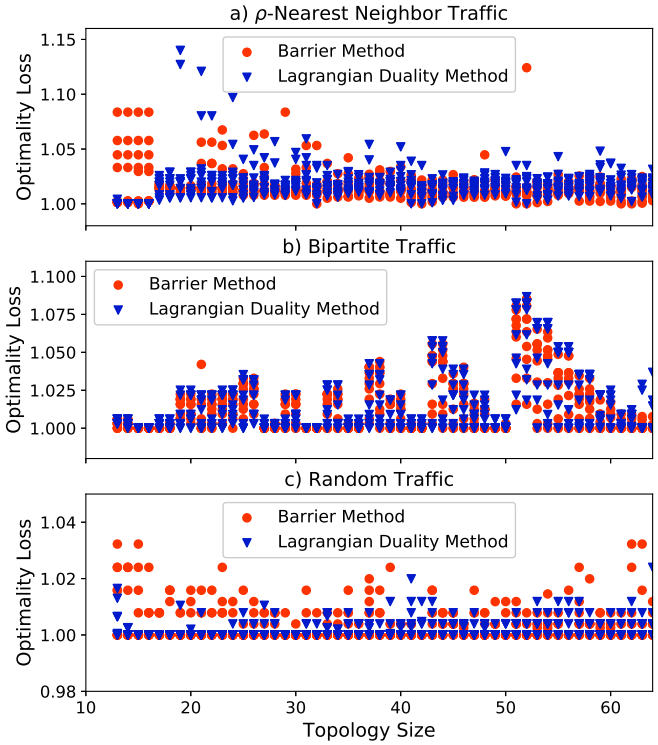

Figure 4: Benchmark on optimality loss for k-nearest neighbor, bipartite, and random traffic patterns

outperforms the barrier method. In general, both methods perform better as the topologies grow in size, which could be attributed to the existence of a larger solution space. It is worth noting that the striping ratio traffic load, about 50% of the topology instances have more than 10% of the constraints being violated when the topology size is under 30 pods.

8.2 Runtime Analysis

The runtime measured in seconds for both Lagrangian duality and barrier penalty methods running 50 iterations are shown in Figure 6. We noticed that while the exact runtime differs for different traffic matrices, the general trend is still similar with runtimes within the same order of magnitude. Therefore, we only show the runtime distribution as a function of fabric size under random traffic load. Note that in general, the barrier method is about twice as fast as the Lagrangian duality method. This speedup is likely linearized approximation of the objective function employed in the barrier method. Both methods completes within $\tilde{1}$50s for even the largest fabric size with >100 pods. These runtimes make both methods suitable for deployment within datacenters equipped with high speed optical switches. Under these conditions, the algorithms are able to map fractional topologies onto the optical switches at high-frequency.

8.3 Barrier Penalty vs Lagrangian Dual

In general, the Lagrangian duality method yields better optimality results and violates fewer matching constraints than the barrier penalty method, at the cost of a higher runtime. This higher runtime can be accredited to the Lagrangian dual method being complicated to implement in practice due to the introduction of its dual variables. The primal objective of the Lagrangian dual method we proposed does not seem to directly minimize matching constraint violation directly, as the constraints violation are only penalized indirectly through the introduction of the Lagrangian relaxation function. Interestingly, Lagrangian dual method has more adaptability due to the dual variables, through which the $\mathbf{p}^{\pm}$ variables and be updated in the iterative algorithm and help satisfy more constraints. The barrier method should in theory penalize constraint violation more harshly due to the quadratic nature of its objective function. However, it does not have the same kind of adaptability that the Lagrangian dual method has.

We envision combining the best features of both methods through an Augmented Lagrangian Dual approach for future work. We posit that this approach could offer both adaptability through the dual variables, while penalizing matching constraints more harshly through higher-order barrier functions of the dual variables. Additionally, it may also be worth trying barrier penalty functions that penalizes over-provisioning of logical links between s_i, s_j less severely than under-provisioning. This is because under-provisioning logical links between every may more negatively impact network congestion, especially when there is a significant amounts of traffic flowing through the s_i, s_j links. That said, it is difficult to say that penalizing over-provisioning less may lead to better results. As is often the case in these NP-hard shared-resources [16] problems, punishing an

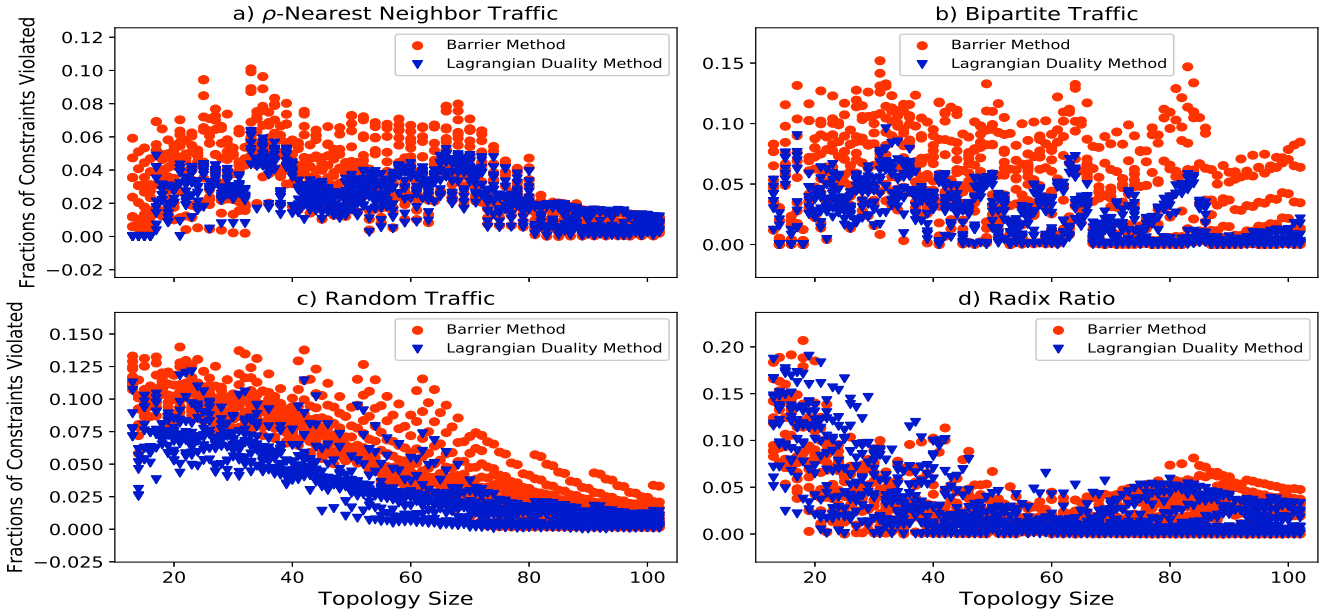

Figure 5: Benchmark on fraction of constraints violated for 4 synthetic traffic patterns

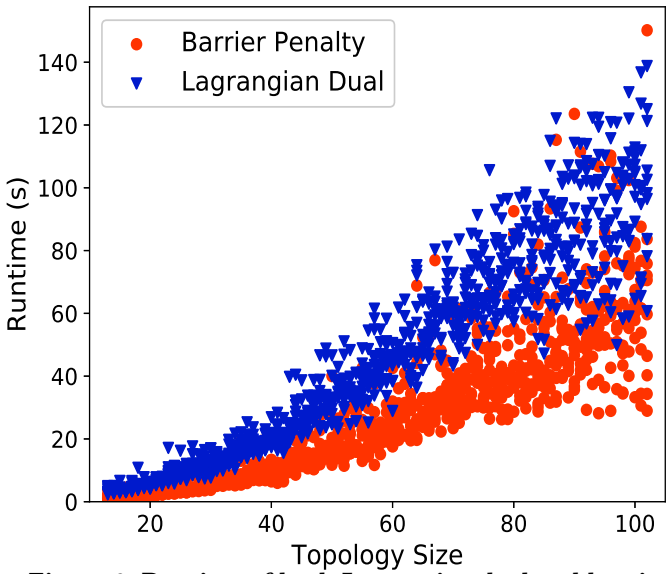

Figure 6: Runtime of both Lagrangian dual and barrier methods with $t_{max} = 50$

over-provisioned logical link less may result in the finite link resources being deprived to other under-provisioned inter-pod links. Nonetheless, these are ideas that are worth exploring in more detail in future literature.

9 PERFORMANCE EVALUATION

9.1 WCMP Traffic Engineering

Given a traffic matrix T and a logical topology $\mathbf{x}$, we employ a two-hop, centralized WCMP [39] routing with perfect traffic knowledge to for routing T over topology $mathbf{x}$. Since we are solely interested in evaluating the quality of the logical topology, perfect traffic knowledge is assumed in order to reduce the impact of routing uncertainties on network performance metrics. The routing weights w_p are derived using:

$$\begin{aligned} \min_{\eta, \boldsymbol{\omega_p}} \quad & \eta \\ \text{s. t: } 1 \quad & \sum_{p \in P_{ij}} \omega_p = t_{ij} \quad \forall\, i, j \in S \\ 2 \quad & \sum_{p \in Pe_{ij}} \omega_p \leq \eta\, b_{ij} \sum_{k \in O} x_{ij}^k \quad \forall\, i, j \in S \end{aligned} \tag{16}$$

Where η denotes the max link utilization, while ω_p denotes WCMP weight allocated to path p. Constraint 1 in (16) states that the WCMP weights through all paths between s_i to s_j to equal the traffic flow from s_i to s_j, ensuring that the routing has to ensure that all traffic flows have to reach their intended destinations. Constraint 2 enforces all inter-pod link utilizations to be bounded by the max link utilization. Note that (16) is essentially traffic engineering, with the objective of minimizing maximum congestion.

9.2 Performance Metrics & Experimental Setup

In our experiments, we simulate the effects of traffic-aware topology engineering on 4 production data-centers using real historical traffic data. Since our evaluations are based on real traffic data, it is very natural to use link-utilizations to measure the network performance. These metrics are computed hourly over a month-long

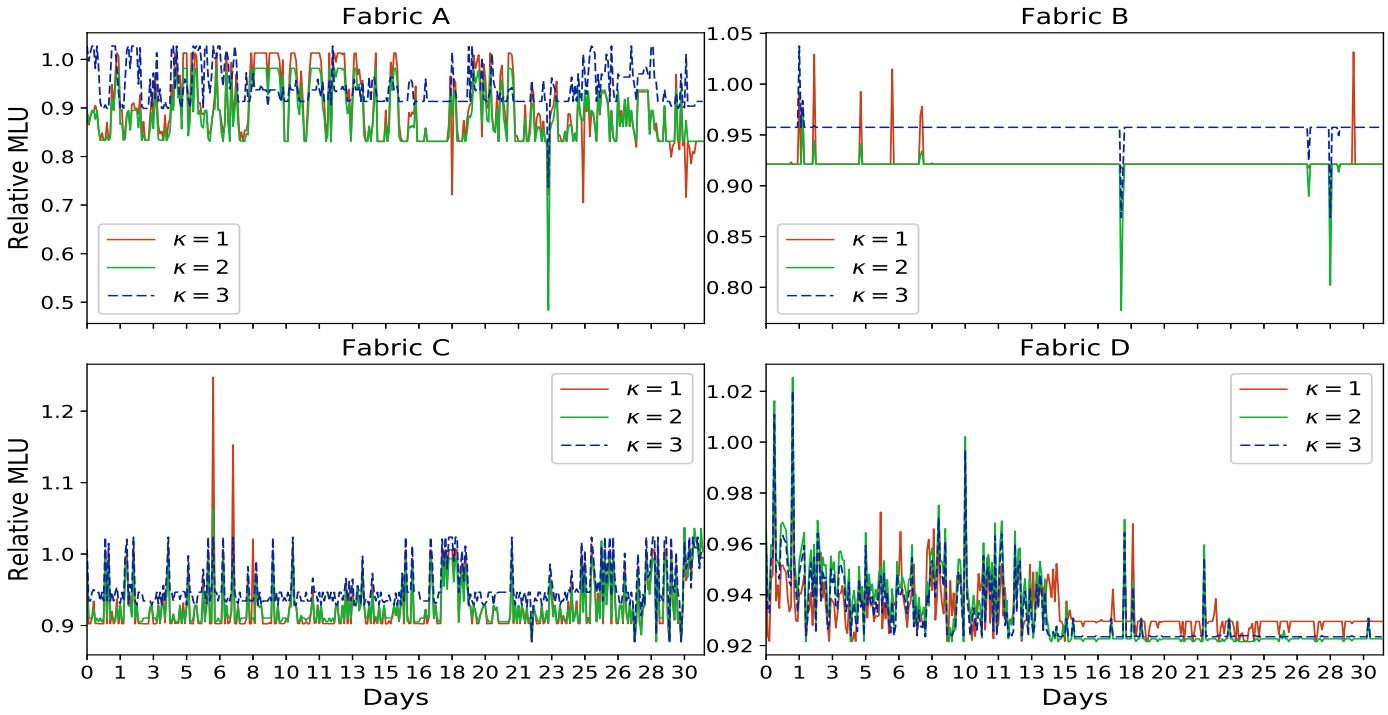

Figure 7: Month-long relative MLU performance with κ = 1, 2, and 3.

duration. The network performance of the same traffic are also evaluated on uniform topologies that only use traffic engineering, as described in (16) for link load-balancing. The evaluated metrics are:

- **Relative Max Link Utilization (MLU)**
- **Relative 90-th Percentile Link Utilization (LU90)**
- **Stretch Factor (SF)**

Link utilization measures the ratio of traffic flowing over a link to its bandwidth, and max link utilization simply the most highly utilized link within the network under a given traffic load. In addition to measuring MLU, which is a good proxy for maximum network link congestion, we also measure the 90-th percentile link utilizations to get a sense of how quickly the link congestion distribution decays. To show the relative impact on link congestion introduced by traffic-aware topology engineering and to anonymize traffic data, we normalize using the link utilization metrics measured using a baseline uniform topology, and call these metrics relative link utilizations.

Additionally, we also compute the stretch factor hourly using $\frac{\sum_{p \in P} l_p \omega_p}{\sum_{p \in P} \omega_p}$. Stretch factor represents the average hop count for each unit of traffic, and is a useful proxy for estimating the per-packet delay. Since we are using two-hop routing, the stretch factor should lie between 1 and 2.

9.3 Discussion

Figure 7 shows the relative MLU performance using topology designed from 1 to 3 traffic clusters. Results in all 4 fabrics show that for most time slots, picking $\kappa = 1$ is sufficient and outperforms $\kappa = 2, 3$. This makes sense, as factoring in more traffic matrices in the topology engineering process not just complicates the model, it also tends to render the topology less optimal for any single traffic matrix. Fabric D shows the usefulness of considering multiple traffic matrices in topology engineering, as the links become noticeably less congested past day 13, indicating that shifted traffic patterns cannot not be accounted for picking just 1 cluster. In fabrics B and C, sporadic significant MLU degradation occur when $\kappa = 1$ but not when $\kappa = 2, 3$. These instances further highlights the need for accounting for multiple traffic-matrices when performing topology engineering to render the topology more robust to potential significant traffic pattern changes.

Figure 8 shows the relative LU90 measurements. Here we can see clearly that LU90 improves rather significantly using topology engineering, since more links can be allocated between pod pairs with higher traffic flows.

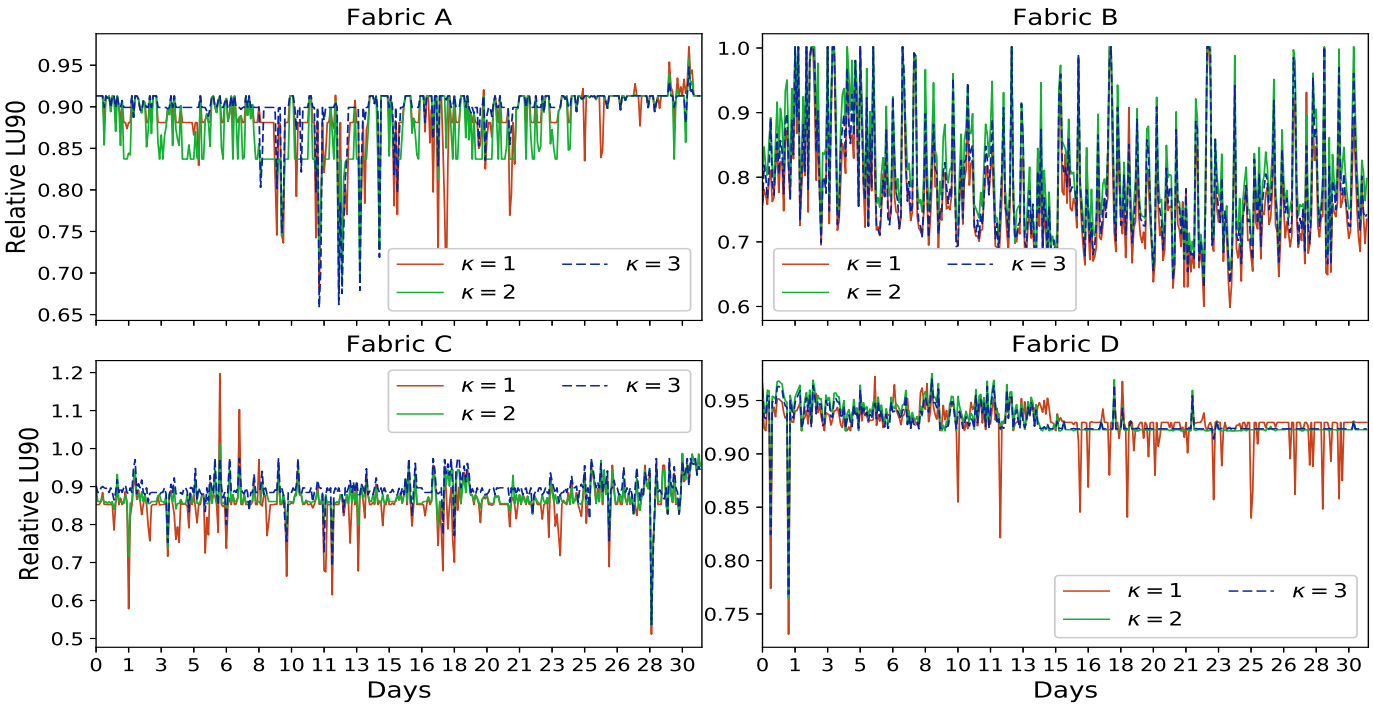

Figure 8: Month-long relative LU90 performance with κ = 1, 2, and 3.

As such, more traffic can take direct-hop paths to reach their destination without driving up the link congestion levels. This is not possible in a uniform topology, since keeping the link congestion levels low would often imply sending traffic via non-minimal paths, therefore generating more network traffic and driving up the average and percentile link congestions.

In addition to link congestion, the performance improvements enabled by using our propose methodology becomes much clearer when measuring the stretch factor, as shown in Figure 9. Here we can see that for all 4 fabrics, almost all of the traffic are allocated to single-hop paths. Besides lowering the average link congestion levels, the lower average hop count implies that the average time a packet spends within the network is less as a result of taking a direct path to its intended destination. This makes topology engineering ideal for improving performances of more latency-sensitive applications.

Our results also points out the inadequacies of traffic engineering alone, since traffic engineering relies trades link de-congestion for increased average hop-count (and thus average packet latency). In fact, the stretch-factor and maximum link congestion are often times mutually-exclusive, since reducing link congestion might require the routers to divert traffic along non-minimal paths. We emphasize that a traffic-aware topology-engineering methodology solves both these issues at once.

10 WHEN TO TRIGGER TOPOLOGY RECONFIGURATION

One potential issue of our approach is that as traffic modes begin to change, it is likely that the network will begin to suffer performance degradation. Therefore, the general rule of thumb is that the entire pipeline shown in Figure 2 should be triggered whenever traffic modes begin to demonstrably change. Events that lead to drastic changes in long-term traffic pattern tends to arise when these is a major DCN expansion. A new logical topology should be computed whenever there is a DCN expansion. Adding new pods into the network can significantly affect the traffic modes. In addition, topology reconfiguration should be triggered when there is a link upgrade for any pod within the network, sicne the . Sometimes, the network performance can degrade over time as the underlying production areas change. A change in production area can upset the traffic modes tremendously, since the different applications can exhibit vastly different network traffic behavior. In general, topology reconfiguration should be triggered whenever there is a change in this .

To reduce the frequency of topology reconfiguration, we need to work on desensitizing the network performance change as traffic modes exhibit drastic changes.

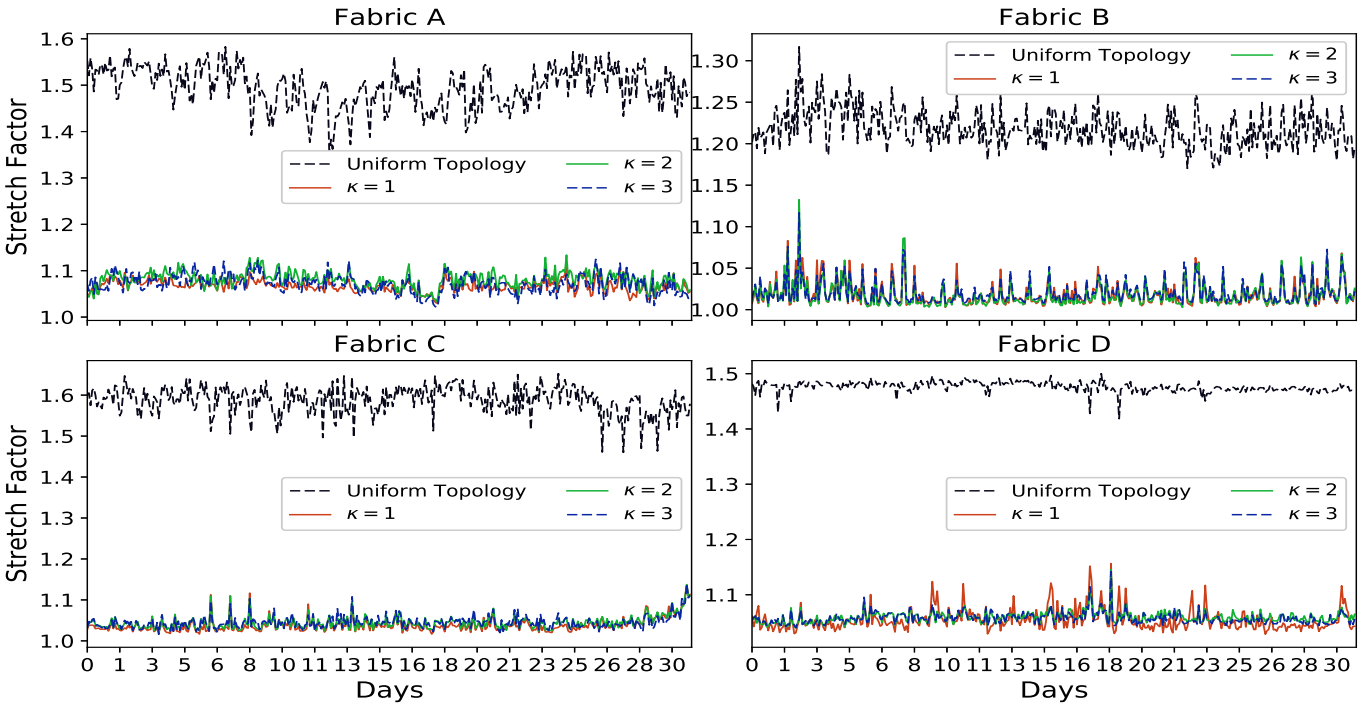

Figure 9: Month-long stretch factor performance with κ = 1, 2, 3, and uniform topology.

This is an ongoing work for us, and thus will be treated more rigorously in future works.

11 CONCLUSION

We propose a traffic engineering methodology with that does not require traffic-prediction. By considering multiple persistent traffic modes that can be derived using historical traffic matrices, our topology engineering solution proves robust under traffic change for long periods of time compared to uniform topologies that rely solely on traffic engineering for network load-balancing. Most importantly, we do not rely on sophisticated and immature circuit-switching hardware, therefore can be readily deployed in today's DCN equipped with cheap commodity patch-panels. Our experimental results using production traffic show that our topology engineering methodology can reduce maximum link congestion, while also keeping most traffic within one-hop. We believe that for future works, different clustering techniques can be looked into in order to select more representative traffic matrices. Similarly, many different methods of mapping a computing logical topology and map it onto a non-fully connected network can should be looked into.

REFERENCES

[1] Jung Ho Ahn, Nathan Binkert, Al Davis, Moray McLaren, and Robert S Schreiber. 2009. HyperX: topology, routing, and packaging of efficient large-scale networks. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. ACM, 41.

[2] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, Vol. 38. ACM, 63–74.

[3] 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.

[4] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. 2009. Understanding data center traffic characteristics. In *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 65–72.

[5] Maciej Besta and Torsten Hoefler. 2014. Slim fly: A cost effective low-diameter network topology. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 348–359.

[6] Ursula Bünnagel, Bernhard Korte, and Jens Vygen. 1998. Efficient implementation of the Goldberg–Tarjan minimum-cost flow algorithm. *Optimization Methods and Software* 10, 2 (1998), 157–174.

[7] Christina Delimitrou, Sriram Sankar, Aman Kansal, and Christos Kozyrakis. 2012. ECHO: Recreating network traffic maps for datacenters with tens of thousands of servers. In *Workload Characterization (IISWC), 2012 IEEE International Symposium on*. IEEE, 14–24.

[8] 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. *ACM SIGCOMM Computer Communication Review* 41, 4 (2011), 339–350.

[9] 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*. Optical Society of America, OTuP1.

[10] Klaus-Tycho Foerster, Manya Ghobadi, and Stefan Schmid. 2018. Characterizing the algorithmic complexity of reconfigurable data center architectures. (2018).

[11] 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 *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 216–229.

[12] 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 *ACM SIGCOMM computer communication review*, Vol. 39. ACM, 51–62.

[13] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. 2009. BCube: a high performance, server-centric network architecture for modular data centers. *ACM SIGCOMM Computer Communication Review* 39, 4 (2009), 63–74.

[14] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. 2008. Dcell: a scalable and fault-tolerant network structure for data centers. In *ACM SIGCOMM Computer Communication Review*, Vol. 38. ACM, 75–86.

[15] 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 *ACM SIGCOMM Computer Communication Review*, Vol. 44. ACM, 319–330.

[16] Robert W Irving and Mark R Jerrum. 1994. Three-dimensional statistical data security problems. *SIAM J. Comput.* 23, 1 (1994), 170–184.

[17] Navendu Jain, Ishai Menache, Joseph Seffi Naor, and F Bruce Shepherd. 2012. Topology-aware vm migration in bandwidth oversubscribed datacenter networks. In *International Colloquium on Automata, Languages, and Programming*. Springer, 586–597.

[18] John Kim, Wiliam J Dally, Steve Scott, and Dennis Abts. 2008. Technology-driven, highly-scalable dragonfly topology. In *Computer Architecture, 2008. ISCA'08. 35th International Symposium on*. IEEE, 77–88.

[19] Hong Liu, Cedric F Lam, and Chris Johnson. 2010. Scaling optical interconnects in datacenter networks opportunities and challenges for WDM. In *High Performance Interconnects (HOTI), 2010 IEEE 18th Annual Symposium on*. IEEE, 113–116.

[20] 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*, Vol. 14. 1–15.

[21] 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.

[22] Vincent Liu, Daniel Halperin, Arvind Krishnamurthy, and Thomas E Anderson. 2013. F10: A Fault-Tolerant Engineered Network.. In *NSDI*. 399–412.

[23] 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.

[24] William M Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, 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*. ACM, 267–280.

[25] Antonios D Papaioannou, Reza Nejabati, and Dimitra Simeonidou. 2016. The Benefits of a Disaggregated Data Centre: A Resource Allocation Approach.. In *GLOBECOM*. 1–7.

[26] George Porter, Richard Strong, Nathan Farrington, Alex Forencich, Pang Chen-Sun, Tajana Rosing, Yeshaiahu Fainman, George Papen, and Amin Vahdat. 2013. *Integrating microsecond circuit switching into the data center*. Vol. 43. ACM.

[27] Sivasankar Radhakrishnan, Malveeka Tewari, Rishi Kapoor, George Porter, and Amin Vahdat. 2013. Dahu: Commodity

switches for direct connect data center networks. In *Proceedings of the ninth ACM/IEEE symposium on Architectures for networking and communications systems*. IEEE Press, 59–70.

[28] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, 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 *ACM SIGCOMM Computer Communication Review*, Vol. 45. ACM, 183–197.

[29] Ankit Singla, Chi-Yao Hong, Lucian Popa, and Philip Brighten Godfrey. 2012. Jellyfish: Networking Data Centers, Randomly.. In *NSDI*, Vol. 12. 1–6.

[30] 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*. ACM, 8.

[31] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. 2017. OSQP: An Operator Splitting Solver for Quadratic Programs. *ArXiv e-prints* (Nov. 2017). arXiv:math.OC/1711.08013

[32] Min Yee Teh, Jeremiah J Wilke, Keren Bergman, and Sébastien Rumley. 2017. Design space exploration of the Dragonfly topology. In *International Conference on High Performance Computing*. Springer, 57–74.

[33] Amin Vahdat, Hong Liu, Xiaoxue Zhao, and Chris Johnson. 2011. The emerging optical data center. In *Optical Fiber Communication Conference*. Optical Society of America, OTuH2.

[34] 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*. ACM, 16.

[35] 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. *ACM SIGCOMM Computer Communication Review* 41, 4 (2011), 327–338.

[36] 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.

[37] Yin Zhang and Zihui Ge. 2005. Finding critical traffic matrices. In *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*. IEEE, 188–197.

[38] 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*.

[39] Junlan Zhou, Malveeka Tewari, Min Zhu, Abdul Kabbani, Leon Poutievski, Arjun Singh, and Amin Vahdat. 2014. WCMP: Weighted cost multipathing for improved fairness in data centers. In *Proceedings of the Ninth European Conference on Computer Systems*. ACM, 5.

[40] Xiang Zhou, Hong Liu, and Ryohei Urata. 2017. Datacenter optics: requirements, technologies, and trends. *Chinese Optics Letters* 15, 5 (2017), 120008.

[41] 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. *ACM SIGCOMM Computer Communication Review* 42, 4 (2012), 443–454.

A APPENDICES

A.1 Simplex-based Patch Panel Reconfiguration

In mapping the fractional graph to the patch-panels such that the logical topology looks identical to the the fractional graph. We consider the logical topology as being identical to the fractional demand if $_{k\in O}\, x_{ij}^{k} \leq c_{ij}^{+}$ and $_{k\in O}\, x_{ij}^{k} \geq c_{ij}^{-}$. To find the integer solution to this SAT problem is NP-hard in general, therefore we employ an LP-relaxation of the SAT-problem. The SAT problem is rewritten as a linear program as follows:

$$\begin{aligned} &\max 0 \\ &\text{s. t} \quad 7,\ 8,\ 9 \end{aligned} \tag{17}$$

This algorithm relies on the simplex method's natural propensity to produce integer solutions, as the pivoting algorithm iteratively searches the optimal solutions on the polytope vertices. Conveniently, the vast majority of integer solutions to (17) do conveniently lie on the vertices, laying well within the simplex algorithm's search space. For the remaining non-integer solutions, we round them into integers using min-cost flow with the the arc-weights of $\left\lceil x_{ij}^{k}{}^{*} \right\rceil - x_{ij}^{k}{}^{*}$.

While certainly intuitive, this method does not work well with the fractional graph optimization derived in for all permutations of traffic matrices. We've tried this method using a variety of fractional graph permutations that are non-uniform and noted that in most cases, a the simplex algorithm crashes due to the inability to find a feasible solution. Since this method breaks down when there is no feasible solution to the SAT problem, it is not suitable for deployment for all permutations of the fractional graph. The simplex method also does not scale well to larger problem sizes due to a significant runtime increase. In fact, the simplex method has a provable exponential worst-case runtime makes this method not robust enough for deployment in real systems.

A.2 Topology Engineering Benefits

We consider 2-hop routing so as to allow some the network to load-balance through routing. Consider the following example:

We posit that there are certain traffic patterns under which traffic-aware topology design will prove most useful. One such example is a M-to-N bipartite traffic pattern, whereby each of the M senders send traffic to all N receivers. These traffic are expected to arise when datacenter architectures become increasingly disaggregated, where there are M pods with purely compute resources that need to write data into storage distributed across

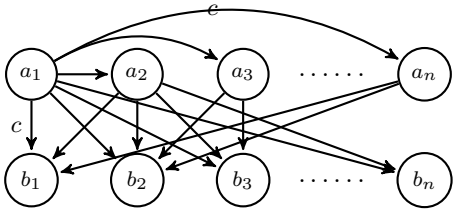

Figure 10: Uniform topology

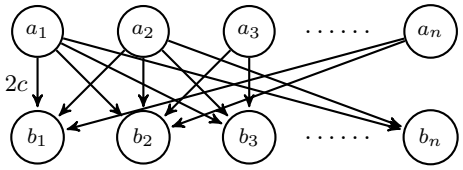

Figure 11: Traffic-aware topology for bipartite traffic

N storage pods. To clarify this concept, we present the following example.

Consider a datacenter network with 2 sets of pods $A = \{a_1, a_2, ..., a_n\}$ and $B = \{b_1, b_2, ..., b_n\}$, whereby the set of all pods is given as $S = A \cup B$. The network topology is that of an all-to-all with a uniform link capacity of c. Suppose we have the traffic demand described by:

$$t_{ij} = \begin{cases} \alpha, \text{ if } a_i \in A \text{ and } b_j \in B \\ 0, \text{ otherwise} \end{cases} \tag{18}$$

We assume 2-hop routing for network load-balancing, and every node can be used as an intermediate hop. Using the available uniform topology shown in Figure 10, the max link utilization with perfect load balancing is $\frac{2n-1\alpha}{c}$. Using traffic-aware topology design, however, we can disconnect the $n-1$ links that each sender in A to other sender nodes in the same set, and connect them to $n-1$ receivers in set B instead, resulting in the topology shown in Figure 11. In this case, the max link utilization under perfect load-balancing is $\frac{n\alpha}{c}$. This shows that with traffic-aware topology designs we can reduce max link utilization to a factor of $\frac{n}{2n-1} \approx \frac{1}{2}$ than that obtained form traffic-agnostic topologies.

- **Singular hotspot** - When one pod sends or receives significantly more traffic than all of its peers, the most congested links will be located at either the egress or ingress links of said pod. Topology engineering cannot alleviate bottlenecks due to physical bandwidth limitations.
- **Uniform traffic** - Under a uniform traffic load, there is no need for special topology design. The ideal topology to handle a uniform traffic pattern is a uniform one.

A.3 Correctness of Minimum-cost Circulation

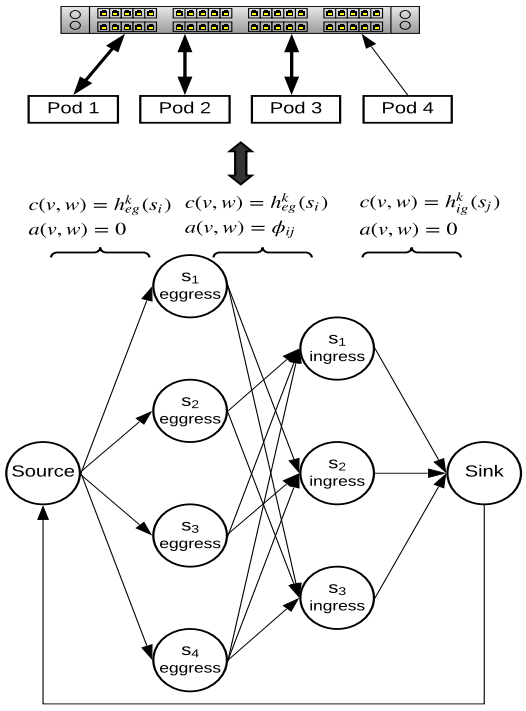

Figure 12: Mapping a patch panel's physical striping onto a network flow graph

Here we prove that the integer optimization of equations (33, 37) is equivalent to a min-cost circulation problem. For any min-cost flow problem, the objective is to find a flow assignment fu, v that minimizes:

$$\begin{aligned} &\min_{v,w \in E} fv, w \cdot av, w \\ &\text{s.t } 1\ lv, w \leq fv, w \leq \min\left(uv, w, cv, w\right) \\ &2 \sum_{u \in V} fu, v = \sum_{w \in V} fv, w \ \forall\ v \in V \end{aligned} \tag{19}$$

Where au, v and cv, w denote the cost per unit flow and capacity for the u, v edge, respectively. The first constraint sets the closed range between which the flow must lie without violating the capacity constraint of each edge, while the second constraint is simply a enforces flow-conservation on any node that is neither the source nor sink node. To prove, we begin by illustrating how a flow digraph, $G_k = V, E$, for patch panel k is generated based on its physical striping. Using this mapping, fs_i, s_j is equivalent to x^k_{ij}.

First, we prove that a solution which satisfies constraints in (19) also satisfies (33) and (37). We know that the flow from each egress vertex satisfies constraint 1 in (19), therefore the following inequality is true: $\sum_{i \in S} fs_i, s_j \leq h^k_{ig} s_j\ \forall\ j \in S$. Similarly, since we've capped the flow capacity of each arc from the dummy source vertex to the ingress vertices by $h^k_{eg} s_i$, we can be certain that a flow that satisfies constraint 1 in (19) is guaranteed to to satisfy the inequality: $\sum_{j \in S} fs_i, s_j \leq h^k_{eg} s_i\ \forall\ i \in S$. Meanwhile, constraint 2 of (19) is simply a flow conservation constraint for the Min-cost Circulation problem and has no relation to problems (33) and (37).

Next, we prove that an optimal solution to (19) is also optimal for (33) and (37). Since $av, w = 0$ for all arcs that either originates from or leads toward the dummy source and sink nodes, respectively, the objective function for the min-cost flow problem reduces to $\sum_{i,j \in S} fs_i, s_j \phi_{ij}$. Therefore, a solution that minimizes this reduced form also minimizes (33) and (37).

A.4 Alternative Fractional Graph Computation

In this section, we detail a routing-topology co-optimization methodology used for computing an optimal fractional topology given a set of traffic matrices. As mentioned briefly in Section 3, we've decided to tackle the optimal logical topology computation by splitting the large problem into two smaller, more easily-solvable problems. In computing the optimal topology, we've decided to ignore the patch panel layer and assume full-connectivity, which is motivated by two reasons: Firstly, considering the patch-panel layer increases the number of variables and constraints required to formulate the LP, which incurs scalability issues when optimizing for larger topologies. Secondly, including the patch-panel layer (which will be considered in Step 3 regardless) brings about negligible improvement (if any) to the optimality of our final integer logical topology.

Step 2 of Figure 2 shows that instead of considering all traffic modes in the same fractional derivation computing, we compute a fractional topology optimal for each traffic mode, and then finally combine all these topologies using LP. This is because formulating an LP that considers all the traffic modes together greatly increases the complexity of the problem. Based on our experience, solutions obtained from LP's tend to be unstable, therefore we employ a quadratic program (QP) to stabilize the fractional topology for each traffic mode. Finally, all the κ fractional topologies are linearly combined using LP.

Given that we assume full-connectivity here, the only physical constraints to be imposed for computing fractional topologies are:

$$\sum_{j \in S} d_{ij} \leq r_{eg}^{i} \; \forall \; i \in S \tag{20}$$

$$\sum_{i \in S} d_{ij} \leq r_{ig}^{j} \; \forall \; j \in S \tag{21}$$

Constraints (20) and (21) upper-bounds the number of links formed by s_i with its peer pods to its total number of egress links and ingress links, respectively. Now, given a traffic mode T^{κ}, we want to obtain an fractional topology that can support the largest scale-up of T^{κ} under multihop routing, which can be formulated as the following LP:

$$\begin{aligned} \max_{\mu_\kappa, \boldsymbol{\omega_p}, \mathbf{d}} \quad & \mu_\kappa \\ \text{s. t: } 1 \quad & \sum_{p \in P_{ij}} \omega_p = \mu_\kappa \; t_{ij}^{k}, \quad \forall \; i, j \in S \\ 2 \quad & \sum_{p \in Pe_{ij}} \omega_p \leq d_{ij} b_{ij}, \; \forall \; i, j \in S \\ 3 \quad & 20 \; , \; 21 \end{aligned} \tag{22}$$

Where $b_{ij} = \min b_i, b_j$ and μ_κ is the traffic matrix scale-up factor for traffic mode κ. Constraint 1 enforces all scaled-up i to j traffic demands to be satisfied via all routable paths, while constraint 4 ensures that the traffic flowing through each link cannot exceed its bandwidth. Unfortunately, the linear nature of (38)'s objective function causes the solution to suffer from stability issues. (38) implies that there may be many different fractional topologies that can support the same scale-up factor, u^{κ}, with some viable fractional topologies perhaps relying more heavily on multi-hop routing rather than direct-hop routing. We know that relying on multihop routing not just increases the delay of each packet, but could also inadvertently congest other network links as overall traffic increases. To solve this issue, we extract the each traffic mode's maximal scale-up factor, μ_{κ}^{*}, by solving (38). Using the computed scale-up factor, we stabilize the fractional topology solution using a the strongly-convex properties of a quadratic program (QP) with linear constraints that acts as a solution "filter". So, for each traffic mode, the objective of this quadratic program is set to minimize multi-hop routing as much as possible while still supporting the same scale-up factor:

$$\begin{aligned} \min_{\boldsymbol{\omega_p}, \mathbf{d}} \quad & \sum_{i,j \in S} \; \sum_{p \in P_{ij} l_p > 1} \left(\omega_p\right)^2 \\ \text{s. t: } 1 \quad & \sum_{p \in P_{ij}} \omega_p = \mu_\kappa^{*} t_{ij}, \quad \forall \; i, j \in S \\ 2 \quad & \sum_{p \in Pe_{ij}} \omega_p \geq \mu_\kappa^{*} \; d_{ij} b_{ij}, \quad \forall \; i, j \in S \\ 3 \quad & 20 \; , \; 21 \end{aligned} \tag{23}$$

Where $P_{ij} l_p > 1 = \{p \mid p \in P_{ij} \cap l_p > 1\}$ denotes the set of routing paths from s_i to s_j with hop-counts greater than 1. To solve (23), we use the OSQP library [31]. By attempting to minimize multihop routing, (23) indirectly forces the fractional topology to "match" the traffic matrix. This has the added benefit of making the κ-th traffic mode's fractional topology to remain fairly optimal for traffic matrices that fall within mode κ's cluster.

With each traffic mode, T^{κ}'s stable fractional topologies computed, we combine them into one fractional topology using linear combination of the form:

$$\begin{aligned} \max_{\alpha, \mathbf{d}^*} \quad & \alpha \\ \text{s. t: } 1 \quad & d_{ij}^{*} \geq \; \alpha \; d_{ij}^{\zeta}, \; \forall \; i, j \in S, \; \zeta \in \{1, 2, ..., \kappa\} \\ 2 \quad & 20 \; , \; 21 \end{aligned} \tag{24}$$

Where α is the scaling factor of each individual fractional graph. Constraint 1 in (24) is imposed so that each i, j fractional link in the combined fractional graph, d_{ij}^{*}, can dominate all other α-scaled $d_{ij}^{\zeta} \; \forall \; \zeta \in \{1, 2, ..., \kappa\}$.

Here we define fractional topology a little more rigorously. A fractional topology describes the number of link counts between two pods in the topology. As its namesake suggests, the number of links between two pods . More concretely, a fractional topology can be described using a matrix, where the entry in i, j denotes the fractional number of links connecting the egress ports of s_i to the ingress ports of s_j. In addition, a fractional topology has to satisfy the degree constraints:

Furthermore, for any pair of T and D, we consider T, D feasible there a

$$\begin{aligned} \min_{\boldsymbol{\omega_p}, \mathbf{d}} \quad & \sum_{i,j \in S} \; \sum_{p \in P_{ij} l_p > 1} \left(\omega_p\right)^2 \\ \text{s. t: } 1 \quad & \sum_{p \in P_{ij}} \omega_p = \mu_\kappa^{*} t_{ij}, \quad \forall \; i, j \in S \\ 2 \quad & \sum_{p \in Pe_{ij}} \omega_p \geq \mu_\kappa^{*} \; d_{ij} b_{ij}, \quad \forall \; i, j \in S \end{aligned} \tag{25}$$

A.5 Effects of Applying QP "filter"

In (23), minimizing the WCMP weights placed on multihop paths indirectly increases the weights placed on direct-hop paths under a given traffic matrix, subject to the constraint that the resulting traffic matrix scale-up factor, μ^{*}, is preserved. Here, we show the utility for (23) using the following Figure 13, and discuss potential implications if a QP filter isn't employed.

In Figure 13, the x-axis denotes the euclidean distance between two traffic matrices, while the y-axis denotes the euclidean distance between the computed fractional topologies for the corresponding traffic matrices. The

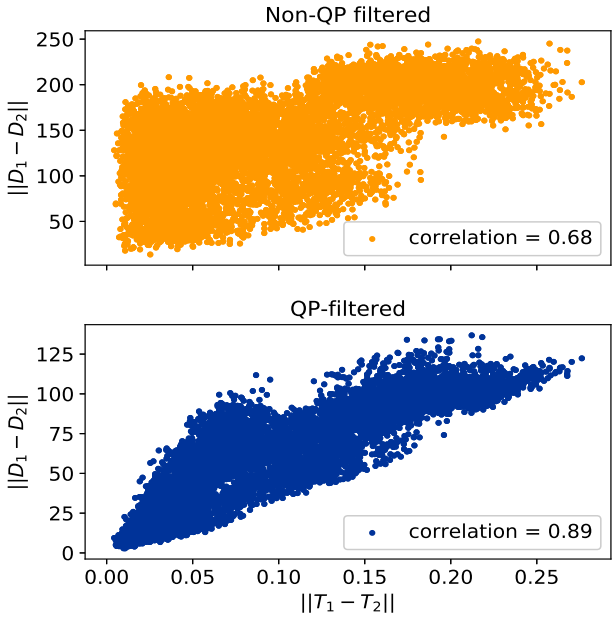

Figure 13: Mapping a patch panel's physical striping onto a network flow graph

traffic matrices are obtained from historical production DCN traffic.

Observing the two scatter plots in Figure 13, it is clear that the QP-filter is doing a good job at producing fractional topology that matches the traffic matrix, as demonstrated by the strongly positive correlation of 0.89. Meanwhile, the non-QP-filtered fractional topology demonstrate a significantly poorer correlation between traffic matrix distance and resulting optimal fractional topology closeness. In fact, for the Non-QP filtered case, the spread of fractional topology distance when traffic matrix distance is close to 0 is rather significant. This spread is due mainly to 2 reasons: 1) there can be many different fractional topologies that could support the same traffic matrix scale up factor, μ^*, with some perhaps utilizing more indirect-hops, and 2) the fractional topology solution given the same traffic matrix obtained from a LP solver is non-deterministic, and depends on the implementation of said solver. As such, employing a QP-filter in as much as an attempt to pick a fractional topology that maximizes direct-hop routing from a set of fractional topologies that could all support the same μ, as it is an effort of reducing the solution oscillation obtained directly from a LP solver.

Most importantly, employing a QP filter helps ensure the fractional topology we compute for a traffic cluster is likely going to be quite optimal for traffic matrices that are in close proximity to the clusters. This has an effect of helping the fractional topology designed from a traffic mode to be able to handle many traffic matrices that exhibit very slight variation from said mode.

A.6 Synthetic Traffic Generation

A.6.1 Bipartite

. The basic idea of the bipartite traffic is to emulate a potential traffic pattern that could arise as data center desegregates compute resources from their storage resources. Since the bipartite traffic pattern is given in (18), we will not repeat the equation here.

A.6.2 κ-Nearest Neighbor

. With this traffic pattern, we attempt to emulate the traffic pattern where pods only send traffic to it's closest peers using circular distance. This traffic pattern shows a "sliding window"-like behavior in the traffic matrix with the center at the source pod due to traffic being sent to a set of closest peers. The traffic equation is given as:

$$t_{ij} = \begin{cases} \alpha, \text{ if } d_{circ}i, j \leq \kappa \\ 0, \text{ otherwise} \end{cases} \tag{26}$$

Where d_{circ} denotes the circular distance between pod indices i and j.

A.6.3 Random

. First, the total flow sent out by each pod is determined through a uniform random variable which we shall denote as $\beta \sim \text{U}0, \alpha$, where α is the maximum posible traffic sent out by one pod. Next, each source pod sends traffic to its closest neighbors in circular distance in a normally-distributed fashion. We define X as a random variable denoting the circular distance between two pods, where $X \sim \text{Norm}0, \frac{n}{4}$. The equation that governs the magnitude of each i, j flow is thus given as:

$$t_{ij} = \begin{cases} 2\beta \cdot \Pr_X\left(x \leq d_{circ}i, j\right), & \text{if } i \neq j \\ 0, \text{ otherwise} \end{cases} \tag{27}$$

A.6.4 Radix Ratio

. The key purpose of introducing the radix ratio traffic pattern is such that each i, j flow will be proportional to the minimum of the radices of the source and destination pods. In a homogeneous topology, the traffic matrix approaches a uniformity.

$$t_{ij} = \begin{cases} r^i_{eg} \cdot r^j_{ig} \cdot \min\left(\frac{1}{\sum_{v \in S\{s_i\}} r^v_{ig}}, \frac{1}{\sum_{w \in S\{s_j\}} r^w_{eg}}\right), & \text{if } i \neq j \\ 0, \text{ otherwise} \end{cases} \tag{28}$$

A.7 Detailed Walkthrough for Lagrangian Dual Method

Here, we introduce the first approach based on Lagrangian Duality [23], with a main (i.e primal) problem of the form:

$$\begin{aligned} \mathbf{P}: \max_{\mathbf{x}} U\mathbf{x} = \max_{\mathbf{x}} \sum_{k\in O} \sum_{i,j\in S} U_{ij}^k x_{ij}^k \\ \text{s.t}: \quad 7,\ 8,\ 9 \end{aligned} \tag{29}$$

Solving for (29) optimally with integer solutions is a challenging combinatorics problem due to the inter-coupling of optimization variables x_{ij}^k through the constraints (7)-(9). The crux here is to transform the primal problem (29) into its Lagrangian dual problem, which first relaxes the matching constraints. It becomes apparent later that writing a objective function that is separable w.r.t x_{ij}^k allows us to decouple the $\mathbf{x}$ variables, and subsequently decompose the dual problem into a series of Min-cost Flow (MCF) subproblems.

At first, we chose an objective function of 0, but this resulted in extremely poor convergence of solution. We also went for a sum of squares of x_{ij}^k, which gave us superior convergence property. However, this objective function causes the logical topology to utilize very few links and negatively affecting network performance. Finally, we went with:

$$U_{ij}^k x_{ij}^k = -\left(x_{ij}^k - h_{ij}^k\right)^2 \tag{30}$$

Where $h_{ij}^k = \min\left(h_{eg}^k s_i, h_{in}^k s_j\right)$, 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. Ideally, we would be able to rewrite (29) in a way that allows us to separate the optimization variables, and solving a collection of smaller problems instead. To this end, we introduce the dual problem.

A.7.1 Dual Problem

. Transforming (29) into its Lagrangian dual problem gets us:

$$\begin{aligned} \mathbf{D}: \min_{\mathbf{p}^+,\mathbf{p}^-} \max_{x_{ij}^k\in\Lambda} L\mathbf{x},\mathbf{p}^+,\mathbf{p}^- \\ \text{s.t } 1 \quad \mathbf{p}^+ \succcurlyeq 0,\ \mathbf{p}^- \succcurlyeq 0 \\ 2 \quad 7,\ 8 \end{aligned} \tag{31}$$

Where $L\mathbf{x},\mathbf{p}^+,\mathbf{p}^- = \sum_{i,j\in S}\Big[\sum_{k\in O} U_{ij}^k x_{ij}^k - p_{ij}^+ \sum_{k\in O} x_{ij}^k - c_{ij}^+ + p_{ij}^- \sum_{k\in O} x_{ij}^k - c_{ij}^-\Big]$ is the Lagrangian of (29), and $\mathbf{p}^+,\mathbf{p}^-$ are the dual variables. Note that the objective function for $\mathbf{D}$ is itself a maximization problem denoted as $B\mathbf{x} = \max_{x_{ij}^k\in\Lambda} L\mathbf{x},\mathbf{p}^+,\mathbf{p}^-$. Plugging the $L\mathbf{x},\mathbf{p}^+,\mathbf{p}^-$ into $B\mathbf{x}$ gives us:

$$B\mathbf{x} = -\sum_{k\in O} Q_k x_{ij}^k + \sum_{i,j\in S} p_{ij}^+ c_{ij}^+ - \sum_{i,j\in S} p_{ij}^- c_{ij}^- \tag{32}$$

Where $Q_k\mathbf{x}^k = \min_{\mathbf{x}} \sum_{i,j\in S} p_{ij}^+ - p_{ij}^- x_{ij}^k + x_{ij}^k - {h_{ij}^k}^2$ is a quadratic program for O_k. Note that we've grouped the terms in such a way that the outermost summation is over all of the patch-panels, allowing us treat each of the patch-panel's configuration as an individually. However, solving for $Q_k\mathbf{x}$ still presents a formidable computation challenge due to it being a QP with integral constraints. To this end, we employ a first order approximation of $Q_k\mathbf{x}^k$, arriving at:

$$Q_k\mathbf{x}^k \approx \min_{\mathbf{x}} \sum_{i,j\in S} \left[p_{ij}^+ - p_{ij}^- - 2h_{ij}^k + 2{x_{ij}^k}^*\right] x_{ij}^k \tag{33}$$

Note that (33) is simply a Min-cost Flow (MCF) problem which can be solved efficiently in polynomial time with integer solutions using the Goldberg-Tarjan algorithm [6]. In the interest of brevity, the correctness of casting (33) as a Min-cost Flow problem, alongside a visual illustration of how each patch-panel can be mapped onto a bipartite netflow graph are shown in Appendix A.3. (33) has a slight problem: the dual variables are only updated once every iteration of the outer loop is taken, as shown in Algorithm 1. As a result, patch panels with the same physical striping will be configured the exact same way, causing the solution to oscillate violently and renders convergence difficult. To solve this, we replace the $\mathbf{p}^\pm$ scores in (33) with:

$$\phi^\pm\tau, k = \mathbf{p}^\pm\tau - \frac{1}{\tau+1}\Big(\pm\mathbf{c}^\pm \mp \sum_{k'\neq k} \mathbf{x}^{k'*}\Big) \tag{34}$$

Since (34) is updated after a MCF problem is solved, it can more effectively fine-tune the arc-costs in the next MCF problem such that the next patch panel may prioritize forming or breaking logical links between i,j pairs that more severely violate their matching constraints.

A.8 Detailed Walkthrough for Barrier Penalty Method

In addition to the Lagrangian dual method, we also propose an alternative barrier penalty method which relaxes the matching constraints directly through an objective function that penalizes matching constraint violation. The optimization problem is written as:

$$\begin{aligned} \min_{\mathbf{x}} U\mathbf{x} = \sum_{i,j\in S}\Big[\sum_{k\in O} x_{ij}^k - c_{ij}^- \sum_{k\in O} x_{ij}^k - c_{ij}^+\Big] \\ \text{s.t: } 7, 8 \end{aligned} \tag{35}$$

The partial derivative of $U\mathbf{x}$ w.r.t x_{ij}^k evaluated at ${x_{ij}^k}^*$ is:

$$\frac{\partial U\mathbf{x}}{\partial x_{ij}^k}\Big|_{x_{ij}^k = {x_{ij}^k}^*} = 2\sum_{k\in O} {x_{ij}^k}^* - c_{ij}^+ + c_{ij}^- \tag{36}$$

Now, expanding and rearranging the objective function of (35), we get the equivalent minimization problem that

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

Result: $\mathbf{x}^{**} \in \mathbb{Z}_{\geq 0}$ - patch-panel switch states

Initialize: $\mathbf{x}^{*} := 0, \mathbf{x}^{**} := 0$;
Build network flow graphs $G_1, ..., G_k$;
for $t \in \{1, 2, ..., \tau_{max}\}$ **do**
  $\delta := \frac{1}{\tau}$;
  for $k \in O$ **do**
    Obtain flow network arc costs by solving (34);
    Solve $\mathbf{x}^k := MCFG_k,\ \boldsymbol{\phi}^{\pm} t, k\ , \mathbf{0},\ \mathbf{h}^k$;
    if $\Psi\mathbf{x}^{*} > \Psi\mathbf{x}^{**}$ **then**
      $\mathbf{x}^{**} := \mathbf{x}^{*}$;
    else
    end
  end
  Solve (13) to update dual variables $\mathbf{p}^{\pm}$;
end

Algorithm 1: Lagrangian duality method

is separable w.r.t k:

$$\min_{\mathbf{x}} U\mathbf{x} \approx \sum_{k \in O} \min \sum_{i,j \in S} \Big[\sum_{k \in O} 2x_{ij}^{k\,*} - c_{ij}^{+} + c_{ij}^{-} \Big] x_{ij}^{k} \tag{37}$$

Note that (37) is a result a linear approximation on the otherwise quadratic (35). Again, we employ a first order linear approximation of the objective function to reduce the otherwise complex polynomial optimization objective function into a min-cost circulation problem. Solving for the linear approximation as a min-cost circulation problem guarantees integer solutions in polynomial run-time, while satisfying the physical constraints (7) and (8). The proof of correctness is omitted here for the sake of brevity, but is included in Appendix A.3 for interested readers. A gradient descent method is employed to iteratively solve each patch-panel's bipartite min-cost circulation problem. Unlike the Lagrangian duality method, every patch-panel optimizes for the same objective function given in (35), making the barrier method operate like a global game in game theory. The pseudocode detailing the steps are is given in Algorithm 2.

A.9 Computing a Good Fractional Topology

In order to compute $\Omega\mu\ T, D$ for any feasible T, D such that μ is maximized, we formulate the follow LP:

$$\begin{aligned} \max_{\mu_\kappa, \boldsymbol{\omega_p}} \quad & \mu_\kappa \\ \text{s. t: } 1 \quad & \sum_{p \in P_{ij}} \omega_p = \mu_\kappa\ t_{ij}^{k}, \quad \forall\ i, j \in S \\ 2 \quad & \sum_{p \in Pe_{ij}} \omega_p \leq d_{ij} b_{ij},\ \forall\ i, j \in S \end{aligned} \tag{38}$$

Data:
- $D = d_{ij} \in \mathbb{R}^{n \times n}$ - fractional demand topology
- τ_{max} - number of iterations

Result: $\mathbf{x}^{**} \in \mathbb{Z}_{\geq 0}$ - patch-panel switch states

Initialize: $\mathbf{x}^{*} := 0, \mathbf{x}^{**} := 0$;
Build network flow graphs $G_1, ..., G_k$;
for $\tau \in \{1, 2, ..., \tau_{max}\}$ **do**
  for $k \in O$ **do**
    Solve (37) for min-cost flow arc costs;
    $u_{ij} := x_{ij}^{k\,*} + 1,\ l_{ij} := x_{ij}^{k\,*}$;
    if *(36)* $\geq$ *0* **then**
      $u_{ij} := x_{ij}^{k\,*},\ l_{ij} := x_{ij}^{k\,*} - 1$;
    else
    end
    Solve $\mathbf{x}^k := MCFG_k,\ \boldsymbol{\phi}^{\pm}\tau, k,\ \mathbf{l},\ \mathbf{u}$;
    if $\Psi\mathbf{x}^{*} > \Psi\mathbf{x}^{**}$ **then**
      $\mathbf{x}^{**} := \mathbf{x}^{*}$;
    else
    end
  end
end

Algorithm 2: Barrier penalty method

Where $b_{ij} = \min b_i, b_j$ and μ_κ is the traffic matrix scale-up factor for traffic mode κ. Constraint 1 enforces all scaled-up i to j traffic demands to be satisfied via all routable paths, while constraint 4 ensures that the traffic flowing through each link cannot exceed its bandwidth. Unfortunately, the linear nature of (38)'s objective function causes the solution to suffer from stability issues. (38) implies that there may be many different fractional topologies that can support the same scale-up factor, u^{κ}, with some viable fractional topologies perhaps relying more heavily on multi-hop routing rather than direct-hop routing. We know that relying on multi-hop routing not just increases the delay of each packet, but could also inadvertently congest other network links as overall traffic increases. This stability issue is why the we further enforce the definition of a best-fitting

fractional topology to a given T using a QP with linear constraints shown in (23).