Traffic-aware topology design using Lagrangian Duality Theory

Min Yee Teh, Shizhen Zhao

ABSTRACT

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

1 INTRODUCTION

The wealth of applications that run on data centers today has brought about irregular traffic patterns, which have confounded data center networking design for a long time. Recent literature have suggested that this wealth of application variety leads to inter-rack traffic patterns that are highly non-uniform [2, 3]. This implies a high degree of traffic locality, and that all-to-all traffic do not arise as frequently as most would assume. This traffic asymmetry is projected to increase in the future as more applications are run on the cloud. What is most surprising is that while there is a general consensus within the data center community on the unpredictability and transient burstiness of inter-rack traffic, studies [9] have found that the traffic locality can persist over a period of months or more.

One way to "even" non-uniform traffic patterns is to load-balance using workload migration [10]. However, as data center tends towards disaggregated hardware resources [14], relying solely on workload placement and migration will become increasingly difficult, since resource disaggregation of compute elements and storage elements will force same workloads to be mapped to physically-separated clusters. Careful job placement even with global knowledge cannot fundamentally even out the strong pairwise traffic pattern generated due to desegregation. Another solution to even-ing out non-uniform traffic is through the use of congestion-aware routing schemes. However, all these routing schemes require traffic flow to be deflected non-minimally to less congested areas first. This adds to the total network traffic and can often create previously unforeseen hotspots in other network areas. We therefore believe that load-balancing through simply intelligent routing or workload placement cannot sufficiently decongest the network under non-uniform traffic permutations.

To address the non-uniform traffic demands, dynamically-reconfigurable topologies like the Firefly [8] and Project-Tor [7] have been proposed using revolutionary hardware, both of which involve wireless solutions using free space optics. These topologies rely heavily on novel hardware architectures that are immature and difficult to control, as free space optics require highly precise laser-to-detector alignment that is prone to failure. The switching time for these fabrics are shown to be on the order of 10's of microseconds, allowing them to in theory react to transient traffic bursts. However, our findings show that the topology need not be reconfigured frequently, since we've observed that the majority of inter-rack traffic flows are long-lived (on the order of weeks), an observation also shared by Delimitrou et al [4]. This observation has led us to believe that a topology reconfigurable on the order of weeks can still significantly improve network performance by serving the long-lived traffic flows.

We propose a traffic-aware topology built using robust and commercially-available technologies that are already deployed in today's data centers [20]. More specifically, we've decided to restrict our fabric design to topologies within the direct connect topology family due to their cost effectiveness and high-bisection bandwidth [15]. To accommodate the incremental growth and need to alter topology based on long-term traffic patterns, we interconnect the server blocks using a layer of patch-panel. Although patch-panels cannot be rapidly reconfigured to serve transient traffic bursts, our production traffic have shown that on a large timescale, traffic patterns do not change much. Our results further show that simply reconfiguring the patch-panel on a monthly-basis is sufficient in increasing network throughput.

While the idea of a traffic-aware reconfigurable topology is simple, finding the optimal solution presents a formidable algorithmic challenge. To facilitate understanding, it is convenient to think of this as a supply-demand matching problem, with inter-rack links being the supply and traffic flows being the demand in our case. Within theoretical computer science, this problem can be mapped onto a multicommodity-flow problem, which is NP-hard [6, 9]. This problem presents a significant combinatorics challenge due to the coupling of network links introduced through the physical topology constraints, i.e links cannot be formed if they are not connected to the same patch-panel. To our knowledge, no literature on dynamic topology to date have treated this problem rigorously while accounting for physical constraints, with most published work resorting to immature hardware technologies [7, 8] to circumvent this problem. Yet finding a solution to this problem is crucial for the deployment of traffic-aware topologies within data centers currently in-production.

Hence, our main contribution in this work is the design of a robust and scalable polynomial time topology-design methodology based on predicted traffic pattern, while taking into account constraints imposed by the physical connectivity. We include this constraint in our problem formulation without reverting to an ILP formulation, and propose a 2-step algorithm to decompose this problem into more manageable subproblems. The first step involves finding an optimal fractional topology which minimizes maximum link utilization under a given traffic load. Using this step, we derive the optimal fractional connectivity between different server blocks, which can be solved quickly using a linear program. Subsequently, we attempt to "match" the topology's connectivity with the optimal fractional connectivity using convex optimization [13]. Using this technique, we show that our integer connectivity matches the optimal fractional topology very closely, without resorting to solving complex integer-linear programs.

Finally, we simulate the effectiveness of our methodology using different metrics across several heterogeneous data center fabrics. Through our simulation results, we show that reconfiguring a topology once a month is enough to lower maximum link utilization by as much as 20%. We further show that our methodology minimizes total network traffic by reducing the average hop count of flows, lowering average link utilization by as much as 50%. Furthermore, our analysis shows that topology engineering also significantly lowers the percentile link utilizations, keeping about 50% of the network links idle under most load conditions. Most importantly, our methodology allows us to achieve all these metrics using only about half the network links, giving us a $2\times$ economic saving on network resources without sacrificing network throughput.

2 RELATED WORK

In the field of data center topologies, several novel, non-traffic aware topologies have been proposed using a graph-theoretic approach, such as Jellyfish [16] and Xpander [18] that are based on random-regular graphs and expander graphs, respectively. These topologies cannot be designed to best cater to the non-uniform traffic that often arise in modern data centers. Although both Jellyfish and Xpander offer excellent incremental expandability, they do not guarantee network diameter as network expands, rendering them suboptimal when running latency-sensitive applications.

Some literature have suggested hybrid topologies [5, 12] to alleviate network hotspots, with the central idea being the electrical network forming the reliable backbone and route mice flows, while using optical reconfigurable circuits to opportunistically route elephant flows. However, these topologies can be expensive to build due to their hybrid architecture presenting significant hardware engineering challenges, which further delays their widespread use in today's data centers. Others dynamic topologies like ProjectTor [7] proposes a "disco ball" device that reconfigures ToR switches by changing mirrors to reflect different laser emissions unto different photodiode receivers. This presents a single point of failure, since a slight misalignment of the disco ball can potentially disconnect the entire data center fabric. FireFly [8] proposes attaching laser sources and photodiode receivers to each ToR and forming interrack links using a ceiling mirror for deflection, with each interrack link being a lightpath between the source ToR and destination ToR. Both papers proposed mirror-reconfiguring algorithm that assume any arbitrary link on a source ToR switch can reach at least one receiver on the destination ToR. This assumption is highly unrealistic in wired networks since links connected to different switches being unconnectable.

Dynamic topologies have also seen its debut in the HPC community through the Flexfly topology [19], which is based on a Dragonfly topology reconfigures its intergroup links using silicon-photonic optical switches. While Flexfly offers immensely short switching time, the algorithm proposed by the author(s) requires examining number of switch states, which grows exponentially with the number of ports in the optical switch. As such, employing the proposed algorithm in any deployed data centers of reasonable size infeasible. In addition, the paper also acknowledges that increasing portcount in silicon photonic switches is technologically-challenging, and therefore severely limiting its ability to be integrated into current high-radix data center fabrics.

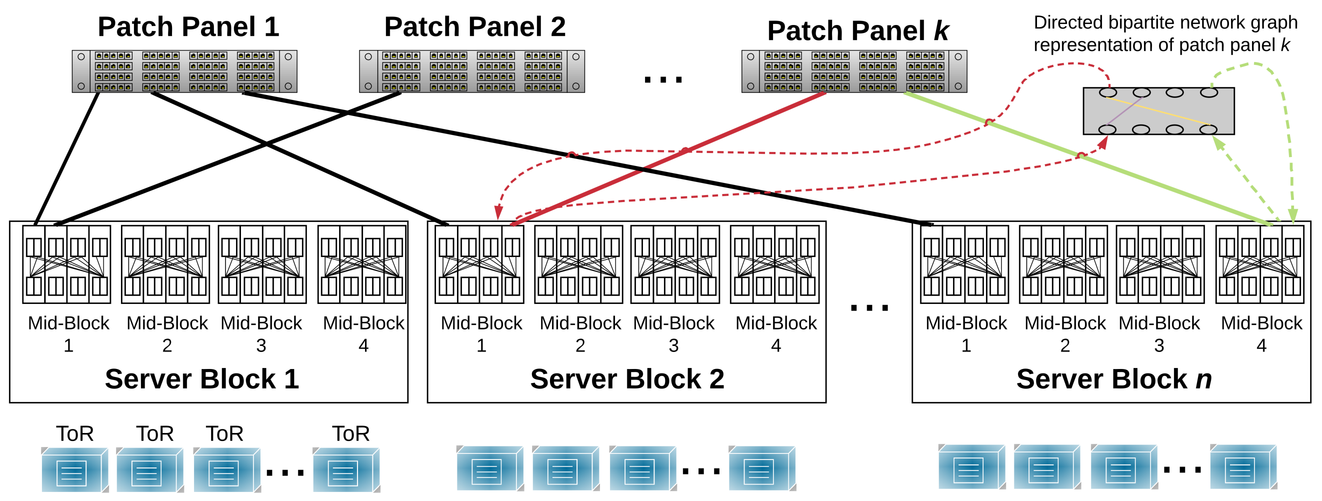


Figure 1: Direct-connect topology interconnected via a patch-panel layer, alongside an illustration on the bipartite network graph representation of patch-panel *k*. All links in solid lines are bidirectional (undirected), while dotted links of the same color are the unidirectional (directed) representation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

S_i, O_k	Server block i, patch-panel k
η	Max link utilization
c_{ij}	Maximum possible bandwidth capacity between S_i and S_j
x_{ij}^k	Number of server block i's northports connected to server block j's southports through patch-panel k
$D \in \mathbb{R}^{n \times n}$	Demand matrix, used interchangeably with the optimal fractional topology
$l_n^k(S_i), l_s^k(S_j)$	Number of physical links connecting S_i to northport of O_k, and S_j to southport of O_k, respectively
$P_k(S_i, S_j)$	Number of port-disjoint paths leading from S_i to S_j via O_k. Therefore, $L_k(S_i, S_j) = \min(l_n^k(S_i), l_s^k(S_j))$
$U_{ij}^k(x_{ij}^k)$	Primal problem's objective function element, generally referring to $-(x_{ij}^k - \min(l_n^k(S_i), l_s^k(S_j)))^2$
ϕ^+	$\phi^+ = \max(0, \phi)$

Table 1: Commonly-used notations

3 DEFINITIONS AND NOTATIONS

In order to rigorously formulate the network reconfiguration problem, we begin by introducing a series of useful notations and definitions.

- **Physical Topology** Describes the physical connectivity between the server blocks and the patch panels within the network, and can be represented as a directed graph.
- **Logical Topology** Describes the connectivity between server blocks, abstracting away the layer of patch-panels.
- **Fractional Topology** The logical topology assuming that link counts between server blocks can occupy non-integer values.
- **Southports, Northports** Southports refer to patch-panel outgoing ports, with an outgoing (south-facing) link towards server blocks. Northports on the other hand are connected to incoming (north-facing) links incident on server blocks.

We assume each patch panel can be represented as a bipartite network flow graph, as shown in Figure **??**. Each bidirectional link are modeled as two directed links, . It turns out, modeling the topology as a directed graph possesses a a key mathematical importance, since our patch-panel reconfiguration methodology does not need to impose the constraint of $x_{ij}^k = x_{ji}^k$ for any $i \neq j$.

4 TRAFFIC-AWARE TOPOLOGY-DESIGN

4.1 Demonstrating Traffic-aware Topology Design Benefits

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

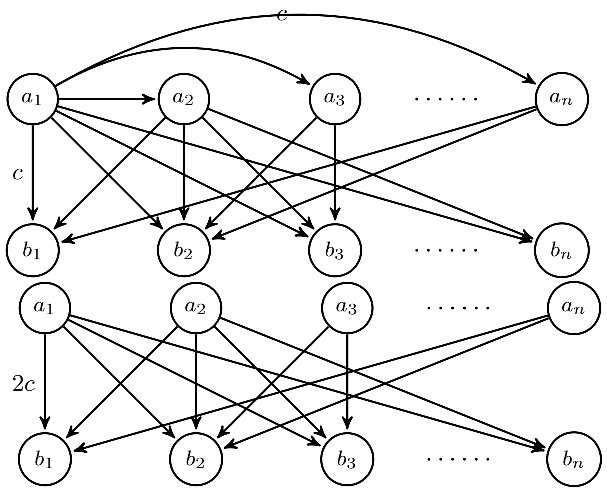
We posit that there are certain traffic patterns under which traffic-aware topology design will prove most useful. One such example is a M-to-N bipartite traffic pattern, whereby each of the M senders send traffic to all N receivers. These traffic are expected to arise when datacenter architectures become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

Consider a datacenter network with 2 sets of server blocks $A = \{a_1, a_2, ..., a_n\}$ and $B = \{b_1, b_2, ..., b_n\}$, whereby the set of all server blocks 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:

$$T(a_i, b_j) = \begin{cases} \alpha, & \text{if } a_i \in A \text{ and } b_j \in B \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

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 4.1, 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 ??. 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 there is a singular hotspot, the most congested links will be either the egress or ingress links of the hottest block. Under these traffic conditions, max link utilization (MLU) cannot be driven down through intelligent topology design, as the congestion is attributed to the server block's physical ability to pushing traffic out or in.
- **Uniform traffic** - Under a uniform traffic load, there is no need for special topology design. The ideal topology to handle a uniform traffic pattern will be to uniformly connect all the server blocks within the topology.



Assuming that every link has capacity c, and

4.2 Deriving the Optimal Fractional Topology

We employ a 2-hop routing scheme to allow for better load-balancing across the network links. Our goal is to minimize the max link utilization under a given traffic load. This can be formulated as a linear program that can be solved easily and quickly.

$$\begin{aligned} \min \quad & \eta \\ \text{s. t: } & 1) \sum_{j \in S \setminus \{i\}} d_{ij} \le a_i \quad \forall\, i \in S \\ & 2) \sum_{i \in S \setminus \{j\}} d_{ij} \le a_j \quad \forall\, j \in S \\ & 3)\ \omega_{ij} + \sum_{k \in S \setminus \{i,j\}} \omega_{ij}^{k} = t_{ij}, \quad \forall\, i, j \in S \\ & 4) \sum_{k \in S \setminus \{i,j\}} \omega_{kj}^{i} + \sum_{k \in S \setminus \{i,j\}} \omega_{ik}^{j} + \omega_{ij} \le \eta\, d_{ij} b_{ij}, \forall i, j \in S \end{aligned} \tag{2}$$

Where $b_{ij} = \min(b_i, b_j)$. Constraints 1 and 2 denote the physical outgoing and incoming link counts. Constraint 3 enforces the rule that all traffic from source i to destination j must be flown through either 1 or 2 hops through an intermediate superblock. Constraint 4 then enforces all link utilizations to be at most equal to that of the max link utilization. The derived fractional topology then acts as a demand matrix for the next matching step.

4.3 Matching the Optimal Fractional Topology

Given a demand matrix, D, we attempt to match the logical topology with that of the demands, subject to physical constraints and connectivity constraints.

4.3.1 *Primal Problem.*

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 southports of O_k to S_j:

$$\sum_{i \in S} x_{ij}^k \leq l_s^k(j) \quad \forall\, k \in O, j \in S \tag{3}$$

By reciprocity, total number of integral flows through O_k originating from S_i cannot exceed the total number of physical links connecting the northports of O_k to S_i.

$$\sum_{j \in S} x_{ij}^k \leq l_n^k(i) \quad \forall\, k \in O, i \in S \tag{4}$$

Connectivity Constraints: In addition to the physical constraints, we also want to "match" the logical topology with the fractional topology. We do this by introducing the following constraint:

$$c_{ij}^L \leq \sum_{k \in O} x_{ij}^k \leq c_{ij}^U, \quad \forall\, i, j \in S \tag{5}$$

Where d_{ij} is the fractional link count between S_i and S_j, $c_{ij}^L = \lfloor d_{ij} \rfloor$ and $c_{ij}^U = \lfloor d_{ij} \rfloor$. Constraint (5) allows us to obtain a logical topology with link counts that are the closest integers to the optimal demand.

Primal Objective Function:

$$\begin{aligned} \mathbf{P} : \max_{d_{ij}^k} \sum_{k \in O} \sum_{i,j \in S} U_{ij}^k(d_{ij}^k) \\ \text{subject to:} \quad (3),\ (4),\ (5) \end{aligned} \tag{6}$$

Special care much be given in picking a objective function, as a good objective function affects not just the quality of the solution, but should also offer meaningful physical interpretation. With this in mind, we propose a concave objective function that would maximize the formation of logical links.

$$U_{ij}^k(x_{ij}^k) = -(x_{ij}^k \ - \ P_k(S_i, S_j))^2 \tag{7}$$

We take advantage of the fact that $L_k(S_i, S_j) \geq x_{ij}^k$ given in constraints (3) and (4) to ensure that the optimal solution maximizes the utilization of physical links. The concavity of (7) also helps remove the problem of obtaining suboptimal solutions due getting trapped in a local extremum.

Solving (6) up front while enforcing integral constraints is provably NP-hard [6], which can be attributed to the coupling of x_{ij}^k variables through constraints (3)-(5). Ideally, we would be able to rewrite (6) 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, D to (6).

4.3.2 *Dual Problem.*

Our proposed dual problem is a Lagrangian dual, as inspired by [13], which can be written as follows:

$$\begin{aligned} \mathbf{D} : \min_{p_{ij}^+, p_{ij}^-} \max_{x_{ij}^k \in \Lambda} L(x_{ij}^k, p_{ij}^+, p_{ij}^-) \\ \text{s.t} \quad p_{ij}^+ \geq 0,\ p_{ij}^- \geq 0 \end{aligned} \tag{8}$$

Where $L(x_{ij}^k, p_{ij}^+, p_{ij}^-)$ denotes the Lagrangian of (6), and p_{ij}^+, p_{ij}^- are the dual variables/Lagrange multiplier terms. The objective function for $\mathbf{D}$ is a maximization problem which we shall denote as $B(x_{ij}^k) = \max_{x_{ij}^k \in \Lambda} L(x_{ij}^k, p_{ij}^+, p_{ij}^-)$. The Lagrangian can be written as:

$$\begin{aligned} L(x_{ij}^k, p_{ij}^+, p_{ij}^-) = \sum_{k \in O} \sum_{i,j \in S} U_{ij}^k(x_{ij}^k) - \sum_{i,j \in S} p_{ij}^+(-c_{ij}^U + \sum_{k \in O} x_{ij}^k) + \\ \sum_{i,j \in S} p_{ij}^-(-c_{ij}^L + \sum_{k \in O} x_{ij}^k) \end{aligned} \tag{9}$$

Plugging (7) into (9) and rearranging the equation terms helps us arrive at the dual objective function:

$$\begin{aligned} B(x_{ij}^k) &= \max_{x_{ij}^k \in \Lambda} [\sum_{k \in O} \sum_{i,j \in S} -(x_{ij} - P_{ij}(S_i, S_j))^2 \ - \\ &\quad \sum_{i,j \in S} p_{ij}^+(\sum_{k \in O} x_{ij}^k - c_{ij}^U) \ + \ \sum_{i,j \in S} p_{ij}^-(\sum_{k \in O} x_{ij}^k - c_{ij}^L)] \\ &= -\sum_{k \in O} \min_{x_{ij}^k \in \Lambda} [\sum_{i,j \in S} (p_{ij}^+ - p_{ij}^-) x_{ij}^k + (x_{ij}^k)^2] \quad + \\ &\quad \sum_{i,j \in S} p_{ij}^+ c_{ij}^U \ - \ \sum_{i,j \in S} p_{ij}^- c_{ij}^L \\ &= \ -\sum_{k \in O} Q_k(x_{ij}^k) \ + \ \sum_{i,j \in S} p_{ij}^+ c_{ij}^U \ - \ \sum_{i,j \in S} p_{ij}^- c_{ij}^L \end{aligned} \tag{10}$$

Where $Q_k(x_{ij}^k) = \min_{x_{ij}^k \in \Lambda} \sum_{i,j \in S} (p_{ij}^+ - p_{ij}^-) x_{ij}^k + (x_{ij}^k)^2$ is a quadratic program for O_k. Note that we've grouped the terms in such a way that the outermost summation is a summation over all of the individual patch-panels in O. This allows us to solve k smaller QP's that are much easily-solvable. We've omitted several intermediate steps in (10) for the sake of brevity. Interested readers may find the detailed derivation steps attached in the appendix section.

4.3.3 *Relaxing $Q_k(x_{ij}^k)$'s Integral Constraint.*

Even after the obtaining a variable decoupling and problem-splitting, solving for $Q_k(x_{ij}^k)\ \forall\ k \in O$ still presents a formidable computation challenge, since we x_{ij}^k have integral requirements. In addition, the number of optimization variables grows as $O(n^2)$, where n is the total number of server blocks, hence rendering a QP highly unsolvable given that our datacenter farbics are rather large, and each patch-panel radix-count numbering in the hundreds. We circumvent this by relaxing the integral constraints, and solving for the following QP:

$$\hat{x}_{ij}^k = \text{argmin}_{x_{ij}^k} \sum_{i,j \in S} (p_{ij}^+ - p_{ij}^-) x_{ij}^k + (x_{ij}^k)^2 \tag{11}$$

Note that $\hat{x}_{ij}^k \in \mathbb{R}$, and are in general not integers. To attain the integer values for $Q_k(x_{ij}^k)$, we perform a Taylor series expansion to linearize $Q_k(x_{ij}^k)$ with respect to x_{ij}^k. Doing so allows us arrive at the following approximation for the quadratic program:

$$Q_k(x_{ij}^k) \approx \min_{x_{ij}^k \in \Lambda} \sum_{i,j \in S} [(p_{ij}^+ - p_{ij}^-) + 2(\hat{x}_{ij}^k)] x_{ij}^k \tag{12}$$

This Taylor expansion is necessary as it allows us to approximate the QP with integer constraints with an LP with integer constraints. At this point, solving for 12 is very simple, since it is simply a Minimal-Cost (min-cost) Flow problem, with $(p_{ij}^+ - p_{ij}^-) + 2(\hat{x}_{ij}^k)$ acting as the cost per unit flow through an arc connecting S_i to S_j. This can be easily solved using polynomial-time network flow algorithms like Edmonds-Karp. It is also known that these algorithms will always yield an integer solution, if such a solution indeed exists.

4.4 Overall Methodology

In this section, we will deliver the overall algorithm that combines all the individual patch-panel optimization problems into one coherent methodology that solves for the demand-matching problem. This requires us to solve the dual problem of the form (8), which has a concave objective function. To solve this, we propose employing an iterative, gradient-projection based solution to force the convergence of solution to the final, optimal solution. Given the concave nature of (10), there exists only one global optima w.r.t p_{ij}^+ and p_{ij}^-. In each iteration, the equation that governs the dual variable values for the next iteration can be written as:

$$\begin{aligned} p_{ij}^+(t+1) &= p_{ij}^+(t) \ - \ \delta \frac{\partial L(x_{ij}^k,\ p_{ij}^+)}{\partial p_{ij}^+} \\ &= p_{ij}^+(t) \ - \ \delta(c_{ij}^U - \sum_{k \in O} x_{ij}^k) \\ p_{ij}^-(t+1) &= p_{ij}^-(t) \ - \ \delta \frac{\partial L(x_{ij}^k,\ p_{ij}^-)}{\partial p_{ij}^-} \\ &= p_{ij}^-(t) \ - \ \delta(-c_{ij}^L + \sum_{k \in O} x_{ij}^k) \end{aligned} \tag{13}$$

(13) seems implies that the dual variables are updated every iteration. This formulation has a serious in that for each iteration, all the patch-panels which share the same physical striping will end up being configured the same way. As a result, the solutions for (8) becomes highly oscillatory. We require a formulation of the dual variables that allows for more fine grained, incremental change as a configure more patch-panels within the same iteration.

In order to capture the topology change as more links are formed and more patch-panels are being configured, we need to change the min-cost flow's edge cost such that not all. As such, we formulate instead using:

$$\begin{aligned} \phi_{ij}^+(t,k') &= p_{ij}^+(t) \ - \ \delta(c_{ij}^U - \sum_{k \neq k'} x_{ij}^k) \\ \phi_{ij}^-(t,k') &= p_{ij}^-(t) \ - \ \delta(-c_{ij}^L + \sum_{k \neq k'} x_{ij}^k) \end{aligned} \tag{14}$$

The formulation given in (14) allows for different flows to be prioritized as we iteratively solve for the switching states of each patch-panel in the inner loop in Algorithm **??**. This property is due to the addition of an extra dimension k', through which we can score significantly under-provisioned flows with lower costs. As a result, the min-cost flow solver will favor flows that would connect more underprovisioned flows, as indicated by higher $\phi_{ij}^-(t,k')$ values and lower $\phi_{ij}^+(t,k')$.

In order to solve the quadratic programs, we utilized the OSQP package [1, 17].

4.5 Algorithmic Complexity

Let us assume that the complexity for solving a QP with n variables to be $f(n)$, where $f(n)$ is a polynomial equation. Kozlov et al. [11] showed that when the objective matrix Q is positive semidefinite, then the ellipsoid method solves the QP in polynomial-time. Given that our objective matrix can be expressed as $Q = cI$, where I is the identity matrix, we are guaranteed that there exists a polynomial-runtime algorithm that solves $Q_k(x_{ij}^k)$.

```
Data:
    • $T = [t_{ij}] \in \mathbb{R}^{n\times n}$ - traffic matrix
    • $t_{max}$ - number of iterations
Result: $\hat{x}^* \in \mathbb{Z}_{\geq 0}$ - patch-panel switch states
1  Build patch-panel bipartite flow graphs $G_f^1, G_f^2, ...G_f^k$;
2  Solve (2) to obtain demand matrix $D = [d_{ij}]$;
3  $\psi := -\infty$ ;
4  for $t \in \{1, 2, ..., t_{max}\}$ do
5      $\delta := \frac{1}{t}$ ;
6      Solve (13) $\forall\, i, j \in S$;
7      for $k \in O$ do
8          Solve (11) to obtain $\hat{x}_{ij}^k$ ;
9          Obtain flow network arc costs by solving (14);
10         Solve bipartite flow $x_{ij}^k := MCF(G_f^k, x_{ij} - x_{ij}^k, \phi_{ij})$ ;
11         $(\tilde{i}, \tilde{j}) := \arg\min_{(i,j)\in S\times S} (\sum_k x_{ij}^k - d_{ij} \mid i, j \in S)$ ;
12         if $\sum_k x_{\tilde{i}\tilde{j}}^k - d_{\tilde{i}\tilde{j}} \;>\; \psi$ then
13             $\psi := \sum_k x_{\tilde{i}\tilde{j}}^k - d_{\tilde{i}\tilde{j}}$;
14             $\hat{x}^* := \{x_{ij}^k \;\forall\, i, j \in S, k \in O\}$;
15     end
16 end
```

Algorithm 1: Overall logical topology reconfiguration algorithm

The complexity for Algorithm **??**.

4.6 Duality Gap and convergence of algorithm

4.7 Convergence of Algorithm

We prove here that Algorithm **??** converges under the sole assumption that there exists a solution of the form $\tilde{x}_{ij}^k \in \mathbb{Z}^+, \tilde{p}_{ij}^+$, and $\tilde{p}_{ij}^- \;\forall i, j \in S, k \in O$ to [8]. Based on this assumption, we claim that by initiating $x_{ij}^k = 0, p_{ij}^+ = 0, p_{ij}^- = 0 \;\forall\, i, j \in S, k \in O$, the final solution $\hat{x}^*$ that minimizes under-provisioning can be obtained. Using this as a starting ground, we prove that our approximation algorithm converges towards the ILP solution.

The proof is twofold: first we have to prove that the linearization of the quadratic program will not cause divergence. Next, we need to prove that replacing the arc weights of 13 with 14 does not cause divergence.

5 PRACTICAL DEPLOYMENT IN REAL-TIMED SYSTEMS

5.1 Accuracy of Relaxation

5.2 Timing Analysis

6 PICKING REPRESENTATIVE TRAFFIC MATRIX

7 SIMULATIONS RESULTS

7.1 Improving Max Link Utilization (MLU)

7.2 Network Decongestion

7.3 Economic Savings

8 CONCLUSION

We propose a traffic-aware topology design methodology based on the co-optimization of both the logical topology and the routing. We've proposed a robust and fast logical topology reconfiguration algorithm by While our proposed methodology is aimed at monthly-reconfiguration, the topology reconfiguration algorithm can is fast and can be used for real-time reconfiguration. In other words, the topology reconfiguration algorithm can be employed within datacenter and even HPC networks that utilize optical switching with faster reconfiguration time.

REFERENCES

[1] G. Banjac, B. Stellato, N. Moehle, P. Goulart, A. Bemporad, and S. Boyd. 2017. Embedded code generation using the OSQP solver. In *IEEE Conference on Decision and Control (CDC)*. https://doi.org/10.1109/CDC.2017.8263928

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

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

[10] 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.

[11] Mikhail K Kozlov, Sergei P Tarasov, and Leonid G Khachiyan. 1980. The polynomial solvability of convex quadratic programming. *U. S. S. R. Comput. Math. and Math. Phys.* 20, 5 (1980), 223–228.

[12] 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.

[13] 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.

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

[15] 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.

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

[17] 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

[18] 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.

[19] Ke Wen, Payman Samadi, Sébastien Rumley, Christine P Chen, Yiwen Shen, Meisam Bahadori, Keren Bergman, and Jeremiah Wilke. 2016. Flexfly: Enabling a reconfigurable dragonfly through silicon photonics. In *High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for*. IEEE, 166–177.

[20] Shizhen Zhao, Rui Wang, Junlan Zhou, Joon Ong, Jeffrey. C Mogul, and Amin Vahdat. 2019. Topology-aware vm migration in bandwidth oversubscribed datacenter networks. In *NSDI*.