Rehoming Edge Links for Better Traffic Engineering

Eric Keller, Michael Schapira, Jennifer Rexford
Princeton University
ekeller@princeton.edu, ms7@cs.princeton.edu, jrex@cs.princeton.edu

***Abstract*—Traditional traffic engineering adapts the routing of traffic *within* the network to maximize performance. We propose a new approach that also adaptively changes where traffic *enters* and *leaves* the network—changing the "traffic matrix", and not just the intradomain routing configuration. Our approach does *not* affect traffic patterns and BGP routes seen in neighboring networks, unlike conventional inter-domain traffic engineering where changes in BGP policies shift traffic and routes from one edge link to another. Instead, we capitalize on recent innovations in edge-link migration that enable seamless rehoming of an edge link to a different internal router in an ISP backbone network—completely transparent to the router in the neighboring domain. We present an optimization framework for *traffic engineering with migration* and develop algorithms that determine which edge links should migrate, where they should go, and how often they should move. Our experiments with Internet2 traffic and topology data show that edge-link migration allows the network to carry 18.8% more traffic (at the same level of performance) over optimizing routing alone.**

I. Introduction

The rapid growth of online services, from video streaming to 3D games and virtual worlds, is placing tremendous demands on the underlying networks. ISP backbone networks carry more traffic than ever. To address these challenges, network operators do *traffic engineering* (TE). Traditionally, traffic engineering involves tuning routing-protocol parameters to control how traffic is routed across the network, to optimize performance and use network resources effectively. Thus, today's traffic engineering adapts routing *within* the network for a given *traffic matrix*, *i.e.*, the volume of traffic between *fixed* traffic ingress and egress points. Traditional traffic engineering assumes that the locations of traffic sources and sinks cannot change over time. Recent innovations challenge this approach.

A recently proposed mechanism—"*router grafting*" [1]—allows an ISP to dynamically rehome its ends of links to neighboring networks. For example, an ISP could move a customer that connects in New York to a different router in nearby New Jersey, transparently to the customer. With this capability, traffic engineering can go beyond adapting the routing protocol to control where traffic enters and exits the network. In effect, an ISP now has the power to *change the traffic matrix* without disrupting its neighbors. The flexibility enabled by router grafting gives rise to new possibilities in traffic engineering. Dynamically relocating traffic end-points can redirect traffic to decrease the traffic traversing a congested bottleneck, or capitalize on unused bandwidth.

In this paper, we introduce *traffic engineering with migration*, and present a framework that addresses the following questions:

- How much can traffic engineering with edge-link migration improve over traditional traffic-engineering techniques?
- Which edge links should migrate, and where should they migrate to?
- How often should traffic edge links migrate?
- Can a "good" placement of edge links be computed efficiently?

We present and analyze traffic engineering with migration in the context of edge-link migration via router grafting in backbone ISP networks.

A. Edge-Link Migration via Router Grafting

An ISP network connects to neighboring networks (customers, peers, or providers) at its perimeter. To establish a link to another network, the ISP selects one of its routers to connect to the adjacent network. Traditionally, the link remains fixed unless there is significant reason for change. This is because changing to a different internal router in real time can be extremely disruptive to the Border Gateway Protocol (BGP) session with the neighboring network, requiring significant coordination such as scheduling a maintenance window. During the transition period, data packets may be lost or delivered out of order, and routers throughout the Internet receive additional BGP update messages.

New mechanisms for rehoming links make the change transparent [1], [2]. Router grafting [1] enables an ISP to move its end of the link without disrupting user performance and without coordination with the neighboring network; the earlier RouterFarm [2] does so with slight downtime. Router grafting rehomes the layer-three link (through signaling in the programmable transport network), migrates the local end-point of the TCP connection to the neighbor's router, and transparently transfers the routing-protocol state to a different internal router. Router grafting has no impact on the neighboring network—the neighbor is not aware that grafting has happened, and sees no change in where traffic enters or leaves its own routers. This is in sharp contrast to traditional inter-domain traffic engineering, where an ISP changes its BGP policies to shift traffic from one edge link to another—triggering both BGP update messages and changes in where traffic enters or leaves neighboring networks [3]–[6].

Router grafting enables an ISP to migrate a link within a few seconds without disruption, allowing network operators to change the ingress and egress points for traffic in real time. The overhead of router grafting is relatively low. Grafting involves the export of state from one router, the transference of state, and the import of state at another router. Changing the network topology requires some routers to repeat the route-selection process, leading to a temporary increase in CPU load. In addition, some routers may change their routing decisions, leading to a temporary increase in BGP update messages. These overheads are short-lived, and do not disrupt the flow of data traffic. As such, network operators can afford to make periodic adjustments to where they terminate the links to neighboring networks.

B. *Traffic Engineering with Edge-Link Migration*

We develop techniques to determine which edge-links should migrate, to where, and how often. We first present a formal framework for traffic engineering with edge-link migration. We show that finding the optimal solution, or even a reasonable approximation of the optimal solution, in this new traffic-engineering setting is computationally intractable. We then present two relatively simple heuristics for traffic engineering with edge-link migration and show that they offer significant performance improvements in practice. Our experiments with Internet2 traffic and topology data show that migration would enable the network to carry 18.8% more traffic at the same level of performance. Importantly, we can achieve close to this level of improvement without frequently re-optimizing the topology—performing this re-optimization every 12 hours still sees a nearly 15% improvement. Similarly, only a small fraction of links need to be migrated in each interval. Migrating about 10% of the links results in the full improvement, but migrating less than 5% of the links still achieves 95% of the benefits.

Organization. After a brief review of traditional traffic engineering, we introduce traffic engineering with migration in Section II. Section III shows that computing an optimal solution is hard, and presents two heuristics for traffic engineering with migration. Section IV presents our experimental evaluation of both of these heuristics. We wrap up with a presentation of related work in Section V, and conclusion in Section VI.

II. Traffic Engineering Model

A. *Traffic Engineering Today*

In traditional traffic engineering, the network is represented by a directed graph $G = (V, E)$, where the vertex set V represents routers, and the edge set E represents the links. Every edge $e \in E$ has capacity $c_e > 0$. We are also given a *traffic matrix* $D = \{d_{ij}\}_{i,j \in V}$, where entry $d_{ij} \geq 0$ is the amount of traffic that vertex i wishes to send vertex j. The goal is to distribute flow across the paths from i to j to minimizing total link usage (TLU).

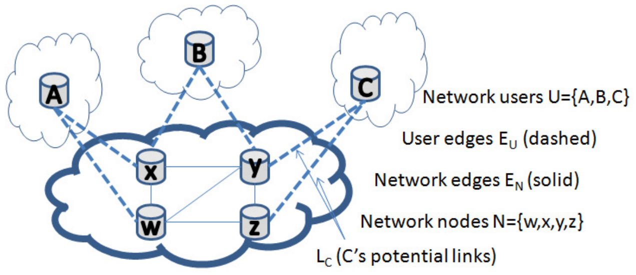

A
B
C
X
Y
W
Z
Network users U={A,B,C}
User edges E_U (dashed)
Network edges E_N (solid)
Network nodes N={w,x,y,z}
L_C (C's potential links)

Fig. 1. Network model for traffic engineering with migration.

TLU minimization reflects a common goal in ISP networks [7]. Each link e has a "cost" that reflects its level of congestion, where lightly-loaded links are "cheap" and links become exponentially more "expensive" as the link becomes heavily loaded. The *cost function* ϕ_e specifies the cost as a function of f_e (the total flow traversing the edge) and c_e (the edge capacity). Every ϕ_e is a piecewise linear, strictly increasing and convex function. We use the cost function from [7], shown below:

$$\phi_e(f_e, c_e) = \begin{cases} f_e & 0 \leq \frac{f_e}{c_e} < \frac{1}{3} \\ 3f_e - \frac{2}{3}c_e & \frac{1}{3} \leq \frac{f_e}{c_e} < \frac{2}{3} \\ 10f_e - \frac{16}{3}c_e & \frac{2}{3} \leq \frac{f_e}{c_e} < \frac{9}{10} \\ 70f_e - \frac{178}{3}c_e & \frac{9}{10} \leq \frac{f_e}{c_e} < 1 \\ 500f_e - \frac{1468}{3}c_e & 1 \leq \frac{f_e}{c_e} < \frac{11}{10} \\ 5000f_e - \frac{16318}{3}c_e & \frac{11}{10} \leq \frac{f_e}{c_e} < \infty \end{cases}$$

The goal is to distribute the *entire* demand between every pair of vertices in a manner that minimizes the sum of all link costs (i.e., $\Sigma_{e \in E}\ \phi(f_e, c_e)$). (Observe that the flow along an edge can exceed the edge's capacity.) TLU minimization can be formulated as *minimum-cost multicommodity flow* and is thus computable using existing algorithms for computing multicommodity flows. Realizing this objective in practice can be done via MPLS and a management system that solves the optimization problem and installs the resulting paths. Network operators often take the indirect approach of tuning Interior Gateway Protocol (IGP) weights to closely approximate the optimal distribution of the traffic [7].

B. *Migration-Aware Traffic Engineering*

We now extend the traffic-engineering model in Section II-A to incorporate migration. Table I summarizes the notation.

Distinguishing users from network nodes: In our model for traffic engineering with migration, the network (see Figure 1) is represented by a directed graph $G = (V, E)$, where the vertex set V is the union of two disjoint subsets, U and N. U is the set of *network users*, that is, originators and receivers of traffic, and N is the the set of *network nodes*, that is, the routers in the network. The term "users" here refers to users of the network and not to end-users. In an ISP network, the set of users U represents routers in neighboring

Notation	Description
G	Network graph $G = (V, E)$
V	Network vertex, union of U and N
E	Network edge, union of E_U and E_N
U	Set of network users
N	Set of network nodes
E_U	Subset of edges that connect user $u \in U$ to network nodes in N, $E_U \subseteq U \times N$
E_N	Subset of edges that connect network node $n \in N$ to network nodes in N, $E_N \subseteq N \times N$
c_e	capacity of edge $e \in E$
L_u	Potential links, $L_u \subseteq E_U$
D	Demand matrix, $D = \{d_{ij}\}_{i,j \in U}$
d_{ij}	Amount of traffic that user i wishes to send user j
ϕ_e	cost function used in TLU minimization, function of f_e/c_e
f_e	Total flow traversing the edge e

TABLE I
SUMMARY OF NOTATION USED IN MODEL OF TRAFFIC ENGINEERING WITH MIGRATION.

networks ("adjacent routers") and the set of network nodes N represents the routers in the ISP's internal network ("internal routers").

User edges are potential links: To capture the ability to migrate, we introduce the notion of *potential links* that represent the locations where the user can *possibly* connect to the network. The edge set E is the union of two disjoint subsets, E_U and E_N, where $E_U \subseteq U \times N$ is the subset of edges connecting users to network nodes, and $E_N \subseteq N \times N$ is the subset of edges connecting network nodes to other network nodes. Each edge $e \in E_N$ has capacity $c_e \geq 0$, which measures the amount of flow that can traverse edge e. We impose no capacity constraints on the edges in E_U (that is, these edges have infinite capacity). We call the set of all edges $L_u \subseteq E_U$ that connect user $u \in U$ to network nodes in N "the set of u's *potential links*" (that is, $\forall u \in U$, $L_u = \{e = (u, v) | \; e \in E_U\}$).

In ISP networks, the set of potential links L_u for each adjacent router (user) u represents the points at which u can connect to the ISP network. This can, in practice, depend on the underlying transport network that can, for example, limit a user to connecting only to network nodes in nearby geographical regions. In addition, the set of potential links can reflect latency considerations, *e.g.*, it is beneficial to home frequently-communicating users near each other.

Demand matrix is user-to-user: Our model distinguishes network users from network nodes, and our demand matrix captures this distinction; we are now given a demand matrix $D = \{d_{ij}\}_{i,j \in U}$, where each entry d_{ij} specifies the amount of traffic *user* i wishes to send *user* j.

Each user must use a single potential link: The high-level goal is, for every pair of users i and j such that $d_{ij} > 0$, to distribute flow from i to j between the routes from i to j in G, subject to the constraint that every user can only connect to the network via a *single* link. That is, for every user $u \in U$, traffic flowing from that user to the other users, and vice versa, can only traverse a single edge in L_u; traffic along all other edges in L_u must be 0. When optimizing the flow of traffic through the network we again consider the TLU minimization objective function.

C. Practical Considerations

Naturally, our formal framework does not capture all the constraints that could arise in practice. We now present several such constraints and discuss how these can be incorporated into our model. We revisit some of these in later sections and leave the others as interesting directions for future research.

Cost of migration. Our framework does not model the cost of migration (in terms of processing, offline time, and more), yet this is expected to be a consideration in practice; we present some indication of the impact of this cost, based on experiments with Internet2 data, in Section IV-E. We can incorporate that cost into our model as follows. The input will include, in addition to the other components, an edge $\overline{e_u} \in L_u$, for every user $u \in U$, that represents the link that user u is *currently* using to connect to the network, and also costs associated with changing each user u's current connection edge to other edges in L_u.

Router limitations. Other practical considerations are the physical limitations of the individual vertices in the network, including the number of links that each vertex can support, and also the capacity of the node (in terms of processing, memory, bandwidth, *etc.*). This can be incorporated into our model through additional constraints (*e.g.*, limits on the number of incoming links per node, node-capacity functions dependent on incoming traffic amount, *etc.*).

Multi-homed users. We did not model the case that users are multi-homed, that is, that users connect to the network at more than one location. This alters our constraint that a single potential link must be chosen per user. To incorporate this into our model we can introduce a variable for each user u that specifies how many links in L_u that user is allowed to send/receive traffic along. It also adds the complexity that changing the ingress point may alter the egress point (*i.e.,* "hot-potato routing" [8]), thus changing the traffic matrix beyond the change introduced with migration. The design and evaluation of heuristics/algorithms for this more general environment is left for future work.

III. Two Edge-Link Migration Heuristics

Ideally, we would be able to find a TLU-minimizing solution in which each user sends/receives traffic along a single potential link in a computationally-efficient manner. Unfortunately, we prove that finding an optimal solution is NP-hard and that even well-approximating the optimal solution is intractable. We thus seek heuristics which fare well in practice. We present two heuristics—the *max-link heuristic* and the *cluster-user heuristic*. We experimentally evaluate these heuristics in Section IV.

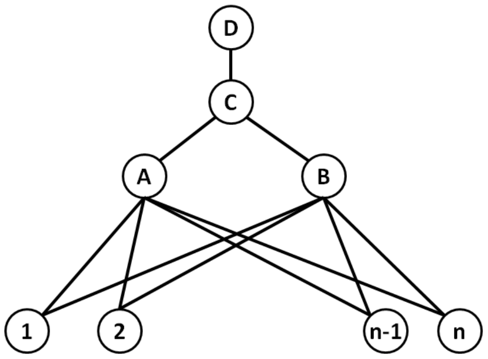


Fig. 2. Example network for proof of NP hardness.

A. *"Good" Solutions are Hard to Compute!*

Computing an optimal solution is hard. In the traditional traffic-engineering setting (see Section II-A), where migration is not considered, existing algorithms for computing multicommodity flow provide the optimal solution in a computationally-efficient manner. Unfortunately, we show that computing the multicommodity flow in our "TE with migration" setting is computationally intractable. In fact, this is true even when every user has at most two potential links.

Theorem 3.1: Minimizing TLU subject to the restriction that each user send/receive traffic along a single potential link is NP-hard even when $|L_u| \leq 2$ for every user u.

Proof: We show a reduction from the NP-hard PARTITION problem. In PARTITION, the input is a set of positive numbers $S = \{a_1, \ldots, a_n\}$ and the objective is to determine whether this set of numbers can be partitioned into two disjoint subsets of numbers, S_1 and S_2, such that the sum of the numbers in each of the subsets is the same (and hence equals precisely $\frac{\Sigma_i a_i}{2}$). Given an input to PARTITION, we construct a network as described in Figure 2. The vertex set consists of the three network nodes $N = \{A, B, C\}$ and also of a set of users $U = [n] \cup \{d\}$. Each user $i \in [n]$ aims to send $\frac{a_i}{3}$ units of traffic to user d and can connect to the network at nodes A and B. User d connects to the network at node C. Let the capacity of both network edges (A, C) and (B, C) be $\frac{\Sigma_i a_i}{2}$.

Now, observe that if there is a way to partition S into two disjoint subsets, S_1 and S_2, such that the sum of elements in each subset is the same, then it is possible to send all traffic to d while utilizing exactly a third of the capacity of both (A, C) and (B, C); simply connect all users whose corresponding numbers lie in S_1 at A and all users whose corresponding numbers lie in S_2 at B. However, observe that in the event that S cannot be partitioned into S_1 and S_2 as above, then more than a third of the capacity of either (A, C) or (B, C) must be exceeded, resulting in a higher TLU value. Thus, an optimal multicommodity flow solution in our TE setting will enable us to distinguish between the case that S can be partitioned as desired to the case that this is impossible. The theorem follows. ■

Approximating the optimal solution is also hard! We now prove that even approximating the optimum within a constant factor better than 1000 is computationally intractable.

Theorem 3.2: Approximating the TLU minimizing solution within a factor better than 1000 subject to the restriction that each user send/receive traffic along a single potential link is NP-hard.

Proof: To establish this inapproximability result, we present a reduction from MAX-LABEL-COVER. An instance of MAX-LABEL-COVER contains a directed regular bipartite multigraph $G = (A, B, E)$ and a finite set of "labels" $L = \{1, \ldots, r\}$. Each edge $e \in E$ leads from a vertex in A to a vertex in B and is associated with a set $L_E \subseteq L \times L$. The goal is to select a "labeling", *i.e.*, to assign every vertex $v \in A \cup B$ a label in $l(v) \in L$, respectively, so as to maximize the number of "satisfied" edges, where an edge $e = (a, b)$ is satisfied if $(l(a), l(b)) \in L_E$. We say that γ-fraction of the edges are satisfiable, for $\gamma \in (0, 1]$, if there exists a labeling that satisfies a γ fraction of the edges. We say that an instance of MAX-LABEL-COVER is "completely satisfiable" if there exists a labeling which satisfies all edges.

We show a reduction from MAX-LABEL-COVER to TE with migration. We create a user u_v for every vertex $v \in A \cup B$. For every user u_v we create r distinct network nodes $N^v = \{n_1^v, \ldots n_r^v\}$, each corresponding to a unique label in L (there is no intersection between such sets of network nodes that correspond to different users). User u_v has a potential link to all nodes in N_v, each with infinite (arbitrarily high) capacity. For every pair of users, u_a and u_b, there exists a directed edge between network nodes n_i^a and n_j^b of capacity 1 if $e = (a, b) \in E$ and $(i, j) \in L_E$. Otherwise, the two network nodes are connected by an edge of capacity $\delta > 0$ otherwise. Every user u_a which corresponds to a vertex $a \in A$ wishes to send $\frac{1}{3}$ units of traffic to every user u_b such that $(a, b) \in E$. All other demands are 0.

We make use of the following result, due to Moshkovitz and Raz [9]. Given an instance of MAX-LABEL-COVER, and $\epsilon > 0$, distinguishing between the case that it is completely satisfiable, and the case that at most an ϵ-fraction of its edges are satisfiable, is NP-hard. Intuitively, every TE solution corresponds to a labeling where the ingress point of each user represents the chosen label, and vice versa. Observe that in the event that the MAX-LABEL-COVER instance is completely satisfiable, it is possible to satisfy all demands at cost $|E|$. This is achieved by connecting each user to the network through the potential link that corresponds the its label in the completely satisfying labeling. Observe also that in the event that at most an ϵ-fraction of the edges are satisfiable every TE solution will have cost at least $\frac{(1-\epsilon)|E|}{3} \times 5000 - \frac{16318\delta}{3}$. This is due to the fact that most of the flow in this case is bound to traverse δ-capacity edges. By choosing ϵ and δ to be small enough, we get that this last expression is greater than $1000|E|$.

Hence, distinguishing between the case that the TLU is at most $|E|$, and the case that it is at least $1000|E|$, makes it possible to distinguish between the case that the original MAX-LABEL-COVER instance is completely satisfiable, and the case that at most an ϵ-fraction of its edges are satisfiable. The theorem follows. ∎

B. Max-Link Heuristic

Intuitively, the max-link heuristic first computes the maximum multicommodity flow in the input network that contains *all* potential links. Then, the heuristic uses this fully-fractional flow (where users' traffic can be split between all their potential links) to choose a single potential link for each user, thus constructing a feasible (integral) solution. To do this, the max-link heuristic throws away, for each user, all potential links but the single potential link along which the user sends and receives the most traffic. The max-link heuristic consists of the following three steps:

- **Step I: Compute multicommodity flow f in the input network G** (that contains all potential links for each user). That is, compute the multicommodity flow without restricting users to sending and receiving traffic along a single potential link. The multicommodity flow solution f tells us how much traffic every user u sends and receives along each of the potential links in L_u. We let $t(l_u)$ denote the sum of traffic that user u sends and receives along the potential link $l_u \in L_u$.
- **Step II: Use the most utilized potential links.** Choose, for every user $u \in U$, the single potential link in $l_u \in L_u$ for which $t(l_u)$ is maximized. (Migrate users' potential links if necessary.)
- **Step III: Output the multicommodity flow in the resulting network**, that is, in the network obtained through the removal from G of all potential links but those chosen above. The algorithm outputs (i) the choice of a single potential link for each user and (ii) the optimal routing of traffic subject to these migration decisions.

C. Cluster-User Heuristic

We now present the cluster-user heuristic, which considers the users in groups, and not individually. Grouping users together is motivated by the fact that, in practice, often large numbers of users can connect to the network at the exact same locations (*e.g.,* via a common access network). In addition, clustering users reduces the network size, thus making computation more efficient.

The intuition behind the cluster-user heuristic is the following. Consider the scenario that the set of users is divided into groups (or clusters) of users, such that every cluster contains a large number of users who all can connect to the network at the exact same locations (network nodes), and such that each user's demands constitute a small fraction of the demands of the cluster as a whole. We observe that, in this case, each cluster can be regarded as a single user with the ability to split traffic among its potential links almost as in the optimal multicommodity flow solution. This follows from the fact each user in the cluster sends/receives a negligible amount of the total traffic, and so users in the cluster can be mapped to outgoing links so as to closely mimic the multicommodity flow solution.

To illustrate this point, consider the example in Figure 3(a). There are six users (labeled A-F) which are grouped into two clusters (cluster 1 and cluster 2). In the cluster user approach, users can (but do not necessarily) belong to the same cluster if their sets of potential links connect to the network at the exact same network nodes.

We create a new network where each user cluster is replaced by a single user (with a set of potential links that connect to the network at the exact same network nodes). We then solve multicommodity flow for this network (allowing traffic to be split between multiple potential links) to get the fraction of traffic flowing over each potential link (shown in Figure 3(a)).

We now use the multicommodity flow solution to map users to potential links, as shown in Figure 3(b). In our example, 0.79 units of cluster 1's traffic should be via w and 0.21 units via x. Hence, we map A (0.60) and B (0.20) to w, and C (0.20) to x. Observe that this is the "best fit" as splitting the traffic exactly as in the multicommodity flow is impossible.

We note that, in general, finding the best fit can easily be shown to be NP-hard, even in the case that every user has 2 potential links, yet good approximations are achievable. In our experiments, we were able to use a brute-force approach to determine the best fit; for each cluster, we go over all possibilities for mapping users to potential links; we pick the mapping that minimizes the sum of gaps between the traffic sent along the potential links in the mapping and in the multicommodity flow solution. In Figure 3(a), for example, a possible mapping is to map user A to node w, and users B and C to node x, which leads to a "penalty" of 0.38 (as $0.38 = |0.79-0.60| + |0.21-(0.2+0.2)|$); mapping users A and B to w and C to x is the optimal solution with penalty 0.02.

IV. Evaluating TE with Migration

We now present our experimental evaluation of the max-link heuristic. The goal of this evaluation is to demonstrate the benefits of using migration in traffic engineering, even with a simple heuristic. We first show in Section IV-C that our max-link heuristic does indeed lead to an improvement in network performance. We then examine two additional concerns relating to more practical questions—how often do links need to be migrated (Section IV-D) and how many links need to be migrated (Section IV-E).

A. Experimental Setup with Internet2 Data

We based all of our experiments on data collected from Internet2 [10], which consists of $N = 9$ core routers and $U = 205$ external routers. We collected one week of data starting January 18, 2010. From each router, we downloaded

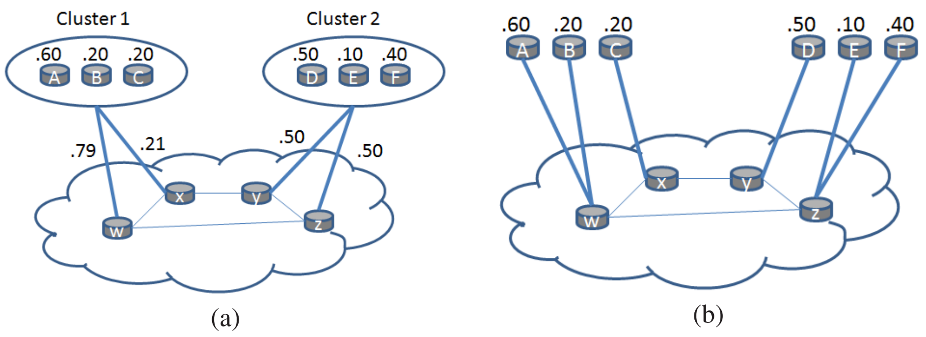


Fig. 3. Illustration of the cluster user approach.

the previously collected NetFlow data which provides summaries of the sampled flows (at the rate of 1/100 packets) in 5-minute intervals (1-week of traffic is 2016 5-minute samples). We also downloaded the routing information base (RIB) and the output for the '*show bgp neighbor*' command, both of which are captured every two hours. Every NetFlow entry contains the incoming interface, which we used to represent an external source user. We used the routing tables for each of the routers to determine the egress router for each flow, along with the specific interface on the egress router that the flow exits the network on, which we used to represent the external destination user. This enabled us to generate an external-user-to-external-user traffic matrix.

Our choice of the set of potential links (the L_u's) in the experiments was based on geographical distance, with the users' locations inferred from which router they are connected to in the original topology—e.g., some users connected in Chicago would have New York as a second potential link (in addition to Chicago), others would have Kansas as a second potential link

Our choice of the set of potential links (the L_u's) in the experiments was based on geographical distance. The first potential link for a given user is to the router the user is connected to in the original topology. The second potential link is to a router randomly selected from the routers nearest the original [1].

B. How to Select the Right Clusters?

Before presenting the experimental results, we will discuss how we selected the clusters. Selecting the "right" cluster of users (and cluster size) is not trivial for the following reasons: (1) Traffic sent from a user in a cluster to another user in the same cluster is not considered (as the entire cluster is treated as a single user). Interestingly, the intuition that the bigger the cluster size is the more unconsidered traffic we have is not necessarily true; sometimes increasing the cluster size changed the division points and resulted in highly communicating users falling into different clusters, thus actually decreasing the unconsidered traffic. (2) Best-fitting users to potential links to match the computed multicommodity flow can lead to lost traffic if the fitting is not perfect (*e.g.*, in the multicommodity flow solution 0.79 units of traffic flow over the link to node w in Figure 3, yet the closest we can come is 0.80 units).

While it is ideal to take both the intra-cluster traffic and the best-fitting penalty into account when selecting clusters, the best-fit penalty is only known after running through the entire heuristic (selecting clusters, running multi-commodity flow, and determining the best-fit). Understanding how to select user clusters (in a computationally-efficient manner) is a direction for future work. We suspect that the cluster user approach is more suitable for larger networks with more users, where we have more freedom in choosing the composition of the clusters. In such networks, we can place users who communicate with each other in separate clusters so as to avoid not considering that traffic. We can also form big clusters with even distribution of traffic, thus making it easier not to lose traffic in the course of best-fitting users to potential links.

For our experiments we formed clusters by minimizing the intra-cluster traffic. In Figure 4 we show the effects of this for the 2 potential links per user case. As expected, smaller clusters have the least amount of intra-cluster traffic and it exponentially rises as cluster size increases, eventually reaching 100% for the case when a cluster consists of all users.

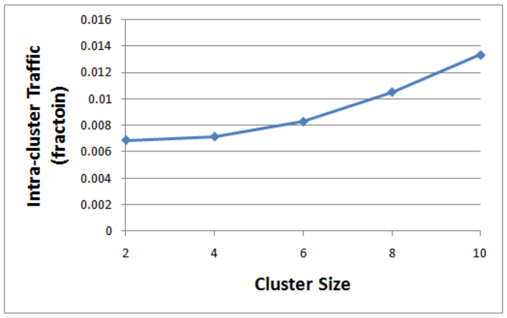


Fig. 4. Intra-cluster traffic for varying cluster sizes.

[1]We do not present results for more than two potential links per-user, as in our small topology almost every two users end up connected to a common network node when there are many potential links, and thus traffic between these users does not traverse the network at all. To elaborate, consider the extreme case in which all users have potential links to all routers. Here, a multicommodity flow solution will give no guidance on which links to use since no traffic will even traverse the network.

C. Migration Improves Network Utilization

The first metric of importance is simply the improvement that can be obtained when utilizing link migration. Here we first define the metric we are evaluating, show that the improvement varies depending on the traffic patterns, and conclude that max-link is slightly better.

1) Defining 'Improvement': In order to determine (i) which heuristic is better, (ii) what is the best cluster size, and (iii) how much improvement we can achieve over a network optimizing only the routing we need to clearly define a metric for which we will compare.

The total link utilization for an example 5-minute period appears in Figure 5. The Figure shows results for the original (optimally engineered) network (the "original topology" line), and for traffic engineering with migration (using max-link) with 2 links per user (the "optimized topology" line).

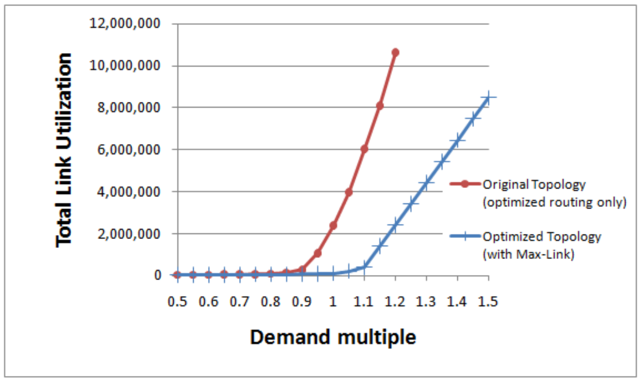


Fig. 5. Evaluation of max-link for a single 5-minute period.

To obtain the graph in Figure 5, we varied the traffic demand by scaling all entries in the demand matrix by a multiplicative factor, plotted on the x-axis, and optimized for the TLU for each. TLU minimization captures the goal of avoiding congestion, and involves an exponentially increasing cost for utilizing a link (see Section II). We used the cost function from [7] as detailed in Section II-A.

Due to the exponentially increasing cost, the network operator will wish to be at a point in the curve that comes before the exponential rise, that is, before the "knee" in the curve. Observe that this "knee" shifted to the right by roughly 20%, and so, with migration, the network can handle 20% more traffic with the same level of congestion.

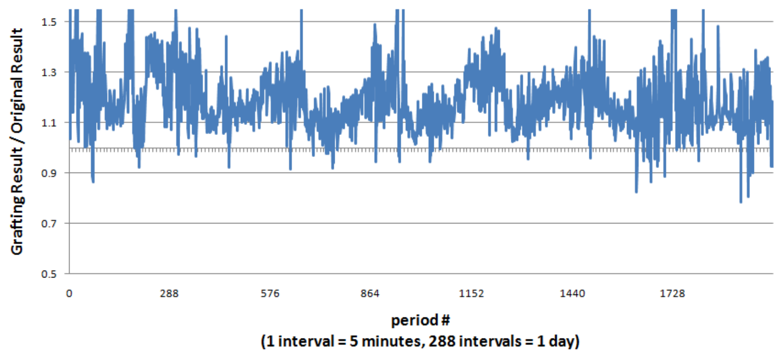
Note that the original topology is currently operating at the knee (since the knee is at 1, which is the actual demand matrix). From this, we define the improvement metric as the amount of traffic the network can carry in the optimized topology at the same level of congestion as the original topology—where the TLU represents the level of congestion. So, from the original topology, we found the minimal TLU with a demand multiple of 1 (i.e., the actual amount of traffic). We then determined which demand multiple in the optimized topology (i.e., with migration) would result in the same TLU. In other words, in terms of the graph in Figure 5, we found the y value for x=1 on the original topology line, and used that y value to find the x value on the optimized topology line.

2) Improvement Varies Between Intervals: Not all periods will see an equal improvement. To expand on this, we present in Figure 6 the results for max-link for all 2016 5-minute intervals (cluster-user is similar). In Figure 6(a), we show a time-series representation where each data point represents the improvement achieved with link migration. Plotted in Figure 6(b) is a cumulative distribution function of the same information. The traffic patterns dictate how much improvement can be achieved—if the network is not that congested during a particular interval, performing edge link migration will have minimal affect since, in that case, it effectively is optimizing an already underutilized network.

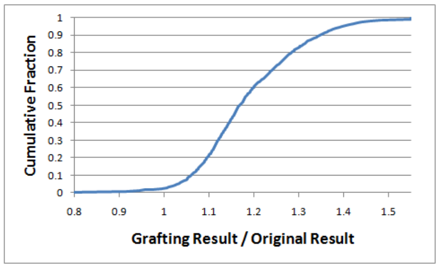
3) Max-link is slightly better than Cluster-user: Shown in Table II is a comparison of the improvements achieved with the max-link heuristic as well as the cluster-user heuristic, for different cluster sizes. From this we can see on average, traffic engineering with migration can increase network utilization by about 18.8% when using max-link, and slightly less when using cluster-user. Note that in both the max-link and cluster-user heuristics, we first calculate the TLU minimizing multicommodity flow of a graph which includes the potential links. The input is a prediction of what the expected demands will be so that a new topology can be optimized for it. For the experiments, we utilized the actual demand matrix, in essence giving perfect predictive power. We rely on an ISP's ability to predict traffic based on past history and the improvement obtainable will be related to the accuracy of the prediction (as we show in Section IV-D, we can still achieve good results over longer intervals, which are easier to predict).

Intuitively this improvement comes from two factors. The first is that by optimizing the homing location based on the demand matrix, users that communicate will tend to get closer together. Without link migration, the homing location must be determined up front and then cannot change. With link migration, we can alter the topology to bring users that communicate a lot closer together. The second factor is that by re-optimizing the topology, we can have a significant impact on congested links. By giving some traffic the ability to avoid the congested link (through migration), we can reduce the congestion on that link. There are, however, a small number of cases (2.6% in the case of max-link) where migrating links actually decreased performance. As mentioned in Section III these are only heuristics which fare well in practice but do not approximate the optimal result within some factor. Therefore, it is expected that there can be conditions which result in poor performance.

The cluster-user heuristic with a cluster size of two is very comparable to max-link. Within the cluster-user heuristic, the performance decreased as the cluster size increased. This suggest there are many factors related to the cluster size and cluster contents which influence performance as the fitting



(a) time-series



(b) cumulative distribution function

Fig. 6. Evaluation of max-link over 7 days of traffic.

heuristic	min	max	mean	#intervals worse (frac.)
max-link	0.783	1.550	1.188	54 (0.0268)
cluster-2	0.860	1.550	1.172	22 (0.0109)
cluster-4	0.565	1.550	1.121	76 (0.0377)
cluster-6	0.718	1.480	1.080	208 (0.1032)
cluster-8	0.790	1.460	1.066	261 (0.1295)
cluster-10	0.622	1.363	1.051	375 (0.1860)

TABLE II
COMPARISON OF THE IMPROVEMENT OVER THE ORIGINAL TOPOLOGY OPTIMIZED FOR ROUTING ONLY WHEN WITH MAX-LINK AND CLUSTER-USER (FOR DIFFERENT CLUSTER SIZES).

interval	min	max	mean	#worse (frac.)
5 mins	0.783	1.550	1.188	54 (0.0267)
30 mins	0.757	1.550	1.166	146 (0.0724)
1 hour	0.777	1.550	1.163	152 (0.0753)
6 hours	0.801	1.550	1.149	182 (0.0902)
12 hours	0.856	1.550	1.141	191 (0.0947)
24 hours	0.806	1.550	1.083	465 (0.2306)

TABLE III
COMPARISON OF THE IMPROVEMENT OVER THE ORIGINAL TOPOLOGY OPTIMIZED FOR ROUTING ONLY WHEN PERFORMING GRAFTING AT DIFFERENT INTERVALS (OVER 7 DAYS TRAFFIC).

penalty decreases with the increase in cluster sizes—a lower best-fit penalty is better as it means we were able to match the optimal multi-commodity flow results more closely. Further research is needed to determine the best cluster composition.

D. *Frequent Migration is Not Necessary*

In Section IV-C, we examined the benefits of utilizing link migration in traffic engineering. We looked at the benefits when we could migrate every interval and knew the traffic in the next interval. However, predicting this can be difficult on that short of a time scale. Here, we examine how frequently we really need to be migrating.

To determine how often migration should occur, we looked at different periods—every 5 minutes, 30 minutes, 1 hour, 6 hours, 12 hours, and 24 hours. The demand matrix used when computing the multicommodity flow as per Step I in the max-link heuristic (i.e., the predicted traffic), was the average demand matrix for the next interval (e.g., the next 6 hours). This was used to determine the optimized topology that would be used for the entire interval. As with the 5-minute case, errors in the prediction affect the results. We note, however, that as the intervals become longer, traffic patterns smooth out and become more predictable. We determined the TLU for each 5-minutes of traffic using this topology and compared the results to the original topology.

Table III shows the results for the different intervals. As could be expected, the longer the interval, the worse the results. However, even re-optimizing the topology every 6 or 12 hours still has good performance.

E. *Only a Fraction of Links Need to be Migrated*

Our formulation of traffic engineering with migration does not currently incorporate the cost of migration. To decide which users to migrate, we can weigh the cost of migrating a user against the gain from migrating that user; when the impact of migrating a user is low (*e.g.*, when that user generates and consumes negligible amounts of traffic), migration might be undesirable. To investigate this, we plotted in Figure 7 the amount of traffic each user sends or receives for an example 5-minute interval. On the x-axis is the index of the user, sorted by the amount of traffic they generate/consume. On the y-axis is the cumulative fraction of the total traffic. We placed markers on each user that our max-link heuristic determined should be migrated. From this we can see that 85% of the traffic comes from the first 42 users, of which, max-link only determined 5 of them should be migrated. Hence, we can still obtain a significant improvement in network performance while migrating only a small number of links.

To evaluate this effect across the entire data sample, we plotted the cumulative distribution functions of the number of links that need to be migrated for three different thresholds—100% (i.e., migrate all links that max-link determined need to be migrated), 95%, and 90%. As can be seen, by not worrying about a small fraction of traffic, we can greatly reduce the number of links that need to be migrated.

V. Related Work

Due to its importance for network performance, there have been much research on traffic engineering. There has

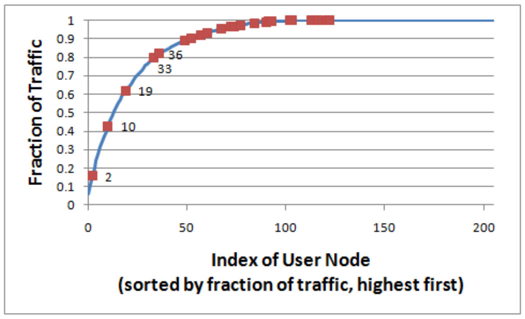


Fig. 7. Fraction of traffic each user node sends in an example 5-minute period.

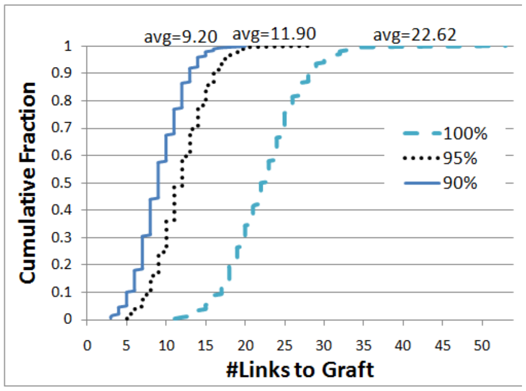


Fig. 8. Cumulative distribution function of the number of links that need to be migrated during each interval over 7 days of traffic (2016 5-minute intervals). Shown are three lines corresponding to different thresholds – only links in the top X% of traffic are migrated.

been much work on schemes for traffic engineering in ISP networks [11] [12] [13] [14] [15]. This work interprets traffic engineering as the adaptation of the routing of traffic within the network so as to optimize performance. We, in contrast, *also* explore how to adapt traffic's ingress and egress points.

User actions can also change the traffic matrix. For example, using overlay routing to circumvent congested links [16], [17] changes the offered load. However, such "selfish" overlay routing can significantly *reduce* the effectiveness of traffic engineering [18].

Previous work on interdomain traffic engineering [3]–[6] considers how to select among a group of fixed egress points for directing traffic to neighboring domains. Interdomain traffic engineering changes the traffic and BGP routes seen in neighboring networks, whereas edge-link migration is transparent. Also, the optimization approaches are different. Interdomain traffic engineering can split traffic over multiple edge links, whereas our "TE with migration" approach must select a *single* edge link for all traffic to and from each user.

Our work builds on earlier research proposing mechanisms for re-homing customers [2] [1]. However, these papers did not explore the implications for traffic engineering.

VI. Conclusions and Future Work

We proposed a new approach to traffic engineering where instead of only optimizing for fixed (predicted) traffic patterns, we also influence where traffic enters and exits the network. We showed that while computing an optimal solution is hard, even relatively simple heuristics can lead to significant performance gains without requiring frequent migration or large numbers of links to migrate. We view our work as a first step in this direction. Incorporating more practical aspects of traffic engineering into our model is a promising direction for future research. Further exploring cluster selection in the cluster user heuristic is also left for future work. Finally, while we focused on edge-link migration in ISP networks, we note that similar capabilities exist in data center networks—namely, virtual machine migration. Exploring our framework in data-center networks, which have different patterns and practical constraints, is also an area for future research.

References

[1] E. Keller, J. Rexford, and J. van der Merwe, "Seamless BGP session migration with router grafting," in *Proc. Networked Systems Design and Implementation*, April 2010.
[2] M. Agrawal, S. Bailey, A. Greenberg, J. Pastor, P. Sebos, S. Seshan, J. van der Merwe, and J. Yates, "RouterFarm: Towards a dynamic, manageable network edge," in *SIGCOMM Workshop on Internet Network Management*, September 2006.
[3] R. Szabó, A. Takács, and A. Császár, "Optimised multi homing – an approach for inter-domain traffic engineering," in *Proceedings of the 2nd International Workshop on Inter-Domain Performance and Simulation (IPS2004)*, Budapest, Hungary, March 2004.
[4] R. Mahajan, D. Wetherall, and T. Anderson, "Negotiation-based routing between neighboring ISPs," in *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation*, Boston, MA, USA, April 2005.
[5] R. Teixeira, T. Griffin, M. G. C. Resende, and J. Rexford, "TIE breaking: Tunable interdomain egress selection," *IEEE/ACM Trans. Networking*, August 2007.
[6] M. Roughan and Y. Zhang, "GATEway: symbiotic inter-domain traffic engineering'," in *The Second International Workshop on Game Theory in Communication Networks*, Athens, Greece, October 2008.
[7] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, 2000.
[8] R. Teixeira, T. Griffin, A. Shaikh, and G. Voelker, "Network sensitivity to hot-potato disruptions," in *Proc. SIGCOMM*, 2003.
[9] D. Moshkovitz and R. Raz, "Two query PCP with sub-constant error," in *Proc. FOCS*, 2008.
[10] "Internet2," http://www.internet2.org.
[11] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "COPE: Traffic engineering in dynamic networks," in *Proc. SIGCOMM*, 2006.
[12] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," in *Proc. SIGCOMM*, 2005.
[13] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS adaptive traffic engineering," in *Proc. IEEE INFOCOM*, 2001.
[14] D. Applegate, L. Breslau, and E. Cohen, "Coping with network failures: Routing strategies for optimal demand oblivious restoration," in *Proc. ACM SIGCMETRICS*, June 2004.
[15] C. Zhang, Z. Ge, J. Kurose, Y. Liu, and D. Towsley, "Optimal routing with multiple traffic matrices: Tradeoff between average case and worst case performance," in *Proc. International Conference on Network Protocols*, Nov. 2005.
[16] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan, "Detour: A case for informed Internet routing and transport," *IEEE Micro*, January 1999.
[17] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. ACM SOSP*, October 2001.
[18] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker, "Selfish routing in Internet-like environments," in *Proc. SIGCOMM*, 2003.