Autonomous Traffic Engineering With Self-Configuring Topologies

Srikanth Sundaresan, Cristian Lumezanu, Nick Feamster
Georgia Tech
{srikanth.sundaresan,lume,feamster}@cc.gatech.edu

Pierre François
Université catholique de Louvain
pierre.francois@uclouvain.be

ABSTRACT

Network operators use traffic engineering (TE) to control the flow of traffic across their networks. Existing TE methods require manual configuration of link weights or tunnels, which is difficult to get right, or prior knowledge of traffic demands and hence may not be robust to link failures or traffic fluctuations. We present a *self-configuring* TE scheme, SculpTE, which automatically adapts the network-layer topology to changing traffic demands. SculpTE is responsive, stable, and achieves excellent load balancing.

1. INTRODUCTION

Network operators use traffic engineering (TE) to manage resource allocation and balance traffic demand across network links. One of the biggest challenges that traffic engineering approaches face is configuring the network topology that establishes the paths between source and destination. Existing approaches rely on setting link weights offline based on long term traffic demand estimations [2] (which is difficult to do accurately in practice and may not be stable), or on establishing tunnels *a priori* and shifting traffic between them in response to fluctuations [6] (which incurs management and configuration overhead, and risk of configuration errors).

We present SculpTE, the first stable traffic engineering approach that is essentially configuration-free. SculpTE balances traffic load across the network by continually adjusting link weights online to expose additional, lightly loaded paths between pairs of endpoints. The link weights in the network topology are derived by periodically sampling the loads on those links; thus, the cost of the link is directly related to the amount of traffic on each link and continually changes as traffic demands change. We have designed these link-weight updates to ensure that SculpTE is both responsive and stable. SculpTE is autonomous, does not require *a priori* topology configuration, and it works well over a wide range of parameter settings, even when nothing is known about traffic demands.

The cost of adjusting topologies online, however, is that the potential for instability is greater than in existing approaches that leave the topology fixed and adjust only how traffic is routed over existing paths. In fact, many schemes that adjust link weights online have encountered stability problems. Nevertheless, adjusting link weights is simple, intuitive, and imposes minimal overhead, so it is worth revisiting this approach. SculpTE offers much promise in this regard: we show that adjusting the topology and letting routing take care of itself can result in a stable system, even in a system that uses destination-based, hop-by-hop forwarding.

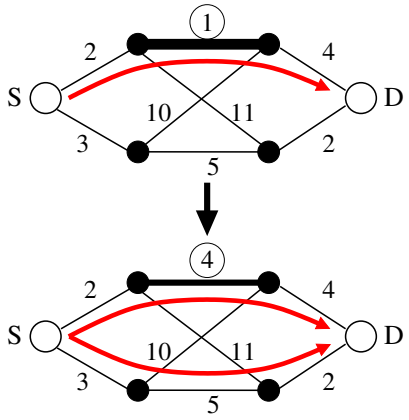

Figure 1: The key idea of SculpTE is to adjust the weight of the most loaded link (thick line) such that additional, lightly loaded paths (red lines) are exposed between a source S and a destination D. Using ECMP routing, SculpTE naturally diverts some of the traffic onto the new shortest path. Labels represent link weights, the cost of a path is the sum of the weights of its links.

2. DESIGN

SculpTE continually balances load across the network by adjusting link weights online to gradually move traffic away from the most loaded link onto alternate, lightly loaded paths. At each iteration, SculpTE identifies the most loaded link, l, and updates its weight with its *key metric* [5]. The key metric of a link for a pair of endpoints is the weight that must be added to the link to remove it from the unique shortest path between the endpoints. We define the *alternate shortest path* for a link l and a pair of endpoints S and D whose traffic traverses l as the shortest path between S and D that does not traverse l. Intuitively, the key metric for l with respect to the SD pair is the difference between the cost of the shortest path and the cost of the alternate shortest path. Adding the key metric to the weight of l increases the cost of the shortest path to the cost of the alternate shortest path. This enables traffic to be distributed evenly across the equal-cost paths with ECMP. In Figure 1, we present a simple example illustrating SculpTE's basic function.

In practice, many pairs of endpoints send traffic through the most loaded link l. Each pair has a different key metric with respect to l. In SculpTE, we must be careful to move just enough traffic to alleviate the load on l, yet preserve stability. To do so, SculpTE applies the minimum key metric of all paths traversing l. By increasing the weight of l with the smallest key metric, only traffic between one pair of endpoints (the pair that yields the smallest key metric) is shifted from the most loaded link. Traffic between all other pairs

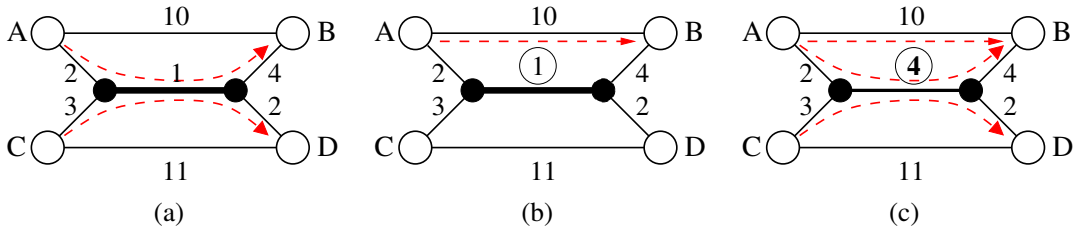

Figure 2: Running SculpTE on a simple topology. (a) We identify the most loaded link l (thick line); AB and BC shortest paths traverse l; (b) We compute the key metric for all pairs of nodes whose shortest path goes through l: $k(A, B, l) = 3$, $k(C, D, l) = 5$; the AB pair yields the minimum key metric (we show the alternate shortest path for AB); (c) We add the value of the minimum key metric to the weight of l; now there are two short paths between A and B; half of the traffic between A and B will move away from l, lowering its utilization. Note that the traffic between C and D does not change; it still traverses l.

is unaffected. In Figure 2, we show how SculpTE chooses the key metric on a simple topology.

Applying the key metric of a link might cause a large amount of traffic to move away from the most congested link. This has the potential to to cause oscillations. To mitigate the effect of destabilizing feedback from link utilization, SculpTE sets up k multiple, independent topologies. Each flow is randomly assigned to one topology. At each iteration, SculpTE updates a single topology. This approach enables SculpTE to (1) prevent reaction to instantaneous traffic patterns, because each topology is updated every k iterations, and (2) reduce the amount of feedback, because flows are divided among k topologies. Prior work by Mitzenmacher [8] shows that having choice helps in achieving good load balancing: multiple topologies increase choice in network paths as different topologies can open up different sets of paths. SculpTE requires that link load and link weight updates are propagated throughout the network once every iteration. This introduces some practical constraints; however, with sub-second IGP weight convergence now possible, this is not expected to be a major problem.

3. PRELIMINARY EVALUATION

We evaluate SculpTE using simulations on six real ISP topologies (the Abilene network and five commercial ISP networks obtained from Rocketfuel [9]). We present results showing the performance of SculpTE compared to standard offline TE algorithms such as IGP-WO [3, 4], proposed by Fortz *et al.*, and InvCap [1]. IGP-WO configures link weights that minimize the maximum utilization given the network graph and an expected traffic matrix. InvCap sets weights inversely proportional to the link capacity. We generate 20 different traffic matrices, using gravity and bimodal models, such that average maximum utilization with optimal routing is between 0.1 and 0.8. This setup captures a wide variety of traffic demands and low as well as high-utilization scenarios. We compute the IGP-WO weights for each traffic matrix using the Totem toolset [7]. Initial weights are set inversely proportional to the capacity of the link, and the frequency of updates is 10 seconds. Figure 3 shows the average deviation from the optimum of SculpTE using three topologies, IGP-WO, and InvCap. The average maximum utilization of SculpTE is much closer to optimum than IGP-WO and InvCap for all networks. The variation in performance obtained across different ASes with the offline schemes is high, in contrast to SculpTE, which performs uniformly well.

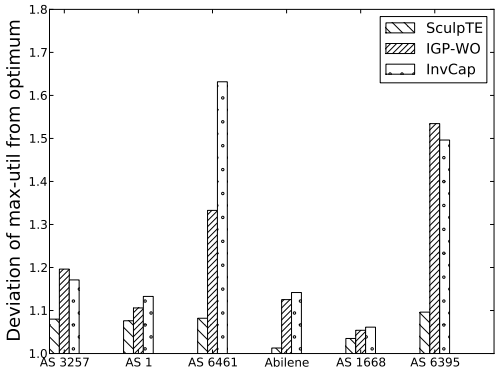

Figure 3: Comparison of various schemes relative to optimal.

REFERENCES

[1] Cisco. Configuring OSPF. http://www.cisco.com/en/US/docs/ios/12_0/np1/configuration/guide/1cospf.html.

[2] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, 2002.

[3] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proc. IEEE INFOCOM*. Tel-Aviv, Israel, 2000.

[4] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE JSAC*, 20(4):756–767, 2002.

[5] P. François, M. Shand, and O. Bonaventure. Disruption-free topology reconfiguration in OSPF networks. In *IEEE Infocom*, 2007.

[6] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *Proc. ACM SIGCOMM*. Philadelphia, PA, 2005.

[7] G. Leduc, *et al.* An open source traffic engineering toolbox. *Computer Communications*, 29(5):593–610, 2006.

[8] M. Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094–1104, 2001.

[9] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *Proc. ACM SIGCOMM*. Pittsburgh, PA, 2002.