Improving Service Differentiation in IP Networks through Dual Topology Routing*

Kin-Wah Kwong, Roch Guérin
University of Pennsylvania
{kkw@seas, guerin@ee}.upenn.edu

Anees Shaikh, Shu Tao
IBM T.J. Watson Research Center
{aashaikh@watson, shutao@us}.ibm.com

ABSTRACT

The convergence on IP of a wide variety of traffic types has strengthened the need for service differentiation. Service differentiation relies on two equally important components: (i) resource allocation, i.e., what resources does a given service class have access to; and (ii) contention resolution, i.e., how is access to shared resources arbitrated between services classes. The latter has been well studied with numerous mechanisms, e.g., scheduling and buffer management, supporting it in modern routers. In contrast, relatively few studies exist on the former, and in particular on the impact of routing that determines the resources a given service class is assigned to. This is the focus of the paper, which seeks to investigate how routing influences a network's ability to efficiently support different service classes. Of particular interest is the extent to which the ability to route service classes *separately* is beneficial. This question is explored for a base configuration involving two classes with either similar or entirely different service objectives (cost functions). The paper's contributions are in demonstrating and quantifying the benefits that the added flexibility of different (dual) routing affords, and in developing an efficient heuristic for computing jointly optimal routing solutions. The former can motivate the deployment of newly standardized multi-topology routing (MTR) functionality. The latter is a key enabler for the effective use of such capability.

1. INTRODUCTION

IP networks today carry traffic of different types, characteristics, and performance requirements. For instance, many ISP's are using IP networks to deliver bundled services that include performance-sensitive applications, such as voice and video, as well as the more "elastic" data services. Large enterprises also rely heavily on IP networks to transfer a mix of both critical (e.g., data center backup) and non-critical data for their business needs. To better support such mixed traffic, service differentiation is often necessary or at least desirable.

Service differentiation is widely available in modern routers, e.g., in the form of scheduling and buffer management mechanisms that arbitrate between service classes in the presence of resource contention. Although "contention resolution" is a critical component in support of service differentiation, it does little to ensure that a network delivers the best possible services to the different service classes it supports, i.e., minimizes the onset of contention whenever possible. In IP networks, this is largely controlled by routing that determines how network resources (link bandwidth) are assigned to carry different traffic flows. In order to optimize routing for the expected traffic demand, network operators often employ traffic engineering (TE) algorithms to determine the paths, which are then implemented using shortest-path routing protocols, such as OSPF or IS-IS.

We investigate the performance benefits of different routing schemes using this traditional TE framework, but with two traffic classes (i.e., high and low priority). We assume a simple contention resolution mechanism—priority queueing (i.e., a separate queue is maintained on each link for each service class, and higher priority classes are always served first). When using common TE approaches in a priority queueing environment, it is straightforward to optimize performance of the high-priority traffic. However, as we show, such optimized performance often comes at the expense of low-priority traffic, whose performance may be severely degraded. Ideally, we seek a routing scheme that ensures the best performance for high-priority traffic, while still maintaining reasonable performance for low-priority traffic.

In this paper, we develop a practical routing scheme that achieves the goal of improving support for service differentiation in IP networks. Our approach is based on multi-topology routing (MTR), in which different

routings are computed for different service classes, allowing greater flexibility in exploiting available network resources to meet their service goals. In traditional IP networks, a single weight is assigned to each link, which produces a *single* routing that is used to forward packets of all traffic classes. We refer to this traditional routing scheme as single-topology routing (STR). In contrast, MTR allows multiple weights on each link, and hence a corresponding number of different routings that can each be assigned to separate service classes. MTR is being standardized [1] and is becoming available on a number of router platforms. In our investigation, we limit ourselves to two topologies, i.e., two distinct weights on each link, and denote the resulting routing as dual-topology routing (DTR). In DTR, high-priority traffic is routed using one set of link weights, and low-priority traffic routed using the other.

While the increased routing flexibility of DTR is clearly beneficial when compared to STR, those benefits come at a cost. The first is the higher complexity of identifying good sets of link weights. This is because the solution space to explore is exponentially larger than with STR (each assignment of link weights for one traffic class needs to be evaluated for all possible combinations of links weights for the other traffic class). Furthermore, the objective functions that need to be optimized by selecting link weights are often different for each traffic class. For example, two commonly used objective functions include one that is based on link utilization and seeks to minimize the average overall network delay, and a second that involves Service Level Agreements (SLAs) in the form of delay bounds that need to be met for each "user" (pairs of source and destination nodes). Computing weight settings that optimize various combinations of such objective functions can be challenging. As a result, realizing any of the benefits of DTR calls for developing computationally tractable approaches for finding good combinations of link weights across these many different scenarios. Another cost of DTR over STR is in the added configuration and computational overhead it imposes on routers, because of the need to configure and disseminate multiple weights for each link and run multiple SPF algorithms in the presence of network changes.

Given the costs associated with DTR, it is vital to both provide techniques to mitigate them when possible, and equally important to quantify the magnitude of the improvements achievable with DTR. The paper makes contributions toward both of these issues, first by developing an efficient heuristic for computing good weight settings for DTR in the presence of various objective functions, and second by performing a comprehensive assessment of the performance benefits that DTR can afford. Through extensive simulations, we show that DTR is able to provide substantial perfor-

mance improvements, especially for low-priority traffic, when compared with STR. These benefits are consistent across a number of topologies, network sizes, traffic patterns, and objective functions in the optimization formulation.

The rest of this paper is structured as follows. Section 2 provides some background on multi-topology routing and reviews a few related works. Section 3 formulates the problem and introduces the two objective functions considered in our problem setting. Section 4 describes the heuristic algorithm designed for link weight computation in DTR. Section 5 applies this algorithm to a wide set of network topologies and traffic patterns, and provides an in-depth discussion of when and why DTR can offer substantial performance improvements above and beyond STR-based solutions. Section 6 summarizes our findings and concludes the paper.

2. RELATED WORK

The problem of finding optimal link weight settings in IP networks that rely on destination-based, SPF routing was first studied in [2], where the problem was identified as NP-hard and a local search heuristic was proposed. Several other heuristics, e.g., [3, 4], were later developed as extensions to this work. Nucci *et al.* [5] studied a similar problem, and their goal was to compute a link weight setting that both met SLA requirements and was robust to link failures. All these works consider routing of only a single traffic class, i.e., an STR setting.

The use of MTR for traffic engineering purposes was proposed in [6] based on dividing traffic matrix into smaller "slices", each routed on a separate topology—the greater the number of slices, the better the performance as it increases the ability to approximate optimal routing. Recently, the use of MTR has been proposed to improve the resiliency of IP routing [7, 8, 9], with different topologies offering backup routes for different failure scenarios. Our work differs from these earlier contributions in that we focus on assessing MTR's benefits in a network that offers service differentiation.

Also related to our work are studies on QoS routing for optimizing multi-class traffic performance, e.g., [10, 11, 12, 13, 14, 15]. However, even though they consider the problem of optimizing network performance across multiple traffic classes and performance objectives, these works focus on a flow-oriented environment where the goal is to minimize the blocking probability of high-priority requests [14, 15]. In contrast, in this paper we assume the standard destination/SPF based routing/forwarding paradigm of traditional IP networks. Similarly, works on multi-class traffic engineering in MPLS [16, 17] have also tackled the problem of network optimization in the presence of different traffic classes. However, they again differ from the work of this paper because of the flow-oriented nature of MPLS

networks.

3. PROBLEM FORMULATION

We model the network as a directed graph $G = (V, E)$ with node set V and edge set E, and C_{ij} denotes the capacity of edge $(i, j) \in E$. The traffic from the high- and low-priority classes are specified through two traffic matrices, $T_H = [r_H(s, t)]_{|V| \times |V|}$ and $T_L = [r_L(s, t)]_{|V| \times |V|}$, respectively, where $r_H(s, t)$ and $r_L(s, t)$ represent the volume of high- and low-priority traffic originating at node s and destined for node t, with $r_H(s, s) = r_L(s, s) = 0$, $\forall s \in V$.

As mentioned earlier, the contention resolution mechanism used to differentiate between service classes is a simple two-priority queueing scheme. The high-priority queue is always served first, hence the low-priority traffic only sees the residual capacity left by the high-priority traffic. For example, if link l with capacity C_l is carrying H_l units of high-priority traffic, the residual capacity seen by the low-priority traffic is $\widetilde{C}_l = \max(C_l - H_l, 0)$. When computing "good" weight settings, we focus on two cost/objective functions that are commonly used in practice in IP networks.

3.1 Load-based cost function

We first consider a load-based cost function that seeks to minimize the overall queueing delay each traffic class experiences in a network. Our model is a generalized version of that of [2]. Specifically, we assume that the performance of both traffic classes on a link can be modeled by an M/M/1 model. Using the piecewise linear approximation suggested in [2], we define the cost incurred by the high-priority traffic on link l as

$$
\Phi_{H,l} = \begin{cases}
H_l, & \frac{H_l}{C_l} \leqslant \frac{1}{3} & \text{(1a)} \\
3H_l - 2/3C_l, & \frac{1}{3} \leqslant \frac{H_l}{C_l} \leqslant \frac{2}{3} & \text{(1b)} \\
10H_l - 16/3C_l, & \frac{2}{3} \leqslant \frac{H_l}{C_l} \leqslant \frac{9}{10} & \text{(1c)} \\
70H_l - 178/3C_l, & \frac{9}{10} \leqslant \frac{H_l}{C_l} \leqslant 1 & \text{(1d)} \\
500H_l - 1468/3C_l, & 1 \leqslant \frac{H_l}{C_l} \leqslant \frac{11}{10} & \text{(1e)} \\
5000H_l - 16318/3C_l, & \frac{11}{10} \leqslant \frac{H_l}{C_l} & \text{(1f)}
\end{cases}
$$

where H_l denotes the amount of high-priority traffic on link l. Similarly, the cost incurred by the low-priority traffic on link l, $\Phi_{L,l}$, can be defined by replacing H_l with the amount of low-priority traffic on link l, L_l, and C_l with the residual capacity on link l seen by the low-priority traffic, $\widetilde{C}_l$.

Based on the above definitions, we define the overall cost for the two traffic classes, denoted by Φ_H and Φ_L, as the sum of all link costs, i.e., $\Phi_H := \sum_{l \in E} \Phi_{H,l}$ and $\Phi_L := \sum_{l \in E} \Phi_{L,l}$. Our goal is to minimize Φ_H and Φ_L, while preserving the priority precedence between the traffic classes. We, therefore, define a lexicographical optimization function such that the high-priority traffic

is optimized first. Specifically, our goal is to

$$
\text{minimize} \quad \mathcal{A} = \langle \Phi_H, \Phi_L \rangle \tag{2}
$$

Note that we use the notation $\langle \cdot, \cdot \rangle$ to denote a lexicographically ordered tuple. Namely, $\langle x_1, y_1 \rangle > \langle x_2, y_2 \rangle$, if and only if $x_1 > x_2$, or $x_1 = x_2$ and $y_1 > y_2$. In other words, we optimize first for the high-priority traffic and then for the low-priority traffic.

3.2 SLA-based cost function

Another commonly used cost function is based on Service Level Agreements (SLA's) [5]. SLA's are typically defined in the form of end-to-end delay targets for each source-destination (SD) pair. SLA-based cost functions are particularly meaningful for high-priority traffic, as customers of this service class pay a premium for their service. Typically, whenever an SLA is violated, a provider pays a certain penalty back to its customers. There is, therefore, a strong incentive on the provider's part to select a routing that minimizes the chance of such occurrences.

The second cost function we consider reflects such an objective. Specifically, its goal is to ensure that delay performance targets are met for all the high-priority traffic SD pairs. Delay consists of both queueing and propagation delays contributed by all links on the path used by a given SD pair. Specifically, the average delay D_l (seen by high-priority traffic) on link l can be computed as

$$
D_l = \frac{s}{C_l} \left(\frac{H_l}{C_l - H_l} + 1 \right) + p_l \approx \frac{s}{C_l} \left(\frac{\Phi_{H,l}}{C_l} + 1 \right) + p_l \tag{3}
$$

where s is the average packet size, p_l is the propagation delay of link l. Note that we use $\Phi_{H,l}/C_l$ to approximate $H_l/(C_l - H_l)$ [18].

Let $\xi_{(s,t)}$ be the average end-to-end delay for SD pair (s, t). Suppose the SLA delay bound of any SD pair is $\theta > 0$. We define the cost function for the high-priority traffic between SD pair (s, t) as

$$
\Lambda_{(s,t)} = \begin{cases}
0, & \text{if } \xi_{(s,t)} \leqslant \theta & \text{(4a)} \\
a + b(\xi_{(s,t)} - \theta), & \text{Otherwise} & \text{(4b)}
\end{cases}
$$

Here, a and b are two positive parameters that determine the SLA penalty; a is a constant penalty incurred by any SLA violation, while b represents the penalty ratio associated with delay in excess of the SLA bound. Without loss of generality, we choose $a = 100$ and $b = 1$. The overall cost function for the high-priority traffic is then defined as $\Lambda := \sum_{(s,t) \in V \times V} \Lambda_{(s,t)}$. In other words, Λ can be considered as the total amount of penalty that the provider pays back to customers because of SLA violations.

For the low priority traffic, we continue to use the load-based cost function Φ_L defined in Section 3.1. In

keeping with the lexicographical ordering that gives precedence to the high-priority traffic cost function, our goal is now to

$$\text{minimize} \quad \mathcal{S} = \langle \Lambda, \Phi_L \rangle \qquad (5)$$

3.3 Discussions

3.3.1 On the feasibility of a joint cost function

A natural question when considering an optimization problem consisting of multiple cost functions, i.e., for different service classes, is to explore whether a unique, joint cost function can be defined to facilitate the selection of a routing solution[1] that accurately captures the goals of the original individual cost functions. One such possibility is a joint cost function of the form $J := \alpha \Phi_H + \Phi_L$ or $J := \alpha \Lambda + \Phi_L$, where by varying $\alpha > 0$ a different trade-off can be realized between the two service classes. When setting $\alpha \to \infty$, J defaults back to the earlier lexicographical cost function. A similar joint cost function was proposed in [19], for a configuration where the two objectives being combined were performance and reliability instead of the performance of different service classes as is the case here.

In spite of the attractiveness of such an approach, especially in terms of computational simplicity[2], identifying how to select α in order to realize a given performance trade-off for high- and low-priority traffic is challenging. Setting α to too low a value can result in a "priority inversion" in the sense that some high-priority traffic can see relatively poor performance compared to what low-priority traffic sees overall. This violates the concept of "high priority" and the associated ordering among traffic classes. Conversely, choosing too large a value for α yields little benefits compared to a lexicographical ordering solution, and in general identifying a value of α that works well across *all* configurations is a challenging task. This is especially so when dealing with different cost functions across service classes, as identifying how to "add" quantities that are expressed in different units is at best un-intuitive. We briefly illustrate these difficulties next through a simple example.

Consider a 3-node triangular network (A, B and C), as shown in Fig. 1. The capacity of each link is 1 unit, and 1/3 and 2/3 units of high- and low-priority traffic, respectively, originate at node A destined for node C. Consider $J := \alpha \Phi_H + \Phi_L$, where $\alpha = 35$. Under STR, both traffic classes then use link $A - C$ to min-

---

[1]Note that the use of a joint cost function only makes sense in a setting where STR is used. If DTR is available, because each class can be routed independently, there is no benefit in coupling their routing through a joint cost function.

[2]A purely lexicographical optimization calls for identifying "all" routings that minimize the first cost function, and then select the one that yields the best value for the second. With a joint cost function, a single optimization is instead performed.
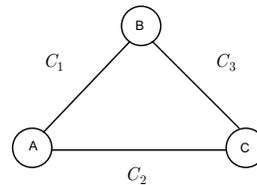

Figure 1: A simple 3-node network

imize J, which results in $\Phi_H = 1/3$ and $\Phi_L = 64/9$. This solution is the same as lexicographically minimizing $\mathcal{A} = \langle \Phi_H, \Phi_L \rangle$ and maintains the priority precedence among the two traffic classes. However, the low-priority traffic performance is very poor, so that one may want to improve it by lowering α to, say, $\alpha = 30$. The solution that minimizes J then calls for evenly splitting both high- and low-priority traffic over links $A - B$ and $A - C$, which yields $\Phi_H = 1/2$ and $\Phi_L = 4/3$. This improves Φ_L by 81%, but also degrades Φ_H by 50%, which results in a "priority inversion" that violates the implicit precedence given to high priority traffic. This illustrates how difficult it is to select a value of α such that both traffic classes achieve good performance, and matters are even worse when considering more complex topologies and different combinations of traffic loads and patterns.

3.3.2 STR with trade-off between traffic classes

Our performance goal is formulated as a lexicographical optimization, which gives strict precedence to the performance of high-priority traffic. With DTR the impact of this strict precedence is "softened" by giving low-priority traffic control on its own routing once the high-priority routing has been set. A natural question is then whether similar benefits are achievable with STR by slightly relaxing the precedence rule.

For load-based cost functions, this can be realized as follows: Denote the minimum cost for high-priority traffic as Φ_H^*; one may then allow STR solutions that satisfy $\Phi_H \leq (1 + \varepsilon)\Phi_H^*, \varepsilon > 0$. In other words, we allow all routing solutions that do not degrade the performance of high-priority traffic by more than ε. This increases the number of routing solutions that can be considered, hence improving the odds of finding one that yields better performance for low-priority traffic. The parameter ε, allows us to tune how much high-priority performance we are willing to sacrifice to improve low-priority performance. For an SLA-based cost function, a similar trade-off can be realized simply by increasing the high-priority delay bound to $(1+\varepsilon)\theta$. As ε grows, more STR routing solutions satisfy the SLA; hence again increasing the odds of finding one that yields better performance for low-priority traffic.

As we shall see, while such relaxation can help improve the overall performance of STR, it does not come

anywhere near what is achievable through DTR. We expand on this in Section 5.3.

4. HEURISTIC SEARCH FOR DTR LINK WEIGHT ASSIGNMENT

For most cost functions, including the two we selected, finding an optimal link weight setting in STR is an NP-hard problem [2]. As alluded to earlier, the problem is even more complex with DTR since in each possible weight configuration for the high-priority traffic, there are exponentially many possible weight settings for the low-priority traffic. To address the problem of computing good sets of link weights for DTR, we develop the following heuristic algorithm.

Let $W = \{W^H, W^L\}$ be the set of link weights for DTR, where $W^H = \{W_l^H\}_{l \in E}$ and $W^L = \{W_l^L\}_{l \in E}$ denote the sets of link weights for the high- and low-priority traffic, respectively. Our goal is to find W that minimizes our lexicographical objective function $\mathcal{L}$, where $\mathcal{L} = \mathcal{A}$ (Eq. (2)) or $\mathcal{L} = \mathcal{S}$ (Eq. (5)). The heuristic is an iterative algorithm. As shown in Algorithm 1, it consists of three routines: The first minimizes the high-priority traffic cost; the second minimizes the low-priority traffic cost, and the third refines the solution obtained from the first two. Details on each routine are provided next.

In the first routine (from lines 3 to 12), the high-priority traffic performance is optimized by searching for a good weight setting W^H (see subroutine `FindH`) given a set of initial link weights for the low-priority traffic, i.e., W^L remains unchanged as W^H is optimized. In the second routine (from lines 15 to 24), W^H is set to the best setting W^{H*} obtained in the first routine, and the low-priority link weights are optimized. After completion of the first two routines, a reasonably good link weight setting $W^* = \{W^{H*}, W^{L*}\}$ has been identified. However, additional improvements are possible, and the third routine (from lines 27 to 38) is used as a refinement step that searches for better overall solutions in a small neighborhood around $W^* = \{W^{H*}, W^{L*}\}$.

In Algorithm 1, diversification is also used to let the search "escape" from a local optimum if there is no cost improvement in M iterations. A small fraction (g_1 and g_2) of weights in W^H and W^L are randomly perturbed in the first and second routines respectively (in lines 9 and 21). In the third routine, the diversification procedure (from lines 33 to 36) randomly perturbs a smaller fraction (g_3) of link weights in both W^H and W^L, so that the search re-starts from a solution sightly different from W^*, while preventing it from jumping to a very sub-optimal one.

The `FindH` subroutine is shown in Algorithm 2. The subroutine first defines a "neighborhood" for the current solution W, and then evaluates solutions in this neighborhood to find the best one to replace W. A neighborhood is defined as follows. With the current solution W, each link l is associated with a lexicographical cost $\mathcal{L}_l$, where $\mathcal{L}_l := \langle \Phi_{H,l}, \Phi_{L,l} \rangle$ (resp. $\mathcal{L}_l := \langle D_l, \Phi_{L,l} \rangle$) if $\mathcal{L} = \mathcal{A}$ (resp. $\mathcal{L} = \mathcal{S}$). Links are then sorted in decreasing order of cost. Intuitively, we want to increase the weight of links with high cost to route traffic away from them, and decrease the weight of links with low cost to attract more traffic onto them [5]. In our heuristic, we select m high-cost links and m low-cost links to form two sets, A and B. A neighbor of the current solution W is identified by randomly selecting a link from A (resp. B) and increasing (resp. decreasing) its weight, both without replacement. As a result, m such neighbors can be identified to form the neighborhood of W. Note that it is not always good to change the weights of the links with the highest or lowest cost, because this may lead to situations where only a few links are perturbed, so that only a small part of the solution space is explored. To avoid this problem, we introduce one more randomization for link selection in `FindH` (as shown in lines 2 to 4 in Algorithm 2). We define two integers k_1 and k_2 randomly drawn from a heavy-tail probability distribution $P(\bar{k}) \propto \bar{k}^{-\tau}$, where $1 \leqslant \bar{k} \leqslant n - m + 1$ and n is the total number of links in the network [20]. Based on this distribution, if $\tau \to 0$, links in A and B are selected independent of their costs, while if $\tau \to \infty$, only those links with the highest or lowest costs are included in A and B. In our study, we use $\tau = 1.5$ so that all links have a chance of being chosen, while keeping some preference for selecting links with very high or low costs.

The routine `FindL` is similar to `FindH` except that we use $\Phi_{L,l}$ for sorting links, because W^L has no effect on the high-priority traffic. For brevity, we omit the pseudo-code for `FindL`.

5. EVALUATION AND FINDINGS

In this section, we evaluate the performance benefits of DTR over STR solutions. Our evaluation spans a broad range of network settings, including different topologies, traffic patterns and intensity levels. Our goal is to investigate when DTR is worth deploying (given its added complexity), and in particular whether the use of different performance objectives for the two traffic classes plays a major role.

5.1 Evaluation settings

5.1.1 Network topology

We use three different types of topologies:

- Random topology: generated by randomly adding links between nodes in a network, all nodes have similar link degrees.

- Power-law topology: generated using the preferen-

Algorithm 1: DTR Weight Search

Input: Network topology, traffic matrix, initial link weight setting W_0
Result: Best link weight setting W^*
1   $W \leftarrow W_0$, $W^* \leftarrow W_0$
2   $\mathcal{L}^* \leftarrow \mathcal{L}(W_0)$; Iteration $\leftarrow 0$
3   **while** *Iteration<N* **do**
4     $W \leftarrow \text{FindH}()$
5   **if** $\mathcal{L}(W) < \mathcal{L}^*$ **then**
6       $\mathcal{L}^* \leftarrow \mathcal{L}(W)$, $W^* \leftarrow W$
7   **end**
8   **if** *No improvement in $\mathcal{L}^*$ after M iterations* **then**
9   Randomly perturb g_1 (%) of link weights in W^H
10   **end**
11   Iteration++
12 **end**
13 Iteration $\leftarrow 0$
14 $W^H \leftarrow W^{H*}$
15 **while** *Iteration<N* **do**
16    $W \leftarrow \text{FindL}()$
17   **if** $\Phi_L(W) < \Phi_L^*$ **then**
18      $\Phi_L^* \leftarrow \Phi_L(W)$, $W^* \leftarrow W$
19   **end**
20   **if** *No improvement in Φ_L^* after M iterations* **then**
21   Randomly perturb g_2 (%) of link weights in W^L
22   **end**
23   Iteration++
24 **end**
25 $W \leftarrow W^*$
26 Iteration $\leftarrow 0$
27 **while** *Iteration<K* **do**
28    $W \leftarrow \text{FindH}()$
29    $W \leftarrow \text{FindL}()$
30   **if** $\mathcal{L}(W) < \mathcal{L}^*$ **then**
31      $\mathcal{L}^* \leftarrow \mathcal{L}(W)$, $W^* \leftarrow W$
32   **end**
33   **if** *No improvement in $\mathcal{L}^*$ after M iterations* **then**
34      $W \leftarrow W^*$
35   Randomly perturb g_3 (%) of link weights in both W^H and W^L
36   **end**
37   Iteration++
38 **end**

Algorithm 2: FindH()

Input: Current link weight setting W
1   Sort the links in the decreasing order of their lexicographical link costs, i.e. $\mathcal{L}_{\Pi(1)} \geqslant \mathcal{L}_{\Pi(2)} \geqslant \ldots \geqslant \mathcal{L}_{\Pi(n)}$, where $\Pi(\cdot)$ means permutation.
2   Draw two random integers, k_1 and k_2, from the distribution $P(\overline{k})$ where $1 \leqslant \overline{k} \leqslant n - m + 1$.
3   Select m links with ranks $\Pi(k_1), \Pi(k_1 + 1), \ldots, \Pi(k_1 + m - 1)$ to form a set A.
4   Select m links with ranks $\Pi(n + 1 - k_2), \Pi(n - k_2), \ldots, \Pi(n - k_2 - m + 2)$ to form a set B.
5   A neighbor is defined as randomly selecting a link l from A and a link l' from B, both without replacement. Then we increase W_l^H and decrease $W_{l'}^H$. Construct m neighbors as the above manner.
6   Find the best neighbor, denoted by $\overline{W}$, from the neighborhood.
7   **if** $\mathcal{L}(\overline{W}) < \mathcal{L}(W)$ **then**
8   return $\overline{W}$.
9   **else**
10   return W.
11 **end**

5.1.2 Traffic matrix

The traffic matrices $T_H = [r_H(s,t)]_{|V| \times |V|}$ and $T_L = [r_L(s,t)]_{|V| \times |V|}$, which correspond to the high- and low-priority traffic classes, are specified as follows.

The low-priority traffic matrix T_L is generated using a gravity model [23, 24, 5]. Specifically, the traffic volume from node s to node t is defined as

$$r_L(s,t) = d_s \frac{e^{V_t}}{\sum_{i \in V \setminus \{s\}} e^{V_i}} \qquad (6)$$

where d_s is the total traffic originating at node s, given by

$$d_s = \begin{cases} \text{Uniform}(10, 50), & \text{with prob } 0.6 \quad (7a) \\ \text{Uniform}(80, 130), & \text{with prob } 0.35 \quad (7b) \\ \text{Uniform}(150, 200), & \text{with prob } 0.05 \quad (7c) \end{cases}$$

Here, Uniform(a,b) denotes a random variable uniformly distributed in $[a, b]$, V_t is a random variable uniformly distributed in $[1, 1.5]$, and can be considered the "mass" of node t. The larger a node's mass, the more traffic it attracts. Using d_s, we generate a heterogeneous traffic demand model with three different levels: (1) low traffic volume originating from 60% of the nodes, (2) medium traffic volume originating from 35% of the nodes, and (3) high traffic volume originating from 5% of the nodes to emulate "hot spots" in the network.

We consider two different models for high-priority traffic. The first is a *random* model, in which we randomly select a fraction (k) of SD pairs to generate high-priority traffic. In other words, k represents the density of high-priority SD pairs. The second is a *sink* model that emulates "popular" servers, e.g,. data centers that are destinations for a large number of client nodes. In

tial attachment model [21] to emulate the power-law degree distribution observed in the Internet topology [21, 22].

- ISP topology: emulating a North American backbone network consisting of 16 nodes and 70 links.

In our study, all link capacities are set equal to 500 Mbps. For the evaluation of the SLA-based cost function, we also need to assign propagation delays to links. For synthesized (i.e., random and power-law) topologies, the propagation delay of each link is randomly selected in the range of 1.2ms and 15ms. This ensures that the simulated links range from very short links to long-haul (coast-to-coast) links. For the ISP topology, we assign a propagation delay between 8ms and 15ms to each link, based on the geographical locations of the corresponding nodes. Unless specified otherwise, the SLA delay bound is set to $\theta = 25$ms.

our simulation, a small number of sinks were selected among nodes with the highest degree. Then, a larger number of client nodes are selected, and bi-directional traffic is assumed between sinks and client nodes.

The volume of high-priority traffic is specified in proportion to the total network traffic volume. Specifically, $0 < f < 1$ denotes the fraction of the total traffic volume that corresponds to high-priority traffic. In other words, if the high- and low-priority traffic volumes are η_H and η_L respectively, $f = \eta_H/(\eta_H + \eta_L)$. The high-priority traffic volume between nodes s and t is then generated as follows. We first assign a uniformly distributed random variable, $m_{(s,t)} \in [1, 4]$, to SD pair (s, t). Given that the total volume of low-priority traffic is $\eta_L = \sum_{s,t \in V} r_L(s, t)$, the volume of high-priority traffic between SD pair (s, t) is set as $r_H(s, t) = \eta_L \frac{f}{1-f} \frac{m_{(s,t)}}{\sum_{i,j \in V} m_{(i,j)}}$. This not only ensures that the total high-priority traffic is a fraction f of the total network traffic, but also generates a heterogeneous traffic volume among different high-priority SD pairs.

5.1.3 Heuristic algorithm settings

In our heuristic algorithm, we define link weights to be between 1 and 30. The relatively small maximum link weight is selected as a trade-off between the effectiveness of the resulting routing solutions and computational complexity.

Algorithm 1 uses two variables N and K to control the number of iterations of each routine. In this study, we select $N = 300000$ and $K = 800000$. In each iteration, the heuristic evaluates $m = 5$ neighbors. We also choose $g_1 = g_2 = 5\%$ and $g_3 = 3\%$ for the diversification phases of the three routines of Algorithm 1; g_3 is smaller because this routine is computationally more complex. The diversification interval is set to $M = 300$. Extensive investigations [25] have shown that these are sufficient to generate good link weight settings for the topologies considered in this paper.

To compare the performance of DTR with that of STR, we adopt the "single weight change" heuristic proposed in [2] to generate STR link weight settings to minimize the objective functions of Eqs. (2) and (5).

5.2 Performance benefit of DTR

In this section, we evaluate the performance benefits of DTR over STR. The focus is not only on quantifying the benefits of DTR, but also on identifying the scenarios where the difference is more significant. Because of space limitations, we only include a representative set of results. Results for many other configurations are available in [25], and are consistent with the subset reported here.

A general and expected observation is that across topologies, the performance of high-priority traffic under DTR and STR is comparable, while the perfor-

mance of low-priority traffic is significantly better under DTR. This holds for both cost functions of Section 3. This can be explained as follows: For either load-based or SLA-based cost functions, the combination of lexicographical optimization and priority queueing ensures that under both STR and DTR the performance of high-priority traffic is optimized first. Furthermore, it also ensures that the high-priority traffic is impervious to the behavior of low-priority traffic, whether routed identically or differently. In contrast, because STR routes the low-priority traffic over the same set of links as the high-priority traffic, it tends to experience higher link loads, hence higher cost. DTR gives it the flexibility to be routed away from overloaded links, hence lowering its cost. To illustrate this effect, we plot in Fig. 3 the distribution of link utilization (with both high- and low-priority traffic) in a random topology for the load- and SLA-based cost functions respectively[3]. As shown in the figure, DTR yields significantly fewer overloaded links than STR does.

To further demonstrate the benefits of DTR, Fig. 2 shows for three different topologies the cost ratios R_H and R_L (computed as the cost under STR over the cost under DTR) for high- and low-priority traffic, using load- and SLA-based cost functions, respectively. For all three topologies, the total traffic demand (represented by the average link utilization[4]) is varied by scaling the traffic matrix. As can be seen from the figures, R_H is approximately equal to 1, while R_L can be relatively large.

In our simulations, we also find that the benefits of DTR can be affected by several factors, including network load and traffic patterns. Some of our major findings are summarized as follows[5]:

- DTR outperforms STR the most when the network is moderately loaded. The difference decreases when network load is either light or heavy;

- The performance benefits of DTR are more significant when the network carries more high-priority traffic;

- When the density of high-priority SD pairs increases, we observe opposite behaviors for the cost functions under consideration: the benefits of DTR decrease for load-based cost function, but increase under SLA-based cost function;

- High-priority traffic patterns also affect the performance of DTR. When high-priority traffic is concentrated "locally" on a small set of nodes, the performance benefits of DTR are less pronounced.

---

[3]Fig. 3 is a "stacked" bar chart so that the DTR bars are drawn on top of the STR bars and do not overlap.
[4]Because the average link utilization is roughly equal under DTR and STR, we use it as a reference of network load.
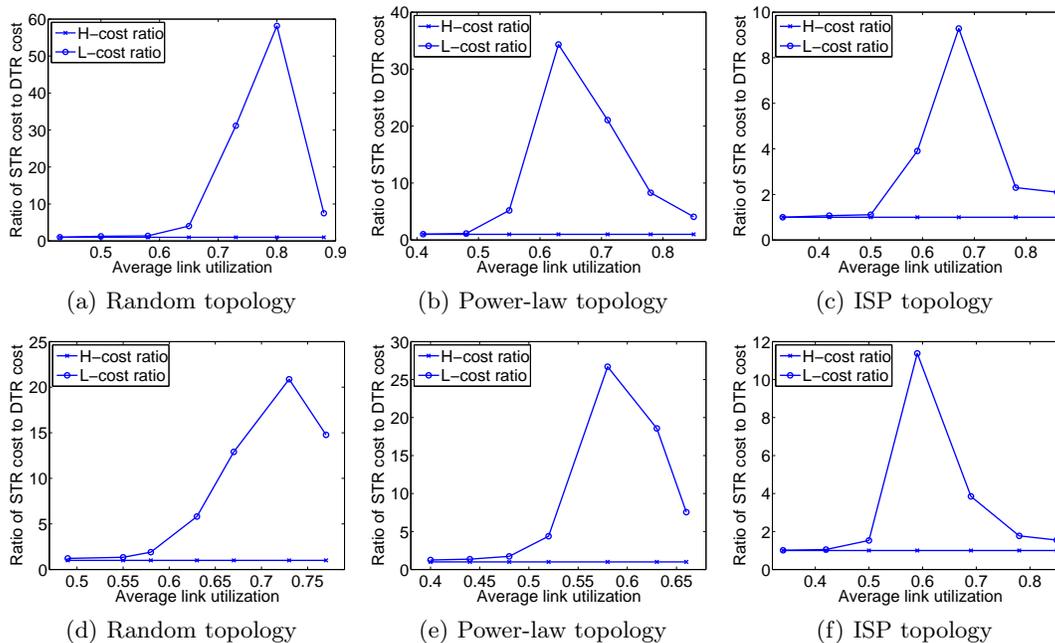[5]Since we always have $R_H \approx 1$, we focus only on R_L.

Figure 2: Cost ratios between STR and DTR when the load-based cost function (a-c) or the SLA-based cost function (d-f) is used: R_L **(denoted as "L-cost ratio") and** R_H **(denoted as "H-cost ratio").** **30% of the traffic is of high-priority (** $f = 30\%$ **); 10% high-priority SD-pair density (** $k = 10\%$ **). A 30-node, 150 link random topology is studied in (a) and (d), a 30-node, 162-link power-law topology is used in (b) and (e), and the ISP topology is used in (c) and (f).**

We expand on these observations next.

5.2.1 Impact of network load

We first study how R_L changes with network load. From Fig. 2, we observe that for both load- and SLA-based cost functions, R_L follows an "increasing-and-decreasing" pattern when network load (represented by the average link utilization) increases. This observation can be explained as follows.

When the overall network load is low, the volume of high-priority traffic is proportionally low (note that $f = 30\%$ and $k = 10\%$ throughout the simulations in Fig. 2). Therefore, although STR routes high- and low-priority traffic on the same set of paths, the residual capacity seen by the low-priority traffic on these paths is relatively large. Hence, the cost for low-priority traffic (Φ_L), which is determined by the residual capacity (see Eq. (1)), is low. In such cases, the difference between STR and DTR is marginal. When network load increases, the volume of high-priority traffic proportionally increases. With STR, the residual capacity seen by the low-priority traffic decreases, hence Φ_L increases. But with DTR, the low-priority traffic can be routed away from the links with low residual capacity, hence allowing Φ_L not to increase as much. As the network load keeps increasing, most links become eventually heavily loaded, so no matter how low-priority traffic is routed,

it experiences a high cost, which limits the potential for improvement of DTR and eventually results in a decrease of R_L.

Next, we investigate the effect on R_L of varying the fraction of network load contributed by the high-priority traffic. Specifically, we conduct simulations similar to those discussed above, but change f from 20% to 40%, while keeping $k = 10\%$. Fig. 4 shows the results for a 30-node, 150-link random topology, when the load-based cost function is used. As the figure shows, the cost ratio R_L becomes higher when f increases from 20% to 40%. This can be understood as follows. As f increases, the links on the shortest paths (for both STR and DTR) become more loaded with high-priority traffic. Since under STR the low-priority traffic is routed over the same links, its performance degrades. In contrast, DTR allows low-priority traffic to be routed away from those highly loaded links, hence avoiding most of the performance degradations associated with the increase of f. As a result, the benefits of DTR are more pronounced (R_L becomes larger) for $f = 40\%$ than for $f = 20\%$. For the same reasons, the above observation also holds for the SLA-based cost function.

5.2.2 Impact of high-priority SD-pair density

We investigate next how R_L varies when the density of high-priority SD pairs (i.e., k) changes. Specifically,
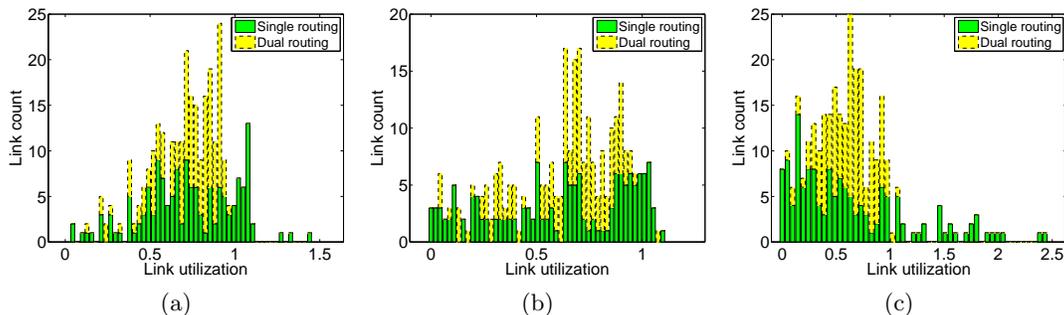
Figure 3: Link utilization comparison between STR and DTR in a 30-node, 150-link random topology. 30% of the traffic is of high-priority ($f = 30\%$). (a) 10% high-priority SD pairs ($k = 10\%$), load-based cost function. (b) 10% high-priority SD pairs ($k = 10\%$), SLA-based cost function. (c) 30% high-priority SD pairs ($k = 30\%$), SLA-based cost function.
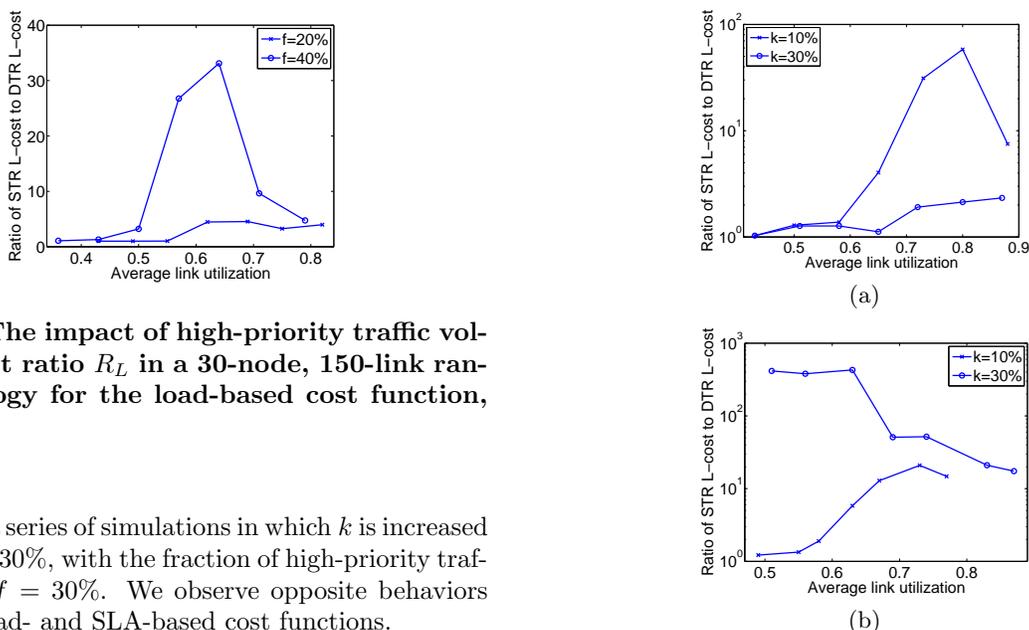

Figure 4: The impact of high-priority traffic volume on cost ratio R_L in a 30-node, 150-link random topology for the load-based cost function, $k = 10\%$.

we conduct a series of simulations in which k is increased from 10% to 30%, with the fraction of high-priority traffic fixed at $f = 30\%$. We observe opposite behaviors under the load- and SLA-based cost functions.

We consider first the load-based cost function. As shown in Fig. 5(a), when k changes from 10% to 30%, R_L decreases. This is because for a fixed fraction f of the high-priority traffic volume, as the number of SD pairs sending high-priority traffic increases, the high-priority traffic is spread on more links. For instance, Fig. 6 shows the high-priority link utilization (sorted in descending order) in a random topology, when STR is used. As shown in the figure, when k is increased from 10% to 30%, the curve "flattens", indicating more evenly distributed high-priority load across the links, and hence lowering the utilization on those once-highly-loaded links. Consequently, the residual capacity on those links is increased, which helps make the performance of low-priority traffic under STR closer to that of DTR, hence reducing R_L.

In contrast, when the SLA-based cost function is used, R_L increases when k changes from 10% to 30%, e.g., see Fig. 5(b). This can be explained as follows: Given that

Figure 5: The impact of high-priority SD-pair density (k) on R_L in a 30-node, 150-link random topology. (a) Load-based cost function. (b) SLA-based cost function.

the high-priority traffic is only a small portion of the total traffic ($f = 30\%$), its queueing delay is nearly insignificant compared to the propagation delay; hence it plays little or no role in the optimization of the objective function Λ in Eq. (4). As the density of high-priority SD pairs initially increases, more high-priority flows concentrate on links with small propagation delays. In STR, because nodes originating high-priority traffic also generate low-priority traffic to the same destination(s), when k increases to 30%, at least 30% of low-priority SD pairs are "forced" onto a small set of links with low propagation delay. Therefore, the low-priority traffic
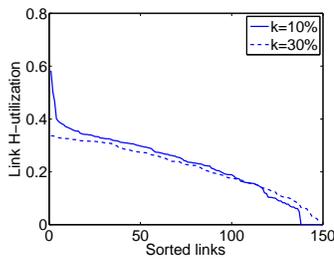
Figure 6: Link load contributed by the high-priority traffic under STR using the load-based cost function. The results for two different high-priority SD-pair densities, $k = 10\%$ and $k = 30\%$, are plotted.
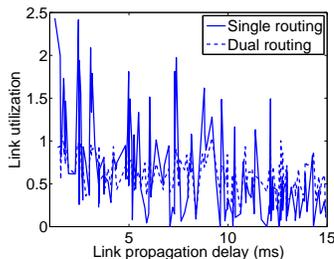

Figure 7: Link load as a function of propagation delay when SLA-based cost function is used.

performance is poorer than when $k = 10\%$. To better illustrate this effect, we plot the link utilization distributions for $k = 10\%$ and $k = 30\%$ in Figs. 3(b) and 3(c) respectively in which a random topology is used. We see that when $k = 30\%$, the tail of distribution is spread to the right under STR, indicating more low-priority traffic has been "dragged" onto congested links. In Fig. 7, we also plot the load of a link as a function of its propagation delay, which shows how links with lower propagation delay tend to have a higher load. This further confirms the role of link propagation delays.

5.2.3   Impact of high-priority traffic patterns

The above discussions focus on the *random* model for the high-priority traffic. We consider next the *sink* model of Section 5.1. We find that in most cases DTR still outperforms STR, and essentially the same observations hold as with random traffic patterns (details can be found in [25]). Nevertheless, there exists one noticeable difference, namely, that the distribution of high-priority SD pairs together with topology, may significantly affect the effectiveness of DTR.

To illustrate this effect, we consider a power-law topology and two different distributions of high-priority node. Specifically, we first select 3 nodes with the highest degrees as sink nodes and then select high-priority client nodes either uniformly distributed in the network or
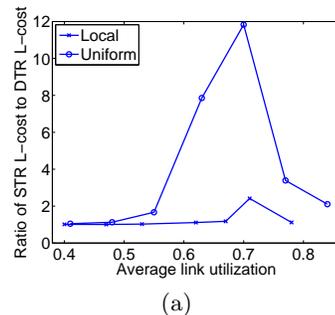
(a)
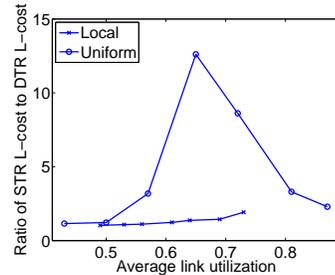
(b)

Figure 8: Performance benefits of DTR for the sink communication pattern (Uniform vs. Local) in a 30-node, 162-link power-law topology. $f = 20\%$ and $k = 10\%$. (a) Load-based cost function. (b) SLA-based cost function.

among nodes located close to the sinks. Fig. 8 shows the performance of DTR and STR under the load-based and the SLA-based cost functions for the two scenarios ("Uniform" and "Local" denote the first and second scenarios, respectively). We see that DTR and STR achieve similar performance ($R_L \approx 1$) in the "Local" scenario, while DTR performs much better than STR in the "Uniform" scenario. There are several reasons for this behavior. When client nodes are close to the sink nodes, high-priority paths remain "local" to the sink nodes, and hence affect fewer low-priority SD pairs overall. In addition, the proximity between client and sink nodes also means that their SLA is easier to meet, which allow more routing solutions, some of which can yield better low-priority performance. Neither of these factors is present when clients are randomly distributed in the network, so that the benefits of DTR remain more pronounced in this case.

5.3   Comparing DTR and STR with tradeoff between traffic classes

Following the discussion of Section 3.3, we study whether and how much low-priority performance can be improved under STR if we are willing to sacrifice some high-priority performance.

Table 1: Low-priority traffic performance in STR with a relaxation. $f = 30\%$, $k = 10\%$.

30-node, 150-link random topology						
R_L	1.03	1.29	1.38	4.04	31.16	58.17	7.51
$R_{L,5\%}$	1.02	1.07	1.14	2.34	19.89	21.84	5.74
$R_{L,30\%}$	1.02	1.04	1.14	2.07	6.91	19.56	5.52
A_D	0.43	0.5	0.58	0.65	0.73	0.8	0.88
30-node, 162-link power-law topology						
R_L	1.03	1.15	5.20	34.32	21.06	8.3	4.07
$R_{L,5\%}$	1.03	1.09	1.3	5.86	15.62	4.69	3.28
$R_{L,30\%}$	1.03	1.09	1.25	3.24	15.45	4.58	3.06
A_D	0.41	0.48	0.55	0.63	0.71	0.78	0.85
ISP topology						
R_L	1	1.07	1.11	3.91	9.28	2.3	2.1
$R_{L,5\%}$	1	1.04	1.09	2.06	7.72	1.58	1.36
$R_{L,30\%}$	1	1.04	1.06	1.42	5.94	1.45	1.35
A_D	0.33	0.42	0.5	0.59	0.67	0.78	0.86

5.3.1 The case of load-based cost function

We first consider the load-based cost function. Relaxing the constraint of strict lexicographical optimization can be easily incorporated into the heuristic used to search for STR solutions. Specifically, denote the link weight setting at iteration n as $W(n)$, and the costs of high- and low-priority traffic associated with it as $\Phi_H(n)$ and $\Phi_L(n)$, respectively. $\Phi_H^*(n)$ and $\Phi_L^*(n)$ are the best costs for high- and low-priority traffic up to iteration n. Assume next that we allow a degradation of ε of the high-priority traffic performance. Then, for every iteration n, if $W(n)$ results in $\Phi_H(n) \leq (1+\varepsilon)\Phi_H^*(n)$ and $\Phi_L(n) < \Phi_L^*(n)$, we record $W(n)$ as a "relaxed" best solution[6]. We study the impact on the low-priority traffic performance of this relaxation for $\varepsilon = 5\%$ and 30%. The results are shown in Table 1 for three different topologies, where A_D is the average link utilization. We observe that the performance of low-priority traffic improves as ε increases in STR, but even for $\varepsilon = 30\%$, a large gap remains between the performance of STR and DTR. Furthermore, this improvement causes significant performance degradation to the high-priority traffic. In contrast, DTR provides both greater improvement for low-priority traffic and no degradation in the performance of high-priority traffic.

5.3.2 The case of SLA-based cost function

In the case of an SLA-based cost function, relaxation takes the form of a looser SLA delay bound. To investigate its impact, we study the performance of STR and DTR when varying the SLA requirement from 25ms to 35ms. In Fig. 9, we report a set of results collected from a random topology, with a random traffic pattern.

Fig. 9(a) shows that independent of the SLA delay bound, the number of high-priority SD pairs that violate the SLA requirement is the same for STR and DTR. We also observe that when the SLA delay bound

---

[6]If there are multiple such $W(n)$'s, we pick the one achieving the lowest $\Phi_L(n)$.

is above 30ms, STR and DTR result in similar performance for the low-priority traffic and similar maximum link utilizations as shown in Figs. 9(b) and 9(c) respectively. In other words, a loosening of the delay bound by 20% allows STR to provide similar performance as DTR for the low priority traffic. The main reason for this improvement is that the looser SLA enables STR to explore more routing solutions, including some that offer good performance to the low-priority traffic while still meeting the (looser) high-priority SLA. Nevertheless, DTR offers these benefits without penalizing high-priority traffic. Furthermore, as with any relaxation technique, identifying the "right" level of relaxation required to allow STR to provide reasonable performance to low-priority traffic is in itself a challenging task. This issue is altogether absent when using DTR.

6. CONCLUSION

This paper investigates how routing can help service differentiation in IP networks. We study the benefits of DTR, which allows separate routing of high- and low-priority traffic classes. Compared to the traditional STR scheme, DTR provides greater flexibility in controlling resource allocation across traffic classes, hence achieving more efficient service differentiation.

To realize the benefits of DTR, we developed a heuristic that overcomes the computational complexity of identifying good link weight settings for each traffic class. The heuristic was tested under a lexicographical optimization model that gives strict precedence to high priority performance, but as shown in the paper, can be easily adapted to other configurations. The heuristic was used to investigate the effectiveness of DTR with two common performance objectives, embodied by load-based and SLA-based cost functions, respectively. The results showed that irrespective of which cost function was used, DTR can substantially improve the performance of low-priority service class without sacrificing that of high-priority class. This was demonstrated through extensive simulations for a broad range of network topologies and traffic patterns.

7. REFERENCES

[1] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault, "Multi-topology (MT) routing in OSPF," IETF RFC 4915, June 2007.

[2] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, 2000.

[3] M. Ericsson, M. Resende, and P. Pardalos, "A genetic algorithm for the weight setting problem in OSPF routing," *J. of Combinatorial Optimization*, vol. 6, pp. 299–333, 2002.

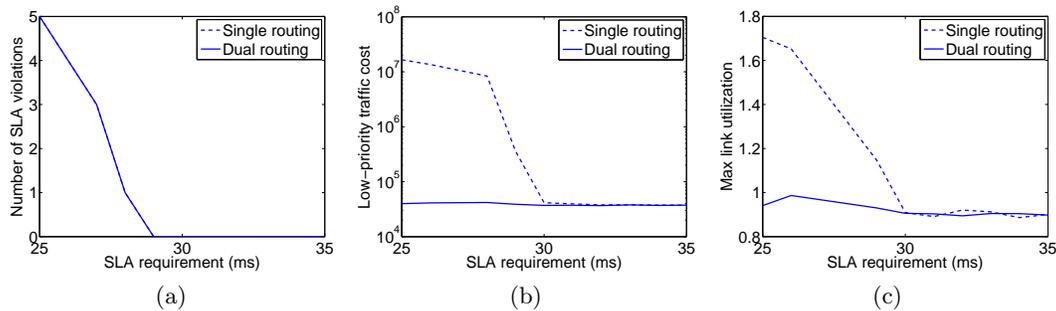[4] L. Buriol, M. Resende, C. Ribeiro, and M. Thorup, "A memetic algorithm for OSPF

Figure 9: The impact of changing the SLA requirement on STR and DTR in a 30-node, 150-link random topology. $f = 30\%$ and $k = 30\%$. **The average link utilization is roughly 0.5 in each SLA requirement setting.**

routing," in *Proc. 6th INFORMS Telecom*, 2002.

[5] A. Nucci, S. Bhattacharyya, N. Taft, and C. Diot, "IGP link weight assignment for operational Tier-1 backbones," *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 789–802, August 2007.

[6] S. Balon and G. Leduc, "Dividing the traffic matrix to approach optimal traffic engineering," in *Proc. IEEE ICON*, 2006.

[7] G. Apostolopoulos, "Using multiple topologies for IP-only protection against network failures: A routing performance perspective," ICS-FORTH, Greece, Tech. Rep., 2006.

[8] S. Gjessing and O. Norway, "Implementation of two resilience mechanisms using multi topology routing and stub routers," in *Proc. AICT-ICIW*, 2006.

[9] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne, "Fast IP network recovery using multiple routing configurations," in *Proc. IEEE INFOCOM*, 2006.

[10] W. C. Lee, M. G. Hluchyj, and P. A. Humblet, "Routing subject to quality of service constraints in integrated communication networks," *IEEE Network*, vol. 9, no. 4, pp. 46–55, Jul/Aug 1995.

[11] E. Crawley, B. R. R. Nair, and H. Sandick, "A framework for QoS-based routing in integrated service networks," *IETF RFC 2386*, 1998.

[12] J. Wang and K. Nahrstedt, "Hop-by-hop routing algorithms for premium traffic," *ACM SIGCOMM Computer Communications Review*, vol. 32, no. 5, pp. 73–88, November 2002.

[13] R. Guérin and A. Orda, "QoS-based routing in networks with inaccurate information: Theory and algorithms," in *Proc. IEEE INFOCOM*, 1997.

[14] Q. Ma and P. Steenkiste, "Supporting dynamic inter-class resource sharing: A multi-class QoS routing algorithm," in *Proc. IEEE INFOCOM*, 1999.

[15] S. Chen and K. Nahrstedt, "An overview of

quality of service routing for next-generation high-speed networks: Problems and solutions," *IEEE Network*, vol. 12, no. 6, pp. 64–79, Nov/Dec 1998.

[16] D. Mitra and K. G. Ramakrishnan, "Techniques for traffic engineering of multiservice, multi-priority networks," *Bell Labs Technical Journal*, vol. 6, no. 1, pp. 139–151, 2001.

[17] S. L. Spitler and D. C. Lee, "Integrating effective-bandwidth-based QoS routing and best effort routing," in *Proc. IEEE INFOCOM*, 2003.

[18] S. Balon, F. Skivée, and G. Leduc, "How well do traffic engineering objective functions meet TE requirements," in *Proc. IFIP Networking*, 2006.

[19] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 4, pp. 756–767, May 2002.

[20] S. Boettcher and A. G. Percus, "Nature's way of optimizing," *Artificial Intelligence*, vol. 119, pp. 275–286, May 2000.

[21] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, October 1999.

[22] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," in *Proc. ACM SIGCOMM*, 1999.

[23] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques compared and new directions," in *Proc. ACM SIGCOMM*, 2002.

[24] S. Bhattacharyya, N. Taft, J. Jetcheva, and C. Diot, "POP-level and access-link-level traffic dynamics in a Tier-1 POP," in *Proc. ACM IMW*, 2001.

[25] K.-W. Kwong, R. Guérin, A. Shaikh, and S. Tao, "Improving service differentiation in IP networks through dual topology routing," University of Pennsylvania, Tech. Rep., June 2007. [Online]. Available: http://einstein.seas.upenn.edu/mnlab