Optimized Network Traffic Engineering using Segment Routing

Randeep Bhatia, Fang Hao, Murali Kodialam, T.V. Lakshman
Bell Laboratories, Alcatel-Lucent
Crawford Hill, NJ 07733
{firstname.lastname}@bell-labs.com

I. Abstract

Segment Routing is a proposed IETF protocol to improve traffic engineering and online route selection in IP networks. The key idea in segment routing is to break up the routing path into segments in order to enable better network utilization. Segment routing also enables finer control of the routing paths and can be used to route traffic through middle boxes. This paper considers the problem of determining the optimal parameters for segment routing in the offline and online cases. We develop a traffic matrix oblivious algorithm for robust segment routing in the offline case and a competitive algorithm for online segment routing. We also show that both these algorithms work well in practice.

II. Introduction

Traditional routing in IP networks is along shortest paths using link weight as the metric. It has been observed that under some traffic conditions, shortest path routing can lead to congestion on some links in the network while capacity is available elsewhere in the network. The objective of traffic engineering is to ensure that traffic is managed such that network capacity is utilized efficiently and in a balanced manner. There are several techniques for performing traffic engineering in IP networks [2], including adjusting link weights based on traffic patterns, using MPLS to control routing paths and using centralized controllers like a Software Defined Network (SDN) controller to control traffic in a centralized manner.

A relatively new approach to traffic engineering is segment routing [6]. The key idea in segment routing is to break up the routing path into segments in order to better control routing paths and hence improve network utilization. A segment is essentially an MPLS label; traditional push, pop and swap actions can be applied by the routers on the path [7]. There are two basic types of segments: node and adjacency. A node segment identifies a router node. Node segment IDs are *globally unique* across the domain. An adjacency segment represents a local interface of a node. Adjacency segment IDs are typically only *locally significant* on each node. The MPLS data plane can be leveraged to implement segment routing essentially without modification since the same label switching mechanism can be used. Segment labels are distributed across the network using simple extensions to current IGP protocols and hence LDP and RSVP-TE are no longer required for distributing labels. As a result, the control plane can be significantly simplified. Moreover, unlike MPLS, there is *no need to maintain path state* in segment routing except on the ingress node, because packets are now routed based on the list of segments they carry. The ingress node has to be modified since it needs to determine the path and add the segment labels to the packet. At a high level segment routing is implemented as follows (See Figure 1): The ingress node adds a set of labels to IP header of the packet with the address of the end point of each segment. These labels are used as temporary destination addresses for the segment. The packet is then routed using the standard shortest path routing algorithm. When the packet reaches the intermediate destination from the ingress, the top level label is popped by the intermediate destination and now the packet is routed from the intermediate node to the end point of the next segment again along the shortest path. A

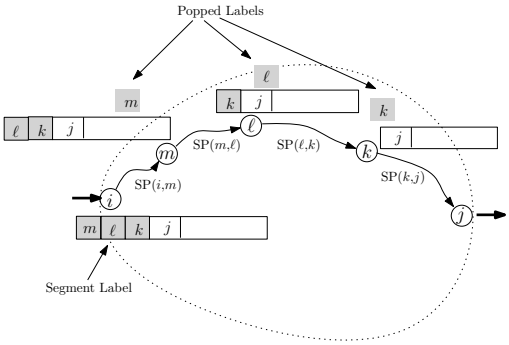

Fig. 1. Illustration of Segment Routing.

segment routing path with k segment paths will be referred to as k-segment routing. If we permit an arbitrary number of segments, we can use arbitrary paths and therefore the problem of routing flows in an n-segment routed network can be viewed as solving a multi-commodity flow problem. This flexibility comes at the expense of increased label overhead and increased label processing in the network. From a traffic engineering point of view, most of the benefits of moving away from shortest path routing can be achieved using just two hops. We show in the experimental section, that 2-segment routing is almost as good as solving the multi-commodity flow problem. Therefore in this paper, we primarily focus on 2-segment routing where traffic between the ingress and egress passes through exactly one intermediate node. The same techniques that are developed in this paper can be extended to segment

978-1-4799-8381-0/15/$31.00 ©2015 IEEE

routing with more than two segments albeit at the expense of increased computation. In the 2-segment routing problem, the key decision that has to be made by the control plane is how to pick the appropriate segments for each flow in order to minimize overall network congestion. We consider three versions of this problem in this paper:

- *Traffic Matrix Aware Segment Routing:* In this problem the traffic matrix is assumed to be known and the objective is to determine the traffic split across different segments for each source destination pair.
- *Traffic Matrix Oblivious Segment Routing:* In this case, the traffic matrix is not known in advance but the traffic splits are designed such that it distributes traffic well for a wide range of traffic matrices.
- *Online Segment Routing:* In this problem, connection requests arrive into the system one at a time had the segments are picked in order to keep the load balanced across the network.

The traffic matrix aware segment routing problem serves as a benchmark for the more practical segment routing problems with unknown traffic matrices. The main difference between the traffic matrix oblivious segment routing and online segment routing is that in the traffic matrix oblivious segment routing, the traffic split parameters are computed ahead of time and are set up once. There is no need for online monitoring of link utilization. In the online segment routing problem, segments are chosen based on current link utilization and therefore there has to be continuous monitoring of link utilization in order to compute the segment for the current flow. The main contributions of the paper are the following:

- A game theoretic like analysis of the segment routing problem results in the formulation of a linear program for solving the traffic matrix oblivious segment routing problem.
- A competitive online algorithm for the online segment routing problem with provable worst case performance guarantee.

There does not seem to be much analytic work on segment routing. There are some similarities of segment routing to pathlet routing [10] where fragments of end-to-end paths are exposed to help in routing. The resemblance to segment routing is in the fact that segments can be viewed as path fragments. However the shortest path structure of the segments lends itself well to modeling and optimization.

III. System Model

The network is represented by a graph $G = (N, E)$, where the nodes are the routers connected by directed links. Each link has a weight $w(e)$ and capacity $c(e)$. We do not assume that the network is symmetric. We assume that the flows in the network are 2-segment routed. Figure 2 is an illustration of 2-segment routing. We use the terms segment routing and 2-segment routing interchangeably in the rest of the paper. In the offline planning problem, instead of focusing on individual flows, we focus on the aggregate traffic between source destination pairs.

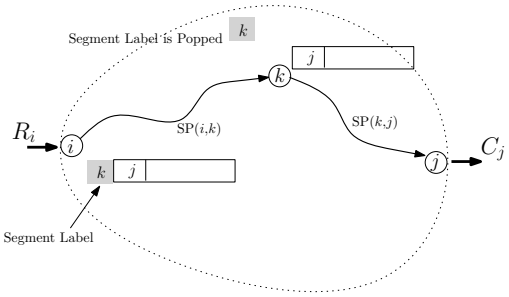

Fig. 2. Illustration of 2-Segment Routing.

The aggregate amount of traffic between nodes i and j is denoted by t_{ij}. In the first part of the paper, we assume that t_{ij} is known to the segment routing planner. In the last two sections, we do not assume that the traffic matrix is known. We assume that this traffic t_{ij} between nodes i and j can be split across multiple intermediate nodes k. We assume that this split is flow based. In other words, we assume that the source node splits the traffic using a hashing scheme that ensures that all packets belonging to the same flow are routed through the same intermediate node (thus maintaining packet ordering). Assume that the link weights are fixed and all routing is along shortest paths using this link weight as the metric. Let $SP(i, j)$ denote the set of links on the shortest path from i to j. Note that when there are multiple shortest paths between nodes, then the network can split traffic across these equal cost paths (ECMP [9]). If the size of individual flows is small compared the the capacity between node pairs and there are a large number of flows in the network then the traffic can be assumed to be split (approximately) equally across these multiple shortest paths. Figure 3 shows how traffic will be split across equal cost paths from i to j. The fraction next to a sub-path represents the fraction of this unit flow that traverses all links in the sub-path. We use $f_{ij}(e)$ to represent the flow on link e when unit flow is routed from i to j. The value of $f_{ij}(e)$ is function of the network topology, link weights and the routing policy. If a single shortest path is used to route traffic from i to j, the $f_{ij}(e) = 1$ for all links e on the shortest path and $f_{ij}(e) = 0$ if e is not this shortest path. If ECMP is used, then $f_{ij}(e)$ can be fractional. Assume that a unit flow from node i to node j is routed through k as the intermediate node. In this case, the flow on link e will be the sum of the flows from i to k and from k to j. We define this fraction

$$g^k_{ij}(e) = f_{ik}(e) + f_{kj}(e).$$

Therefore, $g^k_{ij}(e)$ is the flow that results on link e if a unit flow is routed from i to j through intermediate node k. Note that routing on each segment is along shortest paths. It is easy to show that the value of $g^k_{ij}(e)$ can be computed efficiently for a given set of link weights. We now consider the problem of determining the optimal segment routing scheme when the traffic matrix is given. Though the fact that the traffic matrix is known may seem a little unrealistic, the traffic matrix aware segment routing problem is the benchmark against

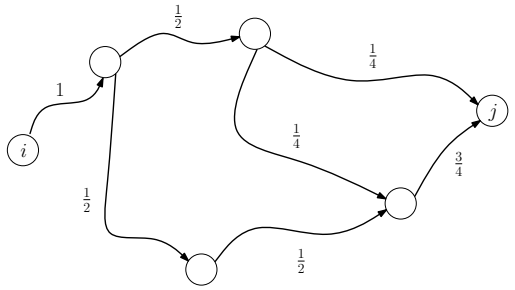

Fig. 3. Fractional values $f_{ij}(e)$ with ECMP routing. All links on some shortest path are shown. The values shown on the path represents the $f_{ij}(e)$ for all links e on that section of the path

which traffic matrix oblivious segment routing schemes will be measured.

IV. Traffic Matrix Aware Segment Routing

If the traffic matrix is given, then it is easy to figure out how traffic should be segment routed. Let t_{ij} denote the traffic between nodes i and j. Let x_{ij}^k denote the amount of traffic between i and j that is routed through node k. Recall that in 2-segment routing we just have to pick one intermediate node through which flow is routed. Intermediate node $k = i$ or $k = j$ represents the shortest path from i to j. If x_{ij}^k amount of traffic is routed from i to j through intermediate node k, then the amount of this flow that traverses link e is $g_{ij}^k(e)\ x_{ij}^k$. The total flow on link e is then is $\sum_{ij}\sum_k g_{ij}^k(e)x_{ij}^k$.

Definition 4.1: A traffic matrix $[t_{ij}]$ is defined to be **2-Segment Routeable** if there exists $x_{ij}^k \geq 0$ for all (i,j) that satisfies the following constraints:

$$\sum_k x_{ij}^k \quad = \quad t_{ij} \quad \forall\,(ij) \tag{1}$$

$$\sum_{ij}\sum_k g_{ij}^k(e)x_{ij}^k \quad \leq \quad c(e) \quad \forall\, e \tag{2}$$

Equation (1) ensures that all the traffic between nodes i and j is routed in the network. Equation (2) constrains the flow on link e to be not greater than the capacity of link e. The set of segment routable matrices will be denoted by $\mathcal{SR}_2$.

Since the number of links in the min weight path between any two nodes has not more than n links, the set of n-segment routable traffic matrices represent the set of traffic matrices that can be routed on the network using arbitrary paths. In other words, these are the traffic matrices that can be routed on the network (without violating link capacities) by solving the multi-commodity flow problem. Let $\mathcal{P}_{ij}$ denote the set of paths between nodes i and j and P denote a generic path.

Definition 4.2: A traffic matrix $[t_{ij}]$ is defined to be n-**Segment Routable** if there exists $x(P) \geq 0$ corresponding to path P such that

$$\sum_{P\in\mathcal{P}_{ij}} x(P) \quad = \quad t_{ij} \quad \forall\,(ij) \tag{3}$$

$$\sum_{P:e\in P} x(P) \quad \leq \quad c(e) \quad \forall\, e \tag{4}$$

Equation (3) ensures that all traffic is routed and Equation (4) ensures that no link capacity is exceeded. The set of routable matrices will be denoted by $\mathcal{SR}_n$. From the definitions, note that $\mathcal{SR}_2 \subseteq \mathcal{SR}_n$. The problem of determining the a 2-segment routing that minimizes the maximum link utilization for a given traffic matrix can be formulated as the following linear program. The variable θ represents the maximum link utilization.

$$\min\ \theta$$

$$\begin{aligned}
\sum_k x_{ij}^k \quad &\geq \quad t_{ij} \quad \forall\,(ij) \\
\sum_{ij}\sum_k g_{ij}^k(e)x_{ij}^k \quad &\leq \quad \theta\ c(e) \quad \forall\, e \\
x_{ij}^k \quad &\geq \quad 0 \quad \forall\,(ij)
\end{aligned}$$

It is easy to solve this linear program or use a combinatorial primal-dual algorithm along the approach pioneered in [8]. We solve this problem in the experimental section but our main interest in outlining this problem is to develop an algorithm for the traffic oblivious segment routing problem.

V. Traffic Oblivious Segment Routing

The traffic pattern in a network varies widely over time and is not easy to estimate accurately. Therefore there is need for developing techniques for setting segment routing parameters in a manner that works well for a wide range of traffic matrices. One approach to solve this problem is to consider a group of traffic matrices and then use some heuristic to derive the segment routing parameters. An alternate approach is to develop techniques where the segment routing parameters are chosen in a traffic matrix oblivious manner. We use a rigorous approach based on game theoretic techniques to derive a traffic matrix oblivious segment routing mechanism. This adversarial approach is part of the game theoretic literature and its use in networking was initiated in [1] for deriving routing parameters for Racke's oblivious routing scheme. The adversarial approach works very well for the traffic oblivious 2-Segment routing problem and the resulting linear programs are relatively easy to solve. Alternative approaches like in [11] that optimizes network performance for expected traffic matrices while bounding the worst case performance can be used for segment routing. In this paper, we restrict attention the traffic oblivious segment routing problems.

A. Comparing the Performance of Traffic Oblivious Segment Routing Algorithms

The most common measure of the performance of a routing algorithm is the maximum link utilization that the algorithm produces. The maximum link utilization is a function of the routing algorithm $\mathcal{A}$ as well as the traffic matrix $\mathbf{T} = [t_{ij}]$. Let $\ell_e(\mathcal{A},\mathbf{T})$ denote the load (flow) on link e when routing algorithm $\mathcal{A}$ is used to route traffic matrix $\mathbf{T}$. We define the resulting maximum link utilization as

$$\rho(\mathcal{A},\mathbf{T}) = \max_e \frac{\ell_e(\mathcal{A},\mathbf{T})}{c(e)}.$$

Given a routing algorithm $\mathcal{A}$, we use $\mathcal{F}(\mathcal{A})$ to be the set of traffic matrices which are routed by the algorithm without any link exceeding its capacity.

$$\mathcal{F}(\mathcal{A}) = \{\mathbf{T} : \rho(\mathcal{A}, \mathbf{T}) \leq 1\}.$$

When comparing two algorithms $\mathcal{A}$ and $\mathcal{B}$, we use

$$\eta(\mathcal{A}, \mathcal{B}) = \max_{\mathbf{T}} \frac{\rho(\mathcal{A}, \mathbf{T})}{\rho(\mathcal{B}, \mathbf{T})}$$

to measure the performance of $\mathcal{A}$ relative to $\mathcal{B}$. For a given traffic matrix $\mathbf{T}$, the ratio represents the ratio of maximum link load realized by algorithm $\mathcal{A}$ to the maximum link load realized by $\mathcal{B}$. The value of $\eta(\mathcal{A}, \mathcal{B})$ is the is ratio of the link loads for the worst case traffic matrix. The link load for any algorithm is a linear function of the traffic matrix. Therefore for some $\lambda > 0$

$$\ell_e(\mathcal{A}, \lambda\mathbf{T}) = \lambda\, \ell_e(\mathcal{A}, \mathbf{T}).$$

This implies that

$$\rho(\mathcal{A}, \lambda\mathbf{T}) = \lambda\, \rho(\mathcal{A}, \mathbf{T}).$$

Therefore when comparing the performance of two algorithms,

$$\begin{aligned} \eta(\mathcal{A}, \mathcal{B}) &= \max_{\mathbf{T}} \frac{\rho(\mathcal{A}, \mathbf{T})}{\rho(\mathcal{B}, \mathbf{T})} \\ &= \max_{\mathbf{T} \in \mathcal{F}(\mathcal{B})} \rho(\mathcal{A}, \mathbf{T}). \end{aligned}$$

Therefore, the performance of algorithm $\mathcal{A}$ relative to $\mathcal{B}$ is the highest link utilization that algorithm $\mathcal{A}$ achieves over all traffic matrices that can be feasibly routed by $\mathcal{B}$. We use $\mathcal{B}$ to be traffic aware segment routing and $\mathcal{A}$ to be traffic oblivious segment routing. Since the performance of traffic aware segment routing will always be not worse than the traffic oblivious scheme, the value of $\eta(\mathcal{A}, \mathcal{B}) \geq 1$. We now outline the structure of the traffic oblivious 2-segment routing algorithm and its implementation. We then derive the optimal parameters of this routing scheme by solving a linear program.

B. Structure and Implementation of Traffic Oblivious Segment Routing

Instead of using flow variables, the main set of variables that we use is the fraction of traffic for each source destination pair that is routed through a given intermediate node. More specifically, let α_{ij}^k represent the *fraction of traffic* from i to j that is routed through intermediate node k. Since all traffic has to be routed we have $\sum_k \alpha_{ij}^k = 1$ for all (i, j) pairs. We call the set of α_{ij}^k the *traffic split variables.* The traffic split variables are picked *independent of the traffic matrix*. In the next section, we outline the technique for picking the traffic split variables. Figure 4, illustrates traffic oblivious segment routing. The figure shows the traffic split variables as well as the actual traffic on a couple of links due to a traffic flow of t_{ij} between nodes i and j. The amount of traffic between nodes i and j that flows on link e is $g_{ij}^k(e)\alpha_{ij}^k t_{ij}$. The first term $g_{ij}^k(e)$ represents the fraction of traffic that flows on link e if one unit of flow is sent from source i to destination j through intermediate node k. Note that $\alpha_{ij}^k t_{ij}$ represents the total amount of flow from i to j that is sent through intermediate node k. Therefore, the total amount of traffic flowing on the link is a function of the topology and link weights (through $g_{ij}^k(e)$) as well as the splitting factors and the traffic matrix. Note that only variable that we can set are the α_{ij}^k values. Once the values of α_{ij}^k have been picked (we outline how to do this in a traffic oblivious manner in the next section) we set up hash table at each ingress node i corresponding to each egress node j as follows: Assume that we have a uniform hash function that takes the flow id and maps to an n array. We assume that $\lfloor aij^k\ n \rfloor$ of these n array returns k. If we assume uniform hashing, note that the probability that k is returned as approximately α_{ij}^k. By choosing a suitably large value of n, the achieved probability can be made as close as we want to α_{ij}^k. We assume that there are a large number of flows between each node pair and the flow size is small compared to the total available capacity. In this case, if t_{ij} is the total traffic (bits/second) between nodes i and j, then the mean traffic between i and j that goes through node k will be $\alpha_{ij}^k\ t_{ij}$. When a new flow arrives at node i, its destination j is first determined. The flow id is hashed into the hash table corresponding to egress node j and this returns the intermediate node k through which the flow is routed. Label k is prepended to the packet header and the flow is routed from i to k. Once the intermediate node for a flow is picked, all subsequent packets for the flow will use the same intermediate node. Once the packet reaches node k, the header is popped and the packet is routed from k to j.

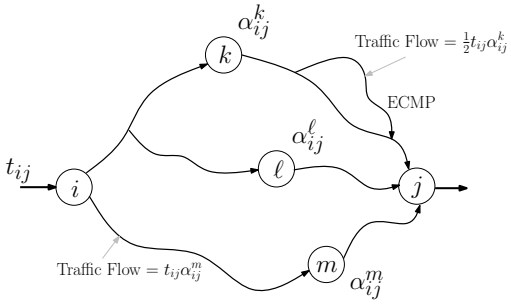

Fig. 4. Traffic split variables for source-destination pair (i, j). Note that α_{ij}^k represents the fraction of traffic from i to j that is routed through node k. Also shown are actual amount of (i, j) traffic carried on a couple of sub-paths. Note that actual flow values depend on the traffic split variables as well as whether ECMP is used.

C. Picking the Optimal Split Ratios

We now consider the problem of optimally picking the traffic split variables. We assume that we want to pick traffic split variables that works well for any 2-segment routable traffic matrix. Recall that

$$\eta(\mathcal{A}, \mathcal{B}) = \max_{\mathbf{T} \in \mathcal{F}(\mathcal{B})} \rho(\mathcal{A}, \mathbf{T}).$$

Let the traffic oblivious 2-segment routing algorithm be $\mathcal{A}$ and the traffic aware 2-segment routing algorithm be $\mathcal{B}$. We now consider a link e' and consider the maximum load on link e'

due to some traffic matrix in $\mathcal{F}(\mathcal{B})$. Note that $\mathcal{F}(\mathcal{B}) = \mathcal{SR}_2$ is the set of all 2-segment routable traffic matrices. Assume that an adversary wants to pick a traffic matrix from $\mathcal{SR}_2$ that maximizes the load on link e'. This can be formulated as the following optimization problem:

$$\max \sum_{ij} \sum_{k} g_{ij}^k(e')\ \alpha_{ij}^k\ t_{ij}$$

$$\begin{aligned} t_{ij} - \sum_k x_{ij}^k &= 0 \quad \forall\ (ij) \\ \sum_{ij} \sum_k g_{ij}^k(e) x_{ij}^k &\leq c(e) \quad \forall\ e \\ x_{ij}^k &\geq 0 \quad \forall\ (ij) \end{aligned}$$

The objective function is the flow on link e' if the split variables are α_{ij}^k, and the traffic between i and j is t_{ij}. The variables in the linear program are the traffic matrix values t_{ij}. We now observe the crucial fact that α_{ij}^k is *independent of the traffic matrix* (and can therefore be treated as a constant in this linear program. The t_{ij}^k can be eliminated from the linear program to get the following:

$$\max \sum_{ij} \left[\sum_k g_{ij}^k(e')\ \alpha_{ij}^k \right] \sum_m x_{ij}^m$$

$$\begin{aligned} \sum_{ij} \sum_k g_{ij}^k(e) x_{ij}^k &\leq c(e) \quad \forall\ e \\ x_{ij}^k &\geq 0 \quad \forall\ (ij) \end{aligned}$$

Associating a dual variable $\pi(e, e')$ with constraint corresponding to link e, we can write the dual to the above linear program as

$$\min \sum_e c(e)\ \pi(e, e')$$

$$\begin{aligned} \sum_e g_{ij}^m(e)\ \pi(e, e') &\geq \sum_k g_{ij}^k(e')\ \alpha_{ij}^k \quad \forall\ (ij)\ \forall e'\ \forall m \\ \pi(e, e') &\geq 0 \quad \forall\ e, e' \end{aligned}$$

Let θ be the maximum link utilization. (We want to minimize θ). We want the solution of the primal linear program to be less than $\theta c(e')$. By strong duality, the optimal solution to the primal and dual are the same and therefore we want $\sum_e c(e)\ \pi(e, e') \leq \theta c(e')$ if we want the maximum link utilization of e' to be less that θ. Therefore the system of inequalities

$$\begin{aligned} \sum_e g_{ij}^m(e)\ \pi(e, e') &\geq \sum_k g_{ij}^k(e')\ \alpha_{ij}^k \quad \forall\ (ij)\ \forall e'\ \forall m \\ \sum_e c(e)\ \pi(e, e') &\leq \theta\ c(e') \quad \forall\ e' \\ \pi(e, e') &\geq 0 \quad \forall\ e, e' \end{aligned}$$

represents the set of conditions that α_{ij}^k has to satisfy if we want the utilization of link e' to be less than θ for *any* traffic matrix in $\mathcal{SR}_2$. We can write one set of inequalities for each link e' along with the condition $\sum_k \alpha_{ij}^k = 1$ for all i, j to formulate the problem of determining the traffic split values to minimize the worst case link utilization as

$$\min \theta$$

$$\begin{aligned} \sum_e g_{ij}^m(e)\ \pi(e, e') &\geq \sum_k g_{ij}^k(e')\ \alpha_{ij}^k \quad \forall\ (ij)\ \forall e'\ \forall m \\ \sum_e c(e)\ \pi(e, e') &\leq \theta\ c(e') \quad \forall\ e' \\ \sum_k \alpha_{ij}^k &= 1 \quad \forall\ (ij) \\ \alpha_{ij}^k, \pi(e, e') &\geq 0 \quad \forall\ (ij)\ \forall e, e' \end{aligned}$$

The objective of the linear program is to minimize the maximum link utilization over all traffic matrices in $\mathcal{SR}_2$. Note that this linear program has $O(n^3 m)$ constraints and $O(n^3 + m^2)$ variables. The variables in the linear program are α_{ij}^k for all node triplets i, j, k and $\pi(e, e')$ for all link pairs e, e'. We solve this linear program using the GNU LP solver to derive the optimal traffic matrix oblivious traffic split values. So far we have considered the case where the routing in done in a distributed manner and once the α_{ij}^k values are set, we do not change the route according to the traffic matrix. However, if there is a centralized Path Computation Engine (PCE) that routes every request and if requests arrive one at time then we can develop an online approach that can use the current network state to segment route traffic. We explore this approach in the next section.

VI. Online Segment Routing

If the network has a centralized controller like a Path Computation Engine (PCE) [5] or an Software Defined Network (SDN) [12] controller that processes every flow arrival and computes a segment routed path for the arrival, then this centralized controller can use an online approach to solve the segment routing problem. In the online approach, we assume that we know the current network state (all the link loads) but we do not assume that we know any future arrivals. The objective of the online routing algorithm is to reject as few requests as possible. The performance of the online algorithm will be compared to be best offline algorithm that knows all requests ahead of time. We show that we can derive online algorithms with guaranteed worst case performance.

A. Online Model

The network set up is the same as in the traffic oblivious routing problem. The main difference is the way requests arrive into the system. In the online segment routing problem, requests arrive to the PCE (or SDN controller) one at a time. Request r requires $d(r)$ units of bandwidth between nodes $s(r)$ and $t(r)$. If request r is accepted, then it is segment routed from $s(r)$ to $t(r)$ and there is a "profit" of $d(r)$. We assume that $s(r), t(r)$ and $d(r)$ are known at the request arrival. The PCE has to determine the 'best" intermediate node k to route this request. All routing is along shortest paths. We assume

that ECMP is used and that the smallest non-zero value of $g_{ij}^k(e) \geq B$ for some constant B. The implication of this assumption is that the flows between source and destination are not split because of ECMP into too many small streams. This assumption generally holds in practice and is needed to prove the competitive ratio of the online algorithm. Once a request is accepted into the system, it stays in the system permanently. This assumption is necessary to prove the theoretical results but we relax this assumption while performing experimental studies. We assume that if the request is rejected, then the system accumulates zero profit. The objective of the system is to maximize the total profit. If all the requests are known ahead of time, then we can formulate an integer programming problem to maximize the total profit.

B. Offline Problem Formulation

Let R denote the set of requests and r denote a specific request. We assume that all the link weights have been specified and Let $X_r^k = 1$ if request r is 2-segment routed through intermediate node k. If $X_r^k = 1$, then the flow on link ℓ will be $g_{s(r)t(r)}^k(\ell)d(r)$. In order to keep the notation simple we use the short-hand

$$g_r^k(\ell) = g_{s(r)t(r)}^k(\ell).$$

Using this shorthand notation, the flow on link e when request r is routed through intermediate node k is $g_r^k(\ell)d(r)$.

$$\max \sum_{r \in R} d(r) \sum_k X_r^k$$

$$\sum_k X_r^k \leq 1 \quad \forall r \tag{5}$$

$$\sum_{r \in R} g_r^k(e) X_r^k \leq c(\ell) \quad \forall \ell \tag{6}$$

$$X_r^k \geq 0 \quad \forall r, k \tag{7}$$

We consider the linear programming relaxation of the above problem where we set $0 \leq X_r^k \leq 1$. The upper bound $X_r^k \leq 1$ is implied by Equation (5) and can therefore be eliminated from the formulation. We now write the dual to the above linear programming relaxation as

$$\min \sum_r \pi(r) + \sum_\ell c(\ell)\theta(\ell)$$

$$\pi(r) \geq d(r) \left[1 - \sum_\ell g_r^k(\ell)\ \theta(\ell)\right] \quad \forall k \ \ \forall r \tag{8}$$

$$\pi(r), \delta(\ell) \geq 0 \tag{9}$$

From Equation (8), we can set

$$\pi(r) = \max_k \ d(r) \left[1 - \sum_\ell g_r^k(\ell)\ \theta(\ell)\right] \quad \forall r. \tag{10}$$

The offline segment routing problem cannot be solved in practice since the requests will not be known ahead of time. We use the above primal and dual problems to develop an online algorithm. The design of this algorithm follows the elegant approach outlined in [3], [4]. We now outline the performance guarantee that the online algorithm provides.

C. Performance Guarantee

Let Z^* denote the optimal solution to the linear programming problem. This is the maximum traffic that is routed by the offline algorithm without exceeding any capacity constraints. Let Z_{ON} represent the solution obtained by the online algorithm. Note that $Z_{ON} \leq Z^*$ since the online algorithm can at best match the offline optimal solution. We say that the online algorithm is β-competitive if there exists a $\beta \leq 1$ such that

$$Z_{ON} \geq \beta \ \ Z^*$$

for *all* possible inputs. In general, we would like to get a competitive ratio that is close to one. This is possible in the case of some simple online optimization problems. For the more complex problems like the online segment routing problem, we can show that a constant factor competitive algorithm is not possible. Instead we define the notion of $[c_1, c_2]$ competitive algorithm [3].

Definition 6.1: An online algorithm is $[c_1, c_2]$-competitive if the online algorithm achieves routes at least $c_1 \leq 1$ of the optimum offline solution while ensuring that the *utilization* of any link does not exceed $c_2 \geq 1$.

For example an online algorithm is $[0.5, \log n]$ competitive if the online algorithm gets at least 0.5 of the revenue of the offline algorithm while none of the links flows exceed $\log n$ of the link capacity. We ideally want none of the link capacities to be exceeded. For many difficult online optimization problems, this logarithmic capacity violation is the best possible guaranteed bound. One way of getting around this excess utilization is to scale down the capacities by $\log n$ so that the capacity will not be exceeded. This may be too pessimistic. Instead, we show how to modify the algorithm to get good online heuristic that works well in practice in Section VI-F.

D. Online Segment Routing Algorithm

The description of the online segment routing algorithm is shown in Figure 5. The algorithm is quite simple. Link ℓ has a dual weight $\theta(\ell)$ that is initialized to zero. The $\pi(r)$ values are only used in the analysis of the algorithm. At each arrival, we find the minimum weight 2-segment path from the ingress to egress for the demand. Link ℓ is weighted with the $g_{ij}^k(\ell)$ values. Note that the routing for each segment is along min weight path according to the link weight metric. If the minimum weight path has a weight greater than one, then we reject the demand. Else the demand is routed along this min weight path and the dual weights of all the links on this path are increased according to Equation (11). in the description of the algorithm and analysis we use ℓ to denote a generic link.

Next we analyze the competitive ratio of the primal-dual scheme.

Online Segment Routing:

- Initialize $\theta(\ell) \leftarrow 0 \ \ \forall \ell \in E$.
- At arrival r
  1) Compute $V = \min_k \sum_\ell g_r^k(\ell)\theta(\ell)$.
  2) If $V > 1$ then reject request r.
  3) If $V \leq 1$ then accept the demand and
     a) Let
     $$k^* = \arg\min_k \sum_\ell g_r^k(\ell)\theta(\ell)$$
     b) Set
     $$\pi(r) \leftarrow d(r)\left[1 - \sum_\ell g_r^{k^*}(\ell)\theta(\ell)\right]$$
     c) Set for all links e
     $$\theta(\ell) \leftarrow \theta(\ell)\left[1 + \frac{g_r^{k^*}(\ell)d(r)}{c(\ell)}\right] + \frac{1}{n(e-1)}\frac{g_r^{k^*}(\ell)d(r)}{c(\ell)} \quad (11)$$

Fig. 5. Description of Online Segment Routing

E. Analysis of Competitive Ratio

The approximation guarantee for the algorithm is proved in three steps as in [3]. First, we show that the dual solution constructed is feasible to the dual linear programming problem. Next we show that the ratio of the primal solution to the dual solution is bounded by a constant. Since the dual provides and upper bound to the primal solution, this implies that the primal solution, if feasible, is within a constant factor of the upper bound and hence the optimal solution. The last step is to show that the scaled primal solution constructed is feasible.

1) **Dual Feasibility:** We first show that the values of $\theta(\ell)$ and $\pi(r)$ generated by the algorithm are dual feasible. Let k^* denote the intermediate node that minimizes $\sum_\ell g_r^k(\ell)\theta(\ell)$. Setting

$$\pi(r) = d(r)\left[1 - \sum_\ell g_r^{k^*}(\ell)\theta(\ell)\right]$$

ensures that $\pi(r) \geq d(r)\left[1 - \sum_\ell g_r^k(\ell)\theta(\ell)\right]$ for all nodes k and hence the solution is dual feasible at this point. Any subsequent increase in the value of $\theta(\ell)$ still maintains the inequality since the value of $\pi(r)$ does not change.

2) **Primal to Dual Ratio:** The increase in primal value at time slot if request r is accepted is $d(r)$. Let $\Delta(\ell)$ denote the change in the value of $\theta(\ell)$ when request r is accepted. The increase in dual value is

$$\pi(r) + \sum_\ell c(\ell)\Delta(\ell).$$

Since $\pi(r) = d(r)\left[1 - \sum_\ell g_r^{k^*}(\ell)\theta(\ell)\right]$ and

$$c(\ell)\Delta(\ell) = g_r^{k^*}(\ell)d(r)\theta(\ell) + \frac{1}{n(e-1)}g_r^{k^*}(\ell)d(r).$$

Therefore

$$\begin{aligned}\pi(r) + \sum_\ell c(\ell)\Delta(\ell) &= d(r)\left[1 + \frac{\sum_\ell g_r^{k^*}(\ell)}{n(e-1)}\right] \\ &\leq d(r)\left[1 + \frac{2}{(e-1)}\right] \quad (12) \\ &= d(r)\left[\frac{e+1}{e-1}\right].\end{aligned}$$

In obtaining Inequality (12), we used $g_r^{k^*}(\ell) \leq 2$ for any link ℓ due to the fact that routing between $s(r)$ and k^* as well as k^* and $t(t)$ is along shortest paths and therefore no link can be traversed more than twice. Therefore the ratio of the increase in the primal to the dual in time period if request r is accepted is

$$\frac{d(r)}{\pi(t) + \sum_i B(i)\Delta(i,t)} \geq \frac{e-1}{e+1}.$$

Since this holds for each request (if a request is not accepted the primal and dual values do not change) and the initial primal and dual solutions are zero, the ratio of the primal to the dual solution after processing any request is $\frac{e-1}{e+1}$.

3) **Primal Feasibility:** We now show that the solution is *almost* primal feasible. Let $F(\ell, r)$ denote the flow on link ℓ after demand r is processed and

$$\rho(\ell, r) = \frac{F(\ell, r)}{c(\ell)}.$$

We use $\theta(\ell, r)$ as the dual variable corresponding to link ℓ after request r is processed. We want to show that

$$\theta(\ell, r) \geq \frac{1}{n}\frac{e^{\rho(\ell,r)} - 1}{e - 1}.$$

We show this via induction. This statement is true initially since $F(e, r)$ as well as $\theta(e, r)$ are both zero when $r = 0$. Assume that the statement is true after request $r - 1$.

$$\begin{aligned}\theta(\ell, r) &= \theta(\ell, r-1)\left[1 + \frac{g_r^{k^*}(\ell)d(r)}{c(\ell)}\right] + \frac{1}{n(e-1)}\frac{g_r^{k^*}(\ell)d(r)}{c(\ell)} \\ &\geq \frac{1}{n(e-1)}\left[e^{\rho(\ell,r-1)} - 1\right]\left[1 + \frac{g_r^{k^*}(\ell)d(r)}{c(\ell)}\right] + \\ &\quad \frac{1}{n(e-1)}\frac{g_r^{k^*}(\ell)d(r)}{c(\ell)} \\ &= \frac{1}{n(e-1)}\left[e^{\rho(\ell,r-1)}\left(1 + \frac{g_r^{k^*}(\ell)d(r)}{c(\ell)}\right) - 1\right] \\ &\geq \frac{1}{n(e-1)}\left[e^{\rho(\ell,r)} - 1\right]\end{aligned}$$

The first inequality follows from the induction hypothesis. If we assume that any particular request takes up only a very

small fraction of the link capacity, i.e., $\frac{g_r^{k^*}(\ell)d(r)}{c(\ell)} \to 0$, then we can use

$$\left(1 + \frac{g_r^{k^*}(\ell)d(r)}{c(\ell)}\right) \approx e^{\frac{g_r^{k^*}(\ell)d(r)}{c(\ell)}}.$$

This combined with the fact that $F(\ell, r) = F(\ell, r-1) + \frac{g_r^{k^*}(\ell)d(r)}{c(\ell)}$ gives the above result. The above expression can be refined if the request size is not small when compared to the link capacity. The solution will have additional terms involving the largest fraction of the link capacity that a request can take up. Since we do not route any request if $\theta(\ell) > B$ (in which case $V > 1$) we can set

$$\frac{1}{n(e-1)}\left[e^{\rho(\ell,r)} - 1\right] \leq B$$

and solving for $\rho(\ell, r)$, we get

$$\rho(\ell, r) \leq \log\left[(e-1)Bn + 1\right].$$

Therefore the algorithm gives an $\left[\frac{e-1}{e+1}, \log\left((e-1)Bn+1\right)\right]$ approximation algorithm. The constants in the approximation ratio can be manipulated by changing the dual update rules. The approximation, more generally is $[O(1), O(\log n)]$.

F. Practical Heuristic

In order to account adapt the algorithm for use in realistic settings, we develop a heuristic based on the analysis of the algorithm. In practice, requests arrive as well as leave the system and therefore the system has to account for request departures. Note that

$$\theta(\ell, r) \geq \frac{1}{n}\,\frac{e^{\rho(e,r)} - 1}{e-1}.$$

At any given point, the controller can track the total amount of flow $F(\ell)$ on link ℓ. We set

$$\theta(\ell) = \frac{1}{n}\,\frac{e^{\frac{F(\ell)}{c(\ell)}} - 1}{e-1}$$

or more generally make the weight of the link an exponential function of the link utilization, i.e., $\theta(\ell) = e^{\frac{F(\ell)}{c(\ell)}}$. The value of $\theta(\ell)$ is updated whenever any flow either enters or leaves the system. This adaptation of the online algorithm works very well in practice.

VII. Experimental Results

We performed experiments to measure the performance of the traffic oblivious offline algorithm as well as the online segment routing algorithm. These are preliminary results and more complete results will be presented in the full version of this paper.

A. Offline Traffic Oblivious Segment Routing

All the experiments were performed on two topologies one with 15 nodes and 56 links and the other with 33 nodes and 100 links. The results were similar and we show the results for the 15 node network. On each topology, the link capacities were either all set equal or picked at random (with a specified minimum). The link weights were either all set to one or picked uniformly at random between 1 and 10. Experiments were done with 50 traffic matrices. The results shown are a combination of the constant capacity, variable capacity, constant weight and variable weight cases. For traffic oblivious segment routing, once the topology and the link weights were specified, the values of $g_{ij}^k(e)$ were calculated taking ECMP into account. The linear program in Section V was solved using GNU LP Solver to get the optimal traffic split values. These values are independent of the traffic matrix. Once a traffic matrix is specified, all link loads can be calculated for the optimal traffic split values. The performance metric for all algorithms was the *maximum link utilization* for each traffic matrix.

1) Shortest Path Routing (SP)
2) Traffic Aware 2-Segment Routing (TA)
3) Traffic Aware n-Segment Routing or Multi-commodity flow (MC)
4) Traffic Oblivious 2-Segment Routing (OB)

The maximum link utilization for the algorithms are $\lambda_{SP}, \lambda_{TA}, \lambda_{MC}$ and λ_{OB} respectively. In Figure 6 we compare traffic aware 2-segment with multi-commodity flow (traffic aware n-segment routing) . Note that n-segment routing is only marginally better than 2 segment routing. Most benefits of path diversity can be captured by 2-segment routing. Figure 7 compares shortest path and traffic oblivious 2-segment routing. Both these techniques do not adapt to the traffic. Note that the performance of traffic oblivious 2-segment routing is better than shortest path routing by over a factor of 2 on the average. In Figure 8 we compare traffic aware 2-segment routing and traffic oblivious 2-segment routing. The worst case performance ratio obtained by solving the linear program is 2.6. Note that the performance ratio is significantly better than better for several traffic matrices.

B. Online Segment Routing

We also conduct an experiment to compare the performance of Online Segment Routing with Shortest Path Routing. Here we use the practical heuristic described in Sec. VI-F. In this setup, we generate 30K bandwidth demands from a Poisson distribution with mean rate of 1 arrival per minute. Source and destinations are randomly selected among the network nodes. Demand holding time is selected from an exponential distribution with mean of 30 minutes. Demand size is selected from a uniform distribution. We vary the mean demand size between 50 Mbps and 1 Gbps. Note that the network link capacity are generally within the range of 1 to 10 Gbps.

Figure 9 shows the percentage of demands being rejected under both Online Segment Routing and Shortest Path Routing

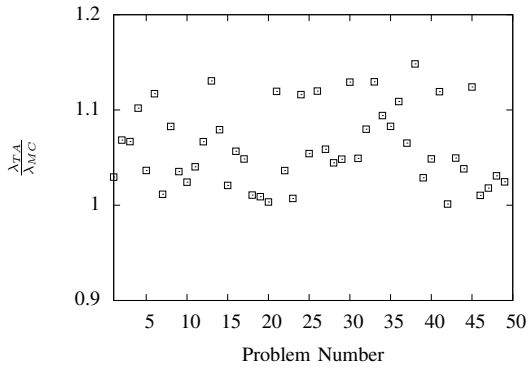

Fig. 6. Maximum Link Utilization Ratio for Traffic Aware Segment Routing (TA) to Multi-commodity Flow (MC). Multi-commodity flow is only marginally better than traffic aware 2-Segment Routing

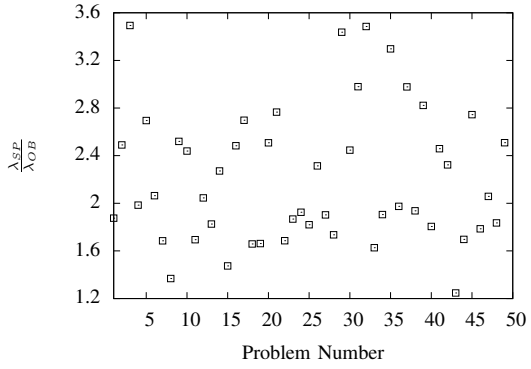

Fig. 7. Maximum Link Utilization Ratio for Shortest Path Routing (SP) to Traffic Oblivious Segment Routing (OB). Note that the maximum link utilization when using shortest path routing is almost twice the maximum link load when using traffic oblivious segment routing. Both these algorithms do not adapt to traffic and hence essentially use the same information.

across different mean demand size values. We observe that Segment Routing performs consistently better than Shortest Path Routing, with lower demand rejection rate. We also measure the percentage of bandwidth of all demands being rejected under both algorithms. The results are similar to Figure 9 and hence are not shown here.

REFERENCES

[1] D. Applegate, and E. Cohen, "Making Intra-Domain Routing Robust to Changing and Uncertain Trafc Demands: Understanding Fundamental Tradeoffs," SIGCOMM, 313-324, 2003.
[2] D. Awduche et. al, "Overview and Principles of Internet Traffic Engineering," IETF, RFC 3272, 313-324, 2003.
[3] N. Buchbinder, J. Naor, "Online Primal-Dual Algorithms for Covering and Packing Problems", 13th Annual European symposium on Algorithms, (2005).
[4] N. Buchbinder, J. Naor, "A Primal-Dual Approach to Online Routing and Packing", 47th IEEE Foundations on Computer Science, (2006).
[5] A. Farrel et. al, "A Path Computation Element (PCE)-Based Architecture," IETF, RFC 4655, Aug. 2006.
[6] C. Filfils et. al, "Segment Routing Architecture," IETF, Internet Draft, Dec. 2013.
[7] C. Filfils et. al, "Segment Routing with MPLS data plane," IETF, Internet Draft, Apr 2014.

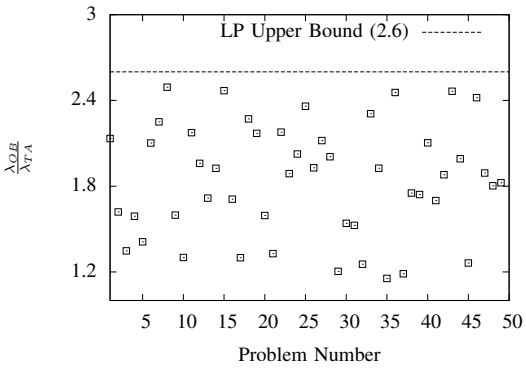

Fig. 8. Maximum Link Utilization Ratio for Traffic Oblivious Segment Routing (OB) to Traffic Aware Segment Routing (TA). Solving the linear program in Section V gives a worst case upper bound of 2.6 for the ratio. Note that the performance is better than the worst case bound for several traffic matrices.

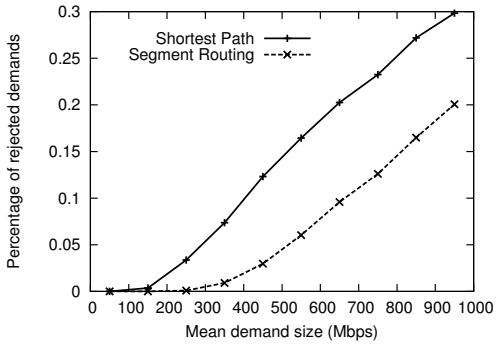

Fig. 9. Comparison of Online Segment Routing and Shortest Path

[8] N. Garg, and J. Konemann, "Faster and Simpler Algorithms for Multicommodity Flow and other Fractional Packing Problems," SIAM Journal on Computing, 37(2):630-652, 2007.
[9] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," IETF, RFC 2992, Nov. 2000.
[10] P.B. Godfrey, S. Shenker, and I. Stoica, "Pathlet Routing," SIGCOMM, Nov. 2008.
[11] H. Wang, H. Xie, L. Qiu, R. Yang, Y. Zhang, A. Greenberg, "COPE: Traffic Engineering in Dynamic Networks," ACM-SIGCOMM, 2006.
[12] "Software-defined networking: the new norm for networks," Open Networking Foundation, 2012.