

# On Traffic Engineering with Segment Routing in SDN based WANs

George Trimponias, Yan Xiao, Hong Xu, Xiaorui Wu, and Yanhui Geng

**Abstract**—Segment routing is an emerging technology to simplify traffic engineering implementation in WANs. It expresses an end-to-end logical path as a sequence of segments, each of which is represented by a midpoint. In this paper, we arguably conduct the first systematic study of traffic engineering with segment routing in SDN based WANs. We first provide a theoretical characterization of the problem. We show that for general segment routing, where flows can take any path that goes through a midpoint, the resulting traffic engineering is NP-hard. We then consider segment routing with shortest paths only, and prove that the traffic engineering problem can now be solved in (weakly) polynomial time when the number of middlepoints per path is fixed and not part of the input. Our results thus explain, for the first time, the underlying reason why existing work only focuses on segment routing with shortest paths. In the second part of the paper, we study practical traffic engineering using shortest path based segment routing. We note that existing methods work by taking each node as a candidate midpoint. This requires solving a large-scale linear program which is prohibitively slow. We thus propose to select just a few important nodes as middlepoints for all traffic. We use node centrality concepts from graph theory, notably group shortest path centrality, for midpoint selection. Our performance evaluation using realistic topologies and traffic traces shows that a small percentage of the most central nodes can achieve good results with orders of magnitude lower runtime.

**Index Terms**—Segment Routing, Traffic Engineering, Graph Centrality, Software Defined Networking

## I. INTRODUCTION

Traffic engineering (TE) is an important task for network operators to improve network efficiency and application performance. TE is commonly exercised in a wide range of networks, from carrier networks [22], [28] to data center backbones [30], [31]. Increasingly, TE is implemented using SDN (Software Defined Networking) given its flexibility. Notable examples include Google’s B4 [31] and Microsoft’s SWAN [30].

Implementing TE in the data plane requires a large number of flow table entries on switches. This is because each switch on the path needs to have an entry for a demand, i.e. ingress-egress switch pair, to forward its traffic to the next hop, and for a large-scale network there can be many demands. Commodity switches, on the other hand, have very limited capacity for flow entries (usually 1-2 thousands of entries [9], [35]) due

to the expensive TCAM (Ternary Content Aware Memory) hardware needed [8], [9], [40]. The use of wildcarding could reduce the number of flow entries, but it is often undesirable as it reduces the ability to implement demand-level policies and monitoring. Therefore it has become a major challenge to practically implement TE on commodity SDN switches.

Segment routing [18]–[20] is a recently proposed routing architecture to tackle this challenge. Its key idea is to perform routing based on a sequence of logical segments formed by some *middlepoints* between the ingress and egress nodes. A segment is the logical pipe between two middlepoints that may include multiple physical paths spanning multiple hops, and ECMP is used to load balance traffic among these paths. Now with segment routing instead of end-to-end paths, intermediate switches only need to know how to reach middlepoints in order to forward packets. They no longer need to maintain per-demand routing information which scales quadratically with the number of nodes. Thus segment routing has the potential to greatly reduce the overhead and cost of TE [5], [28].

Segment routing has been explored with TE in some existing work. For example Bhatia et al. [5] apply 2-segment routing to TE, where any logical path contains only one midpoint and thus two segments. Hartert et al. [28] propose some heuristics to solve various TE problems with segment routing. There lacks a thorough exploration and understanding of applying segment routing to TE, particularly the hardness of the resulting TE problem, and the development of practical TE algorithms with segment routing.

In this paper, we conduct arguably the first systematic study of TE with segment routing in SDN based WANs. We first focus on the theoretical aspects of TE with segment routing. We consider two common types of TE:  $TE_{MF}$  that maximizes total throughput based on multi-commodity flow, and  $TE_{LU}$  that minimizes the maximum link utilization.  $TE_{MF}$  is mostly for data center backbone WANs [30], [31], and  $TE_{LU}$  mostly for carrier networks [22], [28].

We provide new hardness results for TE with segment routing in directed networks. With general segment routing where traffic can take any path that goes through a midpoint, we prove that it is NP-hard to decide if the maximum flow through just one given midpoint is greater than 0 (§IV). Thus  $TE_{MF}$  is NP-hard. Due to the connection between the decision version of maximum flow and  $TE_{LU}$ , this also proves that  $TE_{LU}$  given a single fixed midpoint is NP-hard. We then study a restricted form of segment routing that uses only shortest paths between two segments in §V, and prove that both TE problems now can be solved in (weakly) polynomial time as an LP when the number of middlepoints per path

The work was supported in part by contract research between City University of Hong Kong and Huawei. The corresponding author is Hong Xu.

G. Trimponias and Y. Geng are with Huawei Noah’s Ark Lab, Hong Kong, China (email: g.trimponias@huawei.com, geng.yanhui@huawei.com).

Y. Xiao, H. Xu and X. Wu are with Department of Computer Science, City University of Hong Kong, Hong Kong, China (email: yanxiao6-c@my.cityu.edu.hk, henry.xu@cityu.edu.hk, xiaoruiwu3-c@my.cityu.edu.hk).

is fixed and not part of the input. Our results thus provide a theoretical foundation for existing work that focuses on shortest path based segment routing [5], [28]. Interestingly, imposing acyclic end-to-end paths renders TE NP-hard.

Given our theoretical results, we next focus on the practical problem of how to choose a small but representative set of middlepoints in order to solve TE with shortest path based segment routing (§VI). Existing approaches [5] assume that for each demand, every node in the network is potentially a midpoint candidate, and formulate it as part of the TE problem. This causes the TE to be of a very large scale, which makes it computationally expensive to solve for practical purposes. As we show in §VII-C, it cannot be solved by the ECOS solver [11] after three hours on a medium topology with 100 nodes and 1500 commodity flows, while in practice TE often needs to be re-computed at the granularity of 10 minutes [28], [30], [31], [37].

Thus we propose to apply the centrality concept from graph theory and network analysis [42] to select a few middlepoints to route all traffic in the network. Centrality was first developed in social network analysis [6], [24] to determine the most influential nodes in a social graph. In the context of routing, centrality can be naturally viewed in terms of the node importance when routing the demands along the admissible paths. We explore several centrality definitions based on the network topology only, such as shortest-path, group shortest-path, and degree centralities, and apply them to midpoint selection in networks. We also introduce weighted variants that additionally take into account the link capacities.

We conduct comprehensive performance evaluation of centrality based midpoint selection using real topologies and traffic traces. Our results demonstrate that only a small percentage of around 2.5%–7% of the most central nodes can achieve good TE performance with orders of magnitude lower runtime. Using centrality based midpoint selection methods, one can solve TE problems with up to 3000 flows on a 161-node topology in less than 3 minutes. We also observe that group shortest-path consistently outperforms other centralities for midpoint selection, and may be used as the sole solution in practice for simplicity.

Finally, we comment that our work is of independent theoretical interest for two reasons. First, our theoretical analysis for  $TE_{MF}$  can be used to prove that the flow centralities, first introduced in 1991 by Freeman, Borgatti, and White [23], are NP-hard to compute in directed graphs, thus restricting their practical applicability. Second, we show a profound dichotomy between directed and undirected networks: TE and flow centralities are NP-hard to compute in the former, but they are (at least) weakly polynomial in the latter. These results are included in Appendices at the end.

## II. A PRIMER ON SEGMENT ROUTING

We start by introducing segment routing and the benefit of applying it to TE. We next explain related work on segment routing.

Segment routing [18]–[20] is a recently proposed architecture based on source routing that facilitates packet forwarding

via a series of segments. It can be directly applied to MPLS and IPv6. The key idea is that the ingress switch can break up the end-to-end logical path into segments, and specify this logical path as a series of *middlepoints* to traverse. Figure 1 illustrates an example of segment routing. The ingress switch S embeds a stack of segment labels (MPLS labels for example) into the packet header to specify the entire path. Note here each label just represents a midpoint in the network, i.e. we consider node segments here [18]–[20]. The top label is the active label that instructs packet forwarding. Then the packet is sent to the next label  $M_1$  along the shortest path(s). ECMP is used if there are multiple shortest paths. When the packet reaches  $M_1$ , the top label  $M_1$  is popped and the packet is routed to the next label  $M_2$ . Finally, all the labels are popped and the packet arrives at the egress switch D.

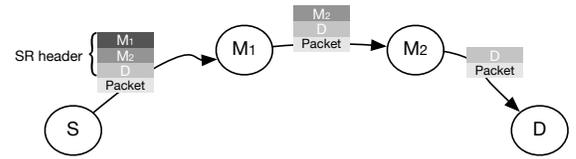


Fig. 1. A segment routing example from S to D through middlepoints  $M_1$  and  $M_2$ .

One key advantage of segment routing is that it can greatly reduce routing cost in terms of number of flow table entries required. To see this, consider the next example shown in Figure 2. Three *demands*<sup>1</sup>, which refer to the aggregated flows between a unique ingress-egress switch pair, are routed through three paths P1, P2, and P3 to their respective destinations. With tunnel-based forwarding in SDN [30], [31], each intermediate switch needs to store flow entries for each demand, and in total 12 entries are needed as shown in Table II. Now if segment routing is applied with node E as the midpoint, the three paths can be represented using just two labels each as in Figure 2, and switches C and D only need to have one entry in order to forward to the midpoint E. The total number of entries is reduced to only 8, with 33.3% saving.

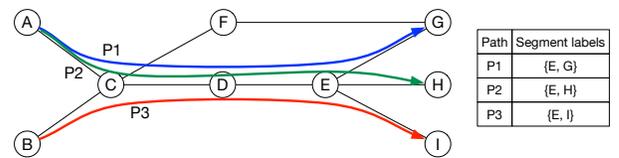


Fig. 2. An example where segment routing saves flow table entries.

Given the potential of segment routing, some recent work has started to investigate how to apply it in TE. In [5], the authors propose solutions for determining the optimal TE with segment routing and ECMP. In their scheme, they regard all nodes except for the source and the destination as candidate middlepoints, and split flows across exactly one midpoint. Although they limit the number of middlepoints to one for each logical path, since they consider all the intermediate

<sup>1</sup>When it is clear from the context, we use the terms *commodities*, *demands*, and *flows* interchangeably.

Node	w.o. segment routing	w. segment routing
A	2	2
B	1	1
C	3	1
D	3	1
E	3	3

TABLE I  
NUMBER OF FLOW TABLE ENTRIES FOR THE EXAMPLE IN FIG. 2.

nodes for one demand, the search space for middlepoints is very large. The algorithm thus cannot scale to handle medium to large scale networks. Hartert et al. [27], [28] studied a similar TE problem with segment routing under a constraint programming framework. Their midpoint selection method also takes every node as a potential candidate on a per-demand basis, and they have to resort to heuristics to reduce the run time of the algorithm.

The exploration of segment routing in TE has been ad-hoc so far. Existing work uses heuristics [28] or some special for of segment routing [5] without theoretical justification. We are thus motivated to conduct a systematic study of segment routing in TE, including the theoretical characterization of the hardness results of various forms of segment routing, and practical algorithms of solving the problems.

Given our focus on TE, in the next section we review the two common types of TE formulations. We subsequently reveal an interesting connection between them, which we utilize later in the proof of several hardness results.

### III. BACKGROUND ON TRAFFIC ENGINEERING

In our work, we focus on two common types of traffic engineering, depending on the objective criterion. The first type maximizes the total throughput subject to the capacity and maximum demand constraints. Since it can be formulated as a maximum flow problem, we call it  $TE_{MF}$ . The second type minimizes the maximum link utilization, which acts as the system bottleneck. For this reason, we call it  $TE_{LU}$ .

#### A. Preliminaries

Assume a directed graph  $G = (V, E)$ , where  $V$  is the set of nodes and  $E$  the set of directed edges. Given a node  $v \in V$ ,  $v^+$  denotes the set of outgoing edges of node  $v$ , i.e., the subset of edges in  $E$  of the form  $(v, u)$ ,  $u \in V$ . Similarly, the set  $v^-$  denotes the set of incoming edges of  $v$  of the form  $(u, v)$ ,  $u \in V$ . The out-degree of  $v$  is defined as the cardinality  $|v^+|$ , whereas the in-degree is defined as the cardinality  $|v^-|$ .

A *flow network*  $G = (V, E, c)$  is defined as a directed graph  $G = (V, E)$ , together with a non-negative function  $c : V \times V \rightarrow \mathbb{R}_{\geq 0}$  that assigns to each edge  $e \in E$  a non-negative capacity  $c(e)$ . If  $(u, v) \notin E$ , then we define  $c(u, v) = 0$ .

A *walk* in a directed graph is an alternating sequence of vertices and edges,  $v_0, e_0, v_1, \dots, v_{k-1}, e_{k-1}, v_k$ , which begins and ends with vertices and has the property that each  $e_i$  is an edge from  $v_i$  to  $v_{i+1}$ . A *path* is a walk where all edges are distinct. A *simple path* is a path where all vertices are distinct. The term *u-v path* (resp., *simple path*) refers to any valid path (resp., *simple path*) from  $u$  to  $v$ .

In flow networks, we usually distinguish between single-commodity and multi-commodity flows. For single-commodity flow, we consider a single commodity that consists of a source  $s \in V$  and a sink  $t \in V$ , where  $s \neq t$ . For multi-commodity flows, we assume  $L$  commodities of the form  $(s_i, t_i)$ , where  $s_i, t_i \in V, s_i \neq t_i$ . Each commodity  $i$  is associated with a non-negative demand  $D_i \geq 0$ . For convenience, we also use the notation  $\mathbf{s} = (s_1, \dots, s_L)$  and  $\mathbf{t} = (t_1, \dots, t_L)$ , and write  $(\mathbf{s}, \mathbf{t})$  to denote the corresponding multi-commodity network.

#### B. TE Type 1: $TE_{MF}$

Let  $\mathcal{P}_i$  be the set of all  $s_i$ - $t_i$  paths, and  $\mathcal{P}_{i,e}$  the set of all  $s_i$ - $t_i$  paths that go through edge  $e$ . Then the maximum multi-commodity flow program can be expressed by the following path-based formulation:

$$\text{maximize} \quad \nu = \sum_{i=1}^L \sum_{p \in \mathcal{P}_i} f_i(p) \quad (1)$$

$$\text{subject to} \quad \sum_{i=1}^L \sum_{p \in \mathcal{P}_{i,e}} f_i(p) \leq c(e), \forall e \in E \quad (2)$$

$$\sum_{p \in \mathcal{P}_i} f_i(p) \leq D_i \quad (3)$$

$$f_i(p) \geq 0, \forall i \in \{1, \dots, L\}, \forall p \in \mathcal{P}_i \quad (4)$$

The commodity flow  $i$  along path  $p \in \mathcal{P}_i$  is  $f_i(p)$ . Constraint (2) is a capacity constraint that the sum of all sub-flows on any edge cannot exceed the edge capacity. Constraint (3) simply describes the maximum demand  $D_i$  for commodity  $i$ . Finally, constraint (4) simply imposes that the flow should be non-negative. For any valid flow  $f$ , the value of a flow  $\nu(f)$  is defined as the total sum of units that all sub-flows  $f_i$  send. The maximum flow is then simply defined as  $\nu_{max}$ .  $TE_{MF}$  is mostly used in data center backbone WANS [30], [31], where traffic is elastic and the main objective is to fully utilize the expansive WAN links.

Note that even though the single-commodity maximum flow accepts various combinatorial algorithms [3], e.g., Ford-Fulkerson or Edmonds-Karp, there is to date no combinatorial algorithm for the maximum multi-commodity flow even though the problem is known to be strongly polynomial due to Tardos [46]. Furthermore, even though single-commodity networks always accept an integer maximum flow, this is not always the case with multi-commodity networks; in fact, the decision problem of integral multi-commodity flow is NP-complete even if the number of commodities is two, for both the directed and undirected cases [14].

#### C. TE Type 2: $TE_{LU}$

$TE_{LU}$  is mostly used in carrier networks [22], [28], where traffic demands are given and inelastic, and the main objective thus is to control the congestion or link utilization in order to

ensure the smooth operation of the network. The general form for this type of TE is:

$$\text{minimize } \theta \quad (5)$$

$$\text{subject to } \sum_{i=1}^L \sum_{p \in \mathcal{P}_{i,e}} f_i(p) \leq \theta \cdot c(e), \forall e \in E \quad (6)$$

$$\sum_{p \in \mathcal{P}_i} f_i(p) \geq D_i \quad (7)$$

$$f_i(p) \geq 0, \forall i \in \{1, \dots, L\}, \forall p \in \mathcal{P}_i \quad (8)$$

The variable  $\theta$  in the objective function (5) refers to the maximum link utilization, which must be minimized. Constraint (6) ensures that  $\theta$  will be at least as large as the maximum link utilization; constraint (7) ensures that each demand is satisfied; and the last constraint (8) is similar to  $TE_{MF}$  in Section III-B.

#### D. An Interesting Connection

Having introduced the two types of TE, a natural question is to ask whether they are related. To answer this question, we introduce the following *decision version* of the maximum multi-commodity flow problem.

**Definition 1.** [Decision version of maximum flow (DMF)] Given a flow network  $G = (V, E, c)$  with a set of  $L$  commodities  $(s, t)$ , each associated with a non-negative maximum demand  $D_i \geq 0$ , decide whether the maximum multi-commodity flow has a value of at least  $\sum_{i=1}^L D_i$ .

Note that is the answer to the decision problem DMF is a “yes”, then by constraint (3) the maximum flow has to be exactly equal to  $\sum_{i=1}^L D_i$ . If the answer is no, then the maximum flow is strictly less than  $\sum_{i=1}^L D_i$ .

We are now ready to establish the following result that reveals the relationship between the two types of TE:

**Lemma 1.** DMF accepts a “yes” answer, if and only if the system (5)–(8) for  $TE_{LU}$  accepts a solution  $\theta^* \leq 1$ .

*Proof.* Assume that DMF accepts a “yes” answer. Then there is a flow that respects constraints (2)–(4). That flow will then trivially satisfy constraints (2)–(4) with  $\theta = 1$ . Since the objective criterion of  $TE_{LU}$  minimizes over  $\theta$ , the optimal solution to the TE program (5)–(8) will accept an optimal solution  $\theta^* \leq 1$ .

For the reverse direction, assume that the system (5)–(8) accepts a solution  $\theta^* \leq 1$ . Then constraint (6) implies that the capacity constraints are satisfied for each edge, thus the corresponding flow is a valid flow for system (1)–(4) with value  $\sum_{i=1}^L D_i$ . The maximum flow has then trivially a value of at least  $\sum_{i=1}^L D_i$ .  $\square$

Lemma 1 shows that solving  $TE_{LU}$  immediately generates a “yes” or “no” answer to the DMF. Thus, the TE naturally

encompasses the general DMF problem of Definition 1. This also suggests that hardness results on the DMF (Proposition 1) immediately imply hardness for  $TE_{LU}$ . We describe this in more detail in the next section.

To conclude this section, notice that even though we assumed a directed network throughout Section III, it is possible to extend the definitions to undirected graphs as well. The main difference is that an undirected edge is associated with a capacity, and flow can travel in both directions of a link, under the constraint that the sum of the flow value in the two edge directions does not exceed the capacity. Despite the innocuous difference, our analysis in Appendix B uncovers a profound dichotomy between the two cases with regard to maximum flow.

## IV. HARDNESS OF GENERAL SEGMENT ROUTING

In this section, we present the first part of our theoretical investigation. We study the general form of applying segment routing to TE, where traffic can take any path that goes through a midpoint. We prove an important new result that it is NP-hard to decide if the maximum flow through just one given midpoint is greater than 0. Due to the connection between the decision version of maximum flow and  $TE_{LU}$ , this also means that  $TE_{LU}$  given a single fixed midpoint is already NP-hard. This then motivates us to consider using segment routing with shortest paths only in the next section.

### A. Hardness of $TE_{MF}$

The maximum multi-commodity flow  $f_{max}$  with value  $\nu_{max}$  refers to the total flow over all possible paths that each commodity accepts. Assume instead that we focus on the maximum flow that can go through a specific network node, e.g.,  $w \neq s, t$ . Let  $\mathcal{P}_i^w$  be the set of all  $s_i$ - $w$ - $t_i$  paths (i.e.  $s_i$ - $t_i$  paths that go through  $w$ ), and  $\mathcal{P}_{i,e}^w$  the set of all  $s_i$ - $w$ - $t_i$  paths that also go through edge  $e$ . The path-based formulation then is:

$$\begin{aligned} &\text{maximize } \nu^w = \sum_{i=1}^L \sum_{p \in \mathcal{P}_i^w} f_i(p) \\ &\text{subject to } \sum_{i=1}^L \sum_{p \in \mathcal{P}_{i,e}^w} f_i(p) \leq c(e), \forall e \in E \\ &f_i(p) \geq 0, \forall i \in \{1, \dots, L\}, \forall p \in \mathcal{P}_i^w \end{aligned}$$

We denote the maximum flow through any node  $w$  as the *maximum  $w$ -flow*  $f_{max}^w$  and denote its value by  $\nu_{max}^w$ . Alternatively, we use the notation  $s$ - $w$ - $t$  flow for single-commodity networks (or  $s$ - $w$ - $t$  for multi-commodity networks). Similarly, for single-commodity flows we also write  $\nu_{max}^w(s, t)$  (or  $\nu_{max}^w(s, t)$  for multi-commodity networks) for the value of the maximum  $w$ -flow.

Note that in the single-commodity case we always assume that  $w \neq s, t$ , even if not explicitly stated. Indeed, if either  $w = s$  or  $w = t$  then  $\nu_{max} = \nu_{max}^w$ . In this case, the problem is strongly polynomial and accepts combinatorial algorithms such as the Ford-Fulkerson algorithm.

A central result in graph theory that we will be using throughout the paper is the two node-disjoint path (2DP) problem due to Fortune, Hopcroft and Wyllie [21].

**Theorem 1** (NP-hardness of 2DP [21]). *Assume a directed graph  $G = (V, E)$  and four distinct vertices  $u_1, u_2, v_1, v_2 \in V$ . It is NP-hard to decide whether there are two node-disjoint paths in  $G$  from  $u_1$  to  $u_2$  and from  $v_1$  to  $v_2$ .*

We are now ready to provide two lemmas.

**Lemma 2.** *Computing whether there is a simple  $s$ - $w$ - $t$  path in a directed graph  $G = (V, E)$ , where  $w, s, t \in V$  and  $w \neq s, t$ , is NP-hard.*

*Proof.* Finding whether there is a simple  $s$ - $t$  path that goes through a node  $w$  is equivalent to determining whether there are two node-disjoint paths from  $s$  to  $w$  and from  $w$  to  $t$  (excluding of course node  $w$ ). We prove that the latter problem is NP-hard by a reduction from the NP-hard 2DP problem.

Assume a directed graph  $G = (V, E)$  and nodes  $u_1, u_2, v_1, v_2 \in V$ . We introduce a new node  $w$  and create the new edges  $e_1 = (u_2, w)$  and  $e_2 = (w, v_1)$ . We now argue that there are two node-disjoint paths from  $u_1$  to  $u_2$  and from  $v_1$  to  $v_2$ , if and only if there is a simple  $u_1$ - $w$ - $v_2$  path.

Indeed, if the former condition is satisfied, then we can know for sure that the path from  $u_1$  to  $u_2$  cannot go through node  $w$  through edge  $e_1$  since otherwise that path would also have to use node  $v_1$  after  $w$  since  $u_2$  is the end node. Similarly we can argue that the path from  $v_1$  to  $v_2$  cannot go through node  $w$  through edge  $e_2$ . But then we can form a new path from  $u_1$  to  $v_2$  by concatenating the path  $u_1$  to  $u_2$ , edge  $e_1$ , edge  $e_2$ , and finally the path from  $v_1$  to  $v_2$ . This path does not repeat any node since the node disjoint paths from  $u_1$  to  $u_2$  and from  $v_1$  to  $v_2$  do not contain  $w$ , hence it is a simple path. For the reverse direction, we just note that if there exists a simple  $u_1$ - $w$ - $v_2$  path, then this path will necessarily contain edges  $e_1$  and  $e_2$ . By removing these two edges, we get two node-disjoint paths, one from  $u_1$  to  $u_2$  and another from  $v_1$  to  $v_2$ , since the  $u_1$ - $w$ - $v_2$  path is simple.  $\square$

**Lemma 3.** *Computing whether there is a  $s$ - $w$ - $t$  path in a directed graph  $G = (V, E)$ , where  $w, s, t \in V$  and  $w \neq s, t$ , is NP-hard.*

*Proof.* Finding whether there is a  $s$ - $t$  path that goes through a node  $w$  is equivalent to determining whether there are two edge-disjoint paths from  $s$  to  $w$  and from  $w$  to  $t$ . We now argue that the latter problem is NP-hard by a reduction from the 2DP problem.

Indeed, assume any graph  $G = (V, E)$ , and three distinct nodes  $s, t, w \in V$ . We construct a new graph  $G' = (V', E')$  from  $G = (V, E)$  in the following manner. For each node  $v \in V$  we introduce two nodes  $v_{in}, v_{out} \in V'$ . For each edge  $e = (u, v) \in E$ , we introduce an edge  $e' = (u_{out}, v_{in}) \in E'$ . Moreover,  $E'$  contains an edge  $e'' = (v_{in}, v_{out})$  connecting each pair of nodes  $v_{in}, v_{out}$  in  $G'$  that we constructed above. We now make the statement that there are two edge-disjoint paths in graph  $G'$  from  $s_{out}$  to  $w_{in}$  and from  $w_{out}$  to  $t$ , if and only if there are two node-disjoint paths in  $G$  from  $s$  to  $w$  and from  $w$  to  $t$ .

Indeed, consider first any two node-disjoint paths in  $G$ , namely,  $s, u_1, \dots, u_l, w$  and  $w, v_1, \dots, v_m, t$ , where all intermediate nodes  $u_i$  and  $v_j$  are distinct. Then it is easy to see that the paths  $s_{out}, u_{1,in}, u_{1,out}, \dots, u_{l,in}, u_{l,out}, w_{in}$  and  $w_{out}, v_{1,in}, v_{1,out}, \dots, v_{m,in}, v_{m,out}, t_{in}$  in  $G'$  are: (1) valid since they use existing edges in  $G'$ , and (2) edge-disjoint since the set of nodes on the first path is disjoint to the set of nodes in the second. For the reverse direction, consider two edge-disjoint paths in  $G'$  from  $s_{out}$  to  $w_{in}$  and from  $w_{out}$  to  $t$ . We then argue that these paths necessarily have the above form  $s_{out}, u_{1,in}, u_{1,out}, \dots, u_{l,in}, u_{l,out}, w_{in}$  and  $w_{out}, v_{1,in}, v_{1,out}, \dots, v_{m,in}, v_{m,out}, t_{in}$ . The reason is that any pair of nodes  $(v_{in}, v_{out})$  can only be reached from other nodes in  $V'$  via  $v_{in}$  and can only reach other nodes in  $V'$  via  $v_{out}$ . So, a path will necessarily consist of consecutive pairs of nodes of the form  $(v_{in}, v_{out})$  (with the exception of the two endpoints). Furthermore, any such pair  $(v_{in}, v_{out})$  can (1) appear at most once on either path, and (2) cannot appear on both paths. The reason is that going from  $v_{in}$  to  $v_{out}$  requires edge  $v_{in}, v_{out}$ , but the two paths are edge-disjoint. But then it holds that the paths  $s, u_1, \dots, u_l, w$  and  $w, v_1, \dots, v_m, t$  in  $G$  are both valid and node-disjoint. This completes the proof.  $\square$

We next provide definitions and results for the maximum  $w$ -flow that are reminiscent of results in traditional single-commodity maximum flow. One significant difference is that the *cut* is now defined as a collection of edges rather than a collection of nodes. We focus first on the  $s$ - $w$ - $t$  flow in single-commodity networks.

**Definition 2.** *A  $s$ - $w$ - $t$  edge-cut is a subset of edges  $C^w \subseteq E$  such that removing the edges in  $C^w$  from the graph results in no  $s$ - $w$ - $t$  paths, i.e., there are no  $s$ - $w$ - $t$  paths in the graph  $G' = (V, E - C^w)$ . The value  $c(C^w)$  of the edge-cut is defined as the sum of the capacities of all edges in  $C^w$ .*

**Lemma 4.** *Let  $f^w$  be any  $s$ - $w$ - $t$  flow, and  $C^w$  any  $s$ - $w$ - $t$  cut. Then  $\nu^w(f^w) \leq c(C^w)$ .*

*Proof.* First, note that the flow  $f^w$  is the sum of individual flows, each going through a distinct  $s$ - $w$ - $t$  path  $p$ . Each of these individual flows must go through at least some edge in  $e \in C^w$ , otherwise there would be a  $s$ - $w$ - $t$  path in the graph  $G' = (V, E - E')$ . So, let  $\mathcal{F}_e$  be the set of flows that go through  $e$ . Then we have:

$$\begin{aligned} \sum_{e \in C^w} \nu^w(\mathcal{F}_e) &\leq \sum_{e \in C^w} c(e) \Leftrightarrow \\ \sum_{e \in C^w} \nu^w(\mathcal{F}_e) &\leq c(C^w) \Leftrightarrow \\ \nu^w(f^w) &\leq c(C^w) \end{aligned} \quad (9)$$

Note that  $\sum_{e \in C^w} \nu^w(\mathcal{F}_e) \leq \nu^w(f)$ , since we argued that the path for each individual flow must go through at least one edge in  $C^w$ .  $\square$

**Lemma 5.** *Given a flow network  $G = (V, E, c)$  and three distinct nodes  $s, w, t$ , the maximum value of any  $s$ - $w$ - $t$  flow is equal to the minimum capacity of any  $s$ - $w$ - $t$  edge-cut.*

*Proof.* Consider a variation of the well-known Ford-Fulkerson algorithm for (single-commodity) maximum flow [43], where at each round the algorithm picks a  $s$ - $w$ - $t$  path rather than just a  $s$ - $t$  path. The augmenting  $s$ - $w$ - $t$  path algorithm terminates, if and only if there is an  $s$ - $w$ - $t$  edge-cut in the graph where each edge  $e \in \mathcal{C}^w$  is saturated. Next, note that the total flow comprised of all individual path flows must be at least as large as the capacity of that edge-cut. But since by Lemma 4 the value of any flow can be at most as large as the value of any  $s$ - $w$ - $t$  edge-cut, it must necessarily hold that the flow that the augmenting path algorithm returns is maximal and equal to the capacity of the minimum  $s$ - $w$ - $t$  edge-cut.  $\square$

**Corollary 1.** *For integral capacities, there is a maximum  $s$ - $w$ - $t$  flow that is integral.*

*Proof.* Note that by Lemma 5 the augmenting  $s$ - $w$ - $t$  path algorithm returns a maximal flow and, furthermore, at each step the flow on any edge is integral. Thus, there must be a maximal flow whose value on any edge is integral.  $\square$

We are now ready to prove that the decision problem of whether the maximum flow through any specific node is greater than 0 is NP-hard under integral demands.

**Proposition 1.** *Given a multi-commodity flow network  $G = (V, E, c)$ , it is NP-hard to even decide whether  $\nu_{max}^w(s, t) > 0$ .*

*Proof.* We show that even the single-commodity version of our problem is NP-hard. Since the single-commodity case can be seen as a special case of the multi-commodity problem, our result immediately implies hardness for the multi-commodity maximum flow as well. Our strategy will be to reduce the  $s$ - $w$ - $t$  path problem in Lemma 3 to the single-commodity maximum flow problem with integral demands.

Indeed, consider a directed graph  $G = (V, E)$  and three distinct nodes  $s, t, w \in V$ . We construct a flow-network  $G'$  from  $G$  in the following manner. We consider one commodity from  $s$  to  $t$ , and we further assume that each edge  $e \in E$  has a capacity of 1. We then argue that there is a path from  $s$  to  $t$  through  $w$ , if and only if the maximum flow through  $w$  in  $G'$  is greater than 0.

First, assume there is a path  $p = (s, e_1, \dots, e_m, t)$ , so that every edge  $e_i$  in the path appears only once and node  $w$  appears in the path. It is then possible to send one unit of flow from  $s$  to  $t$ , given the unit capacities. Thus, the maximum multi-commodity flow will be at least 1, and thus for sure greater than 0. For the reverse direction, assume there is a maximum flow greater than 0 in  $G'$ . Since we only have one commodity, it follows that there has to be a maximum flow where the flow value on every edge is integral. Moreover, note that each edge can be used at most once in that flow due to its unit capacity. Now, consider any path  $p_{(s,w)}$  that carries flow from  $s$  to  $w$ , and any path  $p_{(w,d)}$  that carries flow from  $w$  to  $d$ . Since each edge is used at most once, this means that the path  $p_{(s,w)} \cup p_{(w,d)}$  has the properties: (1) it goes from  $s$  to  $d$  through  $w$ , and (2) it visits any edge at most once. Thus, there must be a path from  $s$  to  $t$  through  $w$ .  $\square$

**Corollary 2.** *Given a flow network  $G = (V, E, c)$  with a single commodity  $(s, t)$  and a node  $w \in V$ , it is NP-hard to compute the minimum  $s$ - $w$ - $t$  cut  $\mathcal{C}^w$ .*

*Proof.* We can show that even for single commodity-networks the problem is NP-hard. Indeed, by Lemma 5 we know that the maximum value of any  $s$ - $w$ - $t$  flow is equal to the minimum capacity of any  $s$ - $w$ - $t$  edge-cut. But since by Proposition 1 it is NP-hard to compute the maximum  $s$ - $w$ - $t$  flow, it must also be NP-hard to compute the value of the minimum  $s$ - $w$ - $t$  cut.  $\square$

Perhaps unexpectedly, the above hardness results do not hold when the underlying graph is undirected. Appendix B discusses this dichotomy in detail.

### B. Hardness of $TE_{LU}$

For the flow network  $G$  with the  $L$  commodities, we define the  $TE_{LU}$  in the following manner:

$$\min \quad \theta \quad (10)$$

$$\text{subject to} \quad \sum_{i=1}^L \sum_{p \in \mathcal{P}_{i,e}^w} f_i(p) \leq \theta \cdot c(e), \forall e \in E \quad (11)$$

$$\sum_{i=1}^L \sum_{p \in \mathcal{P}_i^w} f_i(p) \geq 0 \quad (12)$$

$$f_i(p) \geq 0, \forall i \in \{1, \dots, L\}, \forall p \in \mathcal{P}_i^w \quad (13)$$

Lemma 1 shows that solving the  $TE_{LU}$  immediately generates a “yes” or “no” answer to the corresponding DMF. Thus, the hardness results on the DMF (Proposition 1) immediately imply hardness for the corresponding TE:

**Corollary 3.** *It is NP-hard to solve the  $TE_{LU}$  (10)–(13).*

## V. SEGMENT ROUTING WITH SHORTEST PATHS

Given the NP-hardness of applying general segment routing in TE, here we consider TE with shortest path based segment routing. That is, now traffic is routed only along the shortest paths for a given segment. In this sense, our results provide for the first time theoretical foundation for existing work that focuses on shortest path based segment routing [5], [28].

### A. Network model and TE

Assume there are in total  $K$  middle points available. Each end-to-end path can use up to  $M \leq K$  of these middle points. For a segment  $s \in S$  between an ingress node and a midpoint, two midpoints, or a midpoint and an egress node, there are multiple paths in general. We assume, for simplicity, that routing is done by ECMP over all shortest paths of a segment. This is consistent with prior work [5]. We use  $T_i$  to denote the complete set of logical tunnels formed by segments in  $S$  that can be used for commodity  $i$ , with up to  $M$  middle points. A tunnel involves only ingress/egress switch, and the intermediate midpoints. This can be constructed offline efficiently.

Let  $G_{t,s}$  denote if a tunnel  $t$  uses segment  $s$  or not, and  $I_{p,e}$  denote if path  $p$  uses link  $e$  or not. Furthermore, let  $\hat{P}_s$  be the

set of all shortest paths for segment  $s$ , and  $f_i(t)$  represent the flow in tunnel  $t$  for commodity  $i$ . The split ratio  $x_{i,t}$  for  $i$  on tunnel  $t$  is defined as the ratio  $x_{i,t} = \frac{f_i(t)}{\sum_{t \in T_i} f_i(t)}$ .

The  $TE_{LU}$  problem with segment routing can be formulated similar to Section III-C, where the set of paths  $\mathcal{P}_i$  for commodity  $i$  is now replaced by the set of logical tunnels  $T_i$ :

$$\min \quad \theta \quad (14)$$

$$\text{s.t.} \quad \sum_{i=1}^L \sum_{t \in T_i} \sum_{s \in S_t} \sum_{p \in \hat{P}_s} f_i(t) \frac{I_{p,e}}{|\hat{P}_s|} \leq \theta \cdot c(e), \forall e \in E, \quad (15)$$

$$0 \leq f_i(t), \forall i \in \{1, \dots, L\}, t \in T_i, \quad (16)$$

$$\sum_{t \in T_i} f_i(t) \geq D_i, \forall i \in \{1, \dots, L\}. \quad (17)$$

The capacity constraint (15) indicates that the total traffic routed to link  $e$  from across all flows, tunnels, segments, and shortest paths, cannot exceed  $\theta$  times the link capacity. Since ECMP is used for routing within any segment  $s$ , each shortest path  $p$  of segment  $s$  receives flow equal to  $f_i(t)/|\hat{P}_s|$ .

Regarding the TE asymptotic complexity, we have the following result when  $M$  is fixed and not part of the input:

**Proposition 2.** *For fixed  $M$  with respect to the input graph  $G$ , the  $TE_{LU}$  problem described by Equations (14)-(17) can be solved in (weakly) polynomial time.*

*Proof.* The number of commodities  $L$  cannot exceed  $|V| \cdot (|V| - 1)$ , and the number  $|T_i|$  of tunnels per commodity  $i$  is upper bounded by  $\binom{K}{0} + \dots + \binom{K}{M}$ , where  $K \leq |V|$ . For fixed  $M$  w.r.t. the input graph  $G$ ,  $|T_i|$  has polynomial size w.r.t. the graph. Finally, the number  $S_t$  of segments per tunnel cannot exceed  $K + 1 \leq |V| + 1$ , since a tunnel can use at most all  $K$  middlepoints. For the inner sum  $\sum_{p \in \hat{P}_s} \frac{I_{p,e}}{|\hat{P}_s|}$ , note that it basically denotes the percentage of shortest paths for segment  $s$  that use link  $e$ . However, this can be computed in polynomial time, e.g. by using the techniques in [7].

Thus, we have proved that for fixed  $M$ , the LP has a polynomial number of variables, and a polynomial number of constraints whose coefficients can be computed in polynomial time. The proposition then immediately follows by standard results in linear programming [33], [34].  $\square$

Given that the  $TE_{MF}$  formulation is very similar, we can similarly prove that it can also be solved in (weakly) polynomial time for segment routing with shortest paths.

Finally, we observe that the TE problem is naturally related to the shortest path centrality that we discussed in Section VI. Indeed, the inner part  $\sum_{p \in \hat{P}_s} \frac{I_{p,e}}{|\hat{P}_s|}$  of constraint (15) precisely describes the percentage of shortest paths for segment  $s$  that use a specific edge, and note that we do that for all possible segments. Even though shortest path centrality refers to a node rather than an edge and equally takes into account all possible source-destination pairs, constraint (15) reveals interesting connections between the popular centrality metric and the segment routing problem.

## B. Hardness of Acyclic Segment Routing

So far, we have focused on segment routing where routing on a segment is based on ECMP. One challenge is that this generally produces source destination paths with edge repetitions, i.e., walks. Even in the case of just one middle point per path, it is possible that a (simple) shortest path from the source  $s$  to a middle point  $M$  shares an edge  $e$  with a shortest path from  $M$  to the destination  $d$ . So, even though the paths for any given segment are simple, the resulting  $s$ - $d$  path may not even be a path. This may reduce the performance of segment routing, because it increases the link load on the reused edges and may lead to higher link utilization.

In that case a natural question arises: what if we consider segment routing with shortest paths for segments, under the condition that the resulting walk from the source to the destination is a path or even a simple path? As our subsequent analysis shows, traffic engineering will become NP-hard, even for just one commodity. To prove this fact, we first introduce the following fundamental result due to Eilam-Tzoref [12].

**Theorem 2** (NP-hardness of  $k$ DSP [12]). *Given a graph  $G = (V, E)$  and  $k$  pairs of distinct vertices  $(u_i, v_i)$ ,  $1 \leq i \leq k$ , the  $k$ DSP problem of computing  $k$  pairwise disjoint shortest paths  $P_i$  between  $u_i$  and  $v_i$  is NP-complete, when  $k$  is part of the input. This result holds for all four versions of the  $k$ DSP problem, namely, node or edge-disjoint paths for directed or undirected graphs.*

**Proposition 3.** *The  $TE_{MF}$  and  $TE_{LU}$  problems in a directed or undirected graph with  $K$  middle points (1) using only shortest path segment routing and (2) only allowing paths or simple paths from a source to a destination, are NP-hard, even for just one commodity, when  $K$  is part of the input.*

*Proof sketch.* We can show the statement by making similar arguments as for general segment routing TE in Section IV. For  $TE_{MF}$ , the idea is to first show that we can solve  $k$ DSP if and only if we can solve the corresponding  $TE_{MF}$  formulation.

Indeed, assume a directed or undirected graph  $G = (V, E)$ , two distinct nodes  $s, t$  in  $V$ , and  $K$  distinct nodes  $s \neq M_i \neq t$  in  $V$ ,  $1 \leq i \leq K$ . Consider we do segment routing from  $s$  to  $t$  using nodes  $M_i$  as our  $K$  middle points. We will show that the  $TE_{MF}$  problem is NP-hard by a reduction from the  $k$ DSP problem.

Concretely, assume for instance the  $k$ DSP node-disjoint problem in Theorem 2. We construct a new graph  $G'$  as follows. For each  $i$ ,  $1 \leq i \leq K - 1$ , we introduce a new node  $M_i$  along with the two (directed or undirected) edges  $e_{in}^i = (v_i, M_i)$  and  $e_{out}^i = (M_i, u_{i+1})$ . Moreover, we associate each edge in  $G'$  with a positive capacity, and we assume the single commodity  $(s, t) = (u_1, v_K)$  with a positive demand  $D > 0$ . We now argue that there are  $K$  node-disjoint shortest paths between  $u_i$  and  $v_i$ , if and only if  $TE_{MF}$  with the single commodity  $(s, t)$  and the  $K - 1$  middle points  $M_1, \dots, M_{K-1}$  accepts a positive solution (maximum flow).

But this can be proven using very similar techniques as in Section V-B. The only difference is that the minimum edge cut in this case corresponds to the minimum sum of edge

capacities whose removal results in no path (or simple path) using shortest paths from the source to the destination through the middle points.

Finally, NP-hardness for  $TE_{LU}$  then follows immediately by Lemma 1, in a similar spirit as Corollary 3.  $\square$

Interestingly, Proposition 3 is general and holds for both directed and undirected graphs. We emphasize however that it assumes that the number of middle points  $K$  is part of the input. For  $k = 2$ , [12] provides a polynomial algorithm for the undirected case of  $k$ DSP, whereas the complexity for the directed case when  $k = 2$  remains open.

## VI. CENTRALITY BASED MIDDLEPOINT SELECTION

The previous sections investigated the fundamentals of segment routing, and showed that the TE for unrestricted segment routing is NP-hard. On the other hand, Proposition 2 suggests that if we only allow a fixed number  $M$  of middlepoints per path, then TE with shortest paths is (weakly) polynomially computable.

One approach would then be to consider all nodes as candidate middlepoints, i.e.  $K = |V|$ . However, that results in very large TE programs that are costly to solve. An alternative is to just consider a small number of middlepoints such that  $K \ll |V|$ , that would still produce good output for the TE. Given that this is generally NP-hard [28], in this section we discuss practical middlepoint selection based on alternative centrality measures with polynomial complexity. Note that these centralities are *structural* metrics that look at the graph structure, i.e., the connections among the various nodes. However, they generally do not take into account the flow network and its flow conservation and capacity constraints, which was the case with the NP-hard flow centralities in Appendix A.

**Shortest-path centrality.** We start with *shortest-path* centrality, which characterizes the power of a node in terms of the number of shortest paths that go through that node for a randomly picked source-destination pair. Concretely, assume a directed graph  $G = (V, E)$ . The shortest-path betweenness centrality of a node  $v \in V$  [24] is defined as:

$$\delta(v) = \sum_{s,t \in V | s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}, \quad (18)$$

where  $\sigma_{st}(v)$  is the number of shortest paths from  $s$  to  $t$  that go through  $v$ , and  $\sigma_{st}$  the total number of shortest paths from  $s$  to  $t$ . Calculating the shortest path centrality of all vertices in a graph requires  $\Theta(|V|^3)$  time and  $\Theta(|V|^2)$  space. This can be achieved by augmenting the Floyd-Warshall algorithm for the all-pairs shortest-paths problem with path counting. Brandes's algorithm improves these bounds by only using  $O(|V| + |E|)$  space and running in  $O(|V| + |E|)$  and  $O(|V| \cdot |E| + |V|^2 \log |V|)$  time on unweighted and weighted networks, respectively [7].

**Group shortest-path centrality.** As opposed to the aforementioned *individual* centrality, the *group* shortest-path be-

tweenness centrality of a group of nodes  $C \subseteq V$  refers to the combined centrality of the group [15]. It is defined as:

$$\delta_G(C)(v) = \sum_{s,t \in V | s \neq v \neq t} \frac{\sigma_{st}(C)}{\sigma_{st}}, \quad (19)$$

where  $\sigma_{st}(C)$  the number of shortest paths that go through *any node* in  $C$ . Group betweenness centrality can be approximated within a factor  $1 - \frac{1}{e}$  to the optimal using a greedy incremental algorithm [10]. Brandes' algorithm for computing the betweenness centrality of all vertices can be modified to compute the group betweenness centrality of one group of nodes with the same asymptotic running time [44].

**Degree centrality.** A simple alternative to the family of shortest path centralities is *degree* centrality. The degree centrality of a node  $v \in V$  is defined as the average of its in-degree and its out-degree:

$$d(v) = \frac{|v^+| + |v^-|}{2} \quad (20)$$

Degree centrality captures a node's power by its number of neighbors; the higher that number, the better connected the node and the larger its centrality. Despite its simplicity, degree centrality can capture to a good extent a node's structural importance.

**Weighted centralities.** All aforementioned centralities only employ the graph connectivity information, and treat all links equally. However, in practice links are further characterized by their capacity. We can thus define variants of the previous centralities that additionally take into account the link capacity information. A simple approach is to associate each edge with the non-negative cost  $\frac{1}{c(e)}$ . This is based on the observation that the higher the capacity, the lower the cost of the link since it can accommodate larger flows. The shortest path centrality variants are simple to define, if we note that the cost of a path is the sum of the costs of its constituent links, and the shortest path refers to the path with the minimum cost among all paths. In a similar spirit, we can define the weighted degree of any node as the sum of the costs of the edges that are incident to the node. Intuitively, we expect that the weighted variants should perform better since they take into account both the connectivity and the capacity information. Section VII-E empirically confirms our intuition.

## VII. EVALUATION

In this section, we conduct trace-driven simulations to evaluate the performance of centrality based middlepoint selection methods. The experiments are designed to answer the following important questions:

- What is the best parameter setting for centrality based middlepoint selection? Specially, how many middlepoints per commodity, and how many middlepoints in total should we use?
- How does our centrality based approach compare to existing work, in terms of both performance and complexity?
- How do various centrality definitions perform against each other?

## A. Methodology

We use two network topologies from the dataset provided by DEFO (Declarative and Expressive Forwarding Optimizer) [2] which is used in [28]. One is a synthetic network with 100 nodes (*synth100* in the dataset), and the other is a real network with 161 nodes (*rf3257* in the dataset). Table II provides more details about the networks. The DEFO dataset also contains information of commodity flows (simply referred to as flows hereafter) for these topologies. For the real 161-node topology the flows are provided by the ISP [28]. For the synthetic topology the demand matrices are computed using the approach in [45]. As explained in [28], this approach uses a gravity model fed with i.i.d. exponential random variables. It produces realistic demand matrices as shown in [28], [45].

Type	ID	# nodes	# links	# flows
Synthetic	<i>synth100</i>	100	572	9817
Real	<i>rf3257</i>	161	656	25486

TABLE II  
DATASET SUMMARY.

We perform simulations on servers each with a 2.2 GHz 64-bit 8-Core Xeon processor and 128 GB memory. We use the *cvxpy* [1] modeling language with the ECOS solver [11] to solve the LPs. Our evaluation compares the following schemes:

- *Baseline*: Traditional approach of applying segment routing studied in Sec. IV of [5]. Specifically, the TE problem assumes that every node is a candidate of midpoint, and exactly one midpoint is used for each flow. Only shortest paths are used for segment routing.
- *Random*: Our approach where a total of  $K$  middlepoints are randomly selected and used in TE problems  $TE_{MF}$  and  $TE_{LU}$ .
- *Shortest-path centrality (SP)*: Our approach where middlepoints are selected using shortest-path centrality as explained in §VI for TE.
- *Group shortest-path centrality (GSP)*: Our approach where middlepoints are selected using group shortest-path centrality.
- *Degree centrality (Degree)*: Our approach where middlepoints are selected using degree centrality.

## B. Microscopic Performance

First we aim to understand the microscopic performance of our centrality based approach. There are two key parameters affecting our approach in general: the number of middlepoints per flow  $M$ , and the total number of available middlepoints for all flows  $K$ . Their effects need to be thoroughly understood before we compare our approach to existing methods.

**Number of middlepoints per flow  $M$ :** We begin by trying to answer how many middlepoints should be used for each flow. Note that there is an inherent tradeoff: more middlepoints per flow leads to more flexibility in constructing the paths and balancing the traffic, and thus better performance. On the other hand it also means more overhead in terms of higher complexity of the TE algorithms.

To demonstrate this tradeoff, we use  $TE_{LU}$  and compute the maximum link utilization with our centrality based midpoint selection when the number of middlepoints per flow  $M$  is equal to 1 or 2. We use the *synth100* network with 100 nodes. We choose 1000 flows for the topology randomly for ten times and report the average. We apply *GSP* to select the middlepoints, and vary the total number of available middlepoints  $K$  from 2 to 6. For a given  $K$ , the flows are identical for different values of  $M$ . Note we also experiment with  $TE_{MF}$  and the 161-node network; the results are qualitatively similar and omitted here for space.

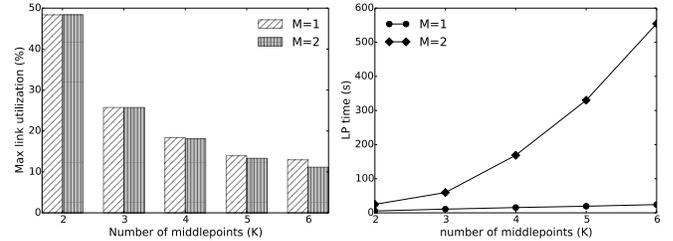


Fig. 3. Maximum link utilization of the 100-node network with 1000 flows and varying  $M$ . Fig. 4. LP solving time of  $TE_{LU}$  on the 100-node network with 1000 flows and varying  $M$ .

Fig. 3 and Fig. 4 depict the results. We find that interestingly, the maximum link utilizations for  $M = 1$  and  $M = 2$  are quite similar. Yet the time of solving the TE with  $M = 2$  is much higher than  $M = 1$  as shown in Fig. 4 (a difference of almost 20x) when  $K = 6$ ). Given that the middlepoints are central, there is indeed a low probability that a bottleneck link exists between two middlepoints. Hence, maximum link utilization is largely determined by the bottleneck links between either the source and the midpoint, or between the midpoint and the destination. Therefore we conclude that 1 midpoint per flow is good enough for performance, and use that throughout the remainder of the experiments.

**Number of total middlepoints  $K$ :** We next run experiments to verify that just a few central middlepoints are sufficient to achieve satisfactory TE performance. We vary the total number of middlepoints  $K$  from 1 to 6 for  $TE_{LU}$  and from 1 to 8 for  $TE_{MF}$ . The middlepoints are selected using *Random*, *SP*, *GSP*, and *Degree* as explained in §VII-A. We use both the 100-node and 161-node networks, and randomly choose 1000 flows and 2000 flows respectively for 10 runs. We report the average and standard deviation results. Since *Random* is non-deterministic, we randomly select 5 sets of middlepoints for each of the 10 flow sets, resulting in 50 runs in total for *Random*. For  $TE_{MF}$ , in order to make the results more readable, we scale the traffic volumes by 10 times for 100-node topology and 40 times for 161-node topology, respectively.

We depict the results in Fig. 5–Fig. 8. As expected, more available middlepoints improve TE performance. For  $TE_{LU}$ , when there is only one available midpoint for the network, every flow has to be routed through it, which severely limits the path choice and the maximum link utilization is way above 1 for *Random* and over 1 for the other schemes. With two middlepoints the maximum link utilization is dramatically reduced by over 50% for most schemes as seen in Fig. 5 and

Fig 6. The same can be observed for  $TE_{MF}$  in Fig. 7 and Fig. 8. The demand satisfaction ratio is improved by around a factor of 2 when  $K$  increases to 2.

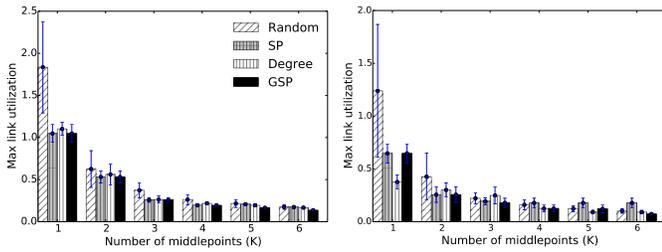


Fig. 5. 100-node network with 1000 flows of  $TE_{LU}$ .

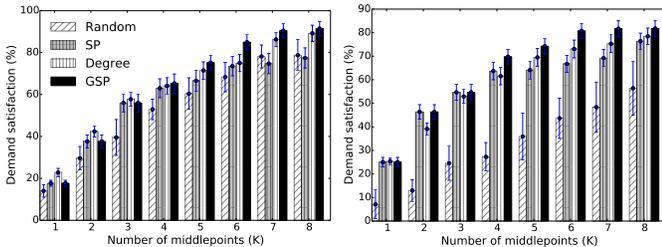


Fig. 7. 100-node network with 1000 flows of  $TE_{MF}$ .

Another important observation is that the TE performance exhibits diminishing marginal gains as  $K$  increases. For  $TE_{LU}$ , when more than 4 middlepoints are used, very limited gains are observed ( $<10\%$ ) in Fig. 5 and Fig. 6. For  $TE_{MF}$ , beyond 7 middlepoints there is little demand satisfaction improvement especially for  $GSP$  in Fig. 7 and Fig. 8. On the other hand the runtime of the TE algorithms increases dramatically due to the growing size of the LP problems. Take the 161-node network for instance. The LP time for  $TE_{LU}$  increases by  $\sim 50\%$  when  $K$  increases from 4 to 6 as shown in Table III, and from 6 to 8 for  $TE_{MF}$  as in Table IV.

Scheme \ $K$	1	2	3	4	5	6
Random	11.25	24.21	36.50	51.29	66.00	81.48
SP	9.79	19.56	29.35	38.31	46.27	54.25
Degree	8.92	17.83	28.49	42.98	55.45	64.96
GSP	9.75	19.54	29.30	39.64	54.85	70.81

TABLE III  
AVERAGE LP TIME (SECONDS) OF 161-NODE NETWORK WITH 2000 FLOWS OF  $TE_{LU}$ .

Scheme \ $K$	1	2	3	4	5	6	7	8
Random	12.10	23.87	32.78	45.91	60.04	76.43	100.02	110.00
SP	10.63	19.49	26.46	35.66	43.36	53.80	72.30	82.26
Degree	9.92	17.68	26.25	41.21	52.52	60.47	78.97	86.62
GSP	10.42	20.00	26.83	38.35	52.79	67.49	81.44	91.88

TABLE IV  
AVERAGE LP TIME (SECONDS) OF 161-NODE NETWORK WITH 2000 FLOWS OF  $TE_{MF}$ .

Based on the above results, we conclude that 4 middlepoints for  $TE_{LU}$  and 7 middlepoints for  $TE_{MF}$  are the sweetspots of the tradeoff between performance and complexity. We thus

use these settings in the rest of the experiments. This confirms the intuition behind our centrality based approach, namely, that it suffices to just use a small fraction of nodes as middlepoints (2.48%–7% of nodes) to achieve satisfactory performance.

### C. Comparison with Baseline

Our motivation of using centrality based midpoint selection is to reduce the high complexity of existing approaches, which takes all nodes in the network as middlepoints [5] as discussed in §I. We now compare our approach against *Baseline* to validate its effectiveness in this regard. The experiments here are performed on the 100-node topology for both TE formulations. The maximum link utilization and LP time of  $TE_{LU}$  are shown in Fig. 9 and Fig. 10, respectively. The demand satisfaction ratio and corresponding LP time are depicted in Fig. 11 and Fig. 12, respectively, for  $TE_{MF}$ . We scale the demands of flows by a factor of 2 for  $TE_{LU}$  and a factor of 40 for  $TE_{MF}$ .

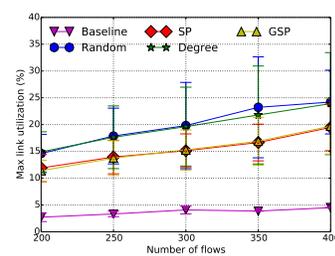


Fig. 9. Performance of  $TE_{LU}$  with different centralities and *Baseline*.

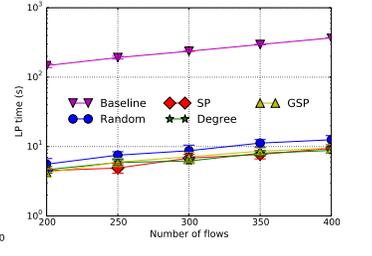


Fig. 10. LP time of  $TE_{LU}$  with different centralities and *Baseline*. Note the log scale of the y-axis.

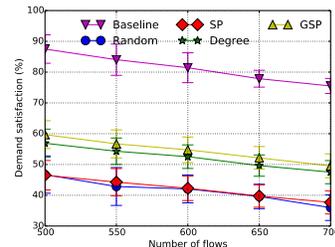


Fig. 11. Performance of  $TE_{MF}$  between different centralities and *Baseline*.

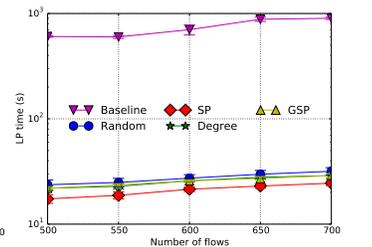


Fig. 12. LP time of  $TE_{MF}$  between different centralities and *Baseline*. Note the log scale of the y-axis.

Notice that with *Baseline*, the TE problems have much more variables and constraints due to the large number of middlepoints. As a result, our machines can only solve  $TE_{LU}$  with  $\sim 400$  flows, and  $TE_{MF}$  with  $\sim 1500$  flows. For problems beyond these scales the solver reports error messages. Recall that with centrality based midpoint selection the solver can easily handle problems with 3000 flows even for the bigger 161-node topology as we will show in §VII-D. In addition, solving  $TE_{MF}$  with *Baseline* and 1500 flows takes more than three hours, far exceeding the time scale (5–10 min) at which TE is performed in practice [28], [30], [31]. Thus we only run *Baseline* with up to 700 flows for  $TE_{MF}$  to make sure the LP time is less than 1000 seconds.

As shown in Fig. 9, the maximum link utilization of our approach is about 4–5 times that of *Baseline*, whereas the LP

time of *Baseline* is at least 40 times worse than any centrality based approach shown in Fig. 10. For  $TE_{MF}$ , the demand satisfaction ratio of *Baseline* is about 1.5 times of ours in Fig. 11 but the LP time is about 60 times higher than ours as in Fig. 12.

Indeed we observe that our centrality based approach sacrifices performance in order to reduce the complexity of TE. We argue that this is a sensible tradeoff to make in most cases, especially for data center backbone WANs that use  $TE_{MF}$  with very short time periods of 5–10 min [28], [30], [31]. Centrality based approach can support much larger topologies and much more flows with orders of magnitude smaller runtime. One can also increase  $K$  to obtain better performance if necessary.

#### D. Comparison of Various Centralities

We now wish to understand the relative performance of various centralities in realistic settings. We use both the 100-node and 161-node topologies with  $M = 1$ . Total number of middlepoints  $K$  is set to 4 for  $TE_{LU}$  and 7 for  $TE_{MF}$  based on our previous experiments. We vary the number of flows and for a given number of flows randomly draw flows 15 times from the traces. For *Random* we perform 5 independent random selections of middlepoints for a given set of flows, resulting in 75 runs in total. For each run we compute the respective performance metrics and report the average and standard deviation. In order to make the results more readable we scale the demands by 10 for 100-node topology and 40 for 161-node topology, respectively for  $TE_{MF}$ .

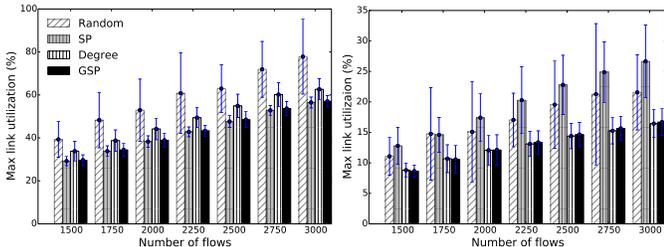


Fig. 13. Performance of  $TE_{LU}$  on the 100-node network with various centralities.  $M=1$  and  $K=4$ .

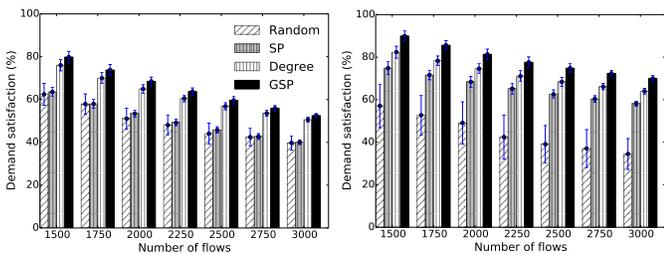


Fig. 15. Performance of  $TE_{MF}$  on the 100-node network with various centralities.  $M=1$  and  $K=7$ .

Fig. 13 and Fig. 14 depict the results for  $TE_{LU}$ , and Fig. 15 and Fig. 16 for  $TE_{MF}$ . We can make several interesting observations. First, for the 100-node network *SP* and *GSP*

perform the best under all settings in Fig. 13. In contrast, for the 161-node topology in Fig. 14 *GSP* and *Degree* perform the best. When considering  $TE_{MF}$ , Fig. 15 shows that *GSP* and *Degree* perform better in the 100-node topology, while in the 161-node topology *GSP* performs best in Fig. 16. Thus, middlepoints chosen by group shortest path centrality consistently outperform those selected by other centralities in terms of TE performance.

The main advantage of *GSP* is that it selects a set of middlepoints whose combined power is strong. In particular, *SP* may select nodes that are individually strong but cover the same set of shortest paths; thus, when combined together these nodes result in poor performance since they share the same shortest paths and are unable to spread out the traffic. This is the reason why *GSP* performs consistently well, while the performance of *SP* can fluctuate from very strong as in Fig. 13 to very poor and even worse than *Random* as in Fig. 14.

Second, *Random* performs the worst in Fig. 13, Fig. 15, and Fig. 16, and it also performs badly in the 161-node network in Fig. 14. This confirms our premise that centrality based midpoint selection generally outperforms a naive random selection scheme. Indeed, *Random* does not utilize any topological information from the network. Further, *Random* fluctuates wildly, which makes it ill-fitted for practical use. As seen from the figures, *Random* has the largest standard deviations among all.

Third, we observe that the performance of *SP* can be worse than *Random* sometimes in Fig. 14. Indeed, *SP* just greedily selects the top- $K$  shortest-path central nodes, even though in reality these nodes may share several shortest paths. *Random*, on the other hand, can do better than *SP* in certain settings since it has a lower probability of choosing overlapping shortest paths.

Another aspect of performance is the runtime of the TE LPs. Table V and Table VI show the average runtimes for  $TE_{LU}$  and  $TE_{MF}$  respectively. *Random* consistently has the worst results. *SP* takes the least time but the difference between *SP*, *GSP*, and *Degree* is little. All of the schemes can finish within 100 seconds even with 2000 flows, which demonstrates that centrality based segment routing can be practically used in large-scale networks.

Scheme \ # flows	1500	1750	2000	2250	2500	2750	3000
Random	37.54	44.76	53.07	60.70	68.18	78.52	88.00
SP	28.16	32.61	38.92	44.42	50.93	56.65	62.14
Degree	30.81	35.75	41.40	47.68	54.56	61.99	68.01
GSP	27.48	33.75	38.49	45.44	50.56	57.38	62.64

TABLE V  
AVERAGE LP TIME (SECONDS) OF 161-NODE NETWORK WITH  $TE_{LU}$ .

Scheme \ # flows	1500	1750	2000	2250	2500	2750	3000
Random	69.15	81.62	96.97	115.41	137.69	153.45	176.62
SP	49.55	58.83	70.53	79.87	93.67	107.57	122.04
Degree	53.79	64.79	76.04	90.49	102.63	112.53	129.58
GSP	54.37	66.32	79.54	94.04	108.38	125.80	140.17

TABLE VI  
AVERAGE LP TIME (SECONDS) OF 161-NODE NETWORK WITH  $TE_{MF}$ .

The reason that *Random* has the longest runtime is that it selects nodes that are not central with possibly many distinct paths and links. This leads to more active optimization variables and constraints for the same LP, thus longer runtime. By the same token, the reason that *SP* has the lowest runtime is that it selects top- $K$  central nodes with many overlapping shortest paths. This results in fewer active links being used for routing, and thus fewer active optimization variables and constraints in the LP.

To summarize, based on the above experimental results and analysis, we find that *GSP* consistently delivers the best TE performance with the least LP time among all centralities we considered.

### E. Comparison of Weighted Centralities

The centralities we have studied so far only considered the connectivity of the network topology. As discuss in §VI, it is also possible to take into account the link capacity information by adding weights to links and using weighted versions of centralities.

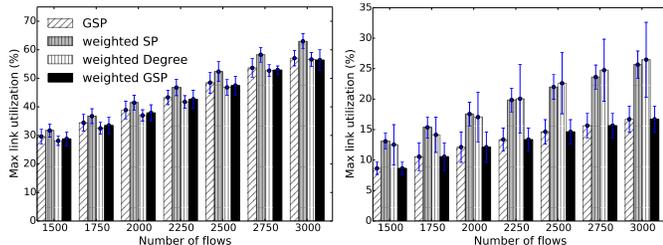


Fig. 17. 100-node network when  $M=1$  and  $K=4$  based on weighted centrality for  $TE_{LU}$ .

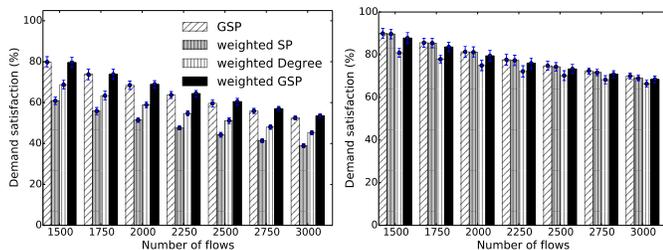


Fig. 19. 100-node network when  $M=1$  and  $K=7$  based on weighted centrality for  $TE_{MF}$ .

We also carry out experiments to compare the performance of weighted *SP*, weighted *degree*, and weighted *GSP* centralities against *GSP*, the best centrality without using weights for midpoint selection. Weighted here means that the three centrality based approaches are weighted by the capacity of each edge. Fig. 17 and Fig. 18 show the performance comparison with  $TE_{LU}$ , while Fig. 19 and Fig. 20 show the comparison with  $TE_{MF}$ . For the 161-node topology, we observe that *GSP* and weighted *GSP* are always the best. In the 100-node topology, *GSP* is sometimes worse than weighted *Degree* and weighted *GSP* although the differences are very small. Therefore, *GSP* without weights is still the

most effective and robust midpoint selection method in all settings.

## VIII. RELATED WORK

We now review related work on segment routing other than those discussed already in §II. Segment routing is a relatively new concept with limited prior work. Aubry et al. [4] propose to use segment routing for continuous monitoring of the data plane of the network with a single box. Segment routing is used to force probe packets to traverse specific paths. Giorgetti et al. [26] propose algorithms for segment routing label stack computation that guarantee minimum label stack depth.

TE has been extensively studied in carrier networks [13], [22], [28], [29], [32], [47], and has also attracted much attention recently in data center backbone WANs [25], [30], [31], [37] with software defined networking [16]. End-to-end paths are usually used while we study segment routing here in TE.

Finally, we note that graph centralities have been applied to routing in some specific SDN problems, such as in service chain embedding [39] and incremental SDN deployment [36], [38]. In a service chain [39], traffic needs to be steered through a set of waypoints, with the goal of admitting a maximum number of routes. In the context of hybrid and incremental SDN deployment [36], a set of middleboxes need to be deployed in order to serve a maximal number of flows, respecting flow rule constraints. Solutions to these problems are based on degree centralities, and there exist greedy approximation algorithms exploiting submodularity as well [38]. Contrary to these works, our paper focuses on the theoretical fundamentals of TE using segment routing and on graph-theoretic practical midpoint selection.

## IX. CONCLUSION

We have conducted the first systematic study of traffic engineering with segment routing in SDN based WANs. We showed that TE for the general segment routing is NP-hard, while segment routing with shortest paths is polynomial when the number of middlepoints per logical path is fixed and not part of the input. We also studied practical TE with shortest path based segment routing, and proposed to select just a few important nodes for all network traffic using graph theoretic centrality concepts. Our performance evaluation demonstrated that just a small percentage of powerful nodes can achieve good results at very low time complexities.

## REFERENCES

- [1] <http://www.cvxpy.org/en/latest/>.
- [2] “Declarative and expressive forwarding optimizer,” <http://sites.uclouvain.be/defo/>, October 2016.
- [3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993.
- [4] F. Aubry, D. Lebrun, S. Vissicchio, M. T. Khong, Y. Deville, and O. Bonaventure, “SCMon: Leveraging Segment Routing to Improve Network Monitoring,” in *Proc. IEEE INFOCOM*, 2016.
- [5] R. Bhatia, F. Hao, M. Kodialam, and T. V. Lakshman, “Optimized Network Traffic Engineering using Segment Routing,” in *Proc. IEEE INFOCOM*, 2015.
- [6] P. Bonacich, “Power and Centrality: A Family of Measures,” *American Journal of Sociology*, vol. 92, no. 5, pp. 1170–1182, 1987.

- [7] U. Brandes, "A faster algorithm for betweenness centrality," *Journal of Mathematical Sociology*, vol. 25, pp. 163–177, 2001.
- [8] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "On the effect of forwarding table size on SDN network utilization," in *Proc. IEEE INFOCOM*, 2014.
- [9] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," in *Proc. ACM SIGCOMM*, 2011.
- [10] S. Dolev, Y. Elovici, R. Puzis, and P. Zilberman, "Incremental deployment of network monitors based on group betweenness centrality," *Inf. Process. Lett.*, vol. 109, no. 20, pp. 1172–1176, 2009.
- [11] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *Proc. European Control Conference (ECC)*, 2013.
- [12] T. Eilam-Tzoref, "The disjoint shortest paths problem," *Discrete Appl. Math.*, vol. 85, no. 2, pp. 113–138, 1998.
- [13] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS Adaptive Traffic Engineering," in *Proc. IEEE INFOCOM*, 2001.
- [14] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, 1975, pp. 184–193.
- [15] M. G. Everett and S. P. Borgatti, "The centrality of groups and classes," *The Journal of Mathematical Sociology*, vol. 23, no. 3, pp. 181–201, 1999.
- [16] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *ACM Queue*, vol. 11, no. 12, pp. 20:20–20:40, December 2013.
- [17] U. Feige, "A threshold of  $\ln n$  for approximating set cover," *Journal of ACM*, vol. 45, no. 4, pp. 634–652, 1998.
- [18] C. Filsfils, S. Previdi, A. Bashandy, and Decraene, "Segment routing with mpls data plane," *Internet Engineering Task Force, Internet Draft (Work in Progress) draft-ietf-spring-segment-routing-mpls-00*, 2014.
- [19] C. Filsfils, P. Francois, and Previdi, "Segment routing use cases," 2013.
- [20] C. Filsfils, N. K. Nainar, and Pignataro, "The Segment Routing Architecture," in *Proc. IEEE Globecom*, 2015.
- [21] S. Fortune, J. Hopcroft, and J. Wyllie, "The directed subgraph homeomorphism problem," *Theoretical Computer Science*, vol. 10, no. 2, pp. 111–121, 1980.
- [22] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, 2000.
- [23] L. C. Freeman, S. P. Borgatti, and D. R. White, "Centrality in valued graphs: A measure of betweenness based on network flow," *Social Networks*, vol. 13, no. 2, pp. 141–154, 1991.
- [24] L. C. Freeman, "A Set of Measures of Centrality Based on Betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977.
- [25] A. Ghosh, S. Ha, E. Crabbe, and J. Rexford, "Scalable multi-class traffic management in data center backbone networks," 2013.
- [26] A. Giorgetti, P. Castoldi, F. Cugini, J. Nijhof, F. Lazzeri, and G. Bruno, "Path encoding in segment routing," in *Proc. IEEE Globecom*, 2015.
- [27] R. Hartert, P. Schaus, S. Vissicchio, and O. Bonaventure, "Solving Segment Routing Problems with Hybrid Constraint Programming Techniques," in *International Conference on Principles and Practice of Constraint Programming*, 2015.
- [28] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. Francois, "A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks," in *Proc. ACM SIGCOMM*, 2015.
- [29] J. He, M. Bresler, M. Chiang, and J. Rexford, "Towards robust multi-layer traffic engineering: Optimization of congestion control and routing," vol. 25, no. 5, pp. 868–880, June 2007.
- [30] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven WAN," in *Proc. ACM SIGCOMM*, 2013.
- [31] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined WAN," in *Proc. ACM SIGCOMM*, 2013.
- [32] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the Tightrope: Responsive Yet Stable Traffic Engineering," in *Proc. ACM SIGCOMM*, 2005.
- [33] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. ACM STOC*, 1984.
- [34] L. Khachiyan, "Polynomial algorithms in linear programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 20, no. 1, pp. 53–72, 1980.
- [35] M. Kuźniar, P. Perešini, and D. Kostić, "What you need to know about SDN flow tables," in *Proc. PAM*, 2015.
- [36] D. Levin, M. Canini, S. Schmid, F. Schaffert, and A. Feldmann, "Panopticon: Reaping the benefits of incremental sdn deployment in enterprise networks," in *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, 2014, pp. 333–345.
- [37] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic engineering with forward fault correction," in *Proc. ACM SIGCOMM*, 2014.
- [38] T. Lukovszki, M. Rost, and S. Schmid, "It's a match!: Near-optimal and incremental middlebox deployment," *SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 1, pp. 30–36, 2016.
- [39] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *Post-Proceedings of the 22Nd International Colloquium on Structural Information and Communication Complexity - Volume 9439*, ser. SIROCCO 2015, 2015, pp. 104–118.
- [40] L. Molnár, G. Pongrácz, G. Enyedi, Z. L. Kis, L. Csikor, F. Juhász, A. Kőrösi, and G. Rétvári, "Dataplane Specialization for High-performance OpenFlow Software Switching," in *Proc. ACM SIGCOMM*, 2016.
- [41] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.
- [42] M. Newman, *Networks: An Introduction*. Oxford University Press, Inc., 2010.
- [43] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., 1982.
- [44] R. Puzis, D. Yagil, Y. Elovici, and D. Braha, "Collaborative attack on internet users' anonymity," *Internet Research*, vol. 19, pp. 60–77, 2009.
- [45] M. Roughan, "Simplifying the synthesis of Internet traffic matrices," *ACM CCR*, vol. 35, no. 5, pp. 93–96, 2005.
- [46] E. Tardos, "A strongly polynomial algorithm to solve combinatorial linear programs," *Oper. Res.*, vol. 34, no. 2, pp. 250–256, 1986.
- [47] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "COPE: Traffic Engineering in Dynamic Networks," in *Proc. ACM SIGCOMM*, 2006.

**George Trimponias** received his PhD from the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology. Prior to that he obtained a five-year diploma in Electrical and Computer Engineering from the National Technical University of Athens, Greece. He is currently a Researcher at Huawei Noah's Ark Lab in Hong Kong. His research interests include machine learning, combinatorial optimization, and game theory.

**Yan Xiao** received the B.Eng. and M.Eng. degrees in College of Computer and Information from Hohai University, Nanjing, China, in 2013 and 2016, respectively. She is currently a Ph.D. student in Department of Computer Science, City University of Hong Kong. Her research interests include data mining, big data, deep learning based software engineering, and graph theory.

**Hong Xu** received the B.Eng. degree from the Department of Information Engineering, The Chinese University of Hong Kong, in 2007, and the M.A.Sc. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Toronto. He joined the Department of Computer Science, City University of Hong Kong in 2013, where he is currently an assistant professor. His research interests include data center networking, SDN, NFV, and cloud computing. He was the recipient of an Early Career Scheme Grant from the Research Grants Council of Hong Kong in 2014. He also received the best paper awards from IEEE ICNP 2015 and ACM CoNEXT Student Workshop 2014. He is a member of ACM and IEEE.

**Xiaorui Wu** received his B.S. degree in Electrical Information Engineering from University of Electronic Science and Technology of China in 2016. He is currently a first-year PhD student in Department of Computer Science, City University of Hong Kong. His research interests include computer networking, machine learning, and big data systems. He is a student member of the IEEE.

**Yanhui Geng** is a senior researcher and project manager at Huawei Noah's Ark Lab (Hong Kong). He received his B.Eng. and M.Eng. in Electronic Engineering and Information Science from the University of Science and Technology of China (USTC) in 2002 and 2005, respectively. He obtained his Ph.D. degree in Electrical and Electronic Engineering from the University of Hong Kong (HKU) in 2009. Before joining Huawei, he was a Post-Doctoral Research Fellow at HKU from 2009 to 2012, and was a Senior Engineer at the Hong Kong Applied Science and Technology Research Institute (ASTRI) from 2012 to 2013. His research interests include performance modeling and analysis of communication networks, SDN, machine learning, big data analytics, cloud computing, and indoor positioning technology. He has filed 15 patents related to communication networks. He has 26 technical publications on international journals and conferences including ACM SIGCOMM, IEEE/ACM Transactions on Networking, IEEE INFOCOM, ICNP, ICDCS, ICDM, etc. and his work on information quality received the IEEE ICC 2010 Best Paper Award.

APPENDIX A  
FLOW CENTRALITIES

A. Preliminaries

The original flow centrality by Freeman et al. [23] defines the flow centrality of a node  $w \in V$  in a flow network  $G = (V, E, c)$  as:

$$\gamma(w) = \sum_{s,t \in V | s \neq v \neq t} \frac{\nu_{max}^w(s,t)}{\nu_{max}(s,t)}, \quad (21)$$

where  $\nu_{max}(s,t)$  the maximum flow in the single-commodity flow network with commodity  $s-t$ , and  $\nu_{max}^w(s,t)$  the maximum flow through node  $w$  in the single-commodity flow network with commodity  $s-t$ . Thus, the flow centrality of node  $w$  represents the percentage of the maximum flow that can go through  $w$  for a demand chosen uniformly at random.

For multi-commodity networks with  $L$  commodities as described in Section III-A, we can provide an alternative definition of flow centrality as follows:

$$\tilde{\gamma}(w) = \frac{\nu_{max}^w(s,t)}{\nu_{max}(s,t)}, \quad (22)$$

which denotes the percentage of the maximum multi-commodity flow  $\nu_{max}^w(s,t)$  that can go through node  $w$  to the maximum multi-commodity flow.

The basic difference in the two definitions is that the former considers equiprobably all possible source-destination pairs, while the latter focuses on the actual commodities in the flow network. Thus, the former is based on the single-commodity formulation, and the latter on the multi-commodity one.

B. Hardness of Flow Centralities

**Corollary 4.** *Given a flow network  $G = (V, E, c)$  and a node  $w \in V$ , it is NP-hard to compute flow centrality  $\gamma(w)$  as in Equation (21). Furthermore, it is NP-hard to compute the flow centrality  $\tilde{\gamma}(w)$  of Equation (22).*

*Proof.* The first statement is straightforward, since it uses  $\nu_{max}^w(s,t)$  which is NP-hard by Proposition 1. For the second statement, note that  $\tilde{\gamma}(w)$  is a fraction of two terms. The denominator is the maximum multi-commodity flow  $\nu_{max}$  which can be computed in strongly polynomial time [46]. If  $\tilde{\gamma}(w)$  could also be computed in polynomial time, then we could simply compute  $\nu_{max}^w$  in polynomial time as the product of  $\tilde{\gamma}(w)$  and  $\nu_{max}$ , which is a contradiction since by Proposition 1, it is NP-hard to compute  $\nu_{max}^w$ .  $\square$

C. Group flow centrality

In this section, we introduce the concept of multi-commodity group flow centrality, which can be seen as generalization of the flow centrality  $\tilde{\gamma}$  that we defined in Equation (22).

**Definition 3.** *The group multi-commodity maximum flow  $\mathcal{GF} : 2^V \rightarrow \mathbb{R}_{\geq 0}$  in a multi-commodity flow network is a function which, for any group of nodes  $C \subseteq V$  of nodes, returns the maximum multi-commodity flow  $\mathcal{GF}(C)$  that can go through any node in  $C$ .*

*Furthermore, we call the maximum group flow that uses at most  $N$  nodes as the  $N$ -group maximum flow.*

The group flow centrality for directed graphs is obviously NP-hard as a generalization of the  $\tilde{\gamma}$  flow centrality which is also NP-hard by Proposition 1. It is however possible to show NP-hardness by reduction from the maximum coverage problem, even for just one commodity. This is very important to also acquire approximability, as we discuss later.

**Proposition 4.** *The  $N$ -group maximum single-commodity flow is NP-hard.*

*Proof.* We prove NP-hardness by reduction from the maximum coverage problem (MCP) [41], which is well-known to be NP-hard. In particular, assume a set  $S$  of  $m$  items  $I = i_1, \dots, i_m$  and  $n$  sets  $S_1, \dots, S_n$  containing elements in  $I$ . Given a positive integer  $N \leq n$ , the MCP tries to select  $N$  sets among  $S_1, \dots, S_n$  such that the maximum number of elements are covered, i.e. the union of the selected sets has maximal size. We can reduce the MCP to the  $N$ -group maximum multi-commodity flow by constructing a directed graph  $G = (V, E)$  as follows.  $V$  contains two dedicated nodes  $s$  and  $t$ , one node  $u_j$  for each item  $i_j$  that appears in  $I$ , and one node  $v_k$  for each set  $S_k$ . We then add one edge  $(s, u_j)$  from  $s$  to every node  $u_j$ , one edge  $(v_k, t)$  from each node  $v_k$  to  $t$ , and finally an edge  $(u_j, v_k)$ , if and only if set  $S_k$  contains item  $i_j$ . We consider that each edge of the form  $(s, u_j)$  has a capacity of 1, while all other edges have infinite capacity. Finally, the demand has an upper bound of  $m$ .

We now prove that the maximum coverage problem has a value equal to  $C_{max}$ , if and only if the  $N$ -group maximum flow has a value of  $C_{max}$ . Assume first that the MCP has a value of  $C_{max}$ . We can then construct a corresponding flow in  $G$  in the following manner. First, we send one unit of flow from  $s$  to node  $u_j$  if and only if item  $i_j$  is covered. Let  $V_1 \subseteq V$  be the subset of nodes  $u_j$  where a unit of flow was sent from  $s$ . For each  $u_j \in U$  we subsequently send one unit of flow to exactly one of the nodes  $v_k$  that  $u_j$  connects to, chosen uniformly at random. Let  $V_2 \subseteq V$  be the subset of nodes  $v_k$  which receive flow from any node in  $V_1$ . We complete the construction by sending  $l$  units of flow from every  $v_k \in V_2$  to  $t$ , where  $l$  is the number of nodes in  $V_1$  that send a unit of flow to  $v_k$ . Now, the constructed flow is valid since it is easy to verify that it respects all capacity and conservation constraints. Based on that flow, we can also form a  $N$ -group flow. Indeed, by definition the MCP contains at most  $N$  sets  $S_i$ , so there can be at most  $N$  nodes of the form  $v_k$  participating in the flow. Since the entire flow has to pass through these nodes, we can then claim that the above flow is a  $N$ -group flow passing through (at most)  $N$  nodes of the form  $v_k$ . So, the  $N$ -group flow is at least  $C_{max}$ .

Furthermore, we can argue that that the  $N$ -group maximum flow cannot be greater than  $C_{max}$ . Indeed, if that were not the case, then that would imply that there is another group of  $N$  nodes that can accept an even larger flow. Without loss of generality, we may assume that these nodes belong to  $V$  only, since we can trivially replace any node in  $U$  by any node in  $V$  that it connects to and achieve a flow that is at least as large

the original one. The key observation is that the group flow of  $N$  nodes in  $V$  has a value  $C$ , if and only if the coverage of the respective sets has a size of  $C$ . So, if there existed a group maximum flow with a greater value than  $C_{max}$ , then that would imply the existence of a maximum coverage of a value greater than  $C_{max}$ , which is a contradiction.

The reverse direction can be proven similarly.  $\square$

The previous result immediately implies that the  $N$ -group multi-commodity flow is also NP-hard. It is interesting that the proof of Proposition 4 does not rely at all on the theory that we developed to show that the individual flow centrality is NP-hard in Proposition 1. It is well-known that the MCP cannot achieve a better approximation ratio than  $O(1 - \frac{1}{e})$ , unless  $P=NP$ . We show below that this is also the case for the group multi-commodity flow by using results from submodular function maximization [17], [41].

**Definition 4.** Consider a finite set of elements  $U$  and a function  $g : 2^U \rightarrow \mathbb{R}_{\geq 0}$ . We call  $g$  monotone if adding an element to a set  $S \in 2^U$  cannot cause the function to decrease, i.e.,  $g(S \cup \{v\}) \geq g(S)$  for all  $v \in U$  and  $S \in 2^U$ . Furthermore, we call  $g$  submodular if the marginal gain from adding an element to a set  $S$  is at least as high as the marginal gain from adding the same element to a superset of  $S$ , i.e.,  $g(S \cup \{v\}) - g(S) \geq g(T \cup \{v\}) - g(T)$  for all  $v \in U$  and pairs of set  $S \subseteq T$ .

**Lemma 6.** The function  $\mathcal{GF} : 2^{V-\{s,t\}} \rightarrow \mathbb{R}_{\geq 0}$  is (1) monotone, and (2) submodular.

*Proof.* For monotonicity, note that adding a node can never decrease the maximum group flow, since an additional node can never decrease the number of available paths to route the flow; in particular, adding a node  $v \neq s, t$  to a set  $S$  can either increase the number of available paths to route the flow, or leave the number of paths unchanged, if all paths that go through  $v$  already go through nodes in  $S$ .

It is also simple to prove submodularity by noticing that the augmenting  $s$ - $w$ - $t$  algorithm can pick the augmenting paths arbitrarily. For  $T \cup \{v\}$ , we can then first saturate paths through  $S$ , then through  $T - S$ , and finally through  $v$ . For  $S \cup \{v\}$ , we can then first saturate paths through  $S$ , and finally through  $v$ . Since paths through  $T - S$  may overlap with paths through  $v$ , it is straightforward that the marginal gain from adding  $v$  in the former case can never exceed the marginal gain from the latter.  $\square$

**Proposition 5.** Consider the greedy algorithm that each time picks the node in  $V - \{s, t\}$  that maximizes the marginal utility to add to the group set. Let  $S^*$  be the subset of size  $k$  of  $V - \{s, t\}$  that maximizes the group  $\mathcal{GF}$ . Then the set  $S^g$  that the greedy selects satisfies the property that  $\mathcal{GF}(S^g) \geq (1 - \frac{1}{e}) \cdot \mathcal{GF}(S^*)$ , i.e.,  $S^*$  provides a  $(1 - \frac{1}{e})$ -approximation. Furthermore, unless  $P=NP$ , no polynomial algorithm can achieve a  $(1 - \frac{1}{e} + o(1))$ -approximation ratio.

*Proof.*  $\mathcal{GF}$  is a non-negative monotone submodular function. The approximation ratio for the greedy algorithm then follows directly from submodular function maximization [41]. The

impossibility result on the approximation ratio follows from [17].  $\square$

However, note that applying the greedy algorithm by Nemhauser et al. [41] to approximate the maximal group flow is NP-hard, since even computing the maximum  $s$ - $w$ - $t$  node is NP-hard by Proposition 4. So, the greedy algorithm will also be NP-hard.

For the undirected group flow centrality, we can get a similar result as in Proposition 5. The main difference is that in the undirected case a  $s$ - $w$ - $t$  path consists of undirected rather than directed edges. In terms of computational complexity, note that now each step of the greedy algorithm runs in (weakly) polynomial time (see Appendix B), so the time complexity for the greedy algorithm will be (weakly) polynomial, as opposed to the directed case where it is NP-hard. In fact, a strongly polynomial algorithm may also be possible, in a similar spirit to the results in [46].

## APPENDIX B THE UNDIRECTED CASE

We demonstrate a very interesting dichotomy between the cases of a directed and undirected graph. In particular, we show that the maximum multi-commodity  $s$ - $w$ - $t$  flow in a undirected graph can be computed in polynomial time. Note that the main difference between the directed and the undirected flow is that the directed assumes separate capacities for each direction  $(u, v)$  and  $(v, u)$  whereas the undirected assumes a single capacity for the undirected edge  $e$  which can be arbitrarily allocated in both directions, which upper bounds the total flow that we can send in both directions (but not in any individual direction).

**Proposition 6.** The maximum multi-commodity flow  $v_{max}^w$  in any undirected graph  $G = (V, E)$ ,  $w \in V$ , can be computed exactly in (weakly) polynomial time.

*Proof.* Assume a multi-commodity undirected graph  $G = (V, E)$  with  $L$  commodities of the form  $(s_i, t_i)$ . For simplicity, we assume infinite maximum demands  $D_i$  so that the demand constraint becomes redundant. We construct a directed graph  $G' = (V', E')$  from  $G$  as follows. We first replace each undirected edge  $(u, v) \in E$  by two directed edges  $(u, v)$  and  $(v, u)$  edges. The capacities on the two directed edges are equal to the original capacity  $c(e)$  of the undirected edge. We next introduce  $L$  new nodes  $z_1, \dots, z_L$  (one for each commodity), and for each  $z_i$  we add the two directed edges  $(s_i, z_i)$  and  $(t_i, z_i)$ . Finally, we introduce a node  $z$  and  $L$  directed edges  $(z_i, z)$  from each  $z_i$  to  $z$ . Thus, we have that  $V' = V \cup \{z_1, \dots, z_L, z\}$  and  $E' = E_G \cup (\cup_i \{(s_i, z_i), (t_i, z_i), (z_i, z)\})$ , where  $E_G$  are the edges that we got by replacing each undirected edge in  $E$  by two directed edges. The capacities of the newly constructed edges of the form  $(s_i, z_i), (t_i, z_i), (z_i, z)$  are infinite.

Next, we claim that the maximum flow in  $G$  can be computed by considering the following arc-based linear program:

$$\begin{aligned} \text{maximize } \mathcal{V} &= \sum_{i=1}^L \sum_{e \in w^+} f_i(e) \\ \text{subject to } \sum_{i=1}^L f_i(e) &\leq c(e), \forall e \in E' \end{aligned} \quad (23)$$

$$\sum_{e \in u^+} f_i(e) = \sum_{e \in u^-} f_i(e), \forall i, \forall u \in V', w \neq u \neq z \quad (24)$$

$$f_i(u, v) + f_i(v, u) \leq c(e), \forall i, \forall e = (u, v) \in E \quad (25)$$

$$f_i(e) \geq 0, \forall e \in E', \forall i \in \{1, \dots, L\} \quad (26)$$

$$f_j(s_i, z_i) = 0, \forall i, j \in \{1, \dots, L\} \text{ with } i \neq j \quad (27)$$

$$f_i(s_i, z_i) = f_i(t_i, z_i), \forall i \in \{1, \dots, L\} \quad (28)$$

The above LP computes the maximum flow from  $w$  to  $z$ , where the total flow is composed of  $L$  separate sub-flows, one for each commodity. The sub-flow for commodity  $i$  can be sent from  $w$  to  $z_i$  through either  $s_i$  or  $t_i$ . Constraints (23), (24), (26) are the link capacity, node conservation and positive flow constraints, respectively. Constraint (25) is necessary to make sure that the sum of flow units in each of the two directed edges does not exceed the capacity of the original undirected edge. Constraint (27) implies that node  $z_i$  can only receive flow from commodity  $i$ . Constraint (28) is especially important because it ensures that the sub-flow for commodity  $i$  sent through  $s_i$  is the same as the one sent through  $t_i$ .

Now, we establish the equivalence between the original problem and the above LP by showing that (i)  $2 \cdot \nu_{max}^w \leq \mathcal{V}^*$ , and (ii)  $\mathcal{V}^* \leq 2 \cdot \nu_{max}^w$ . We start with (i). Assume any maximum flow through  $w$  with value  $\nu_{max}^w$  in  $G$ . We first construct the corresponding flow in  $G'$ . We note that the flow in  $G$  consists of  $L$  sub-flows, one for each commodity  $i$ , that send flow from  $s_i$  to  $t_i$  through  $s_i$ - $w$ - $t_i$  paths. The idea is then to reverse the direction of each sub-flow in the part from  $s_i$  to  $w_i$ , so that it now sends to the opposite direction. We then send  $\nu(f_i)$  units of flow from  $s_i$  to  $z_i$ ,  $\nu(f_i)$  units of flow from  $t_i$  to  $z_i$ , and  $2 \cdot \nu(f_i)$  units of flow from  $z_i$  to  $z$ . Note that that is a valid flow since it respects all constraints in the LP, and it has a value  $2 \cdot (\nu(f_1) + \dots + \nu(f_L)) = 2 \cdot \nu_{max}^w$ . But then the maximum flow will be at least as large, hence  $2 \cdot \nu_{max}^w \leq \mathcal{V}^*$ .

For the reverse direction (ii), assume a maximum flow in  $G'$ . Then for each commodity  $i$  half units are sent from  $w$  to  $s_i$  and half from  $w$  to  $t_i$  (and subsequently  $z_i$  and  $z$ ) due to constraint (28). Again, the idea is to reverse the flow  $f_i$  in all paths from  $w$  to  $s_i$ . For each edge of  $G$  we then send on each direction as many units of flow as we send in  $G'$  (after reversing the direction from  $w$  to  $s_i$ ). The key is that in graph  $G'$  the same amount of flow is sent from  $w$  to  $s_i$  and  $w$  to  $t_i$ , which implies that the constructed flow in  $G$  will respect all capacity and conservation constraints, including for node  $w$ . The value of the flow in  $G$  is half that in  $G'$ , thus for the maximum flow on  $G$  we will trivially have that  $\mathcal{V}^* \leq 2 \cdot \nu_{max}^w$ .

Since the constraints of the LP have a size that is polynomial in  $V$  and  $E$ , and LP runs has a (weakly) polynomial complex-

ity in terms of the LP size [33], [34], we immediately deduce that computing the maximum  $s$ - $w$ - $t$  flow in an undirected graph  $G = (V, E)$  can be solved in (weakly) polynomial time, which concludes the proof.  $\square$