Multicast Traffic Engineering for Software-Defined Networks

Liang-Hao Huang*, Hsiang-Chun Hsu*, Shan-Hsiang Shen*, De-Nian Yang* and Wen-Tsuen Chen*†

*Institute of Information Science, Academia Sinica, Taipei, Taiwan
†Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan

Email: lhhuang@iis.sinica.edu.tw, hchsu0222@gmail.com, {sshen3, dnyang, chenwt}@iis.sinica.edu.tw

***Abstract*—Although Software-Defined Networking (SDN) enables flexible network resource allocations for traffic engineering, current literature mostly focuses on unicast communications. Compared to traffic engineering for multiple unicast flows, multicast traffic engineering for multiple trees is very challenging not only because minimizing the bandwidth consumption of a single multicast tree by solving the Steiner tree problem is already NP-Hard, but the Steiner tree problem does not consider the link capacity constraint for multicast flows and node capacity constraint to store the forwarding entries in Group Table of OpenFlow. In this paper, therefore, we first study the hardness results of scalable multicast traffic engineering in SDN. We prove that scalable multicast traffic engineering with only the node capacity constraint is NP-Hard and not approximable within δ, which is the number of destinations in the largest multicast group. We then prove that scalable multicast traffic engineering with both the node and link capacity constraints is NP-Hard and not approximable within any ratio. To solve the problem, we design a δ-approximation algorithm, named Multi-Tree Routing and State Assignment Algorithm (MTRSA), for the first case and extend it to the general multicast traffic engineering problem. The simulation and implementation results demonstrate that the solutions obtained by the proposed algorithm outperform the shortest-path trees and Steiner trees. Most importantly, MTRSA is computation-efficient and can be deployed in SDN since it can generate the solution with numerous trees in a short time.**

***Keywords*—SDN, multicast, traffic engineering**

I. Introduction

Software-defined networking (SDN) provides a new centralized architecture with flexible network resource management to support a huge amount of data transmission [1]. Different from legacy networks, SDN separates the control plane from switches and allows the control plane to be programmable to efficiently optimize the network resources. OpenFlow [1] in SDN includes two major components: controllers (SDN-Cs) and forwarding elements (SDN-FEs). Controllers are in charge of handling the control plane and install forwarding rules based on different policies, while forwarding elements in switches deliver packets according to the rules specified by the controllers. Compared with the current Internet, routing paths no longer need to be the shortest ones, and the paths can be distributed more flexibly inside the network. It has been demonstrated that SDN provides a better overview of network topologies and enables centralized computation for traffic engineering for multiple unicast flows [2], [3], [4]. However, multicast traffic engineering for multiple multicast trees in SDN has attracted much less attention in previous studies.

Compared to unicast, multicast has been shown in empirical studies to be able to effectively reduce overall bandwidth consumption in backbone networks by around 50% [5]. It employs a multicast tree, instead of disjoint unicast paths, from the source to all destinations of a multicast group, in order to avoid unnecessary traffic duplication. The current Internet multicast standard, i.e., PIM-SM [6], employs a shortest-path tree to connect the source and destinations, and traffic engineering is difficult for PIM-SM since the path from the source to each destination is the shortest one. A shortest-path tree tends to lose many good opportunities to reduce the bandwidth consumption by sharing more common edges among the paths to different destinations. In contrast, to minimize the bandwidth consumption, a Steiner tree (ST) [7] in Graph Theory minimizes the number of edges in a multicast tree. Nevertheless, ST only focuses on the routing of a multicast tree, instead of jointly optimizing the resource allocations of all trees. Therefore, when the network is heavily loaded, a link will not be able to support a large number of STs that choose the link. Most importantly, Group Table of an SDN-FE will be insufficient to store the forwarding entries of the STs due to the small TCAM size [8].

Compared to the shortest-path routing in unicast, unicast traffic engineering in SDN is more difficult to aggregate multiple flows in Flow Table of an SDN-FE, and the scalability has been regarded as a serious issue in the deployment of SDN for a large network [2], [9]. The scalability problem for multicast communications is even more serious since the number of possible multicast groups is $O(2^n)$, where n is the number of nodes in a network, and the number of possible unicast connections is $O(n^2)$. To remedy this issue, a promising way is to exploit the *branch forwarding technique* [10], [11], [12], which stores the multicast forwarding entries in only the *branch nodes*, instead of every node, of a multicast tree, where a branch node in a tree is the node with at least three incident edges. The branch forwarding technique can remedy the multicast scalability problem since packets are forwarded in a unicast tunnel from the logic port of a branch node in SDN-FE [1] to another branch node. In other words, all nodes in the path exploit unicast forwarding in the tunnel and are no longer necessary to maintain a forwarding entry for the multicast group. Furthermore, when a branch node is not multicast capable for a tree (ex. Group Table is full in this paper), local *unicast tunneling* from a nearby multicast capable node has been proposed in MBONE [13] and PIM-SM[1] to allow multiple unicast tunnels to pass through the branch node to other nodes in the tree (an example will be presented later in this section). Nevertheless, compared to multicast, it is envisaged that local unicast tunneling will incur more bandwidth consumption since duplicated packets will be delivered in a link. Therefore, there is a trade-off between the link capacity and node capacity, because each branch node can act as either a *branch state node* with the corresponding multicast forwarding entry stored in Group Table or a *branch*

[1]http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipmulti_pim/configuration/xe-3s/imc-pim-xe-3s-book/imc_tunnel.html

stateless node that exploits the unicast tunneling strategy.

In comparison with the ST problem, *scalable multicast traffic engineering*, which jointly allocates the network resources for multiple trees, is much more challenging because both the *link capacity* and *node capacity* constraints are involved in the problem. The link capacity constraint states that the total rate of all multicast trees on each link should not exceed the corresponding link capacity, while the node capacity constraint ensures that Group Table of each node is sufficiently large to support the multicast trees with the node as a branch state node. Moreover, scalable multicast traffic engineering with branching forwarding and unicast tunneling techniques is able to allocate the network resources more flexibly. When Group Table of a node is full, unicast tunneling moves the resource requirement from the node to its incident links, whereas the rerouting of the tree is also promising by exploiting the resources of the nearby nodes and links. Therefore, it is necessary for scalable multicast traffic engineering to carefully examine both the routing and the allocation of the branch state nodes of all multicast trees. In this paper, we explore the Scalable Multicast Traffic Engineering (SMTE) problem for SDNs. Given the data rate requirement of each multicast tree, SMTE aims to minimize the total bandwidth cost of all trees, by finding a tree connecting the source and destinations of each group and assigning the branch state nodes for each tree, such that both the link capacity and node capacity constraints can be ensured.

Fig. 1 presents an illustrative example. Fig. 1(a) is the original network with the unit bandwidth cost specified beside each link. The bandwidth cost of each link is the total bandwidth consumption of the link multiplied by the unit bandwidth cost. The node capacity of each node is 1. The link capacity of edge $e_{s,a}$ is 1, and the link capacities of the other edges are ∞ in this example. There are two multicast trees with both flow rates as 1. The source of the first tree is $s_1 = s$, and its destination set is $D_1 = \{d_1, d_2, \ldots, d_7\}$. The source of the second tree is $s_2 = s$, and its destination set is $D_2 = \{d'_1, d'_2, \ldots, d'_7\}$. Fig. 1(b) shows the first shortest-path tree (blue) and the second shortest-path tree (red). The branch nodes and branch state nodes of the first tree are $\{c, u, v\}$ and $\{u\}$, respectively. The branch nodes and branch state nodes of the second tree are $\{c, v\}$ and $\{c, v\}$, respectively. Note that v is not assigned as a branch state node of the first tree, and u thus needs to exploit unicast tunneling to d_6 and d_7 directly. Therefore, traffic of the first tree are duplicated in edge $e_{u,v}$. On the other hand, if v was assigned as a branch state node for the first tree, traffic duplication in $e_{u,v}$ would be more serious for the second tree since v has three downstream nodes d'_5, d'_6, d'_7. The total bandwidth cost of the two shortest-path trees in Fig. 1(b) is 99.

Afterward, Fig. 1(c) shows the first Steiner tree (blue) and the second Steiner tree (red), and the branch state nodes of the two trees are also $\{u\}$ and $\{c, v\}$, respectively. The total bandwidth cost of the trees in Fig. 1(c) is 103. Note that the total bandwidth cost in Steiner trees is higher since the assignment of branch state nodes are not carefully examined. Finally, Fig. 1(d) presents the first tree (blue) and the second tree (red) in SMTE with the same branch state nodes specified above. The total bandwidth cost of the trees in Fig. 1(c) is 79, and here u is directed connected to d_1, d_2, and d_3 to avoid unicast tunneling, even though the edge cost is higher (i.e., 2) compared to the cost (i.e., 1) of the edge from v in the first Steiner tree. Therefore, this example manifests that it is necessary to consider the tree routing and the assignment of branch state nodes of multiple trees jointly for scalable multicast traffic engineering.

SMTE is very challenging. The ST problem is NP-Hard but can be approximated within the ratio 1.55 [14] and is thus

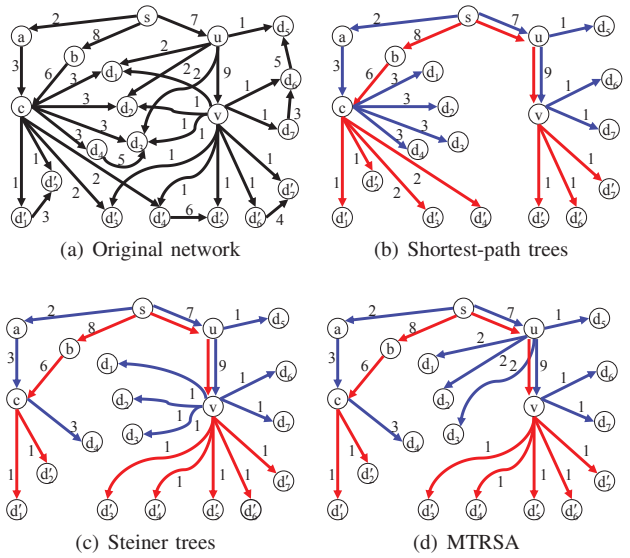


Fig. 1. Comparison of different strategies for multicast traffic engineering

in APX of Complexity Theory. In other words, there exists an approximation algorithm for ST that can find a tree with the total cost at most 1.55 times of the optimal solution. In contrast, we first prove that SMTE-N (i.e., SMTE with only the node capacity constraint, while the link capacity constraint is relaxed) is NP-Hard but cannot be approximated within δ, which denotes the number of destinations of the largest multicast group. Afterward, we prove the SMTE (i.e., with both the link and node capacity) cannot be approximated within any ratio. To solve SMTE-N, we propose a δ-approximation algorithm, named *Multi-Tree Routing and State Assignment Algorithm (MTRSA)*, which can be deployed in SDN-C. MTRSA includes two phases: Multi-Tree Routing Phase and State-Node Assignment Phase, to effectively minimize the total bandwidth cost of all trees according to the node capacity constraint. We first focus on the node capacity (i.e., SMTE-N), instead of the link capacity, because the scalability in Group Table is unique and crucial for SDN and has not been explored in previous studies of multicast tree routing for other networks. Since no $(\delta^{1-\epsilon})$-approximation algorithm exists in SMTE-N for arbitrarily small $\epsilon > 0$, MTRSA achieves the best approximation ratio. Afterward, we extend MTRSA to support SMTE with the link capacity constraint.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 and 4 formulate SMTE with Integer Programming and describe the hardness results. We present the algorithm design of MTRSA in Section 5, and Section 6 shows the simulation and implementation results on real topologies. Finally, Section 7 concludes this paper.

II. Related Work

The issues of traffic engineering for *unicast traffic* in SDN have attracted a wide spectrum of attention in the literature. Sushant et al. [15] developed private WAN of Google Inc. with the SDN architecture. Qazi et al. [16] designed a new system in SDN to control the middleboxes, and Mckeown et al. [1] studied the performance of OpenFlow in heterogeneous SDN switches. Agarwal et al. [2] presented unicast traffic engineering in an SDN network with only a few SDN-FEs, while the other routers in the network followed a standard routing protocol, such as OSPF. However, the above studies focused on only unicast traffic engineering, and multicast

traffic engineering for multiple multicast trees in SDN has attracted much less attention.

To support the multicast communications, the current multicast routing standard PIM-SM [6] relies on unicast routing protocols to discover the shortest paths from the source to the destinations for building a shortest-path tree (SPT). However, SPT is not designed to support traffic engineering. Although the Steiner tree (ST) [7] minimizes the tree cost and the volume of traffic in a network, ST is computationally intensive and is not adopted in the current Internet standard. Overlay ST [17], [18], on the other hand, presents an alternative way to construct a bandwidth-efficient multicast tree in the P2P environment. However, the path between any two P2P clients is still a shortest path in Internet, and it is, therefore, difficult to optimize the routing of the P2P tree. Most importantly, both SPT and ST are designed to find the routing of a tree, instead of jointly optimizing the resource allocation of multiple trees.

Flow table scalability is crucial to support large-scale SDN networks due to the limited TCAM size. Kanizo et al. [9], who showed that the major bottleneck in SDN is the restricted table sizes, proposed a framework called Patette to decompose a large SDN table and distribute its entries across a network. Leng et al. [19] proposed a flow table reduction scheme (FTRS) to reduce flow table usage with omnipotent controller functions. DIFANE [8] distributed the flow entries to multiple SDN switches. Zhang at al. [20] built a multicast topology (single backbone tree) for NFV, while Craig et al. adjusted the link weights for shortest-path trees in SDN [21]. Huang et al. also tried to optimize the routing of single multicast tree in SDN [22], [23]. Nevertheless, the above studies were not designed for minimizing the total resource consumption in multicast traffic engineering with multiple trees subject to both the node and link capacity constraints in SDN.

III. Problem Formulation

In this paper, we explore the *Scalable Multicast Traffic Engineering* (SMTE) problem for SDN. Given the data rate requirement of each multicast group, SMTE aims to minimize the total bandwidth consumption of all multicast groups in the network, by finding a tree connecting the source and destinations of each group, and assigning the branch state nodes for each tree, such that the number of multicast forwarding states will not exceed the size of Group Table in each node, and the total multicast flows on each edge will not exceed the link capacity. Note that a branch node can only facilitate unicast tunneling for a multicast group if it is not assigned as a branch state node in the corresponding multicast tree.

More specifically, given a network $G(V, E)$, where V and E denote the set of nodes and directed edges, respectively, let b_v denote the maximal number of branch state nodes that can be maintained by node v[2]. Let N_v^+ (N_v^-) denote the set of out-neighbor (in-neighbor) nodes of v in G. Node u is in N_v^+ (N_v^-) if $e_{v,u}$ ($e_{u,v}$) is a directed edge from v to u (from u to v) in E, and $c_{u,v}$ is the capacity of $e_{u,v}$, while $k_{u,v}$ is the unit bandwidth cost of $e_{u,v}$. Let $\mathcal{T} = (T_1, T_2, \ldots, T_t)$ denote the set of multicast trees, while s_i acts as the root of tree $T_i \in \mathcal{T}$, i.e., the source with data rate f_i, and the destination set D_i contains the set of destinations in $T_i \in \mathcal{T}$. In the following, we first formally define SMTE, while the derivation of the bandwidth consumption will be explained later in this section in the proposed Integer Programming formulation. Dynamic group membership with user join and leave will be discussed later in Section V-C.

[2]In the following, we first assume that the memory size allocated in Group Table to maintain the branch state node of each multicast tree is the same, and later we extend it to the general scenario that supports different memory sizes for different multicast trees according to the degrees of the node in the trees [1] in Section V-C.

Definition 1. *For network $G(V, E)$ and multicast groups $\mathcal{T}$, SMTE is to find the routing of each tree T_i in $\mathcal{T}$ spanning s_i and D_i and assign the branch state nodes in T_i to minimize the total bandwidth cost, such that each node u acts as the branch state nodes of at most b_u trees, and total multicast bandwidth consumption in each edge $e_{u,v}$ is at most $c_{u,v}$.*

In the following, we present the Integer Programming (IP) formulation for SMTE. SMTE includes the following binary decision variables to find the routing of each multicast tree and the assignment of branch state nodes. Let binary variable $\pi_{i,d,u,v}$ denote if edge $e_{u,v}$ is in the path from s_i to a destination node d in D_i in T_i. Let integer variable $\varepsilon_{i,u,v}$ denote the number of times that each packet of T_i is sent in edge $e_{u,v}$ via multicast (once) or unicast tunneling (multiple times according to the number of tunnels). Let binary variable $\beta_{i,v}$, denote if v is a branch state node in T_i. Intuitively, when we are able to find the path from s_i to each destination node d of T_i with $\pi_{i,d,u,v} = 1$ on every edge $e_{u,v}$ in the path, together with the set of state branch nodes $\beta_{i,v}$, the routing of the tree (the set of edges $e_{u,v}$ with $\varepsilon_{i,u,v} \geq 1$) can be constructed according to the paths from s_i to all destination nodes in D_i.

The objective function of the IP formulation for SMTE is as follows.

$$\min \sum_{1 \leq i \leq t} \sum_{e_{u,v} \in E} f_i \times k_{u,v} \times \varepsilon_{i,u,v}.$$

The objective function minimizes the total bandwidth cost of all multicast trees. For each tree T_i, the following constraints first describe the routing assignment (i.e., $\pi_{i,d,u,v}$) for the path connecting the source s_i and each destination in D_i. Afterwards, we assign the branch nodes (i.e., $\beta_{i,u}$) in different nodes and then derive the bandwidth consumption (i.e., $\varepsilon_{i,u,v}$) of T_i via multicast and unicast tunneling.

$$\sum_{v \in N_{s_i}^+} \pi_{i,d,s_i,v} - \sum_{v \in N_{s_i}^-} \pi_{i,d,v,s_i} = 1, \forall 1 \leq i \leq t, d \in D_i, \quad (1)$$

$$\sum_{u \in N_d^-} \pi_{i,d,u,d} - \sum_{u \in N_d^+} \pi_{i,d,d,u} = 1, \forall 1 \leq i \leq t, d \in D_i, \quad (2)$$

$$\sum_{v \in N_u^-} \pi_{i,d,v,u} = \sum_{v \in N_u^+} \pi_{i,d,u,v},$$
$$\forall 1 \leq i \leq t, d \in D_i, u \in V, u \neq d, u \neq s_i, \quad (3)$$

$$\pi_{i,d,u,v} \leq \varepsilon_{i,u,v}, \forall 1 \leq i \leq t, d \in D_i, \forall e_{u,v} \in E, \quad (4)$$

$$-|D_i|^2 \times \beta_{i,u} + \sum_{v \in N_u^+} \varepsilon_{i,u,v} \leq \sum_{v \in N_u^-} \varepsilon_{i,v,u}, \quad (5)$$
$$\forall 1 \leq i \leq t, u \in V, u \neq s_i,$$

$$\sum_{1 \leq i \leq t} \beta_{i,u} \leq b_u, \forall u \in V, \quad (6)$$

$$\sum_{1 \leq i \leq t} f_i \times \varepsilon_{i,u,v} \leq c_{u,v}, \forall e_{u,v} \in E. \quad (7)$$

The first three constraints, i.e., (1), (2), and (3), are the flow-continuity constraints for each tree T_i to find the path from s_i to every destination node d in D_i. More specifically, s_i is the source node, and constraint (1) states that the net outgoing flow from s_i is one, implying that at least one edge $e_{i,s_i,v}$ from s_i to any neighbor node v needs to be selected with $\pi_{i,d,s_i,v} = 1$. Note that here decision variables $\pi_{i,d,s_i,v}$ and π_{i,d,v,s_i} are two different variables because the flow is directed. On the other hand, every destination node d is the flow destination, and constraint (2) ensures that the net incoming flow to d is one, implying that at least one edge $e_{i,u,d}$ from any neighbor node u to d must be selected with $\pi_{i,d,u,d} = 1$. For every other node u, constraint (3) guarantees that u is either located in the path or not. If u is located in the path, both the incoming flow and outgoing flow for u are at

least one, indicating that at least one binary variable $\pi_{i,d,v,u}$ is 1 for the incoming flow, and at least one binary variable $\pi_{i,d,u,v}$ is 1 for the outgoing flow. Otherwise, both $\pi_{i,d,v,u}$ and $\pi_{i,d,u,v}$ are 0. Note that the objective function will ensure that $\pi_{i,d,v,u} = 1$ for at most one neighbor node v to achieve the minimum bandwidth consumption. In other words, both the incoming flow and outgoing flow among u and v cannot exceed 1.

Constraints (4) and (5) are formulated to find the routing of the tree and its corresponding branch state nodes, i.e., $\varepsilon_{i,u,v}$ and $\beta_{i,u}$. Constraint (4) states that $\varepsilon_{i,u,v}$ is at least 1 if edge $e_{u,v}$ is included in the path from s_i to at least one d, i.e., $\pi_{i,d,u,v} = 1$. The tree T_i is the union of the paths from s_i to all destination nodes in D_i. Constraint (5) is the most crucial one. For each node u in T_i, if it is not a branch state node, i.e., $\beta_{i,u} = 0$, u does not maintain a forwarding entry of T_i in Group Table and thereby facilitates unicast tunneling. In this case, constraint (5) and the objective function guarantee that the number of packets received from an incoming link $e_{v,u}$ must be the summation of the number of packets sent to every outgoing link $e_{u,v}$. By contrast, when $\beta_{i,u} = 1$, constraint (5) becomes redundant because the Left-Hand-Side (LHS) is smaller than 0 and thereby imposes no restrict on the Right-Hand-Side (RHS). In this case, constraint (4) ensures that $\varepsilon_{i,v,u} = 1$ for every incident edge $e_{v,u}$ with $\pi_{i,d,v,u}$ as 1. Therefore, u is multicast capable for T_i, and each packet is delivered once in every incident link.

The last two constraints are capacity constraints. Constraint (6) states that each node u can act as a branch state node of at most b_u trees in $\mathcal{T}$, while constraint (7) describes that the total multicast bandwidth consumption of in each directed edge $e_{v,u}$ cannot exceed $c_{u,v}$.

IV. Hardness Results

In the following, we first show that SMTE-N is very challenging in Complexity Theory by proving that it is NP-Hard and not able to be approximated within δ^c for every $c < 1$, where $\delta = \max_{1\le i\le t} |D_i|$. Afterward, we prove that SMTE cannot be approximated within any ratio.

The Steiner tree problem is a special case of SMTE-N. However, SMTE-N is much more challenging than the Steiner tree problem because the Steiner tree problem can be approximated within ratio 1.55 and is thus in APX in Complexity Theory. In contrast, we find out that SMTE is much more difficult to be approximated. The following theorem first proves that SMTE-N cannot be approximated within δ^c for every $c < 1$, where $\delta = \max_{1\le i\le t} |D_i|$, by a gap-introducing reduction from the 3SAT problem.

Theorem 1. *For any $\epsilon > 0$, there exists no $(\delta^{1-\epsilon})$-approximation algorithm for SMTE-N, where $\delta = \max_{1\le i\le t} |D_i|$, assuming P $\neq$ NP.*

Proof: We prove the theorem with the gap-introducing reduction from the 3SAT problem.

The 3SAT problem is a simplification of the regular SAT problem. An instance of 3SAT is a conjunctive normal form (CNF) in which each clause contains exactly three variables. The 3SAT problem is to decide, given a Boolean expression ϕ in CNF such that each clause contains exactly three variables, whether ϕ is satisfiable.

For any instance ϕ of the 3SAT problem, we build an instance $G(V, E)$ of SMTE-N with two multicast trees, where the destination sets are D_1 and D_2. Let $\mathrm{OPT}(G)$ denote the optimal solution of G for SMTE-N. The goals of the reduction are two-fold. 1) If ϕ is satisfiable then $\mathrm{OPT}(G) \le 4p^{q+1}$. 2) If ϕ is not satisfiable then $\mathrm{OPT}(G) > (4p^{q+1}) \times (\max\{|D_1|, |D_2|\})^{1-\epsilon}$. In the above two goals, n is the number of Boolean variables in ϕ, m is the number of clauses in ϕ, $p = \max\{m, n\}$ and q is a large number (derived later).

To achieve the above goals, we build the instance of SMTE-N from each instance of the 3SAT problem as follows. Given an instance ϕ of 3SAT with n Boolean variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$, we construct a directed graph $G(V, E)$ in the following way. 1) The node set V is partitioned into four node sets $\{s\}$, U, D_1, and D_2. 2) U includes $2n$ nodes $u_1, \overline{u_1}, u_2, \overline{u_2}, \ldots, u_n, \overline{u_n}$ (nodes u_i and $\overline{u_i}$ correspond to the Boolean variable x_i), and for each i with $1 \le i \le n$, there are directed edges (s, u_i) and $(s, \overline{u_i})$. 3) D_1 has mp^q nodes $d_j^{(k)}$, where $1 \le j \le m$ and $1 \le k \le p^q$ (nodes $d_j^{(k)}$, $1 \le k \le p^q$, corresponding to p^q copies of the clause C_j), and there exists a directed edge $(u_i, d_j^{(k)})$ ($(\overline{u_i}, d_j^{(k)})$) if and only if the variable x_i ($\overline{x_i}$, resp.) appears in C_j. 4) D_2 contains np^q nodes $w_i^{(k)}$, where $1 \le i \le n$ and $1 \le k \le p^q$, and there are directed edges $(u_i, w_i^{(k)})$ and $(\overline{u_i}, w_i^{(k)})$ for each i, k with $1 \le i \le n$ and $1 \le k \le p^q$. Note that G only has the directed edges described above, $p = \max\{m, n\}$, and q is the smallest integer such that $q \ge (3 + \log_p 4)/\epsilon$. Fig. 2 presents an illustrative example of an SMTE-N instance.

The cost of each edge from s to U is set as p^q, and the cost of the other edges are set to be 1. The capacity of each node is set as 1. Let s and D_1 be the source node and destination set of T_1 respectively, and let s and D_2 be the source node and destination set of T_2 respectively.

If ϕ is satisfiable, there is a truth assignment to x_i such that ϕ is true. Let $A = \{u_i : x_i \text{ is assigned to be true}\} \cup \{\overline{u_i} : x_i \text{ is assigned to be false}\}$. Consider the tree T_1 rooted at s that includes 1) the edges between s and A, and 2) the edges between $d_j^{(k)}$ and one of its neighbor in A (the existence of its neighbor in A comes from that ϕ is satisfiable). Consider the tree T_2 that includes 1) the edges between s and $U \setminus A$, 2) the edges between $U \setminus A$ and D_2. Then (T_1, T_2) is a feasible solution of SMTE-N, and it can act as an upper bound of SMTE-N in G. The total edge cost of T_1 is $np^q + mp^q$, and the total edge cost of T_2 is $np^q + np^q$. Since the node capacity is sufficient, the total bandwidth cost of this feasible solution is $3np^q + mp^q \le 4p^{q+1}$. Hence, we have $\mathrm{OPT}(G) \le 4p^{q+1}$.

On the other hand, if ϕ is not satisfiable, let (T_1, T_2) be any feasible solution. For $1 \le k \le p^q$, let I_k be the set consisting of every i with $1 \le i \le n$, such that u_i and $\overline{u_i}$ are adjacent to some nodes in $\{d_j^{(k)} : 1 \le j \le m\}$ along the edges in T_1. Since ϕ is not satisfiable, I_k is not empty for each k with $1 \le k \le p^q$. By pigeonhole principle [24], there exists at least one i^* with $1 \le i^* \le n$ such that i^* is in at least $\frac{p^q}{n}$ sets of $\{I_1, I_2, ..., I_{p^q}\}$. In T_1, therefore, u_{i^*} has at least $\frac{p^q}{n} \ge p^{q-1}$ downstream destination nodes, and $\overline{u_{i^*}}$ has at least $\frac{p^q}{n} \ge p^{q-1}$ downstream destination nodes. On the other hand, in T_2, u_{i^*} and $\overline{u_{i^*}}$ need to dominate p^q downstream destination nodes of T_2. If u_{i^*} or $\overline{u_{i^*}}$ is not a branch state node in T_1, then the total cost is at least p^{2q-1}. On the other hand, if u_{i^*} and $\overline{u_{i^*}}$ both are branch state nodes in T_1, since the capacity of node u_{i^*} and $\overline{u_{i^*}}$ are 1, neither u_{i^*} nor $\overline{u_{i^*}}$ are branch state nodes in T_2. The total cost is at least p^{2q}.

Therefore, the total cost of the optimal solution is larger than p^{2q-1}, and we have $\mathrm{OPT}(G) > p^{2q-1} = (4p^{q+1})(p^{q-2-\log_p 4}) = (4p^{q+1})(p^{q+1})^{\frac{q-2-\log_p 4}{q+1}} = (4p^{q+1})(p^{q+1})^{1-\frac{3+\log_p 4}{q+1}} \ge (4p^{q+1})(p^{q+1})^{1-\epsilon} \ge (4p^{q+1})(\max\{|D_1|, |D_2|\})^{1-\epsilon}$. Since ϵ can be arbitrarily small, for any $\epsilon > 0$, there is no $(\max\{|D_1|, |D_2|\})^{1-\epsilon}$ approximation algorithm for SMTE-N, assuming P $\neq$ NP. The theorem follows. ∎

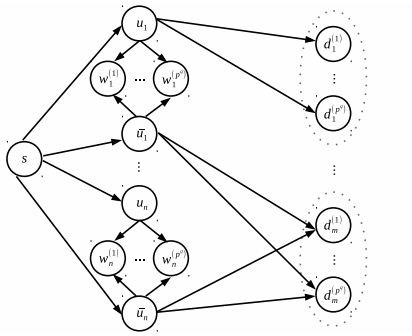


Fig. 2. An illustration of instance building from 3SAT to SMTE-N

In the following, we prove that SMTE cannot be approximated within any ratio.

Theorem 2. *For any polynomial time computable function f, SMTE cannot be approximated within a factor of $f(|V|)$, unless P = NP. In other words, for arbitrary positive integer k, SMTE cannot be approximated within $|V|^k$.*

Proof: Due to the space constraint, the detailed proof is presented in [25]. ■

V. Algorithm Design

In the following, we first propose a δ-approximation algorithm, named *Multi-Tree Routing and State Assignment Algorithm (MTRSA)*, for SMTE-N, where $\delta = \max_{1\leq i\leq t}|D_i|$. Note that we first focus on the node capacity, instead of the link capacity, because the scalability in Group Table is crucial in SDN and has not been explored in previous studies of multicast tree routing for other networks. Since Theorem 1 proves that there is no $(\delta^{1-\epsilon})$-approximation algorithm of SMTE-N for any $\epsilon > 0$, MTRSA achieves the best approximation ratio. Afterward, we extended it to support SMTE. Due to the space constraint, the pseudo-code is presented in [25].

A. Algorithm Description

MTRSA includes two phases: 1) Multi-Tree Routing Phase and 2) State-Node Assignment Phase. Multi-Tree Routing Phase first constructs an initial multicast tree for each multicast group to minimize the total bandwidth consumption and balance the distribution of branch nodes in different trees. State-Node Assignment Phase then finds the branch state nodes for each multicast tree to follow the node constraint.

1) Multi-Tree Routing Phase: Initially, Multi-Tree Routing Phase constructs a shortest-path tree with source s_i and destination set D_i for each tree $T_i \in \mathcal{T}$. A node u is *full* if it acts as a branch node for b_u multicast trees. By contrast, u is *overloaded* if it acts as a branch node for more than b_u trees. In this case, u needs to act as a branch stateless node for some of those trees and thereby will incur more bandwidth consumption. To address this issue, after finding the shortest-path trees, if there is an overloaded node, we adjust the local tree routing nearby the overloaded node to move the branch node to another node that has not been full, in order to balance the distribution of branch nodes among different multicast trees.

More specifically, if any node u is an overloaded node and a branch node in any tree T_i, MTRSA chooses a node v of T_i such that: 1) v is a downstream to u in T_i, 2) v is a branch node or a destination node of T_i, and 3) there is no other branch node or destination node in the path from u to v in T_i. In other words, v is a nearby downstream branch node of u or a destination node. MTRSA reroutes the path (from u to v) to another path (from w to v) as follows, in order to alleviate the storage load in u. Let ℓ denote the total bandwidth cost of the path from u to v in T_i. We find a new path from w to v such that: 1) the total cost of the new path is at most ℓ, 2) the new path does not pass through any exiting node in T_i, and 3) this new path starts from an on-tree node w such that i) it is not a leaf node of T_i, and ii) it will not be overloaded after rerouting. We update tree T_i by substituting the old path from u to v in T_i with the new path from w to v, and the overload situation in u can be alleviated accordingly. Afterward, we process every other downstream branch node v of u until u is no longer a branch node for T_i. The above process is repeated for every tree $T_{i'}$ iteratively until u is no longer overloaded.

Example. Consider the following example in Fig. 3(a). Let $G(V, E)$ be the network with two multicast trees T_1 and T_2 with the data rate as 1. The number on each edge is the unit bandwidth cost of this edge, and the node capacity of each node is 1. The source s_1 of the first tree T_1 is s with the corresponding destination set $D_1 = \{d_1, d_2, \ldots, d_8\}$, while the source s_2 of T_2 is also s, but the destination set is $D_2 = \{d'_1, d'_2, \ldots, d'_6\}$. In Multi-Tree Routing Phase, we first find the blue and red shortest-path trees T_1 and T_2 in Fig. 3(b). Afterward, we adjust the multicast trees for overloaded nodes. Specifically, the node capacity of a is 1, but a is a branch node of both T_1 and T_2. Therefore, a is an overloaded node, and MTRSA examines nodes d_1, d_2, v, c, which are downstream nodes of a in T_1. MTRSA first reroutes the path $\{a, b, c\}$ in tree T_1. Since node y is overloaded, node c cannot be rerouted from node y. In contrast, node v is a full branch node of T_1, and MTRSA reroutes node c from node v for T_1 as shown in Fig. 3(c). Note that the bandwidth cost is efficiently reduced since the new path from v to c is much smaller than the one from a to c. Therefore, Multi-Tree Routing Phase addresses both the node capacity and the bandwidth consumption for scalable multicast traffic engineering.

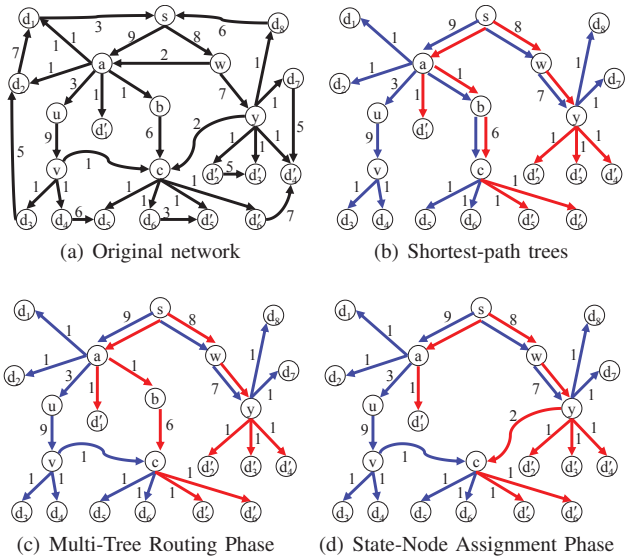
2) State-Node Assignment Phase: It is worth noting when the network is heavily loaded, the first phase may not be able to ensure that every overloaded node can be successfully adjusted to balance the distribution of branch nodes in different trees, and State-Node Assignment Phase is crucial in this case to minimize the increment of bandwidth consumption due to unicast tunneling through branch stateless node. More specifically, State-Node Assignment Phase includes two stages: 1) Greedy Assigning Stage, and 2) Local Search Stage. Greedy Assigning Stage assigns the branch state nodes by iteratively maximizing the reduction of the number of branch state nodes, and later in Section V-B we prove that the number of branch state nodes reduced by the Greedy Assigning Stage is at least half of the number of branch state nodes reduced by an optimal strategy. Local Search Stage then improves the solution by further alleviating the assignment on overloaded nodes and rerouting the trees to further reduce the total bandwidth cost. We detail the two stages as follows.

For each multicast tree T_i obtained in Multi-Tree Routing Phase, let W_i denote the set of branch nodes in T_i, and $W = \cup_{1\leq i\leq t} W_i$. On the other hand, let A_i be the set of branch state nodes in T_i to be decided in this phase, and A_i thereby is a subset of W_i. Let $c(T_i, A_i)$ denote the total bandwidth cost of T_i with the set of branch state nodes as A_i. More precisely, $c(T_i, A_i) = \sum_{v\in A_i\cup D_i} c(P_v)$, where P_v is the path from the closest upstream branch state node in A_i or the source to v, such that all internal nodes of P_v are not in A_i, and $c(P_v)$ is the cost of all edges in P_v. In other words, if there is no branch stateless node in P_v, every packet is delivered only once on every link of P_v. By contrast, if P_v includes a branch stateless node u, each packet is sent multiple times on the links from the closest upstream branch state node to u, corresponding to the unicast tunneling case.

An assignment A of branch state nodes can be defined as follows: A is a 0, 1-matrix with the rows indexed by $\{1, \ldots, t\}$ and columns indexed by W, such that 1) the 1's in row i can only be the columns indexed in W_i, and 2) the number of 1's in column $w \in W$ is no more than the node capacity b_w. We assign a branch state node $w \in W$ to tree T_i if and only if the (i, w) entry of A is 1. In other words, the first condition ensures that a branch state node can only be assigned to a branch node of T_i, while the second condition is the node capacity constraint. Given an assignment A of branch state nodes, let $A_i = \{w \in W :$ the (i, w) entry of A is $1\}$ denote the set of branch state nodes for T_i, and the total bandwidth cost for the set $\mathcal{T}$ of all multicast trees with the state-node assignment A is $c(\mathcal{T}, A) = \sum_{1 \le i \le t} c(T_i, A_i)$. Since an assignment A of branch state nodes can also be regarded as a subset of N, where $N = \{1, \ldots, t\} \times W$, let $\mathcal{M}$ be the family of subsets of N satisfying the above two conditions (hence $\mathcal{M}$ is the family of all feasible assignments of branch state nodes to $\mathcal{T}$), and we use $c(\mathcal{T}, \varnothing)$ to denote the total bandwidth cost of $\mathcal{T}$ without assigning any branch state node. Now let the set function $z : \mathcal{M} \to \mathbb{R}$ such that $z(A)$ represents the cost reduced by assignment A. More formally, $z(A) = c(\mathcal{T}, \varnothing) - c(\mathcal{T}, A)$ for each $A \in \mathcal{M}$.

The above matrix representation plays a crucial role in Greedy Assigning Stage when we prove the quality of the state-node assignment based on Matroid Theory later in Section V-B. This stage starts from a branch state node assignment $\varnothing$ and cost $c(\mathcal{T}, \varnothing)$, and iteratively assigns one branch state node for a tree in $\mathcal{T}$ until no more assignment can reduce $c(\mathcal{T}, A)$. More precisely, in each iteration, if the present branch state node assignment is $A \in \mathcal{M}$, we choose an element x in $N - A$ such that: 1) $A \cup \{x\}$ is in $\mathcal{M}$ and follows the node capacity constraint, and 2) $z(A \cup \{x\}) = \max_{y \in (N-A)} z(A \cup \{y\})$. In other words, the first condition guarantees that the new assignment is feasible, whereas the second condition chooses the node leading to the maximal reduction on $c(\mathcal{T}, A)$. Afterward, Local Search Stage first adjusts the assignment of branch state nodes for overloaded nodes iteratively. In each iteration, we first extract an overloaded node u and then compute the reduction of the bandwidth cost with a branch state node assigned to u for each tree T spanning u, assuming that the state-node assignment of other nodes are not changed. Afterward, this phase sorts the trees according to the bandwidth reduction and chooses the b_u trees with the largest reduction, whereas the branch state nodes are assigned to them accordingly. This stage is repeated until all overloaded nodes are carefully examined. Afterward, this stage reroutes the paths from other branch nodes of a tree in order to find a smaller tree with the same assignment of branch state nodes. More specifically, for any branch node u in tree T_i, we choose nodes v and w of T_i in the same way as the Multi-Tree Routing Phase in order to find a new path from w to v, and w is not full.

Example. In Greedy Assignment Stage of the State-Node Assignment Phase, when there is no branch state node, the total bandwidth cost in Fig. 3(c) is $c(\mathcal{T}, \varnothing) = c(T_1, \varnothing) + c(T_2, \varnothing) = 142 + 92 = 234$. If we assign a branch state node on v for tree T_1, the bandwidth cost of the path s, a, u, v can be reduced by $(4 - 1)$ times since there are 4 downstream destination nodes d_3, d_4, d_5, d_6 of v in T_1. The reduced cost is the largest among all possible branch state node assignments. Therefore, MTRSA first assigns a branch state node on v for tree T_1 with the cost reduced by $(4 - 1) \times (k_{s,a} + k_{a,u} + k_{u,v}) = 63$. It then assigns a branch state node on y to T_2 with the cost reduced by $(3-1) \times (k_{s,w} + k_{w,y}) = 30$. Afterward, node a is assigned as a branch state node for T_2 with the cost reduced by $(3-1) \times (k_{s,a}) = 18$, and node c is assigned as a branch state node for T_2 with the cost reduced by $(2-1) \times (k_{a,b} + k_{b,c}) = 7$.



(a) Original network (b) Shortest-path trees

(c) Multi-Tree Routing Phase (d) State-Node Assignment Phase

Fig. 3. An example of MTRSA

In Local Search Stage, there are three overloaded nodes a, c, and y. For overloaded node a, this phase moves the branch state node on a from T_2 to T_1 without changing the branch state nodes of the other nodes. If we assign a branch state node on a to T_1, it becomes possible to reduce the cost of T_1 by $(3 - 1) \times (k_{s,a}) = 18$. In contrast, if we assign a branch state node on a to T_2, we are able to reduce the cost of T_2 by only $(2 - 1) \times (k_{s,a}) = 9$. Nodes c and y are then processed similarly. Finally, in Fig. 3(d), since node y has been a branch state node, node c can be re-routed to node y in T_2, and the total bandwidth consumption is reduced from 107 in Fig. 3(c) to 93 in Fig. 3(d) accordingly.

B. Approximation Ratio and Time Complexity

In the following, we first examine the quality of assignment for branch state nodes in the second phase. We prove that $(N, \mathcal{M})$ is a matroid and the set function $z : \mathcal{M} \to \mathbb{R}$ is a nondecreasing submodular set function. Therefore, according to the Matroid Theorem for maximizing submodular set function [26], we have the following theorem.

Theorem 3. *The number of branch state nodes reduced by the Greedy Assignment Stage is at least one half of the branch state nodes reduced by the optimal assignment of branch state nodes.*

Proof: Due to the space constraint, the detailed proof is presented in [25]. ■

Then, we prove that MTRSA is a δ-approximation algorithm for SMTE-N, where δ is the maximum size of the destination sets. Since Theorem 1 proves that there is no approximation algorithm with ratio $\delta^{1-\epsilon}$ for any $\epsilon > 0$, the following theorem shows that MTRSA achieves the best approximation ratio. In contrast, since SMTE cannot be approximate within any ratio unless $P = NP$, it is impossible to derive an approximation ratio for any algorithm of SMTE, and we thereby evaluate MTRSA for SMTE in Section VI.

Theorem 4. *MTRSA is a δ-approximation algorithm for SMTE-N, where $\delta = \max_{1 \le i \le t} |D_i|$.*

Proof: Let the set of multicast trees $\mathcal{T}^* = (T_1^*, \ldots, T_t^*)$ with the assignment A^* of branch state nodes be the optimal solution to SMTE-N, and $W^* = \cup_{i=1}^{t} W_i^*$ with W_i^* as the set of branch nodes of T_i^*, whereas $A_i^* = \{w \in W^* :$

the (i, w) entry of A^* is $1\}$ be the set of branch state nodes in T_i^*. Therefore, the optimal bandwidth cost is $c(\mathcal{T}^*, A^*) = \sum_{i=1}^{t} c(T_i^*, A_i^*)$. For MTRSA, Multi-Tree Routing Phase first constructs the shortest-path trees $\mathcal{T}^{(1)} = (T_1^{(1)}, \ldots, T_t^{(1)})$, and the rerouting procedure of the Multi-Tree Phase outputs the new trees $\mathcal{T}^{(2)}$. MTRSA finally generates the trees $\mathcal{T}^{(3)}$ with the assignment $A^{(3)}$ of branch state nodes. Let A be an assignment such that for each non-overloaded node u within $\mathcal{T}^{(1)}$, MTRSA assigns a branch state node on u to each tree in $\mathcal{T}^{(1)}$ with u as a branch node. According to the State-Note Assignment Phase, we have $A \subseteq A^{(3)}$. On the other hand, in Multi-Tree Routing Phase, MTRSA updates the trees only when the bandwidth cost does not increase, and the branch state nodes can only reduce the cost. Therefore, we have $c(\mathcal{T}^{(3)}, A^{(3)}) \leq c(\mathcal{T}^{(2)}, A^{(3)}) \leq c(\mathcal{T}^{(2)}, A)$. In the rerouting procedure of Multi-Tree Routing Phase, suppose we reroute $\mathcal{T}$ to $\mathcal{T}'$. Since each rerouting step does not create any new overloaded node, and the new path in $\mathcal{T}'$ ensures that $c(\mathcal{T}', A) \leq c(\mathcal{T}, A)$. Therefore, $c(\mathcal{T}^{(2)}, A) \leq c(\mathcal{T}^{(1)}, A)$ holds by induction, and we have $c(\mathcal{T}^{(3)}, A^{(3)}) \leq c(\mathcal{T}^{(2)}, A) \leq c(\mathcal{T}^{(1)}, A) \leq c(\mathcal{T}^{(1)}, \varnothing)$. Since the path $P_{s_i,d}^{(1)}$ from the source s_i to node d in D_i in tree $T_i^{(1)}$ is the shortest path from s_i to d in G, and T_i^* has a path from s_i to d, we have $c(P_{s_i,d}^{(1)}) \leq c(T_i^*)$ for every i and every $d \in D_i$. Therefore $c(\mathcal{T}^{(3)}, A^{(3)}) \leq c(\mathcal{T}^{(1)}, \varnothing) = \sum_{i=1}^{t} c(T_i^{(1)}, \varnothing) \leq \sum_{i=1}^{t} \sum_{d \in D_i} c(P_{s_i,d}^{(1)}) \leq \sum_{i=1}^{t} \sum_{d \in D_i} c(T_i^*, A_i^*) = \sum_{i=1}^{t} |D_i| \times c(T_i^*, A_i^*) \leq \delta(\sum_{i=1}^{t} c(T_i^*, A_i^*)) = \delta \times c(\mathcal{T}^*, A^*)$. The theorem follows. ■

Time Complexity. We first find the shortest path between any two nodes in G with Johnson's algorithm in $O(|V||E|+|V|^2 \log |V|)$ time as the pre-processing procedure. Multi-Tree Routing Phase constructs the shortest-path tree for each source s_i and its corresponding destination set D_i, and MTRSA compares the distance from a destination node to all other nodes in $O(|V|)$ time. Processing all $d \in D_i$ requires $O(|V||D_i|) = O(\delta|V|)$ time, and processing all shortest-path trees requires $O(t\delta|V|)$ time. After constructing the shortest-path trees, MTRSA reroutes the paths from each overloaded node to some of its downstream nodes. Since there are at most $t\delta$ branches in the tree set, we reroute at most δ paths for each branch node, and each rerouting requires the comparison of at most $|V|$ distances of paths. Therefore, the above procedure requires $O(t\delta^2|V| \log |V|)$ time, and Routing Phase requires $O(t\delta^2|V| \log |V|)$ time.

Afterward, in Greedy Assigning Stage of State-Node Assignment Phase, there are at most $t|V|$ branch state nodes required to be assigned, and this stage has at most $t|V|$ iterations. In each iteration, we first derive the minimum cost reduced by assigning a branch state node on each node v of every T_i in $O(t|V|)$ time, and we update the cost reduced by assigning each new branch state node to the tree T_i in $O(|T_i||E|) = O(|V||E|)$ time. Therefore, this stage requires $O(t|V|(t|V|+|V||E|)) = O(t|V|^2(t+|E|))$ time. Then, Local Search Stage carefully examines the overloaded nodes. In each iteration, we adjust the branch state nodes on each node u in different trees without changing other branch state nodes to find the new bandwidth cost in $O(t|E|)$ time, and this stage takes $O(t|V||E|)$ time because there are at most $|V|$ iterations. Therefore, State-Node Assignment Phase requires $O(t|V|^2(t+|E|))$ time to allocate the branch state nodes and $O(t\delta^2|V| \log |V|)$ time for rerouting, and MTRSA requires $O(t\delta|V|^2 \log |V|(t+|E|))$ time.

C. Extension to SMTE

For SMTE, since the number of times that each packet is delivered in a link cannot be acquired before assigning the branch state nodes, we first present the concept of *weak edge capacity constraint*. Let $\varepsilon'_{i,u,v} = 1$ if $\varepsilon_{i,u,v}$ is a positive integer in our Integer Programming formulation, and $\varepsilon'_{i,u,v} = 0$ otherwise. MTRSA needs to ensure $\sum_{1 \leq i \leq t} f_i \times \varepsilon'_{i,u,v} \leq c_{u,v}$ holds for $\forall e_{u,v} \in E$ before assigning the branch state nodes. In addition, in the general case of SMTE, the storage size of each branch state node u in Group Table is proportional to the degree of u in the corresponding multicast tree [1]. Therefore, let $\beta_{i,u}$ denote the node weight (i.e., storage size) of assigning a branch state node on u to tree T_i, and b_u here denotes the size of Group Table in u. For SMTE, we extend MTRSA as follows.

Before Multi-Tree Routing Phase, we sort the multicast trees in $\mathcal{T}$ according to their data rates in the ascending order, $f_1 \leq f_2 \leq \cdots \leq f_t$. In Multi-Tree Routing Phase, we find the first shortest-path tree T_1 in $\mathcal{T}$ and decrease the link capacity $c_{u,v}$ for every edge in T_1 by its flow rate f_1. Note that any edge $e_{u,v}$ with insufficient residual capacity to support f_2 will be removed since it cannot support the rest of the multicast flows. The above procedure is repeated for other trees in $\mathcal{T}$.

In the rerouting procedure of Multi-Tree Routing Phase, we reroute each multicast tree T_i according to the weak edge capacity constraint, such that any new path from w to v in Section V-A must have sufficient capacity to support f_i. In Greedy Assignment Stage, we find an element $x = (i, u)$ in $N - A$ according to $\frac{z(A \cup \{(i,u)\}) - z(A)}{\beta_{i,u}} = \max\{\frac{z(A \cup \{(i',u')\}) - z(A)}{\beta_{i',u'}} : A \cup \{(i', u')\} \in \mathcal{M}\}$, which represents the normalized cost reduction. In other words, the node weight $\beta_{i,u}$ is considered during the assignment of branch state nodes.

In Local Search Stage, optimizing the state-node assignment of one node becomes the same as the knapsack problem because each candidate branch state node now has a profit (i.e., cost reduction) and a size (i.e., node weight), and we exploit Polynomial-Time Approximation Scheme for knapsack [27] to find the solution. In the rerouting procedure of Local Search Stage, since now the branch state nodes have been specified, we reroute each multicast tree T_i according to the original edge capacity constraint (7), such that any new path from w to v must have sufficient capacity to support f_i. If the amount of multicast flows in any edge exceeds the capacity constraint, we also reroute its closest upstream state node u in a tree T_i to w, such that the new path from w to v follows the link capacity constraint.

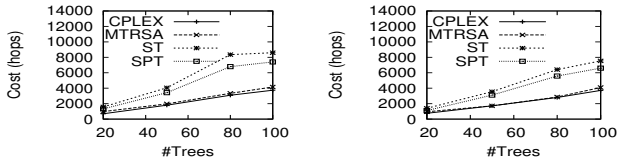
MTRSA can support the dynamic multicast group membership as follows. When a user v joins or leaves a multicast group, MTRSA adds or trims (if no other users are located downstream to the user) the corresponding branch from the upstream branch node u in the same way as Multi-Tree Routing Phase. Afterward, State-Node Assignment Phase adjusts the new branch state node if necessary. Therefore, it does not need to re-compute the whole tree.

VI. Performance Evaluation

In this section, we first compare MTRSA and other approaches with real topologies. Afterward, we deploy our algorithm in a small experimental SDN network with HP SDN switches to evaluate the video performance with YouTube HD traffic that requires a large amount of bandwidth consumption..

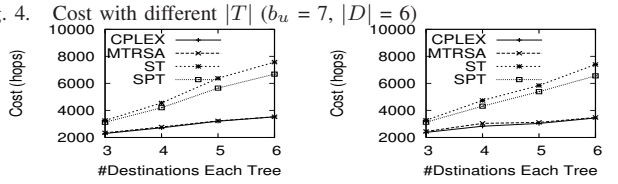
A. Simulation Setup

We simulate our algorithm in two real networks: VtlWavenet2011 and Columbus [28] in the Estinet network simulator [29]. VtlWavenet2011 includes 91 nodes and 96 links, while Columbus has 70 nodes and 85 links. Our simulation



(a) VtlWavenet2011 (b) Columbus

Fig. 4. Cost with different $|T|$ (b_u = 7, $|D|$ = 6)



(a) VtlWavenet2011 (b) Columbus

Fig. 5. Cost with different $|D|$ ($|T|$ = 100, b_u = 7)

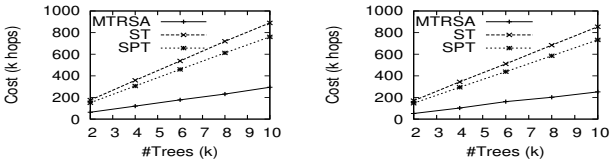
is divided into small-scale and large-scale cases. The numbers of trees in the small-scale cases are smaller than 100, whereas there are more than 2000 trees in the large-scale cases. The link capacity in the topologies is set to the level that the maximal bottleneck link utilization reaches 100% [30], and the edge cost of each link is set as 1. We vary the number of multicast trees, the number of destinations, and node capacity. The source and destinations are chosen randomly from each network.

We compare MTRSA with the following algorithms: 1) the shortest-path tree algorithm (SPT), 2) the Steiner tree (ST) algorithm[3] [7], and 3) CPLEX [31], which finds the optimal solutions of SMTE problem by solving the IP formulation in Section III. In SPT and ST, the branch state nodes of different trees are randomly assigned to a branch node when node is fully utilized, i.e., the number of branch state nodes reaches the node capacity. Each SPT and ST is added to the network iteratively. If an edge does not have sufficient residual capacity to support the multicast flow of a new SPT or ST, it will be removed accordingly to avoid choosing the edge in the SPT or ST. We implement all algorithms in an HP DL580 server with four Intel Xeon E7-4870 2.4 GHz CPUs and 128 GB RAM. Each simulation result is averaged over 100 samples.

B. Small-Scale Evaluation

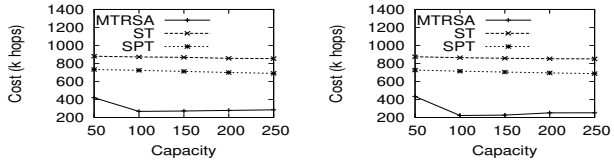
In the small-scale cases, we compare the total bandwidth costs of all trees in MTRSA, SPT, ST, and CPLEX with different number of trees ($|T|$), different node capacity (b_u), and different number of destinations $|D|$. As shown in Fig. 4 and Fig. 5, MTRSA generates a solution with the total bandwidth cost very close to the optimal solution. Although SPT chooses the shortest path to the destinations, it does not carefully examine the node capacity, and its cost is thus higher than MTRSA. Compared with SPT, the distance from the source to a destination in ST is usually higher because the path needs to be deviated from the shortest one in order to share more edges with another path. Therefore, more branch nodes are usually involved in ST. Without a sophisticated allocation of branch state nodes, ST incurs a slightly higher cost than SPT due to the additional bandwidth consumption in unicast tunneling for the branch nodes with full Group Table. The difference becomes more significant when the number of trees increases as shown in Fig. 4. Similarly, Fig. 5 manifests that the total bandwidth cost of each tree increases as the number of destinations grows, because each tree becomes larger in this case.

[3]There are some single-tree multicast routing algorithms with differernt purpuses (such as QoS), but they are not included in this study because ST (i.e., the optimal solution for single tree) outperforms those approaches in terms of the bandwidth consumption.



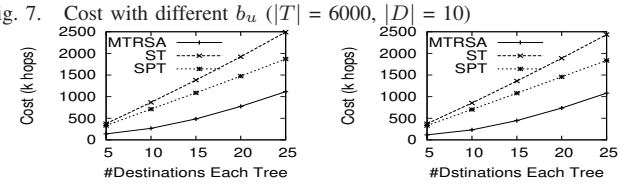
(a) VtlWavenet2011 (b) Columbus

Fig. 6. Cost with different $|T|$ (b_u = 300, $|D|$ = 10)



(a) VtlWavenet2011 (b) Columbus

Fig. 7. Cost with different b_u ($|T|$ = 6000, $|D|$ = 10)



(a) VtlWavenet2011 (b) Columbus

Fig. 8. Cost with different $|D|$ ($|T|$ = 6000, b_u = 300)

TABLE I. RUNNING TIME OF MTRSA (SECONDS)

$\|T\|$	$\|D\|$ = 5	$\|D\|$ = 10	$\|D\|$ = 15	$\|D\|$ = 20	$\|D\|$ = 25
2000	1.00	2.29	3.01	3.43	3.99
4000	3.09	6.33	8.71	11.06	12.39
6000	5.82	12.52	17.94	22.37	25.53
8000	8.80	20.10	29.89	37.52	43.78
10000	12.83	32.32	48.44	57.01	73.40

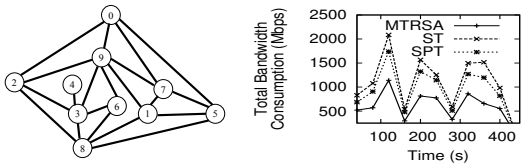
C. Large-Scale Evaluation

In the following, we evaluate MTRSA, ST, and SPT in larger-scale cases, where the number of multicast tree is ranged from 2000 to 10000, the number of destinations is from 5 to 25, and the node capacity is between 50 and 250. Compared with smaller-scale cases, the advantage of MTRSA is more significant in larger-scale cases. In Fig. 6, the total cost increases with the number of trees. For a larger network, the source and any destination are inclined to be located with distantly, but there is also a higher chance to find a node with sufficient capacity as a branch node for rerouting. Therefore, MTRSA effectively reduces the total bandwidth cost by 66% and 59%, respectively, compared to ST and SPT. In addition, Fig. 7 shows that the bandwidth costs can be effectively reduced when we increase node capacity, and setting the node capacity as 100 is sufficient for MTRSA. On the other hand, the bandwidth cost grows with the number of destinations, because all trees are required to span more nodes as shown in Fig. 8.

Table I summarizes the running time of MTRSA with different $|T|$ and $|D|$. With a smaller input, such as 2000 trees and 5 destinations in each tree, the running time for MTRSA is around 1 second. As $|T|$ and $|D|$ increase, MTRSA only requires around 73 seconds in the largest case with 10000 multicast trees. Therefore, it is envisaged that our algorithm is practical to be deployed in SDN.

D. Implementation

To evaluate MTRSA in real environments, we implement it in an experimental SDN with HP Procurve 5406zl OpenFlow-enabled switches. We use Floodlight as the OpenFlow controller to install the multicast forwarding rules in SDN-FEs. MTRSA is running on the top of Floodlight. Our testbed includes 12 nodes and 24 links as shown in Fig. 9, where



(a) Topology (b) Bandwidth consumption

Fig. 9. Implementation results of the experimental SDN

the link capacity and node capacity are set as 50Mbps and 5, respectively. We randomly select 10 nodes as the video multicast sources, where each source is connected to a Youtube proxy to facilitate YouTube multicast. The full-HD test video is in 460 seconds encoded in H.264 with the average bit rate as 10Mbps. For each source, we randomly assign 10 destinations that play videos using the VLC player. Due to the space constraint, the detailed implementation setting is presented in [25]. Fig. 9(b) shows that the total bandwidth consumption during playback, and we average the bandwidth consumption every 40 seconds. The results manifest that the bandwidth consumption of MTRSA is 46% and 35% lower than ST and SPT, respectively. Therefore, MTRSA can effectively support multicast traffic engineering in SDN.

VII. Conclusion

Recent studies on traffic engineering for SDN mostly focus on unicast, while most existing multicast routing algorithms are designed to find the routing of a multicast tree, instead of multiple trees. In this paper, therefore, we have formulated Scalable Multicast Traffic Engineering Problem (SMTE) to minimize the total bandwidth cost according to the link and node capacity constraints for multiple trees in SDN. We have proved that SMTE-N is NP-hard and not able to be approximated within δ, while SMTE cannot be approximated within any ratio. To solve the problem, we have proposed Multi-Tree Routing and State Assignment Algorithm (MTRSA), which is a δ-approximation algorithm for SMTE-N, while MTRSA has been extended to support SMTE as well. Simulation based on real topologies and implementation with Youtube traffic manifest that MTRSA can effectively find the routing of multiple multicast trees and assign the branch state nodes in order to reduce the total bandwidth cost, while the computation time to construct numerous trees is also reasonable for practical SDN. Since the tree obtained by MTRSA is not delay bounded, we will extend it to support QoS multicast in the future work.

Acknowledgment

This work is supported by MOST 104-2622-8-009-001.

References

[1] McKeown et al., "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[2] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *IEEE INFOCOM'13*, pp. 2211–2219.

[3] Lazaris et al., "Tango: Simplifying sdn control with automatic switch property inference, abstraction, and optimization," in *ACM CoNEXT '14*, pp. 199–212.

[4] X. N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, "Optimizing rules placement in openflow networks: Trading routing for better efficiency," in *ACM HotSDN '14*, pp. 127–132.

[5] R. Malli, X. Zhang, and C. Qiao, "Benefit of multicasting in all-optical networks," *Proceedings of the SPIE*, vol. 3531, pp. 209–220, Nov. 1998.

[6] Fenner et al., "Protocol independent multicast - sparse mode (pim-sm)," *IETF RFC 4601*, Dec. 2007.

[7] F. K. Hwang and D. S. Richards, "Steiner tree problems," *IEEE Networks*, vol. 22, no. 1, pp. 55–89, Oct. 1992.

[8] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," in *ACM SIGCOMM '10*, pp. 351–362.

[9] Y. Kanizo, D. Hay, and I. Keslassy, "Palette: Distributing tables in software-defined networks," in *IEEE INFOCOM'13*, pp. 545–549.

[10] D.-N. Yang and W.-J. Liao, "Optimal state allocation for multicast communications with explicit multicast forwarding," *IEEE Trans. on Parallel and Distributed Systems*, vol. 19, no. 4, pp. 476–488, 2008.

[11] D.-N. Yang and W. Liao, "Protocol design for scalable and adaptive multicast for group communications," in *IEEE ICNP'08*, pp. 33–42.

[12] I. Stoica, T. Ng, and H. Zhang, "REUNITE: a recursive unicast approach to multicast," in *IEEE INFOCOM'00*, pp. 1644–1653.

[13] K. Almeroth, "The evolution of multicast: from the mbone to interdomain multicast to internet2 deployment," *IEEE Networks*, vol. 14, no. 1, pp. 10–20, Jan 2000.

[14] G. Robins and A. Zelikovsky, "Improved steiner tree approximation in graphs," in *ACM SODA'00*, pp. 770–779.

[15] Sushant Jain et al., "B4: Experience with a globally-deployed software defined WAN," in *ACM SIGCOMM'13*, pp. 3–14.

[16] Qazi et al., "SIMPLE-fying middlebox policy enforcement using SDN," in *ACM SIGCOMM'13*, pp. 27–38.

[17] D.-N. Yang and W. Liao, "On bandwidth-efficient overlay multicast," *IEEE Trans. on Parallel and Distributed Systems*, vol. 18, no. 11, pp. 1503–1515, Nov. 2007.

[18] E. Aharoni and R. Cohen, "Restricted dynamic steiner trees for scalable multicast in datagram networks," *IEEE/ACM Trans. on Networking*, vol. 6, no. 3, pp. 286–297, Jun. 1998.

[19] B. Leng, L. Huang, X. Wang, H. Xu, and Y. Zhang, "A mechanism for reducing flow tables in software defined network," in *IEEE ICC'15*, pp. 6924–6929.

[20] S. Q. Zhang, Q. Zhang, H. Bannazadeh, and A. Leon-Garcia, "Network function virtualization enabled multicast routing on SDN," in *IEEE ICC'15*, pp. 7217–7223.

[21] A. Craig, B. Nandy, I. Lambadaris, and P. Ashwood-Smith, "Load balancing for multicast traffic in SDN using real-time link cost modification," in *IEEE ICC'15*, pp. 7418–7424.

[22] L.-H. Huang, H.-J. Hung, C.-C. Lin, and D.-N. Yang, "Scalable and bandwidth-efficient multicast for software-defined networks," in *IEEE GLOBECOM'14*, pp. 1890–1896.

[23] S.-H. Shen, L.-H. Huang, D.-N. Yang, and W.-T. Chen, "Reliable multicast routing for software-defined networks," in *IEEE INFOCOM'15*, 2015, pp. 181–189.

[24] P. Cameron, "Combinatorics: Topics, techniques, algorithms." Cambridge University Press, 1994.

[25] L.-H. Huang, H.-C. Hsu, S.-H. Shen, D.-N. Yang, and W.-T. Chen, "Multicast traffic engineering for software-defined networks," *CoRR*, vol. arXiv:1507.08728, 2016.

[26] M. Fisher, G. Nemhauser, and L. Wolsey, "An analysis of approximations for maximizing submodular set functions-II," *Springer Berlin Heidelberg*, vol. 8, pp. 73–87, 1978.

[27] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger, "Approximation algorithms for knapsack problems with cardinality constraints," *European Journal of Operational Research*, vol. 12, no. 2, pp. 333–345, 2000.

[28] "The internet topology zoo," 2014. [Online]. Available: http://www.topology-zoo.org/dataset.html

[29] "EstiNet 8.1 OpenFlow Network Simulator and Emulator," 2014. [Online]. Available: http://www.estinet.com/

[30] D. Xu, M. Chiang, and J. Rexford, "Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering," *IEEE/ACM Trans. on Networking*, vol. 19, no. 6, pp. 1717–1730, Dec 2011.

[31] "IBM ILOG CPLEX," 2014. [Online]. Available: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/