Scalable Routing in SDN-enabled Networks with Consolidated Middleboxes

Andrey Gushchin
Cornell University
avg36@cornell.edu

Anwar Walid
Bell Labs, Alcatel-Lucent
anwar.walid@alcatel-lucent.com

Ao Tang
Cornell University
atang@ece.cornell.edu

ABSTRACT

Middleboxes are special network devices that perform various functions such as enabling security and efficiency. SDN-based routing approaches in networks with middleboxes need to address resource constraints, such as memory in the switches and processing power of middleboxes, and traversal constraint where a flow must visit the required middleboxes in a specific order. In this work we propose a solution based on MultiPoint-To-Point Trees (MPTPT) for routing traffic in SDN-enabled networks with consolidated middleboxes. We show both theoretically and via simulations that our solution significantly reduces the number of routing rules in the switches, while guaranteeing optimum throughput and meeting processing requirements. Additionally, the underlying algorithm has low complexity making it suitable in dynamic network environment.

CCS Concepts

•Networks → Middle boxes / network appliances; Traffic engineering algorithms;

Keywords

Software-Defined Networking, Middlebox, Multipoint-to-Point Tree, Traffic Engineering

1. INTRODUCTION

Middleboxes (e.g. proxies, firewalls, IDS, WAN optimizers, etc.) are special network devices that perform functional processing of network traffic in order to achieve a certain level of security and performance. Each network flow may require certain set of functions. In some cases these functions can be applied only in a particular order, which makes routing in networks with middleboxes under limited resources constraints even a more difficult task. Mechanism of controlling routing through the specified functional sequence is called Service Function Chaining (SFC). Logically centralized traffic control offered by SDN enables traffic routing optimization (in terms of device costs, total throughput, load balancing, link utilizations, etc.), while satisfying a correct traversal of network middleboxes for each flow. Several recent works (e.g. [10], [6], [7]) provide relevant solutions.

Functionality provided by middleboxes can be incorporated in the network in several ways. Traditional middlebox is a standalone physical device that can typically perform one network function, and may be located at an ingress switch. With the development of the Network Function Virtualization (NFV), middleboxes may be implemented using Virtual Machines (VMs) that can be flexibly installed at the Physical Machines (PMs). In addition, virtualization enables implementation of the consolidated middleboxes [12], where a flow receives all of its required service functions at a single machine. The consolidated middlebox model simplifies traffic routing and helps reduce the number of routing rules in the switches.

In this paper, we follow the model in [4], and assume that each middlebox function is an application that can be installed at certain VMs within the PMs. It is also assumed that every flow obtains all its required functional treatment at a single PM, and thus the consolidated middlebox model is implied in the paper. Network function consolidation and flexible implementation of middleboxes were previously discussed, for example in [7], [8], [2] and [12].

Depending on the network traffic environment, two types of routing schemes can be developed: offline, where all required traffic demands are given or can be estimated (for example, using a service level agreement between the customer and the provider), and online, where demands are unknown and a routing solution for each incoming flow is made based on the flow class and the current state of the network. A solution obtained by a routing scheme can be converted into a set of routing rules that are installed in the switches. Different criteria can be used to characterize the achievable performance of a routing scheme: total throughput, average delay, maximum PM utilization, etc. Besides achieving a desired network performance, a routing scheme must also satisfy resource and routing constraints. Additionally, three new constraints are of a special interest in the SDN-enabled networks with middleboxes.

- **Switch memory capacities:** number of rules installed in a single switch is limited by its memory capacity. Ternary Content-Addressable Memory (TCAM) used in SDN switches is a scarce resource which is expensive both in terms of cost and power consumption.
- **Middlebox processing capacities:** load on each middlebox should not exceed its processing capacity. Overload of middleboxes has to be avoided since it may cause loss of traffic, delay, incorrect traversal sequence or other problems.
- **Traversal constraints:** required network functions have to be applied to any given flow in a correct order.

The switch memory constraint is important: flow table overflow is a serious problem that can significantly degrade

HotMiddlebox'15, August 17-21, 2015, London, United Kingdom

DOI: http://dx.doi.org/10.1145/2785989.2785999

$m(v_j)$	capacity of switch $v_j \in V_{sw}$
$r(v_j)$	number of rules in switch $v_j \in V_{sw}$
$b(v_j)$	capacity of PM $v_j \in V_{pm}$
$g(e)$	capacity of link e
com_i $< s_i, t_i, d_i, c_i >$	commodity i with source s_i, destination t_i, demand d_i, and class c_i
p_i	cost (in PM resources) of com_i
M	total number of commodities
C	number of different traffic classes
V_T	set of distinct destinations

Table 1: Main notations.

network performance and, therefore should be avoided. Because this constraint is of integer type, it makes the problem of finding an optimal routing solution hard. If, in addition, middleboxes are added to the network, finding such routing becomes even harder.

In this paper we present an approach based on multi-point-to-point trees that efficiently finds a routing with a guarantee on the maximum number of rules in a single switch, while satisfying all other network constraints. Moreover, our routing solution scales well with the network size: the explicit bound $C+2|E_0|+|V_T|-2|V_{pm}|$ on the number of rules is additive and depends linearly on the number of destination nodes ($|V_T|$), links ($|E_0|$) and flow classes (C) in the network.

2. PROBLEM FORMULATION

2.1 Network Topology and Resources

We assume that the network topology is defined by a directed graph $G_0 = (V_0, E_0)$, where V_0 is the set of its nodes and E_0 is the set of edges. Each node corresponds either to a switch or to a PM, and each edge is a link connecting either two switches, a switch with a PM, or a PM with a switch. We denote by V_{sw} and V_{pm} the node sets corresponding to switches and PMs, respectively, so that $V_{sw} \cup V_{pm} = V_0$, and $V_{sw} \cap V_{pm} = \emptyset$. It will be assumed for simplicity that each PM is connected with a single switch by bi-directional links as shown in Fig. 1a. Let $V_{sw \to pm}$ be the subset of nodes in V_{sw} that are directly connected to the PM nodes ($V_{sw \to pm} = \{sw1, sw2, sw6\}$ in Fig. 1a).

Each switch has a certain memory capacity that can be expressed as a number of rules that it can accommodate. We will denote this number by $m(v_j)$ for a switch located at node v_j ($j = 1, \ldots, |V_{sw}|$), where $|A|$ is the cardinality of a set A. Additionally, let $r(v_j)$ be the number of rules in this switch in a routing solution.

Although a PM may have several types of resources (e.g. memory, CPU), it will be assumed for simplicity that each PM is characterized by a single resource capacity that will be denoted by $b(v_j)$ for a PM located at node v_j ($j = 1, \ldots, |V_{pm}|$). Similarly, each link $e_k \in E_0$ ($k = 1, \ldots, |E_0|$) has an associated link capacity that will be denoted by $g(e_k)$.

2.2 Network Functions and Commodities

There exist several types of network functions (firewall, IPS, IDS, WAN optimization, etc.), and each function has its own cost per unit of traffic in terms of PM resources. Although in this work we assume that this processing cost is the same for all PMs, it is easy to generalize it to the case when the costs are distinct for different PMs.

Additionally, there is a set of M traffic demands or "commodities" that have to be routed in the network. We will use the terms traffic demand and commodity interchangeably. Commodity com_i is defined by a four-tuple $com_i = < s_i, t_i, d_i, c_i >$, where $i = 1, \ldots, M$. Here $s_i \in V_{sw}$ and $t_i \in V_{sw}$ are, respectively, source and destination nodes, d_i is an amount of flow that has to be routed for commodity com_i, which we will call the commodity's demand, and c_i is an ordered set of network functions required by this commodity. Any such ordered set of network functions defines the class of a commodity. We will denote by C the total number of different classes of traffic demands. Due to various functional requirements, different commodities may have different per unit of traffic costs in terms of PM's processing power. Let $p(i)$ be such cost per unit of traffic for traffic demand com_i.

Each PM hosts at most C VMs, where a single VM corresponds to a single commodity class. It is assumed that when a packet from a commodity of class k arrives to a PM, it is transfered to the virtual machine associated with class k, and all network functions of class k are applied to this packet in a correct order. Distribution of each PM's processing capacity among C VMs has to be determined. It is assumed, however, that positions of PMs (nodes V_{pm}) are given as an input and are not subject to change.

By V_T we will denote the set of distinct destinations, then $|V_T| \leq M$, $|V_T| \leq |V_{sw}|$. Main notations are summarized in Table 1.

2.3 Routing via Integer Linear Optimization

In this work we employ the idea of consolidated middleboxes, and each packet belonging to com_i gets all functional treatment specified by c_i at a single PM. It is allowed, however, that a single commodity's traffic is split into several paths from s_i to t_i, and distinct paths may traverse distinct PMs. We point out that splitting occurs at the IP flow level and not at the packet level. This is similar to Equal Cost Multipath [5] in Data Centers, where hashing is used to split traffic at the IP flow level for routing on multiple paths.

If the traffic demands are known in advance, an optimization problem can be posed whose feasible solution defines a routing that satisfies all network constraints. The variables of this optimization problem $f_i^x(e)$ are the amount of traffic of commodity com_i on edge $e \in E_0$, $i = 1, \ldots, M$. Here superscript $x \in \{0, 1\}$ with zero value corresponds to the traffic that has not visited a consolidated middlebox, and unit value is used to denote the traffic that has been processed by the required network functions. There are thus $2 \cdot M \cdot |E_0|$ variables in this optimization problem. Let $d_i(v)$ be the demand from a node $v \in V_{sw}$ for the commodity com_i. Note that $d_i(v) = d_i$ if $v = s_i$, and is zero, otherwise. The problem is formulated as follows.

ILP Optimization (1):

$$\textbf{min} \sum_{\substack{e \in E_0, 1 \leq i \leq M, \\ x \in \{0,1\}}} f_i^x(e),$$

$$\forall v \in V_{sw},\ \forall i:\ 1 \leq i \leq M:$$

$$\sum_{(v,w) \in E_0} f_i^0(v,w) - \sum_{(u,v) \in E_0} f_i^0(u,v) = d_i(v), \tag{1a}$$

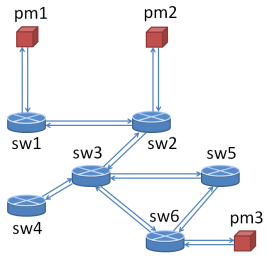
$$\forall v \in V_{sw}:\ v \neq t_i,\ \forall i:\ 1 \leq i \leq M:$$

$$\sum_{(v,w) \in E_0} f_i^1(v,w) - \sum_{(u,v) \in E_0} f_i^1(u,v) = 0, \tag{1b}$$

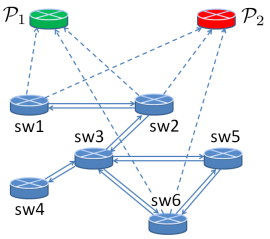
$$\forall v \in V_{pm},\ \forall i:\ 1 \leq i \leq M:\ f_i^1(u,v) = 0, \tag{1c}$$

$$\forall v \in V_{pm},\ \forall i:\ 1 \leq i \leq M:\ f_i^0(u,v) = f_i^1(v,u), \tag{1d}$$

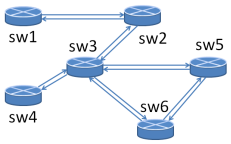
$$\forall v \in V_{pm}:\ \sum_{1 \leq i \leq M} p(i) \cdot f_i^0(u,v) \leq b(v), \tag{1e}$$



(a) Example of a network topology defined by graph $G_0 = (V_0, E_0)$.



(b) Grpah $G_1 = (V_1, E_1)$ constructed from graph G_0 at Step 1.



(c) Graph $G_2 = (V_2, E_2)$ constructed from graph G_1 at Step 2.

Figure 1: Example of a given graph G_0 and graphs G_1 and G_2 constructed at the first and the second steps of our algorithm, respectively.

$$\forall e \in E_0 : \sum_{1 \le i \le M} f_i^0(e) + \sum_{1 \le i \le M} f_i^1(e) \le g(e), \quad (1f)$$

$$\forall v \in V_{sw} : \; r(v) \le m(v), \quad (1g)$$

$$\forall e \in E_0, \; \forall i : \; 1 \le i \le M, \; \forall x \in \{0,1\} : \; f_i^x(e) \ge 0. \quad (1h)$$

Constraints (1a) and (1b) are flow conservation constraints for switches, constraint (1c) forbids the traffic that has already been processed by a middlebox (PM), to visit a middlebox again. Next constraint (1d) says that all unprocessed traffic becomes processed at the PM associated with node $v \in V_{pm}$. Further, constraint (1e) is a PM processing capacity constraint. The following constraint (1f) is a link capacity constraint, and condition (1g) corresponds to the switch memory constraint. Finally, (1h) requires that all flow values are nonnegative. The objective function of this optimization problem is the total flow over all edges. This choice of the objective function guarantees that no cycles will exist in an optimal solution. Notice that there is no constraint $f_i^0(v,u) = 0$ similar to constraint (1c), because it will be automatically satisfied due to the optimization's objective function.

Solution to this optimization problem expressed in terms of variables $f_i^x(e)$ can be translated to a path-flow formulation [1], and the routing rules in switches can be obtained that implement this path-flow solution. Each routing rule in a switch corresponds to a single path in the path-flow solution. Notice that in the solution to the optimization problem, more than one source-destination path can be used to transfer traffic for a single commodity.

The optimization problem formulated above contains integer switch memory constraints (1g) and thus belongs to the class of Integer Linear Programs (ILP). This problem, therefore, is NP-hard, and it is extremely difficult to obtain its solution. In this work, we adapt the idea of multipoint-to-point trees to construct a feasible routing scheme for SDN-enabled networks with middleboxes and known traffic demands. Although the integer switch memory constraints are not explicitly incorporated into our solution, we can obtain the worst case bound on the number of rules in each switch. Moreover, we show that this bound scales well with the network size and is low enough for our routing scheme to be implemented in the networks with existing switches.

3. SOLUTION OVERVIEW

3.1 MPTPT Approach

In this work we take advantage of the capabilities provided by SDN to design efficient routing. In particular, SDN facilitates global design optimization based on inputs and measurements collected from various points of the network, and the ability to translate design solutions into rules which can be downloaded to the switches. One of the major components of our routing solution is MPTP trees that were previously used, for example, by the label based forwarding mechanism of MPLS [11]. Each MPTP tree is rooted at some node, and all its edges are oriented towards this root node. Such trees can be used to route traffic from several sources to a single destination, and each tree is assigned with its own tag which is used to label all traffic belonging to this tree. Utilization of MPTPTs helps to reduce the number of routing rules in the whole network [3].

Our solution contains two main steps. These steps are purely computational (not actual routing steps), and allow to determine how the traffic for each commodity is labeled and routed. At the first step we route all traffic from the sources s_i, $(i = 1, \ldots, M)$ to PMs. At the second step, we route all traffic that has been processed by the required network functions during the first step from the PMs to the corresponding destinations t_i, $(i = 1, \ldots, M)$. Both steps involve construction of MPTP trees: there are C roots for MPTP trees built at the first step, where each root corresponds to a particular flow class, and there are $|V_T|$ roots for the trees at the second step. There can be in general more than one MPTP tree rooted at a single node. In Fig. 2 we show the schematic of our MPTPT-based routing algorithm.

3.2 Step 1: Routing from Sources to PMs

At the first step we consider a graph $G_1 = (V_1, E_1)$ which is obtained from the initial graph G_0 as follows: we add C additional nodes $\mathcal{P}_1, \ldots, \mathcal{P}_C$ such that node $\mathcal{P}_k$ corresponds to the traffic class k. This set of C new nodes is denoted by $V_{\mathcal{P}}$, and $|V_{\mathcal{P}}| = C$. We further remove "PM" nodes belonging to the set V_{pm}, together with the edges going to and from these nodes. Then, we connect each node from $V_{sw \to pm}$ by edges to every node from $V_{\mathcal{P}}$. These new edges are not assigned with capacities explicitly, but the maximum amount of flow on them will be determined by the capacities of PMs and the capacities of removed links from graph G_0 that were connecting nodes in $V_{sw \to pm}$ with nodes V_{pm}. The vertex set of graph G_1 is a union of node sets V_{sw} and $V_{\mathcal{P}}$: $V_1 = V_{sw} \cup V_{\mathcal{P}}$. Number of links in the graph G_1 is $|E_1| = |E_0| + |V_{pm}| \cdot (C-2)$. In Fig. 1 we show an example of a network topology defined by a graph G_0 (Fig. 1a) and corresponding constructed graph G_1 (Fig. 1b). In this example it is assumed that there are two classes of flows and the nodes $\mathcal{P}_1$ and $\mathcal{P}_2$ are associated with flow classes one and two, respectively. In Fig. 1b the new added links are shown by dashed arrows.

We additionally modify destinations of the given commodities. In particular, destination of all traffic demands of class k is node $\mathcal{P}_k$, $k = 1, \ldots, C$. Therefore, for each commodity com_i, its destination is one of the nodes in $V_{\mathcal{P}}$. We can now formulate an LP optimization problem that we solve at the first step of our method. In contrast to the commodity-based ILP problem (1), the optimization here is in a tree-based formulation, and we do not distinguish traffic from different sources if they are for the same destination, i.e. if they belong to the same network class. Let $\hat{v}$ denote

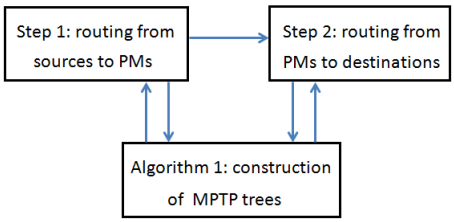


Figure 2: Schematic of the MPTPT-based routing algorithm.

a PM connected to node $v \in V_{sw\to pm}$ in graph G_0 (for example, $\hat{v} = pm3$ for $v = sw6$ in the example from Fig. 1), and $p(t)$, where $t \in V_{\mathcal{P}}$, denotes the cost of PM resources per unit of traffic of class corresponding to the node t.

LP Optimization (2) of Step 1:

$$\min \sum_{e\in E_1, t\in V_{\mathcal{P}}} f_t(e),$$

$$\forall t \in V_{\mathcal{P}},\ \forall v \in V_1,\ v \neq t:$$

$$\sum_{(v,w)\in E_1} f_t(v,w) - \sum_{(u,v)\in E_1} f_t(u,v) = d_t(v), \tag{2a}$$

$$\forall e \in E_1 \cap E_0: \sum_{t\in V_{\mathcal{P}}} f_t(e) \le g(e), \tag{2b}$$

$$\forall v \in V_{sw\to pm}: \sum_{t\in V_{\mathcal{P}}} f_t(v,t) \le \min\{g(v,\hat{v}), g(\hat{v},v)\}, \tag{2c}$$

$$\forall v \in V_{sw\to pm}: \sum_{t\in V_{\mathcal{P}}} p(t)\cdot f_t(v,t) \le b(\hat{v}), \tag{2d}$$

$$\forall e \in E_1,\ \forall t \in V_{\mathcal{P}}:\ f_t(e) \ge 0. \tag{2e}$$

In this optimization problem variable $f_t(e)$ is an amo-unt of flow to destination $t \in V_{\mathcal{P}}$ on link $e \in E_1$. Constraint (2a) is a flow conservation at node v, condition (2b) is a link capacity constraint that should be satisfied for any link that belongs to the both edge sets E_0 and E_1 of graphs G_0 and G_1, respectively. Further, constraint (2c) is a link capacity constraint for the links that connect switches with PMs in graph G_0. This constraint is necessary for feasibility of the solution to optimization problem (2) in the original graph G_0. Notice that in the right hand side of (2c) there is a minimum between capacities of the links going from a switch to a PM and from a PM to a switch. It will guarantee that all traffic processed at a PM can be sent back to a switch connected to this PM. Next constraint (2d) is a PM capacity constraint, and by (2e) we require that flow on each link is nonnegative. As in the ILP optimization (1), we minimize the total network flow to avoid cycles.

Solution to the optimization problem (2) determines how the traffic is routed from the sources to the PMs. Using Algorithm Flow2Trees(t) from [3] that is listed as Algorithm 1 below for completeness, from a basic feasible solution [1] $f_t(e)$ to the LP (2) we construct MPTP trees rooted at the destination nodes from $V_{\mathcal{P}}$, so that all network traffic in the solution is distributed among these trees. Each tree contains traffic of the same class, leafs of a tree are the sources for this traffic class, and amount of traffic from each source in any tree can be determined. It is possible that several $V_{sw\to pm}$ nodes belong to the same tree, i.e. one tree can route traffic to several PMs. Algorithm 1 is iteratively applied to construct trees to each destination $t \in V_{\mathcal{P}}$. We will provide an upper bound on a total number of trees in the subsection 3.4. We refer the reader to [3] for the details and analysis of Algorithm 1.

Algorithm 1: Flow2Trees(t)

Input : $G = (V, E)$, t, $f_t(e)$ $(\forall e \in E)$.
Output: Set of MPTP trees rooted at t and containing all traffic to t.

while *there is a source s with demand to t* **do**
  using only edges e with flow to t ($f_t(e) > 0$), construct a tree R to t spanning all sources with demand to t;
  move as much flow as possible to R;
end

3.3 Step 2: Routing from PMs to Destinations

At the second step of our algorithm we use MPTP trees to route traffic from the PMs to destinations in graph G_2 obtained from G_1 as follows. First, nodes $V_{\mathcal{P}}$ and links to them are removed from the network. Therefore, the node set of the resulting graph $G_2 = (V_2, E_2)$ only contains nodes from V_{sw}: $V_2 = V_{sw}$. Number of links in graph G_2 is $|E_2| = |E_1 \cap E_0| = |E_0| - 2\cdot|V_{pm}|$. Second, the link capacities are updated: for each link e, the amount of traffic on it in the solution to (2) is subtracted from this link's initial capacity $g(e)$. We will denote by $\bar{g}(e)$ the updated capacity of link e. Graph G_2 corresponding to graph G_0 from Fig. 1a is shown in Fig. 1c.

We then create a set of commodities for the second step. It is assumed that all traffic processed at a PM $\hat{v}$ returns to switch $v \in V_{sw\to pm}$ connected to it. Therefore, all traffic at Step 2 is routed from the nodes $V_{sw\to pm}$ to the destinations t_i, where $i = 1, \ldots, |V_T|$. Solution to optimization (2) determines amount of traffic of every class and from every source arriving to each PM. However, amount of traffic to each destination t_i arriving to a PM, in some cases can not be determined unambiguously. This can happen when there exist more than one commodities with the same source and of the same class but with different destinations. We will use the following heuristic to determine the traffic distribution by destination at each node $v \in V_{sw\to pm}$. Let R be a set of trees obtained at Step 1 of our algorithm that carry traffic of the same class c from a source node s to the root node $\mathcal{P}_c$ corresponding to this traffic class. In addition, let T be the set of destinations of commodities with source s and of class c, and $d_1, \ldots, d_{|T|}$ are corresponding demands. By the definition of a tree, in each tree R_i from set R, there is a unique path from s to $\mathcal{P}_c$, and therefore, all traffic from s in the same tree obtains functional treatment at a single PM. According to our heuristic, in each tree R_i, amount of traffic to destination $t \in T$ is proportional to the fraction of traffic to this destination in the total amount of traffic to all destinations, i.e. proportional to $d_t / \sum_{i=1}^{|T|} d_i$. We refer the reader to the extended version of our paper [9] for a more detailed description of the heuristic and an example.

Using this distribution heuristic, we form a set of commodities for the second step of our algorithm. At the Step 2 we do not distinguish traffic from different sources and from different network classes if they have the same destination. We construct MPTP trees with the roots at the destinations t_i, $i = 1, \ldots, |V_T|$. Similarly to Step 1, we first solve the following LP:

LP Optimization (3) of Step 2:

$$\min \sum_{e\in E_2, t\in V_T} f_t(e),$$

$$\forall t \in V_T,\ \forall v \in V_2,\ v \neq t:$$

$$\sum_{(v,w)\in E_2} f_t(v,w) - \sum_{(u,v)\in E_2} f_t(u,v) = d_t(v), \tag{3a}$$

$$\forall e \in E_2 : \sum_{t \in V_T} f_t(e) \leq \bar{g}(e), \quad (3b)$$

$$\forall e \in E_2, \forall t \in V_T : f_t(e) \geq 0. \quad (3c)$$

Here (3a) and (3b) are flow conservation and link capacity constraints, respectively, and (3c) is a requirement for flows to be non negative on each link. Using a basic feasible solution to this problem, we apply again Algorithm 1 and obtain another set of MPTP trees. Complete version of our MPTPT-based routing approach is summarized in Algorithm 2.

Algorithm 2: MPTPT-Based Routing

Input : $G_0 = (V_0, E_0)$, commodities com_i $(i = 1, \ldots, M)$.
Output: Set of MPTP trees rooted at PM nodes and destination nodes.

Step 1: routing from sources to PMs:
construct graph $G_1 = (V_1, E_1)$ from $G_0 = (V_0, E_0)$;
obtain commodities for Step 1;
find a basic feasible solution to LP (2);
find MPTP trees for the solution to LP (2) using Algorithm 1;
Step 2: routing from PMs to destinations:
construct graph $G_2 = (V_2, E_2)$ from $G_1 = (V_1, E_1)$;
obtain commodities for Step 2;
find a basic feasible solution to LP (3);
find MPTP trees for the solution to LP (3) using Algorithm 1.

After both steps of our algorithm are performed, we can determine for any initial commodity $< s_i, t_i, d_i, c_i >$ what trees carry its traffic to the destination t_i. Each commodity's packet is assigned with two tags at the source switch: one for a tree label from Step 1, and another one for a tree label from Step 2. The first label can be removed from a packet during functional processing at a PM, and therefore the maximum number of routing rules in a single switch does not exceed the total number of MPTP trees of both steps. As suggested in previous works (e.g. [10]), VLAN and ToS fields of a packet header can be used for labels.

3.4 Analysis

In this subsection we provide and prove an upper bound on the total number of MPTP trees generated by Algorithm 2. Each tree has its own label and any switch may contain at most one routing rule corresponding to this tree. The bound, therefore, also limits the number of routing rules in any switch.

PROPOSITION 1. *Number of MPTP trees produced by Algorithm 2 does not exceed* $C + 2|E_0| + |V_T| - 2|V_{pm}|$.

PROOF. It was shown in [3] that when Algorithm 1 is iteratively applied to a basic feasible solution of the multi-commodity flow problem (3), the maximum possible number of created trees is $|V_T|+|E_2|$, i.e. bounded above by the sum of number of destinations and number of links in a network. The second term in this sum ($|E_2|$) corresponds to the number of bundle constraints in LP. A constraint is called bundle if it involves variables for different destinations. In optimization problem (3) link capacity constraints (3b) are bundle, and there are $|E_2|$ such constraints. Although optimization problem (2) is slightly different from (3), a similar bound for it can also be established. Number of bundle constraints in (2) is $|E_0| - 2 \cdot |V_{pm}| + |V_{pm}| + |V_{pm}| = |E_0|$, and number of destinations is equal to the number of traffic classes C. Therefore, the total number of trees produced by Algorithm 2 is $C + |E_0| + |V_T| + |E_2| = C + 2|E_0| + |V_T| - 2|V_{pm}|$. □

Notice that while our bound depends on the number of classes C, it does not depend on the number of commodities, because $|V_T|$ is bounded by $|V_{sw}|$. The bound is additive and thus scales well with the network size. Moreover, as shown by simulations, the real number of routing rules obtained by our algorithm is generally much smaller than this worst case bound. It is crucial that a basic feasible solution is used as an input to the Algorithm 1 at both steps of Algorithm 2. We refer the reader to [1] and [3] for a more detailed discussion of basic feasible solutions and bundle constraints.

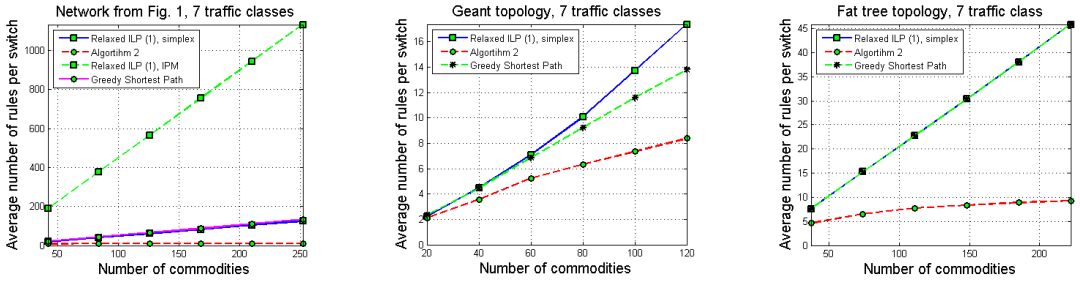
Therefore, Algorithm 2 efficiently solves a routing problem (it contains two linear optimizations and Algorithm 1 with polynomial time complexity) with a guarantee that the number of routing rules in each switch is limited by an additive bound.

4. EVALUATION

In this section we evaluate the performance of Algorithm 2 and compare it with three other routing schemes. The first routing scheme is defined by optimization problem (1) with relaxed integer switch memory constraint, and a basic feasible solution for it is found using simplex method. The second scheme uses the same relaxed LP, but an interior point method (IPM) is applied to find a solution. Finally, the third scheme is based on a greedy shortest path approach. In this approach the commodities are initially sorted in descending order by their total PM capacity requirement. Then, iteratively for each commodity a shortest path is found from its source to a PM, and then a shortest path from the PM to commodity's destination. If link and PM capacity constraints on the shortest path do not allow to send commodity's total demand, a maximum possible fraction of it is sent along this path, and the remaining traffic is sent along the next shortest paths until all commodity's demand is routed. If at some point there is no path available to send commodity's residual demand, the algorithm stops.

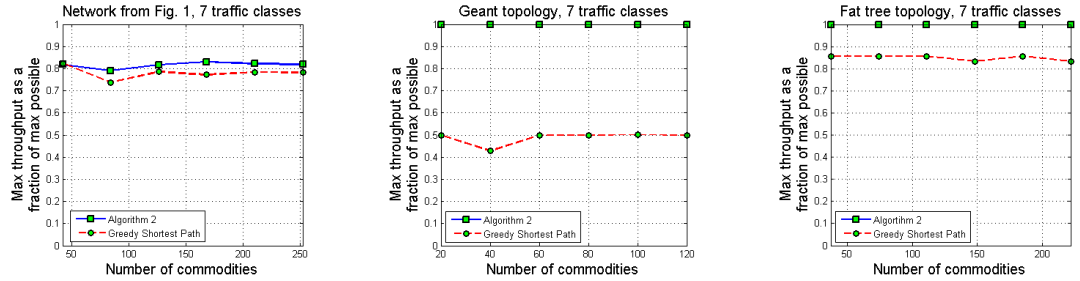
Our evaluation analysis consists of two experiments. In the first experiment we find routing solution using each of four algorithms and calculate an average number of routing rules in switches for each solution. Second experiment allows to estimate for each routing algorithm the maximum total throughput that it can route. Both experiments are carried out for three network topologies: example from Fig. 1, Geant topology, and fat tree topology. Geant network contains 41 switch nodes and 9 additional PM nodes that are connected to 9 switch nodes having the highest nodal degree (so that each PM is connected to exactly one switch). Fat tree topology consists of 22 switch nodes (2 core, 4 aggregation and 16 edge switches), and 6 PM nodes such that each PM node is connected to a single core or aggregate switch node. Link and PM capacities were fixed in each simulation, and took values, respectively, 100 and 500 for the network on Fig. 1, and 500, 500 for Geant topology. For the fat tree topology links between core and aggregation switches had capacities 200, links between aggregation and edge switches had capacities 10, and links between switches and PMs were fixed at 100. In addition, each PM had capacity 500.

Experiment 1: Average Number Of Routing Rules. In the first experiment we varied number of classes and number of commodities, and each commodity's source, destination and class were generated randomly. The demands of the commodities, however, were all equal and fixed at 0.2. Results of Experiment 1 for 7 traffic classes are shown in Fig. 3. It can be observed from the results that Algorithm 2 allows to reduce average number of routing rules in switches by a factor of up to 10. We did not add plots corresponding to the interior point method solution for Geant and fat tree topologies because in the IPM solution average number of rules is much higher compared to the other algorithms.



(a) Network topology from Fig. 1. (b) Geant topology. (c) Fat tree topology.

Figure 3: Comparison of average numbers of routing rules in switches for seven traffic classes.



(a) Network topology from Fig. 1. (b) Geant topology. (c) Fat tree topology.

Figure 4: Comparison of maximum total throughputs for seven traffic classes.

We also performed simulations for one, three and five traffic classes, and the results look similar to Fig. 3. The values of bounds on the maximum number of rules in switches are 43, 295 and 137 for the topologies in the same order they are presented in Fig. 3 and for 7 traffic classes. These values were obtained under assumption that $|V_T| = |V_{sw}|$ and therefore, limit the number of routing rules in each switch for any arbitrary large number of commodities.

Experiment 2: Maximum Total Throughput. In the second experiment we measured the maximum total throughput that can be routed in a network by the Algorithm 2. Notice, that ILP (1) with relaxed switch memory constraint always finds a routing solution when it exists. Therefore, we used the relaxed LP (1) to determine the maximum possible network throughput. For a given set of commodities, we increased iteratively demands of all commodities by the same value until the relaxed LP (1) became unfeasible. We stored this maximum demand value, and repeated the procedure for the Algorithm 2 and also for the Greedy Shortest Path algorithm. Results provided in Fig. 4 demonstrate that a loss in maximum throughput of the Algorithm 2 is relatively small.

5. CONCLUSION

In this work we proposed a multipoint-to-point tree based algorithm for SDN-enabled networks with middleboxes and given required traffic demands. We showed both theoretically and experimentally that in the routing solution obtained by our algorithm, the maximum number of routing rules in a single switch is bounded, and this explicit bound scales well with the network size. Moreover, the low complexity of the algorithm allows its application the algorithm in dynamic network environment.

Acknowledgements

The research is partially supported by ONR under N00014-12-1-1055.

6. REFERENCES

[1] R. K. Ahuja, T. L. Magnanti and J. B. Orlin. Network flows: theory, algorithms, and applications, 1993.

[2] J. W. Anderson, R. Braud, R. Kapoor, G. Porter and A. Vahdat. xOMB: extensible open middleboxes with commodity servers. In *Proc. ANCS*, 2012.

[3] D. Applegate and M. Thorup. Load optimal MPLS routing with N+ M labels. In *Proc. IEEE INFOCOM*, 2003.

[4] M. Charikar, Y. Naamad, J. Rexford and X. K. Zou. Multi-Commodity Flow with In-Network Processing.

[5] M. Chiesa, G. Kindler and M. Schapira. Traffic Engineering with ECMP: An Algorithmic Perspective.

[6] S. K. Fayazbakhsh, V. Sekar, M. Yu and J. C. Mogul. FlowTags: enforcing network-wide policies in the presence of dynamic middlebox actions. In *Proc. HotSDN*, 2013.

[7] A. Gember, R. Grandl, A. Anand, T. Benso and A. Akella. (2012). Stratos: Virtual middleboxes as first-class entities. UW-Madison TR1771.

[8] A. Greenhalgh, F. Huici, M. Hoerdt, P. Papadimitriou, M. Handley and L. Mathy. Flow processing and the rise of commodity network hardware. In *Proc. ACM SIGCOMM CCR*, 2009.

[9] A. Gushchin, A. Walid and A. Tang. (2015) Scalable routing in SDN-enabled networks with consolidated middleboxes, arXiv preprint.

[10] Z. Qazi, C. Tu, L. Chiang, R. Miao, V. Sekar and M. Yu. SIMPLE-fying middlebox policy enforcement using SDN. In *Proc. ACM SIGCOMM*, 2013.

[11] E. Rosen, A. Viswanathan and R. Callon. Multiprotocol label switching architecture, RFC 3031, 2001.

[12] V. Sekar, N. Egi, S. Ratnasamy, M. Reiter and G. Shi. Design and implementation of a consolidated middlebox architecture. In *Proc. NSDI*, 2012.