NFV Middlebox Placement with Balanced Set-up Cost and Bandwidth Consumption

Yang Chen
Temple University
yang.chen@temple.edu

Jie Wu
Temple University
jiewu@temple.edu

ABSTRACT

Network Function Virtualization (NFV) changes the way that we implement network services, or middleboxes, from expensive hardwares to software functions. These software middleboxes, also called Virtual Network Functions (VNFs), run on switch-connected commodity servers. Efficiently placing such middleboxes is challenging because of their traffic-changing effects and dependency relations. Private (used by one single flow) middleboxes can save more link bandwidth resources while shared (used by multiple flows) middleboxes cut down server resource expenses. This paper formulates the resource usage trade-off between bandwidth consumption and cost of middlebox placement as a combined cost minimization problem. After proving the NP-hardness of our problem in general topologies, we narrow down to a specific kind of topology: tree-structured networks. We study two kinds of constraints: traffic-chaining ratio and middlebox dependency relations. With homogeneous flows, we propose optimal greedy algorithms for the placement of a single middlebox first, and then multiple middleboxes without order. We also introduce a dynamic programming algorithm for the placement of a totally-ordered middlebox set. A performance-guaranteed algorithm is designed to handle heterogeneous flows. Extensive simulations are conducted to evaluate the performance of our proposed algorithms in various scenarios.

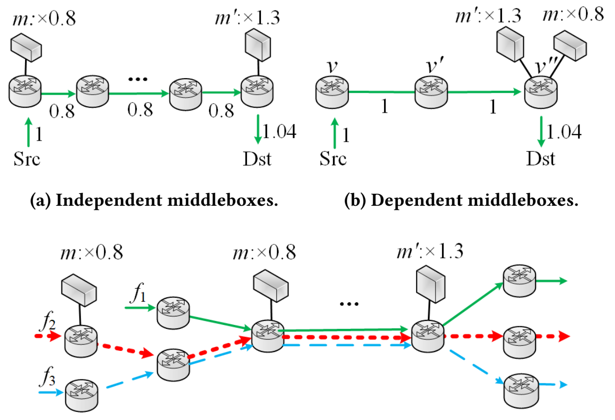
KEYWORDS

Network bandwidth, NFV, middleboxes, resource allocation, SDN.

ACM Reference Format:
Yang Chen and Jie Wu. 2018. NFV Middlebox Placement with Balanced Set-up Cost and Bandwidth Consumption. In *ICPP 2018: 47th International Conference on Parallel Processing, August 13–16, 2018, Eugene, OR, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3225058.3225068

1 INTRODUCTION

Network Function Virtualization (NFV) has been proposed to transform the implementation of network functions from expensive hardwares to software middleboxes, called Virtual Network Functions (VNFs) [9]. Software middleboxes are most commonly provisioned in modern networks, demonstrating their increasing importance [25]. As Software Defined Networking (SDN) emerges, so does a tendency to incorporate SDN and NFV in concerted ecosystems [8]. SDN manoeuvres through NFV traffic and allows middleboxes to choose service locations from multiple available servers; traditional hardwares, on the other hand, offer no such option [20].



(a) Independent middleboxes. **(b) Dependent middleboxes.**

(c) Multiple flows with shared middleboxes.

Figure 1: Middlebox placement with different constraints.

In this paper, we study the NFV middlebox placement problem in a SDN network with a given set of flows on a given topology. We assume that each flow needs to go through a given set of middleboxes, with or without a particular order. In addition, when a flow passes through a middlebox, its traffic may expand or diminish depending on the type of middlebox (this phenomenon is called *traffic-changing effect*) [16]. Multiple flows can share the same middlebox to save middlebox setup costs. The overall objective is to minimize the cost of setting up middleboxes and the cost of total bandwidth consumption by these flows. However, the flexibility that SDN offers also creates new challenges for appropriate middlebox placement because of three reasons:

(1) **Traffic-changing effects of middleboxes** [16]. For example, the Bose-Chaudhuri-Hocquenghem encoder used for satellite signaling messages adds 31% to traffic volume due to checksum overhead [18]. The Citrix CloudBridge Wide Area Network optimizer reduces traffic volume by up to 80% by compressing traffic [5].
(2) **Potential dependency relations among middleboxes** [17]. Some middleboxes have a serving order. For example, a flow might go through a Middlebox Intrusion Detection System before the proxy server [21]. An Internet Protocol Security Decryptor must always be placed before a Network Address Translation gateway [4]. We classify the dependency relations of a middlebox set into three categories: if all middleboxes must be placed in a specific sequence, the set is

ICPP 2018, August 13–16, 2018, Eugene, OR, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6510-9/18/08…$15.00
https://doi.org/10.1145/3225058.3225068

called *totally-ordered*; if there is no order requirement, the set is called *non-ordered*; if certain middleboxes requires a placement sequence, the set is called *partially-ordered.*

(3) **Sharing middleboxes**. Different flows requesting the same network function can share the processing volume of a middlebox. We assume no volume limit of a middlebox.

However, most research only focuses on reducing middlebox setup cost by sharing middleboxes without considering bandwidth consumption. The traffic-changing effect due to the middlebox placement may complicate the scheduling policy. Fig. 1 illustrates the complexity of the placement problem with different constraints. Non-order and totally-ordered middleboxes for one flow are shown in Figs. 1(a) and 1(b), respectively, and sharing middleboxes among multiple flows is shown in Fig. 1(c). The cylindrical nodes are switches and the cubical nodes are middleboxes. Middleboxes are assigned to servers (not shown in the figures) that are attached to switches. All flows need to be served by two middleboxes: m and m'. The traffic-changing ratios, which are the proportions of a flow's traffic rate before and after being processed by the middlebox, are 0.8 (diminishing traffic) and 1.3 (expanding traffic), respectively. Their paths are pre-determined, shown in different lines.

Fig. 1(a) is the optimal placement without order. Our insight is to place middleboxes that diminish traffic near the source of the flow while middleboxes that expand traffic are stationed close to the destination. Fig. 1(b) shows the optimal result with the totally-ordered constraint: m' must be served before m. Since the product of these two ratios is more than 1, we place the middleboxes as late as possible. Fig. 1(c) illustrates a more complicated case where multiple flows may share middleboxes. Flow f_2 monopolizes one m while f_1 and f_3 share one m. All these flows share one m'. This happens when f_2 traffic is so large that the reduced cost on bandwidth consumption before the shared m is more than the cost of setting up a private m. Sharing a middlebox lowers setup costs, but may be non-optimal for some flows in terms of traffic reduction. Therefore, there is a delicate trade-off between sharing middleboxes and placing a private middlebox for traffic reduction.

This paper is the first to study a middlebox placement optimization problem for multiple flows with constraints of middlebox traffic-changing ratios and their dependency relations. We prove the NP-hardness of optimally placing even a single type of middlebox in a general network topology. We then focus on tree-structured networks. For homogeneous flows with the same bandwidth, we propose three optimal algorithms for three different cases, including placing a single middlebox (as a basic solution), a non-ordered middlebox set, and a totally-ordered middlebox set. For heterogeneous flows with different bandwidths, we introduce a performance-guaranteed algorithm. Extensive simulations show the efficiency and effectiveness of our algorithms.

Our main contributions are summarized as follows:

- We prove the NP-hardness of middlebox placement in general topologies, even for placing a single type of middlebox. As a result, we narrow down to tree-structured topologies.
- We propose three optimal algorithms for homogeneous flows in tree-structured topologies under three different cases: (1) a single middlebox, (2) a non-ordered middlebox set, and (3) a totally-ordered middlebox set.
- A performance-guaranteed algorithm is introduced to handle heterogeneous flows for a non-ordered middlebox set.
- Extensive simulations are conducted to evaluate the efficiency of our proposed algorithms.

The remainder of this paper is organized as follows. Section II surveys related works. Section III describes the model and formulates the problem. Section IV introduces our algorithms for placing a single middlebox with homogeneous flows. In Section V, we handle cases with a middlebox set with homogeneous flows. Section VI studies heterogeneous flows. Section VII includes the experiments. Finally, Section VIII concludes the paper.

2 RELATED WORK

NFV frameworks have recently drawn a lot of attention, especially in the area of middlebox placement problem. We give a brief review of state-of-the-art works. For placing a single middlebox for all flows, Casado et al. propose a placement model and present a heuristic algorithm to solve the placement problem [2]. Sang et al. in [23] study the joint placement and allocation of a single middlebox, where flows can be split and served by several middlebox instances. They propose several performance-guaranteed algorithms to minimize the number of middlebox instances. However, neither study considers the middlebox traffic-changing effects. Moreover, there is always a middlebox set of various types that need to be placed.

For placing multiple types of middleboxes, most research on middlebox placement focus on placing a totally-ordered set as a service chain. Mehraghdam et al. in [17] propose a context-free language to formalize the chaining of middleboxes and describe the middlebox resource allocation problem as a mixed integer quadratically constrained program. Rami et al. locate middleboxes in a way that minimizes both new middlebox setup costs and the distance cost between middleboxes and flows' paths. They provide near optimal approximation algorithms to guarantee a placement with a theoretically proven performance [6]. Both [13] and [14] aim to maximize the number of requests for each service chain. Kuo et al. in [13] propose a systematic way to tune the proper link consumption and the middlebox setup costs in the joint problem of middlebox placement and path selection. Li et al. present the design and implementation of NFV-RT, a system that dynamically provisions resources in an NFV environment to provide timing guarantees so that the assigned flows meet their deadlines [14]. However, none of these works consider the traffic-changing effects.

Ma et al. in [16] are the first to take the traffic-changing effects into consideration. It targets load balancing instead of middlebox setting-up costs. It proposes a dynamic programming based algorithm to place a totally-ordered set, an optimal greedy solution for the middlebox placement of a non-ordered set, and proves the NP-hardness of placing a partially-ordered set. However, this work only processes a single flow and always builds new, private middleboxes without sharing with other flows, which excessively increases the setup costs of middleboxes. Sharing middleboxes among multiple flows makes the placement more challenging. In this paper, we consider not only the traffic-changing effects, but also the dependency relations in the placement of a single middlebox or various types of middleboxes for multiple flows.

Table 1: Symbols and definitions.

Symbols	Definitions
V, E, F, M	the set of vertices, edges, flows, and middleboxes
v, f, m	a vertex, a flow, and a middlebox
$e_{vv'}, h_{vf}$	edge from v to v', hop count from f's source to v
$r_f, p_f, M_f, w(f)$	f's traffic rate, path, middlebox set, and cost
c_m, λ_m	m's setup cost, traffic-changing ratio
$b_f^e, w(b_f^e)$	e's bandwidth consumption, cost
x_{vm}, f_{vm}	indicator functions of m on v, f using m on v

3 MODEL AND PROBLEM FORMULATION

In this section, we first propose our network model and then formulate our problem. We also prove the NP-hardness of placing even a single type of middlebox in general topologies. Thus, we narrow the middlebox placement down to a specific kind of topology: tree-structured networks.

3.1 Network Model

We first present our network model as a directed graph, $G = (V, E)$, where $V = \{v\}$ is a set of vertices (i.e., switches) and $E = \{e\}$ is a set of directed edges (i.e., links). We use v to denote a single vertex and $e_{vv'}$ as the edge from vertex v to vertex v'. $M = \{m\}$ is the set of middleboxes. Each middlebox $m \in M$ has a constant setup cost c_m (including the cost of the later server resource occupation) and a pre-defined traffic-changing ratio $\lambda_m \geq 0$ that serves as the ratio of a flow's traffic rate before and after being processed by m. We use an indicator function, x_{vm}, to represent a middlebox m placed on v. We express that a middlebox m' depends on m by using $m \rightarrow m'$, meaning m must process flows before m' [16].

We are given a set of unsplittable flows $F = \{f\}$ because flow splitting may not be feasible for applications that are sensitive to TCP packet ordering (e.g. video applications). Additionally, split flows can be treated as multiple unsplittable flows. We use f to denote a single flow that has an initial traffic rate of r_f, and a required middlebox set of M_f. Its path p_f is an ordered set of edges from the source of f to its destination. All flows' paths are predetermined and valid. We use b_f^e and $w(b_f^e)$ to denote f's traffic rate and cost on e. Then the total bandwidth cost of f is $w(f) = \sum_{e \in p_f} w(b_f^e)$. h_{vf} is the hop count from f's source to a vertex v, measured by the number of edges. The total hop count of f is $|p_f|$. We introduce another indicator function, f_{vm}, to express that the flow f uses the middlebox m placed on the vertex v. In other words, m at the vertex v is effective to f.

We assume each packet in a flow is served by a type of middlebox only once, even if there are several middleboxes of the same type along its path. This is because being served by any middlebox will add an extra transmission delay, which should be avoided as much as possible. The traffic-changing effects of middleboxes are accumulative. We can generate the relationship between f's initial traffic rate r_f and its current rate on e along its path p_f as: $b_f^e = r_f \prod_m \lambda_m, \forall v, e \in p_f, v \prec e$, and $f_{vm} = 1$. The notation $v \prec e$ means v appears before e in p_f. Different weight values corresponding to the importance of various middleboxes can also be attached to the traffic-changing ratios. However, such weight factors are not necessary since we can alternatively scale the cost of setting up new instances c_m for simplicity.

For a better understanding, we illustrate the notations using the example in Fig. 1(b). There are three vertices along the flow's path, denoted (from left to right) as v, v', and v''. The flow f has an initial traffic rate $r_f = 1$. Then, the path is expressed as $p_f = \{e_{vv'}, e_{v'v''}\}$. We have $h_{vf} = 0$ and $|p_f| = h_{v''f} = 2$. There are two types of middleboxes, m and m', with traffic-changing ratios as $\lambda_m = 0.8$ and $\lambda_{m'} = 1.3$, respectively. Their setup costs are $c_m = 0.4$ and $c_{m'} = 0.8$, respectively. Since m' must be served before m, f's middlebox set is simply represented by $m' \rightarrow m$. From the placement plan in the example, we have $x_{v''m} = 1$ and $x_{v''m'} = 1$, and the x values of the rest are all 0. Since f uses both of the middleboxes, $f_{v''m} = 1$ and $f_{v''m'} = 1$. After passing m and m', f's bandwidth on the last edge e is expressed as $b_f^{e_{v'v''}} = r_f \times \lambda_m \times \lambda_{m'}$. f's total bandwidth cost is $w(f) = \sum_{e \in p_f} w(b_f^e) = w(b_f^{e_{vv'}}) + w(b_f^{e_{v'v''}})$.

3.2 Problem Formulation

Based on the above network model, we formulate the middlebox placement as an optimization problem for minimizing the cost of link and server resource usage as follows:

$$\min \sum_{m \in M} \sum_{v \in V} c_m x_{vm} + \sum_{f \in F} w(f) \tag{1}$$

$$\text{s.t. } w(f) = \sum_{e \in p_f} w(b_f^e) \qquad \forall f \in F \tag{2}$$

$$b_f^e = r_f \prod_m \lambda_m \qquad \forall v, e \in p_f, v \prec e, f_{vm} = 1 \tag{3}$$

$$\sum_{v \in p_f} f_{vm} = 1 \qquad \forall f \in F, m \in M_f \tag{4}$$

$$x_{vm} = \{0, 1\}, f_{vm} = \{0, 1\} \qquad \forall v \in V, m \in M \tag{5}$$

Eq. (1) is our objective: minimizing the total costs of middlebox setup and bandwidth consumption. The cost of middlebox setup is the sum of setting up middleboxes. The cost of bandwidth consumption is the sum of each flow's bandwidth cost. A flow's bandwidth cost equals the sum of its bandwidth cost on each link along its path, shown in Eq. (2). Eq. (3) states the relationship between a flow f's initial traffic rate r_f and the bandwidth on e along its path p_f. Eq. (4) requires that each flow $f \in F$ be served by all the required middleboxes in the set M_f once and only once. Eq. (5) shows that x_{vm} and f_{vm} can only be assigned the values 0 and 1.

From Eq. (3), we can see that the effect of each middlebox on each link is multiplicative and cumulative along the flow's path, which is difficult to handle. However, conversion from a non-linear to a linear function can make the problem more tractable. We observe that the effect is log-linear so we apply the translog function as the link bandwidth cost function, i.e. $w(b_f^e) = \log(b_f^e)$. Additionally, the logarithmic function has promising characteristics such as monotonic as an increasing function [7]. Logarithmic cost functions like the ones used by the Cisco EIGRP [3], and OSPF [19] protocols, are common. For the middleboxes, $\log \lambda < 0$, $\forall \lambda \in (0, 1)$ implies that the traffic-diminishing middleboxes decrease the bandwidth consumption cost; $\log\lambda = 0$ if $\lambda = 1$ implies that they do not influence the flow's bandwidth; $\log\lambda > 1$, $\forall\lambda \in (1,\infty)$ implies that the traffic-expanding middleboxes increase the bandwidth consumption cost. The cost of $f \in F$ on the edge $e \in p_f$ can be calculated as follows for all m satisfying:

$$w(b_f^e) = \log(b_f^e) = \log(r_f \prod_{f_{vm}=1} \lambda_m) = \log(r_f) + \sum_{f_{vm}=1} \log(\lambda_m) \quad \forall v \prec e \ (6)$$

Since $\log r_f$ and $\log\lambda_m$ are frequently used, we simplify the notations by replacing with r_f and λ_m. Then, the cost of each flow is calculated as:

$$w(f) = \sum_{e\in p_f} w(b_f^e) = |p_f| r_f + \sum_{f_{vm}=1} (|p_f| - h_{vf})\lambda_m \quad \forall v \prec e \quad (7)$$

Multiplication calculation is changed into summation by the selection of our link bandwidth cost function in Eq. (6), which is a linear function of $\log\lambda_m$. From Eq. (7), we find that the effects of middleboxes on different edges are independent of flows and middleboxes, and that they only relate to the hop count h_{vf} between f's source and the effective middlebox. This simplifies our analysis. Note that we can also add a weight factor to each part in order to show the different importance of each middlebox.

3.3 NP-hardness Proof

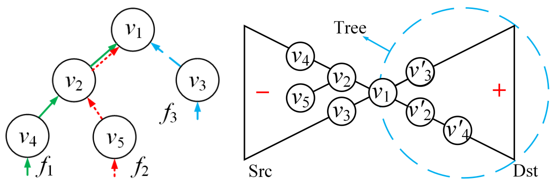
In this subsection, we show that in general network topologies, it is NP-hard to place middleboxes to minimize the cost.

Theorem 3.1. *Middlebox placement for multiple flows is NP-hard in a general topology, even when we place only one type of middleboxes without any traffic-changing effects.*

Proof: We construct a polynomial reduction from the set-cover problem. Assume we have the network (V, E) and a set of flows $F = \{f\}$ that each flow only needs to be processed by one middlebox m, i.e. $M_f = \{m\}, \forall f \in F$. Middleboxes with no traffic-changing effects mean that $\lambda_f = 1$ and $\log\lambda_f = 0, \forall f \in F$. This is the special case of the middlebox placement problem with no bandwidth consumption cost that is shown in Eq. (1). Our objective is reduced to minimizing the cost of setting up middleboxes. Since there is only one type of middlebox, the total setup cost is only related to the number of middleboxes. Our problem is simplified to placing the smallest number of middleboxes m to ensure that each flow passes through at least one m. This problem is equivalent to the set-cover problem. The elements are all the flows $F = \{f\}$. A middlebox m on a vertex v in the network can cover a set of flows whose path passes v, i.e. $S_v = \{f | v \in p_f\}$. We need to find the minimum number of subsets whose union equals the universe set. Since the set cover problem is NP-hard, our placement problem is also NP-hard. ■

3.4 Tree Topologies

Since our problem is NP-hard in a general topology, we narrow it to tree-structured networks, such as the one shown in Fig. 2(a). (The vertices are numbered by Breadth-First Search (BFS).) Tree-structured topologies are extremely common in streaming services, Content Delivery Networks (CDNs) [24], and tree-based tiered topologies like Fat-tree [1] or BCube [11] in data centers. More generally, data centers always use symmetric, hierarchical topologies to balance traffic load [15]. Because of the bi-directional links, the



(a) Complete tree. (b) Hierarchical data center.

Figure 2: Tree-structured topologies in data centers.

topology can be abstracted as two connected, complete trees, as shown in Fig. 2(b). The up and down links separate a hierarchical physical data center topology into two virtual tree-structured networks, whose two parts are separately shown in Figs. 3 (a) and (b). We call this kind of structure a shared-root-double-tree topology. The connection point of the two triangles is the highest level node (core switch), and the two side nodes are the same. Each of the triangles is also a complete tree topology, as shown in Fig. 2(a). The source and destination of each flow are two side nodes.

Here we introduce two classic structure definitions: "fork" and "join" [26]. With flows from the left-most side to the right-most side, Fig. 2(b) can be treated as a precedence graph with the dependence of flows' paths. Flows' transmission process in the left-most complete tree is the procedure of "join" because all the flows will merge at its connection point. For example, all flows passing v_2 or v_3 will meet at the node v_1. After merging at v_1, flows start to separate consistently until they reach their destinations; this process is called a "fork". Since we have already noted that traffic-diminishing middleboxes should be placed near flows' sources, we place them in the left triangle. Similarly, we place traffic-expanding middleboxes in the right triangle. In the physical network view, either traffic-diminishing or traffic-expanding middleboxes can be placed in one node, but traffic-diminishing middleboxes process flows from their sources to the root and traffic-expanding middleboxes process flows from the root to their destinations.

4 PLACEMENT OF A SINGLE MIDDLEBOX WITH HOMOGENEOUS FLOWS

In this section, we study the simple case of placing a single type of middlebox for all flows in a tree-structured topology. We treat all flows with the same source and destination as a single flow with a traffic rate of the sum of their traffic rates. First, we discuss the conditions based on two parts of a shared-root-double-tree topology and middlebox traffic-changing effects. Then we propose optimal solutions of two non-trivial conditions.

4.1 Conditions on Traffic-changing Effects

If the middlebox m is unable to change the traffic rate (i.e. $\lambda = 1$), the bandwidth consumption cost is a constant number $|p_f| r_f, \forall f \in F$, because $\log\lambda_f = 0$. Since all middleboxes of the same type have an identical unit price, our objective is equivalent to minimizing the number of m. In tree topologies, we only need to place middleboxes as closely to the root as possible because the root is the "join" point of all flows. If there are multiple types of such middleboxes, one

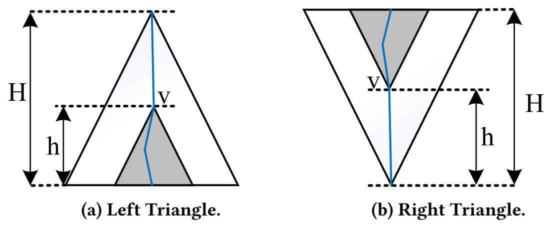


Figure 3: Illustration of variables H and h in two triangles.

optimal solution with the minimum cost is to sort the unit prices of the middleboxes and then sequentially place each type of middlebox in decreasing order by unit price. In the following subsections, we only study the placement of traffic-changing middleboxes.

Based on flow directions and middlebox traffic-changing effects, we classify our problem into four cases: (1) If all the flows move from the root to the leaf nodes (right triangle in Fig. 3(b)), it is trivial to place m when $\lambda_m < 1$. The optimal placement is to place the middleboxes as closely to the root as possible since both bandwidth consumption and middlebox setup cost are the lowest. (2) If all flows move from the leaf nodes to the root (left triangle in Fig. 3(a)) and $\lambda_m > 1$, the optimal placement is the same as case (1). (3) In the subsection 4.2, we look at the placement of middleboxes when all flows move from the leaf nodes to the root and $\lambda_m < 1$. (4) If all flows move from the root to the leaf nodes and $\lambda_m > 1$, the analysis of this scenario is shown in the subsection 4.3.

4.2 Placing a Traffic-diminishing Middlebox

We propose Left Greedy Algorithm (LGA), shown in Alg. 1, to solve the problem of case (1) in a level-by-level manner. The insight of LGA is: placing a middlebox at the root of a subtree can cover all flows passing through it. For each internal node v, we select a better placement with the lower cost between placing one middlebox on v and using the placements under the subtrees of its two children. Since tree topologies are level-structured, we start with the leaf nodes and move in a bottom-up manner to check all levels until the root. We denote the minimum total cost of placing all middleboxes under a node v, shown in the grey area in Fig. 3(a), as LGA(v). In lines 1-2, the cost of placing one middlebox at each leaf node v is $c_m + |p_f|\lambda_m + |p_f|r_f$. In lines 3-4, for each internal node v, suppose $H = |p_f|$ and $h = h_{vf}$, which is illustrated in Fig. 3(a). We have only two choices: (1) place one m at v with the total cost $c_m + 2^{H-h} \times (H \times r_f + (H-h) \times \lambda_m)$ or (2) combine the two placements under the subtrees of its left and right children v' and v'', whose sum of costs is LGA(v') + LGA(v''). The cost of the first choice is generated as: setting up a middlebox m costs c_m; the bandwidth consumption cost is equal to the bandwidth cost of each flow $w(f)$ times the number of flows because all flows are homogeneous with the same traffic rate. We have $w(f) = H \times r_f + (H-h) \times \lambda_m$ from Eq. (7). The number of flows in the subtree of v is equal to the number of leaf nodes, which is 2^{H-h}. The worst time complexity of Alg. 1 is $O(|V|)$ because we go through all nodes ($|V|$ in total) and the cost calculation of each node needs a constant time.

Algorithm 1 Left Greedy Algorithm (LGA)

```
In: Sets of vertices $V$, edges $E$, flows $F$ and middleboxes $m$
Out: The placement plan;
1: for each leaf node $v$ do
2:     Cost of placing one $m$ at node $v$ is LGA($v$) = $c_m + H \times \lambda_m + H \times r_f$
3: for each non-leaf node $v$ with depth $h$ from bottom up do
4:     Select the placement plan with LGA($v$) =min{$c_m + 2^{H-h} \times (H \times r_f + (H-h) \times \lambda_m)$, LGA($v'$) + LGA($v''$)}
5: return The placement plan of the root.
```

A multiple "covered" situation such as Fig. 1(c) happens when traffic is so heavily unbalanced that adding extra private middleboxes to some flows decreases the cost of the bandwidth consumption more than setting a new middlebox. LGA is not optimal for an arbitrary traffic distribution in tree topologies. However, the following theorems state that LGA is optimal in some special cases.

Theorem 4.1. *LGA is optimal in perfect trees for placing a single kind of traffic-diminishing middlebox with homogeneous flows.*

Proof: All levels in the perfect tree are full. If all flows have the same traffic rates and are generated from leaf nodes that are forwarded to the root, the optimal placement plan is obviously symmetric. Specifically, placements of any internal node's two subtrees are identical. Middleboxes should be placed at all the vertices in the same depth. All flows can be served with the minimum cost of the whole-level placement. Therefore, there is no need to place two middleboxes along any flow's path. LGA goes through all the possible plans of the whole-level placement situations and selects the one with the lowest cost, which is optimal. ■

Theorem 4.2. *LGA is optimal in complete tree for placing a single kind of traffic-diminishing middlebox with homogeneous flows.*

Proof: Every level of a complete tree, possibly except the last level, is completely filled. All leaf nodes are as far left as possible. We first prove that each flow passes only one middlebox m in the optimal solution, even in the most unbalanced traffic. The most unbalanced traffic happens to a complete tree topology with homogeneous flows when: the left and right subtrees of the root are perfect binary trees, but one subtree has a depth that is one level deeper than the other. The numbers of flows in two such subtrees are the least equal because the difference between their numbers of leaf nodes is the largest. From the view of the root vertex, we have two choices: (1) If we do not place one m on the root, the placement plan combines the placements of its two subtrees. Theorem 4.1 proves that the optimal placement for a perfect tree places middleboxes at all the vertices with the same depth. It ensures that the twice-covered situation does not exist. (2) If we place one m on the root, we prove the twice-covered impossibility by contradiction. As long as one flow passes two m, all flows in the same subtree should also pass two m on the same level in an optimal solution due to the symmetry of the perfect tree. Then, m on the root is only used by the other subtree. We can move m at the root one level lower in the subtree, which is able to reduce the bandwidth consumption cost without changing the server's resource cost. Thus, placing m on the root is not optimal,

Algorithm 2 Right Greedy Algorithm (RGA)

```
In: Sets of vertices $V$, edges $E$, flows $F$ and middleboxes $m$
Out: The placement plan;
1: Placing one $m$ at root $v$ costs RGA($v$) $= c_m + 2^H(H\lambda_m + Hr_f)$;
2: for each non-root node $v$ with depth $h$ from bottom up do
3:     Select the placement plan with RGA($v$) $= \min\{c_m + 2^h \times (Hr_f + (H-h)\lambda_m)$, RGA($v'$) + RGA($v''$)$\}$;
4: return The placement plan of the root.
```

which contradicts our assumption. Then, there is no need to place two middleboxes along any flow's path. LGA is optimal because it checks all the possible combinations of placement plans and selects the one with the minimum cost. ■

4.3 Placing a Traffic-expanding Middlebox

A traffic-expanding middlebox should be placed in a right triangle, which is illustrated in Fig. 3(b). All flows are from the root to each leaf node. We propose an algorithm, Right Greedy Algorithm (RGA), shown in Alg. 2. We denote the minimum cost of placing all middleboxes above a node v as RGA(v). In line 1, the cost of placing one middlebox at root v is RGA(v) $= c_m + 2^H \times (H \times \lambda_m + H \times r_f)$. In lines 2-3, for each internal node v, suppose $H = |p_f|$ and $h = h_{vf}$, which is illustrated in Fig. 3(b). We have only two choices: (1) place one m on v with the total cost $c_m + 2^h \times (H \times r_f + (H-h) \times \lambda_m)$ or (2) combine the placements of the subtrees of its left and right children v' and v'',, whose sum of costs is RGA(v')) + RGA(v''). The reasons are similar to the last subsection. Setting up a middlebox m costs c_m; the bandwidth consumption cost is equal to the number of flows times the cost of each flow. The cost of each flow is $H \times r_f + (H-h) \times \lambda_m$ from Eq. (7). The number of flows is equal to the number of leaf nodes in the subtree of v, which is 2^h shown in the grey area in Fig. 3(b). We select a better placement with the lower cost each time. The time complexity of Alg. 2 is also $O(|V|)$.

5 PLACEMENT OF A MIDDLEBOX SET WITH HOMOGENEOUS FLOWS

Based on the dependency relations among multiple types of middleboxes, we classify them into three situations: the non-ordered middlebox set, the totally-ordered middlebox set, and the partially dependent middlebox set.

5.1 Non-ordered Middlebox Set Placement

All types of middleboxes are independent in a non-ordered set. For placing a non-ordered middlebox set, we propose an algorithm, called Combined Local Greedy Algorithm (CLGA), which is extended from our LGA and RGA algorithms. CLGA applies LGA for all traffic-diminishing middleboxes in the left triangle and RGA for all traffic-expanding middleboxes in the right triangle, and combines all the placements. We have:

THEOREM 5.1. *CLGA is optimal for placing a non-ordered middlebox set in a complete tree topology.*

Proof: We can place each type of middlebox optimally by applying LGA and RGA. Because of the infinite server capacities, each

Table 2: An illustration of calculating $OPT(i,j)$ values.

$OPT(i,j)$	Placed middlebox number j			
	$\emptyset$	m_1	$m_1 \to m_2$	$m_1 \to m_2 \to m_3$
$i=1$	5.00/5.00	4.12/4.12	5.62/5.62	5.59/5.42
$i=2$	4.00/4.00	4.19/4.19	3.98/3.98	4.89/4.89
$i=3$	1.00/1.00	0.23/0.23	2.09/2.09	0.89/∞
$i=4$	2.00/2.00	1.56/1.56	2.83/2.83	2.63/∞
$i=5$	2.00/2.00	1.56/1.56	2.83/2.83	2.63/∞

type of middlebox can be placed in its optimal location independently. The placement with the lowest cost is the integration of each middlebox's optimal placement. All middleboxes are placed optimally, which indicates CLGA's optimality. ■

5.2 Totally-ordered Middlebox Set Placement

Flows are likely to pass through several middleboxes in a particular order, known as the service chain [13]. A service chain is a totally-ordered middlebox set. We propose a Dynamic Programming (DP) algorithm to achieve the optimal placement plan with a finite server capacity (infinite as a special case). Each vertex is numbered sequentially by BFS and each middlebox is numbered in service chain order. Suppose $n = |V|$. OPT(i,j) denotes the minimum cost of the placement in tree with the root v_i when we have placed the first j middleboxes for all paths from leaf nodes to v_i. If the capacity is not enough to place j middleboxes, the cost is ∞. The optimal substructure gives a recursive formula in the left triangle:

$$\mathrm{OPT}(i,j) = \begin{cases} \min\limits_{0 \le k \le j} \{\mathrm{OPT}(2i,k) + \mathrm{OPT}(2i+1,k) \\ + \sum\limits_{k<l\le j} c_l + \lfloor \log i \rfloor \sum\limits_{k<l\le j} \lambda_l\}, & 1 \le i \le \lfloor \frac{n}{2} \rfloor. \\ \sum\limits_{0 \le l \le j} c_l + \lfloor \log i \rfloor r_f, & \lfloor \frac{n}{2} \rfloor < i \le n. \\ \infty & \text{if not enough node capacity.} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

After placing in the left triangle, we place the remaining middleboxes in the totally-ordered set in the right triangle in the reverse order of the service chain. The nodes are also numbered by BFS. The recursive formula for the placement in the right triangle is shown in Eq. (9). Little difference exists between Eq. (8) and Eq. (9). For simplicity, we only discuss Eq. (8) in the following.

$$\mathrm{OPT}(i,j) = \begin{cases} \min\limits_{0 \le k \le j} \{\mathrm{OPT}(2i,k) + \mathrm{OPT}(2i+1,k) + \sum\limits_{k<l\le j} c_l \\ + (\lfloor \log n \rfloor - \lfloor \log i \rfloor) \sum\limits_{k<l\le j} \lambda_l\} & 1 \le i \le \lfloor \frac{n}{2} \rfloor. \\ \sum\limits_{0 \le l \le j} c_l + \lfloor \log i \rfloor r_f, & \lfloor \frac{n}{2} \rfloor < i \le n. \\ \infty & \text{if not enough node capacity.} \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The insights of the dynamic programming methods are: for each leaf node v_i, as $\lfloor \frac{n}{2} \rfloor \le i \le n$, OPT($i,j$) is simply the cost of placing the first j middleboxes on the vertex v_i. For an internal node v_i, the optimal placement of the tree with the root v_i is selected from all its subtrees' possible placements. If we place the first j middleboxes, we can place the first $k \in [0, j]$ middleboxes in its subtrees and place the following $j - k$ middleboxes on v_i.

To better understand the DP algorithm, we use the topology in Fig. 2(a) to illustrate the placement procedure in the left triangle. There are 5 switches on 3 levels. There are 3 homogeneous flows with the same initial traffic rate $r = 1$, whose sources are the leaf switches and whose destinations are v_1. Suppose we have the same service chain for all flows with 3 middleboxes m_1, m_2, and m_3 for each flow. $\lambda_1 = 0.8, \lambda_2 = 1.5$, and $\lambda_3 = 0.5$. $c_1 = 0.2, c_2 = 0.1$, and $c_3 = 0.8$. The dependency constraint is $m_1 \rightarrow m_2 \rightarrow m_3$. Tab. 2 lists the value of $OPT(i, j)$. The first value in each cell is a case with an infinite server capacity; the second is one with a finite capacity. In the second case, the first level node can hold 1 middlebox and other level nodes are able to hold 2 middleboxes. $OPT(4, 0) = \lfloor \log 4 \rfloor \times r_f = 2$. Similarly, we can calculate $OPT(3, 0)$ and $OPT(5, 0)$. $OPT(2, 0) = OPT(4, 0) + OPT(5, 0)$ and $OPT(1, 0) = OPT(2, 0) + OPT(3, 0)$. Without the constraint of servers' capacity, OPT(2, 1) equals the smaller cost between OPT(4, 0) + OPT(5, 0) + $c_1 + \log \lambda_1$ and OPT(4, 1) + OPT(5, 1) + 0 + 0. The cost of the optimal placement is OPT(1, 3). According to the result of the DP algorithm, the optimal placement places m_1 for each leaf node and places m_2 and m_3 on v_1. In the second case with the servers' capacity constraint, since v_2, v_4, and v_5 can hold at most 2 middleboxes, OPT(2, 1) equals the smaller cost between OPT(4, 0) + OPT(5, 0) + $c_1 + \log \lambda_1$ and OPT(4, 1) + OPT(5, 1) + 0 + 0. However, OPT(3, 3) is ∞ due to inefficient node capacity. The optimal placement of the second case is to place m_1 for each leaf node and m_2 and m_3 on v_1.

Theorem 5.2. *The DP algorithm is optimal for placing the same chain of middleboxes for homogeneous flows in complete topologies with or without server capacity constraint.*

Proof: Theorem 4.2 states that each flow is "covered" only once, when we need to place a single type of middlebox. As a result, if both of the subtrees of v_i have placed the first q middleboxes, we do not need to place these middleboxes again. It breaks the complex problem down into a collection of simpler sub-problems. The detailed proof is omitted due to the optimality of the dynamic programming method. ■

In the infinity server capacity case, the time complexity of the DP is $O(|V||M|^3)$. We separate the chain into two parts to be placed in two triangles, which have $|M|$ possibilities. The dynamic programming table has $|V|$ rows and $|M|$columns. It takes up to $O(|M|)$ time to calculate each table entry. Suppose the largest server capacity is c. In the limited server capacity case, the dynamic programming table at most has $|V|$ rows and c columns, and it takes up to $O(c)$ time to calculate each table entry. If $c \geq |M|$, its time complexity is $O(|V||M|^3)$; Otherwise, its time complexity is $O(c^2|V||M|)$.

5.3 Partially-ordered Middlebox Set Placement

Ma et al. [16] prove that placing a partially dependent middlebox set with the minimum cost is NP-hard even for a single flow. Thus, we propose a heuristic solution to transform a partially-ordered middlebox set into a totally-ordered one. Simply speaking, the transformation treats the middleboxes with dependencies as a single middlebox whose traffic-changing ratio is the product of all their ratios. If there are two middleboxes dependent on the same middlebox, the order of the two is in the non-increasing order of traffic-changing ratios. For multiple dependency relations, we generate a topological order with non-increasing ratios. We sort ratios of the new middlebox set in a non-increasing order leading to a totally-ordered set. We use the dynamic programming method discussed in subsection 5.2 to achieve an efficient placement.

For the transformation part, we provide an example. We have 4 middleboxes m_1, m_2, m_3, and m_4 for each flow. $\lambda_1 = 0.8, \lambda_2 = 1.5, \lambda_3 = 0.5$, and $\lambda_4 = 0.8$. The dependency constraints are $m_1 \rightarrow m_3$ and $m_1 \rightarrow m_4$. Since m_1 is dependent on m_3 and m_4, and $\lambda_3 < \lambda_4$, the order of these three is $m_1 \rightarrow m_3 \rightarrow m_4$. They serve as a new middlebox with a traffic-changing ratio of 0.32, which is less than $\lambda_2 = 1.5$. The final transformation result is $m_1 \rightarrow m_3 \rightarrow m_4 \rightarrow m_2$.

Algorithm 3 Group Flows by Initial Bandwidths (GFIB)

```
In: $V, E, F$ and $M$;
Out: The placement plan;
1: for $i = 1$ to $\lfloor \log_2 \frac{\max r_f}{\min r_f} \rfloor + 1$ do
2:     Find the $i$-th group of flows that satisfy:
       $\{f \in F | 2^{i-1} \times \min r_f \leq r_f < 2^i \times \min r_f\}$.
3:     For flows in the $i$-th group, approximate all of their initial
       traffic rates to be $2^{i-1} \times \min r_f$ (convert to homogeneity).
4:     Call CLGA to place $M$ for all flows in the $i$-th group.
5: return The placement plan in all flow groups.
```

6 MIDDLEBOX PLACEMENT WITH HETEROGENEOUS FLOWS

The last two sections study homogeneous flows. This section explores middlebox placement under the flows' initial traffic rate heterogeneity. We only discuss the placement of a non-ordered middlebox set M in details; its optimal solution, CLGA, is discussed in Section 5.1. For placing a single middlebox, the method is similar except for calling LGA instead of CLGA.

The main idea for handling heterogeneity is to group flows according to their initial traffic rate, i.e., flows with similar initial rates are grouped together. In each flow group, initial rates are approximated to be the same. Then, CLGA algorithm places the middleboxes in this group. The subtle design is the group criterion in line 1 of Alg. 3. $\lfloor . \rfloor$ is the round down operator. Let $\min r_f$ and $\max r_f$ be the minimum and maximum flow initial traffic rate in F, respectively. The i-th flow group consists of flows with initial rate from $2^{i-1} \times \min r_f$ to $2^i \times \min r_f$, i.e., flows are grouped exponentially with respect to their rates in line 2. Note that each flow belongs to exactly one flow group. In line 3, the traffic rates of flows in the i-th group are approximated to their lower bound, $2^{i-1} \times \min r_f$. Flows in the i-th group are approximated to have identical initial rates that can be converted in the flows' homogeneity case. Consequently, CLGA algorithm is called to place the required middlebox in line 4. Finally, the placement plans in all flow groups are returned together in line 5.

Note that the time complexity of Alg. 3 is $\max\{O(|V| \log |V|), O(|V|(\lfloor \log_2 \frac{\max r_f}{\min r_f} \rfloor + 1))\}$. This is because the group mechanism in line 2 takes $O(|V| \log |V|)$ for all groups by sorting all flows'

Table 3: Middlebox settings.

Middlebox types m	m_1	m_2	m_3	m_4
Traffic-changing ratio λ_m	0.7	0.8	1.1	1.2
Setup cost c_m	0.4	0.6	0.2	0.8

initial rates ($O(|V|) = O(|F|)$). We have determined that the time complexity of LGA is $O(|V|)$. Line 3 only needs a constant time for each group. Line 4 costs $O(|V|(\lfloor \log_2 \frac{\max r_f}{\min r_f} \rfloor + 1))$ for all groups. If a group does not include a flow, it is ignored.

Theorem 6.1. *Alg. 3 guarantees an approximation ratio of* $\lfloor \log_2 \frac{\max r_f}{\min r_f} \rfloor + 1$ *to the optimal algorithm.*

Proof: Let GFIB and OPT denote the costs of setting up middleboxes and bandwidth consumption taken by Alg. 3 and the optimal algorithm, respectively. Let GFIB_i denote the cost of placing the middleboxes in the i-th flow group of GFIB. By definition, we have $\text{GFIB} = \sum_i \text{GFIB}_i$. Let OPT_i denote the cost of the optimal algorithm for only flows in the i-th group. Since OPT_i does not ensure that all flows will be served by the required middleboxes, $\text{OPT}_i \leq \text{OPT}$. We claim that $\text{GFIB}_i \leq \text{OPT}_i$. This is because CLGA algorithm is optimal. Since Alg. 3 has at most $\lfloor \log_2 \frac{\max r_f}{\min r_f} \rfloor + 1$ groups, we have:

$$\begin{aligned} \text{GFIB} &= \sum_i \text{GFIB}_i \leq \sum_i \text{OPT}_i \leq \sum_i \text{OPT} \\ &\leq (\lfloor \log_2 \frac{\max r_f}{\min r_f} \rfloor + 1) \times \text{OPT} \qquad (10) \end{aligned}$$

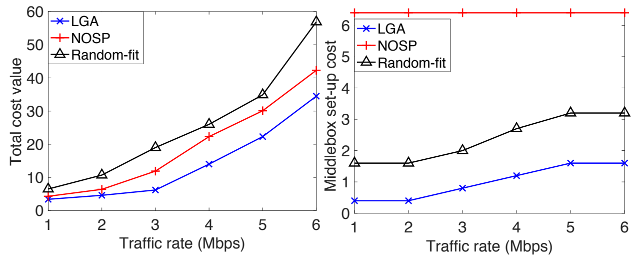
The proof completes. ■

The key insight of Theorem 6.1 is that flows are divided into a limited number of groups. Flows in the same group have similar initial bandwidths, and thus can be resolved by CLGA algorithm . Theorem 6.1 can be further improved by incorporating the flows' initial traffic rate distribution, which enlarges its range of application. For example, if d_i is exponentially distributed, then $\lfloor \log_2 \frac{\max d_i}{\min d_i} \rfloor + 1$ becomes a constant. To better understand Alg. 3, we propose a concrete example. For all given flows $f \in F$, $\min r_f = 1$ and $\max r_f = 64$. Then, we divide all flows into $\lfloor \log_2 \frac{\max r_f}{\min r_f} \rfloor + 1 = \lfloor \log_2 \frac{64}{1} \rfloor + 1 = 7$ groups. If there is one flow f with $r_f = 10$, it belongs to the $\lfloor \log_2 \frac{10}{1} \rfloor + 1 = 4$ group. The approximation ratio of Alg. 3 is 7.

7 EXPERIMENTAL EVALUATION

7.1 Settings

Our experiments are divided into four parts to evaluate the four proposed algorithms: LGA, CLGA, DP, and GFIB. We do simulations in a perfect five-layer binary tree with 31 switches for LGA and GFIB. All flows' sources are leaf nodes and all destinations are the root. The simulations for CLGA and DP are in a shared-root-double-tree topology, as shown in Fig. 2(b). The shared-root-double-tree is symmetric so that each side is a perfect five-layer binary tree with 31 switches in total. We adopt the flow size distribution of Facebook datacenters, which is collected in 10-minute packet traces of three different node types: a Web-server rack, a single cache follower and a Hadoop node [22]. More than 80% flows are less than 6 Mbps. As a result, the traffic rate ranges from 1 to 6 Mbps with a stride of 1 Mbps in this paper. The sources of all flows are the leaf nodes and their destinations are the root. We assume each link is bidirectional and has enough bandwidth to hold all flows, which eliminates congestion and ensures that the routing of all flows is successful. This is because routing failure is not the concern in this paper. The selections of the parameters are based on [10].



(a) Total cost value. (b) Middlebox setup cost.

Figure 4: Single middlebox under bandwidth homogeneity.

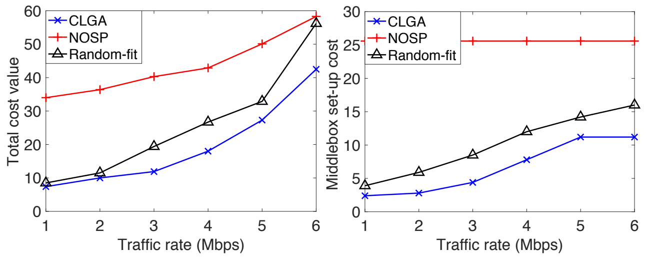
For most cases, the capacities of all servers are infinite since the numbers of middleboxes are relatively small compared to the servers' volumes. When we test the influence of server capacity, the capacity constraint is set to 2 for each server. We use two performance metrics, the cost of middlebox placement and server's utilization, for benchmark comparisons. The cost of middlebox placement is measured as the value of our objective function in Eq. (1). We also evaluate the server's utilization by using the cost of setting up new middleboxes as the metric.

The setting is in accordance with our previous discussion. For homogeneous flows, we change the variability of the initial bandwidth. We test the totally-ordered cases with and without the node capacity constraint. For heterogeneous flows, the initial bandwidth of each flow is generated randomly. Internet Engineering Task Force (IETF) non-exhaustively list 10 middlebox types and show the service chain length is usually small (3 to 5) [12]. As a result, we adopt from [16] a single type of middlebox with a traffic-diminishing ratio of 0.7 and a setup cost of 0.4, and the set of multiple types of middleboxes with a traffic-diminishing ratio of 0.7, 0.8, 1.1, and 1.2 and setup costs 0.4, 0.6, 0.2 and 0.8. For anything but a totally-ordered set, we assume the dependency relation is $0.8 \rightarrow 1.1 \rightarrow 0.7 \rightarrow 1.2$. The middlebox settings are listed in Tab. 3.

7.2 Comparison Algorithms

There are few existing works that study middlebox placement with traffic-changing middleboxes, and we include two benchmark schemes in our simulations:

(1) Ma et al. propose NOSP for the non-ordered case and TOSP for the totally-ordered case in [16]. NOSP sorts the middleboxes based on their traffic-changing ratios; it applies traffic-diminishing middleboxes near the flow's source and traffic-expanding middleboxes near the flow's destination. TOSP is a dynamic programming method.Since NOSP and TOSP are for single flows, we assume they



(a) Total cost value. (b) Middlebox setup cost.

Figure 5: Non-ordered middlebox set.

only handle flows one at a time. For each, we select one flow and run the algorithms until all flows have been selected.

(2) *Random-fit* randomly places middleboxes on random nodes on the paths until all flows are "covered". The middleboxes are sorted by their traffic-changing ratios.

7.3 Evaluation for Homogeneous Flows

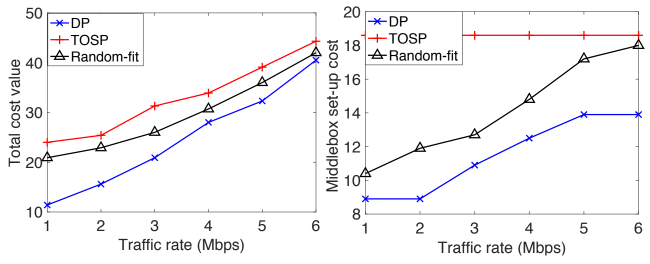
We start with homogeneous flows. The independent variable in x-axis is the unified bandwidth for all flows. First, we evaluate a single type of middlebox placement using LGA in Fig. 4. All curves in Fig. 4(a) are increasing because heavier traffic consumes more bandwidths. LGA achieves the lowest cost value, and on average, it costs about 20.3% less than NOSP and 35.1% less than Random-fit. In the analysis in Section IV, we state that LGA is optimal when placing a single type of middlebox for homogeneous flows in complete tree topologies. In terms of middlebox setup, the cost of NOSP is a constant because it places a required middlebox in the source of each flow. If there are 16 leaf nodes, NOSP needs 16 middleboxes, which costs $16 \times 0.4 = 6.4$. This is the largest number of middleboxes needed to ensure that all flows are "covered". Hence, the middlebox setup cost of NOSP is the largest. The middlebox setup of our algorithm is the smallest because LGA considers not only bandwidth consumption, but also middlebox setup cost.

Second, we evaluate a slightly more complicated case of independent middleboxes using CLGA in Fig. 5. As described in the last subsection, all flows need to be served by all middleboxes in the non-ordered middlebox set $\{0.7, 0.8, 1.1, 1.2\}$. Because of the infinite server capacity and the independence of the middleboxes, we apply LGA to each type of middlebox and the final optimal solution is the combination of the optimal placement of each middlebox. As a result, CLGA performs best in both total cost and middlebox setup cost. The superiority of CLGA is more obvious in Fig. 4 than that of LDA. This is because our algorithms perform better when the setup cost is relatively large compared to NOSP, which only considers the bandwidth consumption. The advantages of LGA and CLGA lie in not only bandwidth consumption, but also in middlebox setup cost. The performance of the Random-fit algorithm is not smooth enough. When the traffic load is heavy, the total cost of Random-fit is 27.0% more than that of CLDA.

Next, we show the placement of a totally-ordered middlebox set using DP with and without the node capacity constraint in Fig. 6 and Fig. 7. In both cases, with the desired optimality property of

Table 4: Different dependencies with $r_f = 3$ Mbps.

Totally-ordered middleboxes	Total cost	Set-up cost
$0.8 \rightarrow 1.1 \rightarrow 0.7 \rightarrow 1.2$	20.9	10.4
$1.1 \rightarrow 0.7 \rightarrow 0.8 \rightarrow 1.2$	23.7	12.0
$0.7 \rightarrow 1.2 \rightarrow 1.1 \rightarrow 0.8$	22.8	9.6
$0.7 \rightarrow 0.8 \rightarrow 1.1 \rightarrow 1.2$	11.9	4.4
$1.2 \rightarrow 1.1 \rightarrow 0.8 \rightarrow 0.7$	24.7	10.2



(a) Total cost value. (b) Middlebox setup cost.

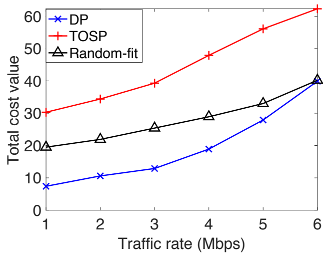
Figure 6: Totally-ordered set without node capacity.

the dynamic programming, our proposed DP has the lowest cost in both metrics. The cost is a little larger than that of the non-ordered middlebox set. This is because the dependency relations make it more difficult to place a single type of middlebox at its optimal location. We also find that when the traffic becomes larger, the Random-fit algorithm performs much worse than it does with the independent set. The dependency relations limit the placement more, so the random placement needs more middleboxes. Random-fit performs a little better in the second case. The limitation of the server capacity eliminates the location possibilities of Random-fit's middleboxes. As a result, the performance difference among these three methods is decreased, especially when the traffic rate is large.

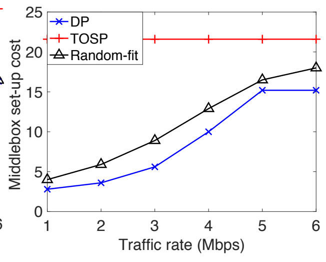
We also test the first case with no node capacity constraint under different dependency relationships, as shown in Table I. The table illustrates that the dependency truly affects both of the metrics. The set with the lowest cost is $0.7 \rightarrow 0.8 \rightarrow 1.1 \rightarrow 1.2$. In the case of the non-ordered middlebox set, the optimal placement sequence is also $0.7 \rightarrow 0.8 \rightarrow 1.1 \rightarrow 1.2$, which verifies the correctness of our DP. The largest total cost belongs to the sequence $1.2 \rightarrow 1.1 \rightarrow 0.8 \rightarrow 0.7$, but its middlebox setup cost is not the largest. This is because our objective is related not only to middlebox setup cost, but also to bandwidth consumption.

7.4 Evaluation for Heterogeneous Flows

We show the case under flow bandwidth heterogeneity with GFIB in Fig. 8. The bandwidths of the flows are generated randomly with an average of 1 to 6 Mbps and a stride of 1 Mbps. Each time, we generate 300 flows. The results show that GFIB consistently achieves the smallest total cost and smallest middlebox setup cost. On average, the total cost is saved about 36.9% and 34.0% compared to the NOSP and Random-fit algorithms, respectively. This is because the dependency relations make it more difficult to place a single type of middlebox at its optimal location. Additionally, the middlebox

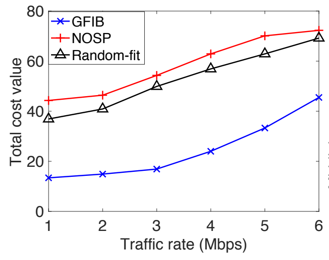


(a) Total cost value.

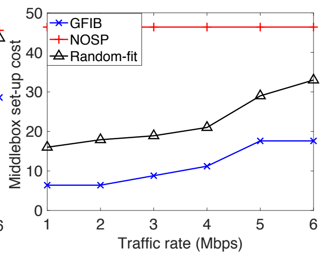


(b) Middlebox setup cost.

Figure 7: Totally-ordered middlebox set with node capacity.



(a) Total cost value.



(b) Middlebox setup cost.

Figure 8: Non-ordered set under bandwidth heterogeneity.

setup cost of GFIB is much lower than that of the other two due to it addressing the middlebox-sharing issue. Though it is not optimal, GFIB is still worth applying to the middlebox placement. It demonstrates the efficiency and effectiveness of our algorithms.

In summary, the experiments verify the correctness and efficiency of our proposed algorithms in the complete tree topologies and in the shared-root-double-tree topology. They also show that only considering bandwidth consumption is too one-sided because sharing middleboxes among flows saves a lot of server resources. Taking both the bandwidth consumption and the server resource usage into consideration, it is worth mentioning that our LGA and CLGA can be used as efficient, greedy algorithms with significant insights in all kinds of tree topologies and traffic distributions. Additionally, our shared-root-double-tree can be embedded in tree-structured data centers using the up-and-down process. The simulation results show that our greedy algorithms and the dynamic programming algorithm empirically perform excellent in trees.

8 CONCLUSION

We study the middlebox placement with the constraints, including traffic-changing effects and dependency relations of middleboxes. Private middleboxes save more flow bandwidth while shared middleboxes cut down the middlebox setup cost. We first formulate the dilemma as a cost minimization problem. We prove it is NP-hard to optimally place even a single middlebox in general topologies and then narrow down to tree-structured networks. With homogeneous flows, we propose three optimal algorithms for several special cases: a single middlebox, a non-ordered middlebox set, and a totally-ordered middlebox set. With heterogeneous flows, we introduce a performance-guaranteed algorithm. Extensive simulations show efficiency and effectiveness of our algorithms.

REFERENCES

[1] Al-Fares, M., Loukissas, A., and Vahdat, A. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev. 38*, 4 (Aug. 2008), 63–74.
[2] Casado, M., Koponen, T., Ramanathan, R., and Shenker, S. Virtualizing the network forwarding plane. In *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow* (New York, NY, USA, 2010), PRESTO '10, ACM, pp. 8:1–8:6.
[3] Cisco. Cisco eigrp protocol.
[4] Cisco. Cisco: Nat order of operation.
[5] Citrix. Citrix cloudbridge product overview, 2015.
[6] Cohen, R., Lewin-Eytan, L., Naor, J. S., and Raz, D. Near optimal placement of virtual network functions. In *2015 IEEE Conference on Computer Communications (INFOCOM)* (April 2015), pp. 1346–1354.
[7] Dixit, S. *IP over WDM: building the next-generation optical Internet.* John Wiley & Sons, 2004.
[8] Fayazbakhsh, S., Sekar, V., Yu, M., and Mogul, J. Flowtags: Enforcing network-wide policies in the presence of dynamic middlebox actions. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking* (New York, NY, USA, 2013), HotSDN '13, ACM, pp. 19–24.
[9] Gember-Jacobson, A., Viswanathan, R., Prakash, C., Grandl, R., Khalid, J., Das, S., and Akella, A. Opennf: Enabling innovation in network function control. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (New York, NY, USA, 2014), SIGCOMM '14, ACM, pp. 163–174.
[10] Guerzoni, R. *Network Functions Virtualisation: An Introduction, Benefits, Enablers, Challenges and Call for Action, Introductory white paper.* 2012 SDN and OpenFlow World Congress, 2012.
[11] Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., and Lu, S. Bcube: A high performance, server-centric network architecture for modular data centers. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication* (New York, NY, USA, 2009), SIGCOMM '09, ACM, pp. 63–74.
[12] Inc, C. S. Service function chaining (sfc) architecture, 2015.
[13] Kuo, T., Liou, B., Lin, K., and Tsai, M. Deploying chains of virtual network functions: On the relation between link and server usage. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications* (April 2016), pp. 1–9.
[14] Li, Y., Phan, L. T. X., and Loo, B. T. Network functions virtualization with soft real-time guarantees. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications* (April 2016), pp. 1–9.
[15] Liu, Y., Muppala, J., Veeraraghavan, M., Lin, D., and Hamdi, M. Data center networks: Topologies, architectures and fault-tolerance characteristics. Springer Science & Business Media.
[16] Ma, W., Sandoval, O., Beltran, J., Pan, D., and Pissinou, N. Traffic aware placement of interdependent nfv middleboxes. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications* (May 2017), pp. 1–9.
[17] Mehraghdam, S., Keller, M., and Karl, H. Specifying and placing chains of virtual network functions. In *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)* (Oct 2014), pp. 7–13.
[18] Miller, M., Vucetic, B., and Berry, L. *Satellite communications: mobile and fixed services.* Springer Science & Business Media, 1993.
[19] Moy, J. Ospf version 2.
[20] P. Zave, P., Ferreira, R., Zou, X., Morimoto, M., and Rexford, J. Dynamic service chaining with dysco. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (New York, NY, USA, 2017), SIGCOMM '17, ACM, pp. 57–70.
[21] Quinn, P., and Nadeau, T. *Service function chaining problem statement.* Active Internet-Draft, IETF Secretariat, Internet-Draft draft-ietf- sfcproblem-statement-05, 2014, 2014.
[22] Roy, A., Zeng, H., Bagga, J., Porter, G., and Snoeren, A. C. Inside the social network's (datacenter) network. In *SIGCOMM 2015.*
[23] Sang, Y., Ji, B., Gupta, G., Du, X., and Ye, L. Provably efficient algorithms for joint placement and allocation of virtual network functions. *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications* (2017), 1–9.
[24] Seyyedi, S., and Akbari, B. Hybrid cdn-p2p architectures for live video streaming: Comparative study of connected and unconnected meshes. In *2011 International Symposium on Computer Networks and Distributed Systems (CNDS)* (Feb 2011), pp. 175–180.
[25] Sherry, J., and Ratnasamy, S. A survey of enterprise middlebox deployments. Tech. Rep. UCB/EECS-2012-24, EECS Department, University of California, Berkeley, Feb 2012.
[26] Wu, J. Distributed system design. CRC press, 1999.