LETTER

Gradual Switch Clustering Based Virtual Middlebox Placement for Improving Service Chain Performance

Duc-Tiep VU†a), *Nonmember and* **Kyungbaek KIM**†b), *Member*

SUMMARY Recently, Network Function Virtualization (NFV) has drawn attentions of many network researchers with great deal of flexibilities, and various network service chains can be used in an SDN/NFV environment. With the flexibility of virtual middlebox placement, how to place virtual middleboxes in order to optimize the performance of service chains becomes essential. Some past studies focused on placement problem of consolidated middleboxes which combine multiple functions into a virtual middlebox. However, when a virtual middlebox providing only a single function is considered, the placement problem becomes much more complex. In this paper, we propose a new heuristic method, the gradual switch clustering based virtual middlebox placement method, in order to improve the performance of service chains, with the constraints of end-to-end delay, bandwidth, and operation cost of deploying a virtual middlebox on a switch. The proposed method gradually finds candidate places for each type of virtual middlebox along with the sequential order of service chains, by clustering candidate switches which satisfy the constraints. Finally, among candidate places for each type of virtual middlebox, the best places are selected in order to minimize the end-to-end delays of service chains. The evaluation results, which are obtained through Mininet based extensive emulations, show that the proposed method outperforms than other methods, and specifically it achieves around 25% less end-to-end delay than other methods.

key words: *middlebox, placement problem, gradual switch clustering, service chain, NFV, SDN*

1. Introduction

In these days, various network services have been developed and some network applications require a set of network services, which is called as a service chain for a network application. A network service is supported by deploying a middlebox which is a separate network appliance that performs a specialized function such as filtering, load balancing, and packet inspection. Earlier middleboxes are deployed with dedicated hardware in networks and the traffic of a service chain should go through middleboxes in a given specific order which requires a complicated routing scheme. As the scale of network and the variety of service chains grow quickly, more efficient way to support service chains is required.

With the popularity of SDN (Software Defined Networking) technique, some past works have focused on supporting efficient service chains with middleboxes by configuring routing paths of each service chain dynamically [1]. Moreover, some works assume that the considered middlebox is a consolidated middlebox which provides multiple service functions within one middlebox in order to improve the scalability of the dynamic routing configuration [2]. However, these works basically assume that a middlebox has the fixed location, and there are limitations of enhancing the performance of service chains.

Recently, the rapid development of NFV (Network Function Virtualization) allows deploying a middlebox function as an application which can be installed at a VM (Virtual Machine) running on a common hardware. A virtual middlebox may support a single service function and multiple service functions (a consolidated middlebox) [3]. A virtual middlebox can be easily moved to a different location on a network by migrating its virtual machine from its original host machine to another host machine. With the flexibility of locating virtual middleboxes, the position of virtual middleboxes can be optimized to improve the performance of service chains, and it is considered as positioning problem of virtual middleboxes.

Some recent works have been studied on how to place middleboxes, especially consolidated middleboxes, for improving the performance of service chains [4]–[6]. However, the consolidated middlebox is a special case of a virtual middlebox, and when we consider the case where a virtual middlebox provides a single function, the placement problem becomes much more complex. Moreover, some of these works did not consider important parameters such as switch memory, virtual machine capacity, and end-to-end delay.

In this paper, we propose a new heuristic method, the gradual switch clustering based virtual middlebox placement method, in order to minimize end-to-end delay of network flows corresponding to service chains while satisfying the constraints of service chains (the corresponding flow must visit a specific type of virtual middlebox in pre-defined order) as well as the constraints of network resources (bandwidth, switch memory and virtual machine capacity). Because the complexity of the placement problem of virtual middlebox with multiple constraints is very high, the proposed heuristic method gradually solves the placement problem step by step. The proposed method finds candidate locations for each type of virtual middleboxes along with the sequential order of service chains by clustering switches which satisfy the constraints on each step. After gathering candidate locations, the best location is selected for each type of virtual middlebox among the corresponding

Manuscript received November 17, 2018.
Manuscript revised March 31, 2019.
Manuscript publicized June 5, 2019.

†The authors are with the Department of Electronics and Computer Engineering, Chonnam National University, Gwangju, Korea.
a) E-mail: ductiep91@gmail.com
b) E-mail: kyungbaekkim@jnu.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2018EDL8240

candidates in order to minimize the end-to-end delay of service chains. Through Mininet based emulation, it is shown that the proposed method outperforms than other methods.

2. Related Works

In [4], the placement of consolidated middleboxes was addressed with delay and bandwidth constraints. This work proposed a greedy solution, which assigning locations of middleboxes based on the degree of usage of switches. Each switch counts how many times it is used for the shortest paths between ingress and egress switches for service chain flows. Then, consolidated middleboxes are assigned to switches in the order of usage of middleboxes and switches. However, this work did not consider some important resource constraints such as switch memory and virtual machine capacity.

In [5], a consolidated middlebox positioning problem was addressed with consideration of end-to-end delay, switch memory and virtual machine capacity. This work proposed a flow clustering method for placing a consolidated middlebox. However, this method is not applicable to the placement problem of virtual middleboxes providing a single function which should consider the order of middleboxes of a service chain. Also, this method lacks the consideration of bandwidth constraints of service chains.

In [7], a virtual machine placement scheme based on mutual bandwidth usage between virtual machines was proposed in order to minimize the product of traffic rate and the number of switches on each flow path. However, this method did not consider end-to-end delay and other resources such as switch memory and server capacity.

3. Gradual Switch Clustering Based Virtual Middlebox Placement

We consider a graph $G = (V_0, E_0)$, where V_0 is the set of nodes and E_0 is the set of edges. A node $v \in V_0$ corresponds to a switch, and an edge $e \in E_0$ corresponds to a link connecting two switches. We assume that the bandwidth of a link (b_e) is given and the delay of a link (d_e) is also given. A node v can hold routing rules, and the number of rules of a node v is denoted as r_v and the maximum number of rules of a node v is R_v. Also, we assume that every switch supports NFV which deploys any virtual middleboxes.

A network service chain is defined as a flow f and the set of flows is denoted as $F = \{f_1, f_2, \ldots, f_m\}$. A flow f is described as $f = \{v_{src_f}, v_{dest_f}, m^{t_1}_{1_f}, m^{t_2}_{2_f}, \ldots, m^{t_l}_{l_f}, BW_f\}$, where v_{src_f} is the source switch and v_{dest_f} is the destination switch. $m^{t_i}_{i_f}$ means the ith virtual middlebox, and deployed switches of virtual middleboxes will be decided by the proposed placement algorithm described later. l is the number of virtual middleboxes required by this service chain and t_i is the type of the ith virtual middlebox. We assume that there are n different types of virtual middleboxes, and the number of available virtual middleboxes for type t can be limited as q_t. BW_f is the required bandwidth of the flow f, and it represents how much bandwidth the flow is expected to occupy per unit of time (Mbps).

The end-to-end delay of a flow f (d_f) is determined by the sum of the delay between the source switch and the first virtual middlebox, the delay between the last virtual middlebox and the destination switch, and the delays of sequential pairs of virtual middleboxes in the corresponding network service chain. Accordingly, it is denoted as $d_f = d(v_{src_f}, m^{t_1}_{1_f}) + d(m^{t_l}_{l_f}, v_{dest_f}) + \sum_{i=1}^{l-1} d(m^{t_i}_{i_f}, m^{t_{i+1}}_{(i+1)_f})$.

The bandwidth usage of a link e is calculated as the summation of the bandwidth allocated to all of the flows which travel through the link e. Accordingly, it is denoted as $\sum_{f \in F_e} BW_f$, where F_e is the subset of F whose member flows travel over a given link e.

Each virtual middlebox may have different processing power which is how much network bandwidth it can deal with per unit of time, and the processing power of a type t virtual middlebox is denoted as $O_{m^t_j}$, where $1 \le j \le q_t$.

With the given G, F and constraints of virtual middleboxes (n, q_t), our placement problem is finding suitable switches for the required virtual middleboxes of flows, so that the end-to-end delay of each flow is minimized and resource constraints (bandwidth of a link, processing power of a virtual middlebox and number of rules of a switch) are satisfied. The problem formalization is as follows:

$$\underset{\forall f \in F}{\text{minimize}} \quad d_f \tag{1}$$

subject to

$$\sum_{f \in F_e} BW_f \le b_e, \forall e \in E_0 \tag{2}$$

$$\sum_{f \in F_t} BW_f \le \sum_{j=1}^{q_t} O_{m^t_j}, 1 \le t \le n \tag{3}$$

$$r_v \le R_v, \forall v \in V_0 \tag{4}$$

The objective (1) represents our primary goal to minimize the end-to-end delay of flows. The constraint (2) is the bandwidth constraint of a link, and the total bandwidth consumption of flows which travel through a link should be lower than the given bandwidth capacity of the link. The constraint (3) is the processing power constraint of a virtual middlebox, and the total processing demand of flows corresponding to type t virtual middleboxes (F_t) should be lower than the maximum processing capacity of the type t virtual middleboxes. The constraint (4) is the switch memory constraint, and a switch should have available memory to store all of the required flow table entries. Accordingly, the formalized problem is a kind of a knapsack problem with multiple constraints, and it is NP-complete [8]. In a brute force manner, the complexity of evaluating constraints of every switch selection takes around $O({}_{|V_0|}C_{\sum_{t=1}^{n} q_t} \times |F|)$.

To solve the problem, we propose a heuristic method which selects proper switches for virtual middleboxes in a gradual manner by using a switch clustering algorithm. Details of the proposed method is shown in Algorithm 1 and

Algorithm 1 *Virtual Middlebox (VMB) Placement with Gradual Switch Clustering*

```
Input: $G = \{V_0, E_0\}, F, l, n, q_t$ where $1 \le t \le n$
Output: $P = \{P_1, P_2, \dots, P_n\}$ where $P_t = \{p_1, p_2, \dots, p_{q_t}\}$, $p_i \in V_0$
  for $t \leftarrow 1$ to $n$ do                    ▹ $P_t$ : final mapping of switches to type $t$ VMB
    $P_t \leftarrow \emptyset, T_t \leftarrow \emptyset$                  ▹ $T_t$ : candidates of switch selection for $P_t$
  for $i \leftarrow 1$ to $l$ do                     ▹ Selecting candidate switches gradually
    for $t \leftarrow 1$ to $n$ do                            ▹ Handling each type of VMB
      $F_t \leftarrow \emptyset, S_t \leftarrow \emptyset$
      for each $f \in F$ do                  ▹ Sorting out flows for type $t$ VMB ($F_t$)
        if typeof($m^{t_i}_{i_f}$) == $t$ then
          add $f$ to $F_t$
      for each $f \in F_t$ do                           ▹ Preparing source switches ($S_t$)
        if $i == 1$ then             ▹ Adding previous switch for the first VMB
          add $v_{src_f}$ to $S_t$
        else               ▹ Adding candidate previous switches for other VMB
          $x \leftarrow$ typeof($m^{t_{(p-1)}}_{(p-1)_f}$)             ▹ Checking type of previous VMB
          add $T_x$ to $S_t$             ▹ Adding corresponding candidate switches
      $C_t$ = SwitchClustering($S_t, G, q_t$)        ▹ Partitioning $q_t$ switch clusters
      for each $C_{t_j} \in C_t$ do                  ▹ $C_t = \{C_{t_1}, C_{t_2}, \dots, C_{t_j}\}, 1 \le j \le q_t$
        $A_{t_j} \leftarrow \emptyset, F_{t_j} \leftarrow \emptyset$
        for each $v \in C_{t_j}$ do  ▹ Preparing adjacent switches of clusters ($A_{t_j}$)
          $V_{adj_v}$=GetAdjacentSwitches($v$, $F_t$, $G$)
          add $V_{adj_v}$ to $A_{t_j}$                    ▹ $V_{adj_v}$ : Adjacent switches of $v$
          $F_v$=GetCorrespondingFlows($v$, $F_t$, $T$)       ▹ $T = \{T_1, T_2, \dots, T_n\}$
          add $F_v$ to $F_{t_j}$                         ▹ $F_v$ : Flows corresponding to $v$
        for each $v \in A_{t_j}$ do            ▹ Calculating flow delays for candidates
          $d^{tot}_v \leftarrow 0$
          for each $f \in F_{t_j}$ do
            $d^{tot}_v \leftarrow d^{tot}_v + d(v_{src_f}, v) + d(v, v_{dest_f})$
        Sort $A_{t_j}$ in increasing order with $d^{tot}_v$                     ▹ Objective 1
        for each $v \in A_{t_j}$ do                ▹ Choosing candidate switches ($T_t$)
          if $r_v \ge R_v$                                           ▹ Constraint 4
            or $\sum_{f \in F_{t_j}} BW_f \ge O_{m^t}$                              ▹ Constraint 3
            or $\exists e \in E$ such that $\sum_{f^e \in F_{t_j}} BW_f \ge b_e$ then      ▹ Constraint 2
            continue
          else
            add $v$ to $T_t$                    ▹ adding $v$ as candidate switches ($T_t$)
  for $t \leftarrow 1$ to $n$ do    ▹ Final selection of $q_t$ switches for type $t$ middlebox
    $C_t$ = SwitchClustering($T_t, G, q_t$)         ▹ Clustering candidate switches
    for each $C_{t_j} \in C_t$ do
      $v \leftarrow$ GetCentroid($C_{t_j}$)            ▹ Finding the centroid of each cluster
      add $v$ to $P_t$                ▹ Centroid of each cluster as final selection
```

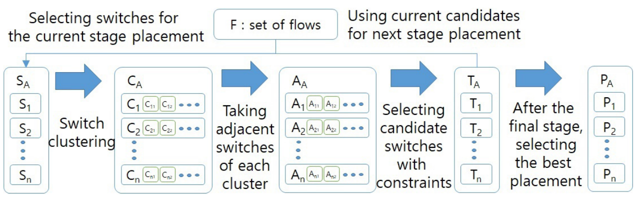

Fig. 1 Overall procedure of the proposed algorithm.

Algorithm 2 *SwitchClustering(S, G, k)*

```
Input: $S, G = \{V_0, E_0\}, k$                        ▹ $S = \{s_1, s_2, \dots, s_n\}, s_i \in V_0$
Output: $C = \{C_1, C_2, \dots, C_k\}$           ▹ $C_i = \{c_1, c_2, \dots, c_m\}, m \le n, c_i \in S$
  Preparing the initial centroids $Z = \{z_1, z_2, \dots, z_k\}$ with $k$ random switches
  Initializing $C = \{C_1, C_2, \dots, C_k\}$                      ▹ $z_i$ is the centroid of $C_i$
  repeat
    for each $s_i \in S$ do                                          ▹ Pre-Clustering
      $d_{min} \leftarrow \infty, q \leftarrow 0$
      for each $z_j \in Z$ do
        Calculating $d(s_i, z_j)$      ▹ Delay between a switch and the centroid
        if $d(s_i, z_j) < d_{min}$ then      ▹ Finding a cluster with minimum delay
          $d_{min} \leftarrow d(s_i, z_j), q \leftarrow j$
      Add $s_i$ to $C_q$                  ▹ Assigning a switch to the nearest cluster
    for each $C_i \in C$ do              ▹ Updating centroids with pairwise delay
      $d_{min} \leftarrow \infty, q \leftarrow 0$
      for each $c_j \in C_i$ do                          ▹ Calculating pairwise delay
        Calculating $d^{tot}_{c_j} \sum d(c_j, c_x), \forall c_x \in C_i, c_x \ne c_j$
        if $d^{tot}_{c_j} < d_{min}$ then          ▹ Finding a switch with minimum delay
          $d_{min} \leftarrow d^{tot}_{c_j}, q \leftarrow j$
      $z_i \leftarrow c_q$                                                  ▹ $c_q \in C_i$
  until No changes in $Z$
```

the overall procedure of the algorithm is depicted in Fig. 1. The proposed algorithm selects candidate switches of virtual middleboxes (T_t) along with the sequence of network service chains, and the selected candidates are used for the next stage of placement. That is, at first, the proposed method selects candidate switches for the first required virtual middleboxes of flows ($m^{t_1}_{1_f}$), then this procedure continues until finding candidate switches for the last required virtual middleboxes of flows ($m^{t_l}_{l_f}$).

The procedure of selecting candidates is composed of gathering source switches for the current target middleboxes (S_t), clustering source switches (C_t), finding adjacent switches of each switch cluster (A_t) and selecting candidate switches with the given constraints (T_t). When gathering the source switches for the second or other virtual middleboxes, the previously selected candidate switches (T_t) are used as source switches for the further procedure. The objectives of switch clustering (Algorithm 2) are 1) selecting the given number of switches for the specific type of virtual middleboxes (q_t) and 2) minimizing the total end-to-end delay of flows corresponding to candidate switches for the given virtual middleboxes.

After finding all the candidate switches for every required virtual middleboxes of flows, the final selection of switches for each t type of virtual middleboxes (P_t) are obtained by using switch clustering algorithm. Consequently, with this gradual heuristic method, the evaluation of candidate switches is conducted along with the sequence of service chain (l) and different types of virtual middlebox (n, q_t), and the complexity of finding proper switches can be relaxed down to $O(l \times n \times q_t \times |F|)$.

4. Evaluation

In order to evaluate the proposed method, an SDN/NFV testbed with Opendaylight SDN controller and Mininet is implemented. On this testbed, the locations of virtual middlebox on a given network topology are deployed by using various placement methods (the random placement, the most used switch placement [4] and our proposed method). Then, the network traffic of service chains are emulated with Iperf and the performance parameters such as end-to-end delay per flow, bandwidth consumption and the number of rules per a switch are measured.

Table 1 Parameters setting in the performance evaluation.

Parameter	Value/Range
FatTree-4 Link delay (ms)	From 1 to 30
The number of type of virtual middlebox	5
The number of middleboxes in a service chain	3
The number of flows	20, 30, 40, 50
The bandwidth demand of each flow (Mbps)	From 0.1 to 5
The processing capacity of each middlebox (Mbps)	100
Maximum bandwidth of each port in a switch (Mbps)	100
Maximum number of rules in each switch	30

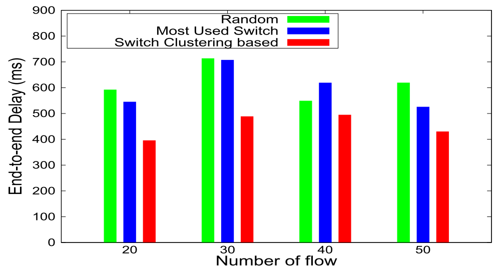

Fig. 2 Average end-to-end delay per service chain flow.

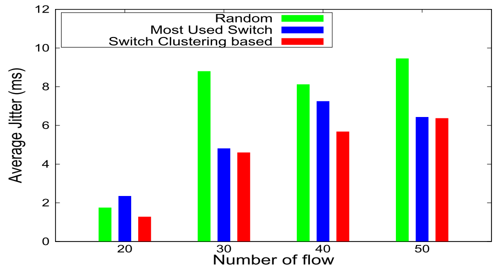

Fig. 3 Average jitter per service chain flow.

For a network topology, the well-known FatTree network topology is used [4]. Particularly, the FatTree-4 with 20 switches and 32 links is used. Through the evaluation, we assume that there are various kinds of service chains and various number of flows corresponding to a service chain. Details of parameter settings are summarized in Table 1.

The evaluation results are illustrated in Figs. 2, 3, 4, and 5. In the results, the proposed method achieves the best performance. Especially, though the average number of rules per switch of the most used switch method is similar to our proposed method (Fig. 5), our proposed method achieves around 25% lower end-to-end delay (Fig. 2) and around 20% lower bandwidth consumption (Fig. 4) than the most used switch method.

5. Conclusion

In this paper, a new heuristic method with gradual switch clustering is proposed for solving placement problem of virtual middleboxes in order to improve service chain performance. Through extensive evaluation, it is demonstrated that the proposed method outperforms the random method

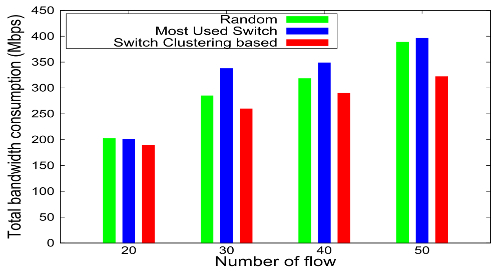

Fig. 4 Total bandwidth consumption.

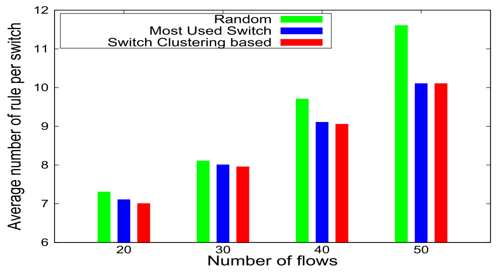

Fig. 5 Average number of rules per switch.

and the most used switch method. The proposed heuristic method can be used for real-time provisioning service chains.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2017R1A2B4012559).

References

[1] Z. Cao, M. Kodialam, and T.V. Lakshman, “Traffic steering in software defined networks: Planning and online routing,” Proc. ACM DCC, pp.65–70, 2014.

[2] A. Gushchin, A. Walid, and A. Tang, “Scalable routing in SDN-enabled networks with consolidated middleboxes,” Proc. ACM SIGCOMM HotMiddlebox, pp.55–60, 2015.

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

[4] J. Liu, Y. Li, Y. Zhang, L. Su, and D. Jin, “Improve service chaining performance with optimized middlebox placement,” IEEE Trans. Services Comput., vol.10, no.4, pp.560–573, 2017, doi: 10.1109/TSC.2015.2502252.

[5] D.T. Vu and K. Kim, “Flow clustering based efficient consolidated middlebox positioning approach for SDN/NFV-enabled network,” IEICE Trans. Inf. & Syst., vol.E99-D, no.8, pp.2177–2181, 2016.

[6] M. Huang, W. Liang, Z. Xu, and S. Guo, “Efficient algorithms for throughput maximization in software-defined networks with consolidated middleboxes,” IEEE Trans. Network and Service Management, vol.14, no.3, pp.631–645, Sept. 2017.

[7] X. Meng, V. Pappas, and L. Zhang, “Improving the scalability of data center networks with traffic-aware virtual machine placement,” Proc. IEEE INFOCOM, pp.1–9, 2010.

[8] H. Kellerer, U. Pferschy, and D. Pisinger, “Introduction to NP-completeness of knapsack problems,” Knapsack Problems, pp.483–493. Springer, Berlin, Heidelberg, 2004.