2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics

Joint Consolidated Middleboxes and Virtual Switch Deployment and Routing for Load Balancing in Software-Defined Network

Juncheng Ge[1] , Gongming Zhao[2] , Xuwei Yang[2]
[1] *School of Software Engineering,*
[2] *School of Computer Science and Technology*
University of Science and Technology of China
Hefei, China
e-mail: sa517087@mail.ustc.edu.cn

***Abstract*—Network Function Virtualization (NFV) is a rapidly developing network architecture concept, in which virtualization technologies transform network functions in hardware devices to software middleboxes. For reasons such as improving security and performance, flows in the network need to traverse specific service function chains (SFCs). Due to the rapid increases in traffic volume, traffic variety, and service requirements, it becomes much complex for SFC routing. The existing solutions for joint middlebox placement and routing still consume a lot of forwarding rules and result in overloaded controllers, which decrease the scalability of middlebox networks. To solve these problems, this paper proposes a joint deployment scheme, which uses the consolidated middlebox to simplify the processing of SFC and uses the virtual switch (e.g., OVS) to reduce TCAM usage. Specifically, We formulate the joint optimization of consolidated middleboxes and the virtual switch placement, as well as routing (JPR) problem into an integer linear programming model, and prove its NP-hardness. An efficient algorithm is proposed to solve the JPR problem. Extensive simulation results show that the proposed solution can achieve better load balancing while using fewer forwarding rules than other existing algorithms. For example, our algorithm can reduce the maximum flow entries by 69%, and reduce the maximum link load by 45% compared with the state-of-the-art approaches.**

Keywords-Software-Defined Networking; Network Function Virtualization; Consolidated Middleboxes; Load Balancing; Virtual Switch Deployment

I. INTRODUCTION

Today's enterprise networks rely on various middleboxes, including firewall, Intrusion Detection System (IDS), WAN optimizer, and Deep Packet Inspection (DPI), to improve the performance and security of network services [1]. However, hardware middleboxes are implemented based on dedicated devices, which are expensive and complicated. To solve this problem, Network Function Virtualization (NFV) is proposed, which replaces hardware middleboxes with virtual network functions implemented by generic servers, switches, and storage [2]. Therefore, NFV can make network services more flexible and scalable while reducing operational efficiency (OPEX) and capital expenditure (CAPEX) [3].

In middlebox networks, flows need to go through several middleboxes in a specific order to meet its processing requirements, also called Service Function Chaining (SFC) [4]. For instance, a flow might go through firewall, IDS, and proxy server in order to filter insecure services and detect suspicious requests. Due to middleboxes in the SFC can be implemented by software running on different Physical Machine (PM) nodes, with the rapid increase of traffic volume, traffic variety, and service requirements, the routing problem in the middlebox networks is more complicated than that of in the traditional networks.

As a result of the Software-Defined Network (SDN) can provide centralized control and flow-level fine-grained scheduling [5], it has become an emerging technology for solving complex SFC routing. However, this also faces the following two critical challenges. 1) **Insufficient TCAM Resources On Data Plane**. Since fine-grained scheduling provided by SDN is based on the implementation of flow entries on SDN switches, flow entries are generally stored in energy-consuming and expensive Ternary Content-Addressable Memory (TCAM) [6]. Due to the high energy consumption and high cost of TCAM, the mainstream SDN switches on the market currently only support the storage of thousands of flow entries [6]. On account of the limited size of the TCAM-based forwarding table and each flow needs to traverse multiple middlebox nodes, thousands of forwarding entries are not enough to match millions of flows in a moderate-sized data center [7]. 2) **Higher Controller Load On Control Plane**. With more new flows arriving at the network, the switch will send more Packet-In messages to the controller, and then the controller computes the route path and sent corresponding entry-installment commands to switches along its route path. It will lead to a higher controller load [8].

In order to solve the above problems, several works have designed efficient solutions [9], [10], [11]. However, these solutions also face some critical disadvantages. They did not consider the forwarding table size constraints, nor did they consider using fewer forwarding rules when routing.

This paper proposes a new scheme to satisfying using fewer forwarding rules consumption and less control overhead for routing in middlebox networks. The main contributions of this paper are as follows:

- We use the consolidated middlebox to simplify the SFC

978-1-7281-6024-5/20/$31.00 ©2020 IEEE

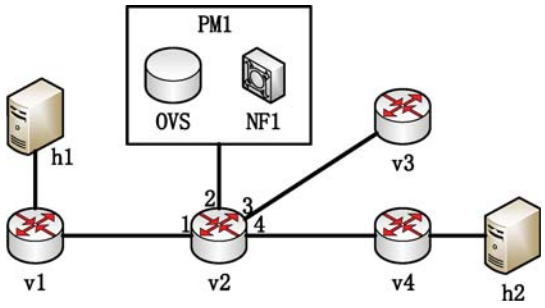

Figure 1. Assume that there are six flows on v2 switch go through NF1, and four flows go directly to the next-hop switch without passing through PM1, then 16 flow entries for v2 are required (6 for $v2-PM1$; 10 for $v2-other\ switches$), exceeding the flow entry constraint. If an OVS is added to the PM1 so that all flows go to OVS, 11 flow entries are required.

in order to solve the problem of complex SFC routing, and then use the virtual switch to reduce the forwarding rules.

- Considering the constraints of network resources and computing resources, we study **J**oint consolidated middleboxes, virtual switch **P**lacement, and **R**outing (JPR) Problem. Given a set of demands, the goal is to minimize the maximum link load. We then propose an effective algorithm that can meet the resource demands of user requests and use fewer flow entries.
- We conduct extensive experiments to verify the efficiency of the proposed algorithm. The results show that our algorithm can reduce the maximum flow entries by 48% and 69%, and reduce the maximum link load by 45% when compared to the baseline algorithms.

The remainder of this paper is organized as follows: In Section II, we introduce the related work and motivation. In Section III, we propose a formal problem definition. In Section IV, we propose the algorithm. We verify the efficiency of the algorithm through simulations in Section V and conclude the paper in Section VI.

II. Background

A. Related Work

To ensure the network security and performance requirements of computer networks, Sekar et al. designed the CoMb [12] architecture. CoMb was a network function consolidation platform, where a flow/request could be processed by all required middleboxes on a single hardware platform. In [13], the authors considered a two-stage local solution for joint optimization of service chain placement and request scheduling. However, they did not consider whether the switch capacity was sufficient, nor did they consider the path selection. Works in [14], [15] mainly focused on middlebox deployment to minimize resource (e.g., bandwidth) consumption, without considering traffic routing. The authors in [9] minimized the distance cost between the client and the network function, as well as the setup costs of these functions, works in [10], [11] considered the joint optimization over network function chain placement and routing. However, these works did not consider resource constraints, such as switch forwarding table size constraints. In [16], the authors studied the problem of VM placement and path selection, intending to make a trade-off between link and server usage. However, they thought that network functions implemented on a single VM could satisfy multiple requests that need to go through different service function chains. The authors in [17] considered the bandwidth and computing resource requirements, and then studied the problem of maximizing SDN network throughput by allowing as many user requests as possible. However, this work is different from ours, because they only used the switch forwarding table size as a constraint and did not reduce the use of flow entries. Besides, when many requests were rejected due to no routing path found, the control plane using reactive mode which would encounter lager response time.

B. Motivation

The flow entry is the core of the packet forwarding on the switch in the software-defined network. Pre-installing a suitable flow entry in the switch facilitates the fast-forwarding of the packet and reduces the load on the controller. Therefore, it is important to lower the flow entry occupation of the switch by reducing the forwarding rules consumption.

The virtual switch (e.g., OVS) is implemented by software, so it has a large number of flow entries and more flexible scheduling capabilities [18]. Because the request needs to go through the middlebox to improve network performance, most switches that consume a lot of TCAM resources are connected to the middlebox. To make the flow entries consume less, we can deploy virtual switch (vSwitch) while deploying the middlebox. However, the deployment of vSwitches consumes computing resources. Therefore, we propose to joint the placement of middleboxes and vSwitches to trade-off computing resources and flow entries consumption.

To better understand this deployment method, an example is presented in Fig. 1. We assume that there are 10 flows traverse switch v2, of which 6 flows go through NF1, and 4 flows go directly to the next-hop switch. A flow entry is required when a flow passes through a switch. At this time, the number of remaining flow entries on switch v2 is 15. Without OVS, we need 16 flow entries (6 for $v2-PM1$; 10 for $v2-other\ switches$), which exceeds the limit on the number of remaining flow entries. If OVS is installed on PM1, only 11 flow entries are required, one wildcard flow entry (i.e., $in\ port=1, output=2$) and 10 flow entries (10 for $v2-other\ switches$).

III. Problem Formulation

A. System Model

SDN network topology can be modeled by $G=(V\cup L,E)$, where V is a group of switches and L is a set of physical machines. For example, l_3 means to deploy a physical machine on switch v_3. In the PM, the specific middleboxes are deployed by VM. E is a set of links connecting switches. Each switch $v\in V$ is equipped with a TCAM forwarding table, which can

accommodate up to $c(v)$ forwarding rules. Each link $e \in E$ has a bandwidth capacity b_e. The controller can be a controller cluster, which helps to balance the processing overhead of each controller. Since we focus on data plane load balancing. For convenience, we assume that there is a logically centralized SDN controller for network G that installs forwarding rules on the switch's forwarding table, assigns middleboxes and vSwitch to the PM, and allocates links bandwidth in order to collect and process user requests.

B. User Requests

User requests can be scheduled by the centralized SDN controller. Each user request $\gamma_i \in \Gamma$ has a specific bandwidth. it needs to traverse a service chain, for example, $\gamma_i =< s_i, t_i, b_i, SC_i >$, where s_i and t_i are the source switch and destination switch, respectively, b_i is its bandwidth demand and SC_i is its service chain. The set of feasible paths for r_i is p_{γ_i}, which will be aggregated at the traffic egress switch along with the traffic ingress switch, passing SC_i. If two requests need to be processed by the same SC_i, the PMs that the feasible path may pass are also consistent.

Virtualization can put the service chain into a consolidated middlebox [12]. Therefore, routing can be simplified, and routing rules for switches in their service chain will be reduced. Following the same assumption as in [19], [20], and [21], we assume that middleboxes in SC_i are run in independent VM and different VMs serving various requests can be consolidated to a single PM. Specifically, when a user request $\gamma_i =< s_i, t_i, b_i, SC_i >$ in the network, where b_i is its bandwidth. γ_i will go from switch s_i to PM_i hosting the VM for its service chain SC_i and then to switch t_i, where the middleboxes in SC_i are deployed in the specified order in VM. Deploying the vSwitch and consolidated middleboxes both consumes the resources of the PM. Each PM $l \in L$ has a finite capability R_l, which can be specified by system operators or measured in real systems. For example, R_l can be measured by CPU resources of PM.

C. Problem Definition

Given a $G = (V \cup L, E)$, the forwarding table capacity $c(v)$ for each switch $v \in V$. A flow entry is required for a request to pass through a switch. the bandwidth capacity b_e for each link $e \in E$ and a set of user requests $\gamma \in \Gamma$. The problem of minimizing the maximum link load in G is to admit as little link load as possible while the bandwidth demand d_γ, the capacity constraints of PM, and resource constraints of G are met. We jointly consider the consolidated middlebox and vSwitch placement and routing. The full notations list is shown in TABLE I. Our objective is to minimize the maximum link load, i.e., $min\ \lambda$.

We formalize the joint placement and routing (JPR) problem in the middlebox network as follows:

$$\min\ \lambda$$

TABLE I
NOTATIONS

Notation	Definition
$G(V \cup L, E)$	network topology
V	set of switches
E	set of links
L	set of Physical Machines
i	a service chain, $i \in I$
l	a physical machine, $l \in L$
Γ	set of requests
γ	a request γ, $\gamma \in \Gamma$
d_γ	the bandwidth of γ
b_e	bandwidth constraint on link e, $e \in E$
p_γ	feasible paths set of request γ
$c(v)$	flow table size of switch v, $v \in V$
R_l	CPU cores of PM l, $l \in L$
$f_{i,l}$	CPU consumed by SC_i deployed on PM l
$g_{o,l}$	CPU consumed by vSwitch deployed on PM l
$x_{i,l}$	whether SC_i is deployed at PM l
$y_{o,l}$	whether vSwitch is deployed at PM l
$y_{\gamma,p}$	whether request γ to select path p
$T_{p,v}$	number of times switch v appears in path p
$\alpha_{\gamma,i}$	whether request γ is processed by SC_i

subject to the following constraints

$$\sum_{i \in I} x_{i,l} \cdot f_{i,l} + y_{o,l} \cdot g_{o,l} \leq R_l, \forall l \in L \quad (1)$$

$$\sum_{p \in p_\gamma} y_{\gamma,p} = 1, \forall \gamma \in \Gamma \quad (2)$$

$$\sum_{\gamma \in \Gamma} \sum_{v \in p : p \in p_\gamma} y_{\gamma,p} \leq c(v), \forall v \in V \quad (3)$$

$$\sum_{\gamma \in \Gamma} \sum_{v \in p : p \in p_\gamma} y_{\gamma,p} \cdot T_{p,v} - y_{o,l} \cdot 2 \times |\Gamma| \leq c(v), \forall v \in V \quad (4)$$

$$\sum_{i \in I} \sum_{l \in L} \alpha_{\gamma,i} \cdot x_{i,l} \geqslant y_{\gamma,p}, \forall \gamma \in \Gamma, \forall p \in p_\gamma \quad (5)$$

$$\sum_{\gamma \in \Gamma} d_\gamma \cdot \sum_{e \in p : p \in p_\gamma} y_{\gamma,p} \leq \lambda \cdot b_e \quad (6)$$

$$x_{i,l} \in \{0, 1\}, \forall i \in I, \forall l \in L \quad (7)$$

$$y_{o,l} \in \{0, 1\}, \forall l \in L \quad (8)$$

$$y_{\gamma,p} \in \{0, 1\}, \forall \gamma \in \Gamma, \forall p \in p_\gamma \quad (9)$$

The first inequality indicates that the number of CPU cores of the PM that SC_i and vSwitch deployed on PM l is less than the number of CPU cores of PM l. The second set of constraints tells that each request only chooses one routing path from the feasible paths. The third set of constraints means that the flow entries deployed by all switches do not exceed the flow table size. The fourth set of inequalities denotes that after PM is deployed on switch v, the flow entries of the switch are limited. Because a request requires a flow entry through a switch, there are at most $2 \times |\Gamma|$ flow entries. The fifth set of inequalities denotes that each request must be processed by the specific SC_i. The sixth inequality measures that the traffic load on each link e. The objective is to achieve the load balancing on the links, that is, $min\ \lambda$.

Theorem 1: The JPR problem is NP-hard.

Proof 1: We assume that all switches have no TCAM capacity limit and all links have no link bandwidth constraints, that is, $c(v) = +\infty$, $b_e = +\infty$, and the controller does not need to consider choosing the optimal path. Therefore, this problem is simplified to minimize the cost of consolidated middlebox placement. Assuming that the number of CPU cores of each PM is equal, placing consolidated middlebox into the PM consumes CPU resources. Then PMs can be regarded as a fixed-capacity bins, and consolidated middleboxes can be regarded as items of different sizes, which is known to be NP-hard [2]. Hence, we prove that it is NP-hard.

IV. Algorithm Design

A. Algorithm for JPR

In the proposed algorithm, in order to minimize the maximum link load, the controller adopts a greedy strategy to deploy consolidated middlebox and vSwitch, as well as routing. Specifically, the controller first chooses three least costly PMs that the feasible path passes, and then calculates the link bandwidth. Moreover, deployment and routing strategies with smaller link load λ_i are selected for each service chain that requests need to traverse. Finally, select a deployment and routing strategy that meets network and computing resource constraints.

The detailed design is shown in Algorithm 1. The whole process can be divided into four steps: (1) Calculate the feasible path and link bandwidth (lines $1-13$). First, calculate the deployment cost based on the cost of the deployment service chain and the length of the request path. Then, select three low-cost PMs as the PMs that the feasible path set may go through. Finally, the link bandwidth is calculated based on the feasible path of each request. (2) Sort the link bandwidth according to the requests traverse different service chains (lines $14-21$). As the request γ_i traverses the feasible path of its service chain SCi, the bandwidth of the feasible path is added to B_i. then, λ_i of each service chain is ordered in non-decreasing degree order. (3) Update the forwarding table capacity (lines $22-28$). For all switches $v \in V$, it will check if its forwarding table capacity is larger than the flow table size and determine whether PM that attached to switch v needs to deploy vSwitch or not. (4) Calculate the number of CPU cores actually used by each PM (lines $29-34$). For all PMs $l \in L$, if the number of remaining CPU cores is less than the number of available cores, the corresponding paths, vSwitch, and the deployed consolidated middleboxes will be removed.

B. Algorithm Complexity Analysis

In algorithm 1, the complexity of the first step is $O(|\Gamma|(|L|+3|E|))$, where $|E|$ is the number of edges, $|L|$ is the number of PMs and $|\Gamma|$ is the number of requests. the complexity of the second step is $O(3|I||\Gamma|)$, where $|I|$ is the number of service chains. The complexity of other steps in algorithm 1 are $|V|$ and $|L|$, where $|V|$ is the number of switches. Therefore, the overall complexity of Algorithm 1 is $O(|\Gamma|(|L|+3(|E|+|I|))+|V|+|L|)$.

Algorithm 1 Algorithm for joint placement and routing

Input: network topology G($V \cup L$, E), R_l, $c(v)$, b_e, $\gamma_i =< s_i, t_i, b_i, SC_i > \in \Gamma$

Output: $x_{i,l}$, $y_{o,l}$ and the selected path of each γ_i

Step1 : Calculate feasible paths and link bandwidth

1: **for** all $\gamma \in \Gamma$ **do**
2: **for** all $l \in L$ **do**
3: $cost_{\gamma,l} \Leftarrow$the cost of request γ traverse PM l
4: Select three PMs with low cost as the feasible path PMs
5: $P_\gamma \Leftarrow$ the set of feasible paths
6: **for** all path in P_γ **do**
7: **for** all $e \in E$ **do**
8: **if** γ traverse e **then**
9: $B_e \Leftarrow B_e + d_\gamma$
10: **if** $\{B_e\} > b_e$ **then**
11: remove path from P_γ, continue
12: $B_{path} \Leftarrow B_{path} \cup \{B_e\}$
13: $P = P \cup P_\gamma, B = B \cup \{B_{path}\}$

Step2 : Sort the link bandwidth according requests traverse different service chains

14: **for** $SC_i \in SCs$ **do**
15: $X_i = \{\}$ //the set of $x_{i,l}$
16: $\Gamma_i \Leftarrow$ the flow set of traverse SC_i
17: **for** all $\gamma \in \Gamma_i$ **do**
18: **for** all path $\in P_\gamma$ **do**
19: $B_i = B_i \cup B_{path}$
20: Order λ_i in non-decreasing degree order and set $X_i \leftarrow x_{i,l}$,$P_i \leftarrow y_{\gamma,p}$
21: $\lambda = \lambda \cup \lambda_i$

Step3 : Update the forwarding table capacity

22: **for** all v $\in$ V **do**
23: $y_{o,l} = 0$
24: **if** the forwarding table capacity of switch v does not satify Eq.(3) **then**
25: remove $x_{i,l}$ from X_i and $y_{\gamma,p}$ from P_i
26: **else if** $y_{o,l} == 0$ is not satified Eq.(4) **then**
27: $y_{o,l} = 1$
28: $Y = y \cup y_{o,l}$

Step4 : Calculate the number of CPU cores actually used by each PM

29: **for** all $l \in L$ **do**
30: **for** $SC_i \in SCs$ **do**
31: **if** the number of CPU cores of PM l is satisfied Eq.(1) **then**
32: $R_l \Leftarrow R_l - (x_{i,l} \times y_{i,l} + y_{o,l} \times y(o,l))$
33: **else**
34: remove $y_{o,l}, x_{i,l}$ and P_i

V. Simulation Results

A. Experimental Environment

To evaluate the proposed scheme, we use Ryu [22] as the central controller and Mininet [23] to emulate the tested topologies. All simulation experiments are conducted on

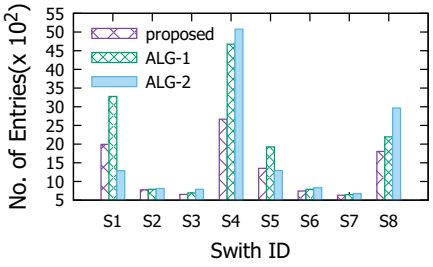

Figure 2. No. of Entries vs. Each Switch on Telstra

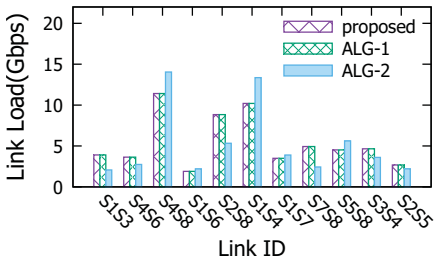

Figure 3. Link Load of Each Link on Telstra

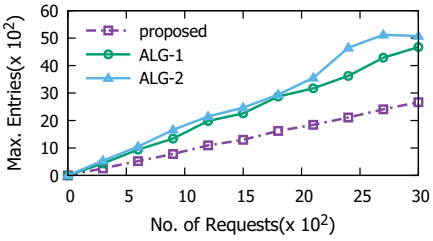

Figure 4. Max. Entries vs. No. of Requests on Telstra

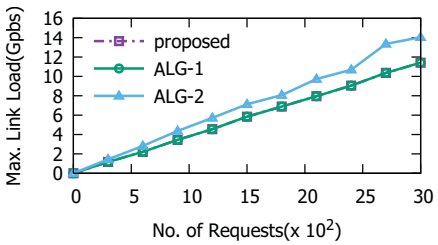

Figure 5. Max. Link Load vs. No. of Requests on Telstra

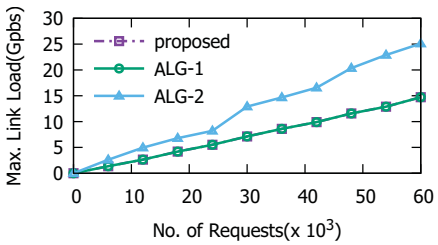

Figure 6. Max. Link Load vs. No. of Requests on Ebone

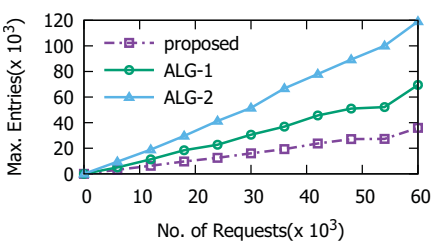

Figure 7. Max. Entries vs. No. of Requests on Ebone

Ubuntu 16.04 in the VMWare workstation. We adopt commonly used, real network topologies including Telstra and Ebone [24] in the simulations, where Ebone is a campus network consisting of 87 switches and 348 servers. We assume the number of CPU cores of PMs is 4. Since these topologies do not provide any PM information, we deploy 4 and 10 PMs respectively for Telstra and Ebone. We consider five types of middleboxes: Firewall, Proxy, NAT, IDS, and Load Balancing. Each service chain SC_i constructs a consolidation middlebox by randomly selecting middleboxes in turn. We assume that the number of consolidated middleboxes of the two topologies is 5 and 20, respectively. We generate different numbers of flows in the network and the average flow intensity is 5Mbps. Furthermore, to simulate the realistic scenario, 20% elephant flows and 80% mice flows are generated in the network.

We compare Algorithm 1 with the other two benchmarks for evaluation. The first benchmark is the ALG-1, which adopts the same deployment strategy and routing method as our proposed algorithm but does not deploy vSwitch. The second benchmark is the most related work, ALG-2 [25], which is an SDN-based network throughput maximization heuristic algorithm. ALG-2 first propose a cost model to construct an auxiliary graph G, and then find the shortest path in a series of auxiliary graphs derived from G.

B. Small-scale Simulations

In this section, we implement our proposed algorithm on a small-scale topology Telstra from the Rocketfuel dataset [24]. We execute each simulation 50 times and average the numerical results.

Fig. 8 shows the deployment of the consolidated middlebox and vSwitch when the number of requests is 3000. The results in Fig. 2 show that our proposed algorithm requires fewer flow

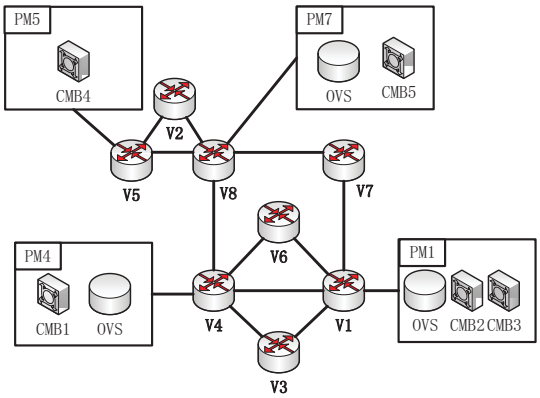

Figure 8. Telstra Topology for small-scale testing. PMs represent physical machines, OVS represents vSwitch and CMBs represent consolidated middleboxes. It contains 8 switches (from V1 to V8), 4 PMs (PM1, PM4, PM5, PM8), 5 CMBs (from CMB_1 to CMB_5) and 3 vSwitches on PM.

entries than ALG-1 and ALG-2. The rule installation speed of a physical switch is about 50ms/rule (e.g., 50.25ms/rule on HP 5130 switches [4]). So our algorithm can achieve lower update delay because of the fewer required entries. Fig. 3 shows the link load for each link in a small topology with 3000 requests. Because the consolidated middleboxes of the proposed algorithm and ALG-1 are placed the same and the routes are the same, the link load of the two algorithms is similar, and the maximum link load is less than ALG-2. Fig. 4 illustrates the algorithm performances in terms of the number of flow entries. It is shown that the proposed algorithm performs much better than the baseline algorithms. In particular, when there are 2400 requests, the number of maximum flow entries of the proposed algorithm can be 42% and 55% less than the ALG-1 and ALG-2

respectively. Fig. 5 presents the link load of the algorithms in the evaluation instances with a different number of requests. We can find that our proposed algorithm is 15% less than ALG-2 when the number of entries is 2400.

C. Large-scale Simulations

We implemented large-scale simulations called Ebone from the Rocketfuel project [24]. This topology includes 87 switches and 348 hosts. We execute each simulation 50 times and average the numerical results.

Fig. 6 illustrate the algorithm performances in terms of the biggest link load. We claim that our proposed algorithm can save bandwidth resources through a well-designed routing strategy. For example, when there are 30×10^3 requests, our proposed algorithm can reduce the maximum link load by about 45% compared with ALG-2. In Fig. 7, with the increasing number of requests, the maximum number of required entries increases for all algorithms. In comparison, our proposed algorithm uses much fewer entries than the other two algorithms. For example, when there are 30×10^3 requests, the maximum flow entries used by our proposed algorithm is 16,016 among all switches, while ALG-1 and ALG-2 use 30,560 and 51,546 entries, respectively. In other words, Our proposed algorithm can reduce the maximum number of required entries by about 48% and 69% compared with ALG-1 and ALG-2, respectively.

VI. Conclusion

In this paper, we use consolidated middlebox to simplify the processing of SFC routing and use vSwitch to reduce the forwarding rules. Then, We have studied how to achieve load balancing through efficient consolidated middlebox and vSwitch deployment and routing in an SDN. We formulate the Joint optimization of the Placement and Routing problem as an integer linear program model. We show that this problem is NP-hard. To tackle this problem, we propose a joint placement and routing algorithm to find a solution whose network resource constraints and computing resource capacity meet the expectation. Extensive simulation results show that the proposed solution can achieve better load balancing and consume less forwarding rules than the baseline algorithm.

References

[1] M. Huang, W. Liang, Z. Xu, and S. Guo, "Efficient algorithms for throughput maximization in software-defined networks with consolidated middleboxes," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 631–645, 2017.

[2] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted nfv service chain deployment based on affiliation-aware vnf placement," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[3] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, 2015.

[4] G. Zhao, H. Xu, J. Liu, C. Qian, J. Ge, and L. Huang, "Safe-me: Scalable and flexible middlebox policy enforcement with software defined networking," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE, 2019, pp. 1–11.

[5] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, 2013, pp. 15–26.

[6] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," in *Proceedings of the ACM SIGCOMM 2011 conference*, 2011, pp. 254–265.

[7] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements & analysis," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, 2009, pp. 202–208.

[8] P. Wang, H. Xu, L. Huang, C. Qian, S. Wang, and Y. Sun, "Minimizing controller response time through flow redirecting in sdns," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 562–575, 2018.

[9] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 1346–1354.

[10] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 98–106.

[11] L. Guo, J. Pang, and A. Walid, "Joint placement and routing of network function chains in data centers," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 612–620.

[12] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture," in *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, 2012, pp. 323–336.

[13] Q. Zhang, Y. Xiao, F. Liu, J. C. Lui, J. Guo, and T. Wang, "Joint optimization of chain placement and request scheduling for network function virtualization," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 731–741.

[14] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying nfv and sdn to lte mobile core gateways, the functions placement problem," in *Proceedings of the 4th workshop on All things cellular: operations, applications, & challenges*, 2014, pp. 33–38.

[15] A. Gupta, M. F. Habib, P. Chowdhury, M. Tornatore, and B. Mukherjee, "Joint virtual network function placement and routing of traffic in operator networks," *UC Davis, Davis, CA, USA, Tech. Rep*, 2015.

[16] T.-W. Kuo, B.-H. Liou, K. C.-J. Lin, and M.-J. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," vol. 26, no. 4. IEEE, 2018, pp. 1562–1576.

[17] M. Huang, W. Liang, Z. Xu, M. Jia, and S. Guo, "Throughput maximization in software-defined networks with consolidated middleboxes," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*. IEEE, 2016, pp. 298–306.

[18] A. Wang, Y. Guo, F. Hao, T. Lakshman, and S. Chen, "Scotch: Elastically scaling up sdn control-plane using vswitch based overlay," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, 2014, pp. 403–414.

[19] A. Gushchin, A. Walid, and A. Tang, "Scalable routing in sdn-enabled networks with consolidated middleboxes," in *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, 2015, pp. 55–60.

[20] M. Huang, W. Liang, Z. Xu, M. Jia, and S. Guo, "Throughput maximization in software-defined networks with consolidated middleboxes," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*. IEEE, 2016, pp. 298–306.

[21] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "Simple-fying middlebox policy enforcement using sdn," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, 2013, pp. 27–38.

[22] "The ryu sdn framework," http://osrg.github.io/ryu/.

[23] "The mininet platform," http://mininet.org/.

[24] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 133–145, 2002.

[25] M. Huang, W. Liang, Z. Xu, and S. Guo, "Efficient algorithms for throughput maximization in software-defined networks with consolidated middleboxes," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 631–645, 2017.