IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 26, 2019, accepted July 17, 2019, date of publication July 22, 2019, date of current version August 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2930295

Online Parallelized Service Function Chain Orchestration in Data Center Networks

GANG SUN[1], ZHENRONG CHEN[1], HONGFANG YU[1], XIAOJIANG DU[2], (Senior Member, IEEE), AND MOHSEN GUIZANI[3], (Fellow, IEEE)

[1]Key Laboratory of Optical Fiber Sensing and Communications (Ministry of Education), University of Electronic Science and Technology of China, Chengdu 611731, China
[2]Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA
[3]Department of Computer Science and Engineering, Qatar University, Doha, Qatar

Corresponding authors: Gang Sun (gangsun@uestc.edu.cn) and Hongfang Yu (yuhf@uestc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61571098, and in part by the 111 Project under Grant B14039.

ABSTRACT In recent years, much attention has been focused on deploying service function chains (SFCs), each of which is composed of a set of virtual network functions (VNFs) in a specified order. This is a promising approach for enabling cloud service providers to deploy user service requests more flexibly while saving costs. However, less effort has been directed toward meeting heterogeneous needs, such as high throughput or low latency of user service requests with heterogeneous bandwidth demands, especially in data center networks (DCNs). In this paper, we propose an efficient orchestration algorithm for online SFC requests. It first splits a large flow into a number of subflows and replicates the same number of sub-SFCs. Each subflow is redirected to one of these ''parallelized'' sub-SFCs, which is termed a sub-user request. Then, each sub-user request is deployed based on a worst-fit strategy, and VNFs in the same SFC are instantiated on the same server to the greatest possible extent. Our algorithm is expected to enable network load balancing, reducing the delay experienced by small flows while improving the acceptance ratio for user requests. Finally, the simulation results show that the proposed algorithm outperforms other comparable algorithms.

INDEX TERMS Service function chain, splitting, data center network, orchestration, load balancing.

I. INTRODUCTION

With the development of 5G, the increasing number of users using smart devices are expected to grow dramatically in the next few years. This in turn will require to access to the network and diverse user requirements that will need to be met by the cloud-based infrastructures [1]. However, because traditional network functions (e.g., firewalls, network address translation, and proxies) are tightly coupled with the physical infrastructure, it is difficult and costly for network service providers to deploy new services. Furthermore, specially trained workers are required to deploy and maintain these hardware-based network functions [2]–[4]. The past few years have witnessed increasing efforts in both academia and industry related to network function virtualization (NFV), a promising paradigm in which network functions are decoupled from dedicated hardware-based appliances. By running virtual network functions (VNFs) in virtual machines (VMs) or containers that are instantiated on commodity (e.g., x86-based) servers across a network, NFV promises to enable the network to deploy VNFs in a more flexible and cost-efficient manner. In this way, capital and operational expenditures (CAPEX/OPEX) can be reduced to meet network service providers' needs [5]–[9]. In particular, the emergence of software-defined networking (SDN), which allows the separation of the control plane from the forwarding plane, enables a network to flexibly route traffic when changes occur via SDN controllers that are able to collect global network information [10]–[14]. Furthermore, user traffic should travel through a sequence of VNFs in a particular order, i.e., a service function chain (SFC) [15].

In a data center network (DCN), traffic flows from user service requests vary from small, short-lived flows, called mouse flows, to large, long-lasting flows, called elephant flows [16]–[19]. Latency-sensitive mouse flows generated

The associate editor coordinating the review of this manuscript and approving it for publication was Kashif Saleem.

by searches, web browsing, *etc*., account for most of the total number of flows, while throughput-sensitive elephant flows generated by video streaming, VM migration and so on carry the majority of bytes of the overall traffic [16]. When these two types of flows are guided onto the same paths, the mouse flows suffer from long queueing delays and poor flow completion times (FCTs) because they are often queued behind elephant flows in switch egress ports [20], [21]. As a result, meeting the different demands of these heterogeneous flows is a nonnegligible challenge when solving the problem of SFC deployment in a DCN.

A. MOTIVATION

To the best of our knowledge, no previous study has considered the issue mentioned above; instead, the traffic rates of user requests are assumed to be fixed [22] or to vary in small ranges [14], or mouse flows are simply ignored [23], [24]. In addition, few related studies have focused on the differences in the performance of the proposed algorithms for mouse flows and elephant flows. For example, in paper [25], the ORBIT algorithm is proposed, in which paths are selected based on equal cost multipath (ECMP) routing, and in paper [26], a hash-based multipath routing and BHT algorithm is introduced that selects VNF instances based on the hash values of the flows. These algorithms may be suboptimal, and their load balancing performance may degrade, when hash collisions occur among elephant flows. In other words, when two elephant flows have the same hash value, they may share the same link or instance, which will cause the shared link or instance to be heavily loaded, even though the others' loads are light. Although the HATS-Flowcell algorithm, presented in paper [27], was designed to solve this problem of hash collision, this algorithm incurs problems involving packet reordering.

B. RESEARCH CONTRIBUTIONS

In this paper, we deploy the SFCs of dynamically arriving user requests by considering "parallelized" VNF chaining [28]. Large flows are split into multiple subflows, and the original SFCs are replicated into a number of sub-SFCs equal to the number of subflows; then, the subflows are passed through the replicated sub-SFCs, such that an original user request may be split into multiple sub-user requests, as shown in Fig. 1. In this way, the problem of hash collision of elephant flows can be alleviated, and a better experience can be provided for mouse flows. However, it is still a challenge to deploy sub-user requests in a DCN while achieving network load balancing and reducing the deployment cost while also satisfying the network resource constraints, which mainly include link bandwidth and node computing resource constraints.

After generating the sub-user requests, we deploy each sub-user request using a proposed heuristic algorithm called ONP_SFO, which is mainly based on a worst-fit strategy for selecting links, to realize network load balancing while deploying all VNFs of the same sub-SFC on the same server

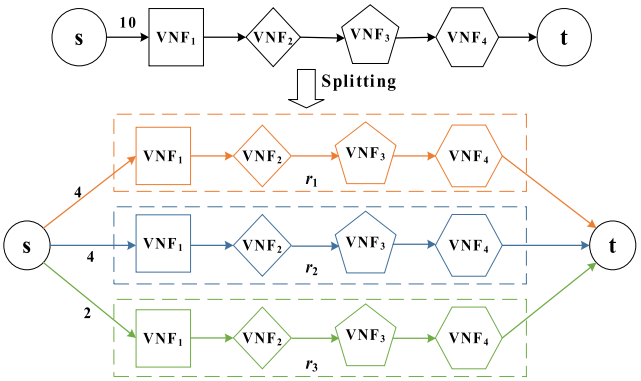


FIGURE 1. Example of splitting an SFC.

to the greatest possible extent to minimize the link mapping cost. Furthermore, in our algorithm, a basic operation called the *downOperation* is designed to search for the optimal VNF deployment from upper layers of the network topology to the server layer. When the available computing resources of the current server cannot satisfy the computing resource demand of the VNF to be deployed, another basic operation, the *upOperation*, is called to find an appropriate switch to allow the downOperation to find another server that not only meets the resource requirements but also is as close as possible to the current server (i.e., the number of hops between them is minimal). The former requirement can be guaranteed by defining a variable that is maintained by each physical node and records the nodes. Finally, a user request is accepted only if all sub-user requests of that user request can be successfully deployed. Otherwise, the request is blocked. The main contributions of this paper are as follows:

- We study the problem of SFC orchestration in a DCN, considering the splitting of large flows, to meet the different needs of heterogeneous flows in the network. Furthermore, we model this problem using an integer linear programming ILP) model based on queueing theory.
- We propose an online heuristic algorithm called ONP_SFO, which is able to improve load balancing while saving mapping cost and deployment time by adopting a worst-fit strategy.
- By means of two proposed basic operation algorithms, we effectively deploy SFC requests in the DCN and minimize the number of hops between VNF instances of the same sub-user request.
- We evaluate our algorithms by varying the network load, the flow splitting specifications and the proportion of elephant flow requests; then, we analyze the simulation results and identify the advantages and disadvantages of our algorithms.

C. STRUCTURE OF THIS PAPER

The remainder of this paper is organized as follows. Section II summarizes and reviews the related work. Section III describes the problem of parallelized SFC orchestration in

a DCN. Section IV describes the details of our splitting method and our designed online heuristic algorithm for deploying sub-user requests. Section V presents the performance evaluation of our solution. Finally, Section VI concludes this work.

II. RELATED WORK

A. VIRTUAL NETWORK EMBEDDING

Virtual network embedding (VNE) has been extensively studied and shows many similarities to SFC mapping. For example, virtual network (VN) and SFC requests are both composed of virtual nodes and virtual links, and mapping these requests means allocating their virtual nodes or links to elements of the physical network topology. Since the problem of VNE is known to be an NP-hard problem [29]–[31], a heuristic algorithm is necessary to adapt to a large-scale network topology. As described in [32]–[35], the workflow of a typical VNE algorithm is to first apply a node-ranking approach that considers only node capacity and adjacent link bandwidth, then map the virtual nodes onto physical nodes by applying a greedy algorithm, and finally apply a Dijkstra or multicommodity flow algorithm to map the virtual links onto the substrate network [36]. Inspired by Google's *PageRank* algorithm, Cao *et al*. [36] proposed a novel node-ranking approach considering two other network topology attributes, namely, node location and propagation delay, to solve the VNE problem. However, when the shared substrate network is provided by multiple infrastructure providers (InPs), the problem becomes one of VNE across multiple domains, which is more difficult because of the lack of global network topology information. To overcome this problem, in [37], Dietrich *et al*. studied how to utilize limited information disclosure to partition VN requests into several segments based on a traffic matrix and then map each segment onto the substrate network. The authors of [38] proposed a mixed integer linear programming (MILP) model that models the problem of VNE across multiple domains with the objective of minimizing the mapping cost while respecting resource and security constraints. Sun *et al*. [39] developed two efficient heuristic algorithms to address the issue of hybrid VNs based on multiple InPs. However, the above articles focused only on VNE instead of SFC mapping, which is a greater challenge because the traffic needs to pass through the VNFs in a certain desired order.

B. SFC MAPPING IN DCNS

Over the past few years, many studies have addressed the NFV resource allocation problem in a single DCN, with optimization goals ranging from energy savings to end-to-end delay reduction to the improvement of system reliability or availability. In [40], the authors proposed a binary integer programming (BIP) model and a heuristic algorithm for solving the problem of optimal VNF placement in packet/optical data centers. The VNFs of the same SFC were placed in the same pod to the greatest possible extent to minimize the number of expensive O/E/O conversions. Similarly, in order to reduce the number of used PMs, the research in [41] modeled the VNF placement problem in cloud datacenters as an ILP model taking into account the time-varying workloads and the basic resource consumptions. Paper [28] focused on effectively increasing the end-to-end service reliability and reducing the backup of VNFs by considering flow and VNF parallelism in a DCN. In paper [23], the NFV-RT algorithm was proposed to maximize the number of accepted requests given end-to-end delay constraints in a DCN. To improve the availability of the SFCs in a given DCN, paper [42] proposed an approximation algorithm for estimating the number of backups while reducing resource usage considering the heterogeneous failure and repair of physical devices. Reference [24] proposed an SFC deployment algorithm that considers the tradeoff between the path length of the corresponding SFC and the reuse of VMs in different DCN topologies. However, this algorithm assumes that at most one VNF can be instantiated on a VM, which can be shared by multiple demands. The authors in [43] designed an efficient online algorithm to deploy SFCs across different data centers and aimed at achieving multi objectives including the minimization of the operating cost and end-to-end delay.

Due to the importance of network load balancing, especially in DCNs, a few studies have focused on achieving load balancing when SFCs are deployed to avoid network congestion and network performance degradation. Paper [44] presented the NF-LGT algorithm with the objective of achieving network and server load balancing when chaining VNFs in a DCN throughout which the VNFs are randomly located. As the first phase of the algorithm, the nearest first (NF) phase uses a greedy strategy to select the next VNF instances as those with the smallest delay from the current location. In the second phase, called the local-global transformation (LGT) phase, the algorithm tried to find a better solution than that found in the NF phase by replacing the VNF instances selected in the NF phase with other available instances and exchanging the order of the VNFs of the SFCs if allowed.

C. SFC MAPPING BY UTILIZING MULTIPLE PATHS

In addition, the methods presented in [25], [28], [45]–[47] consider the selection of multiple paths to carry the total traffic flow of one SFC and the instantiation of multiple instances of VNFs in the same SFC to distribute the traffic load, which is obviously beneficial for load balancing. Considering the availability of several heterogeneous throughput instances for each VNF provided by service providers, [45] discussed the distribution of multiple instances of VNFs in an SFC to overcome the limitations imposed by the chain's traffic demand and better distribute the traffic flow demand to solve the problem of SFC deployment with the goal of improving resource utilization. In [46], Carpio *et al*. first proposed the concept of "replication" when solving the VNF deployment problem in a mobile core network to achieve network load balancing. In this approach, it is assumed that some VNFs are

replicable, while the rest are nonreplicable, and that a node can hold only one function.

Moreover, the authors of [25] developed the ORBIT algorithm by considering ECMP routing to solve the online network load balancing problem for NFV. However, the ECMP approach may cause the hash collision of elephant flows when the hash values of different elephant flows are not considered. To solve the problem of service chaining in a DCN, in the method presented in [27], VNF instances are selected and paths are constructed at soft edge switches by hashing packet headers instead of triggering the centralized controller. In this way, the number of flow entries in the system is reduced while achieving network and VNF load balancing. Specifically, the HATS-Flowcell algorithm proposed in [27] breaks elephant flows into equally sized bursts of packets called flowcells to avoid the hash collision of elephant flows. The work presented in [28] considers the splitting of flows and the replication of VNFs; in this sense, it exhibits some similarities with our article. However, the goal of that work is to increase the end-to-end service reliability rather than to achieve network load balancing.

III. PROBLEM STATEMENT AND FORMULATION

A. PROBLEM STATEMENT

In this paper, we study the problem of online ''parallelized'' VNF chaining in a data center by splitting each original user request into a set of sub-user requests and deploying each sub-user request on the substrate network, where ''online'' means that the detailed information of user requests' cannot be known in advance until they arrive dynamically. First, the question of how to allocate the sizes of the subflows or determine the number of subflows for each flow to be split poses a challenge. After splitting, multiple physical paths from the source node to the destination of an original user request need to be selected to hold its sub-SFCs, guaranteeing that the VNFs required by each sub-SFC are deployed on the servers of the corresponding path. In addition, the physical links' bandwidth resources and servers' computing resources constraints must be satisfied. Finally, this study aims at realizing network load balancing, reduce the queueing delay experienced by mouse flow requests, and improve the acceptance ratio for elephant flow requests.

B. MODEL DEFINITION

1) SUBSTRATE NETWORK

We consider a fat-tree topology as shown in Fig. 2, which is commonly used in data centers. This topology can be represented by an undirected graph $G=(N,E)$, where $N=\{n_i, 1 \le i \le |N|\}$ denotes the physical node set and $E=\{e_i, 1 \le i \le |E|\}$ is the physical link set. The switches and servers in the network are represented by $N_{sw} \in N$ and $N_h \in N$, respectively. Note that a fat-tree topology can be viewed as a multilayer architecture that is composed of three layers of switches and a layer of servers. We use layer (n_i) to denote the layer in which a physical node n_i is located. When layer $(n_i)=1$, $n_i \in N_h$. For $n_i \in N_{sw}$, when layer $(n_i)=2, 3,$ or 4, n_i corresponds to an edge switch, aggregation switch, or core switch, respectively. In addition, $Adj(n_i)$ is the set of all nodes adjacent to n_i in the network topology.

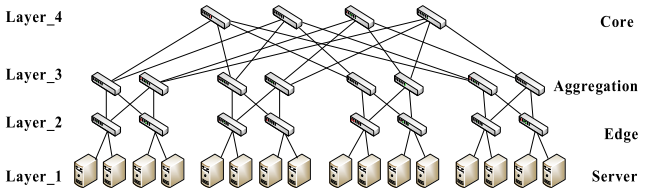


FIGURE 2. A 4-pod fat-tree topology.

We denote the network resource capacity by $RC=(C_N, C_E)$. Here, C_N describes the resource attributes of physical nodes, specifying the computing resource capacity, $c(h_i)$, and the unit node computing resource cost per unit of time, $uCost(h_i)$, for each physical node $h_i \in N_h$. Similarly, the attributes of physical links are described by C_E, including the bandwidth resource capacity, denoted by $c(e_i)$, and the unit link bandwidth resource cost per unit of time, denoted by $uCost(e_i)$, for each physical link $e_i \in E$.

2) SFC REQUESTS

Let $R=\{r_i, 1 \le \mathrm{i} \le |\mathrm{R}|\}$ be the set of arrived user SFC requests. We formulate an SFC request $r_i \in R$ as a four-tuple $r_i =< s_i, t_i, SC_i, b_i >$, layer $(s_i)=$ layer $(t_i)=4$, where s_i and t_i are the source and destination switches, respectively, of the SFC request, both of which are located in the core switch layer, and b_i denotes the bandwidth demand of r_i. Since user requests arrive dynamically, each user request has two additional time-related attributes, service request arrival time denoted by T_i^{arr} and expiration time T_i^{exp}. We consider the traffic flow of r_i to be an elephant flow when $b_i > K_{elep}$, where K_{elep} is a constant; otherwise, the traffic flow is a mouse flow. $R = R_{elep} \cup R_{mouse}$, where R_{elep} and R_{mouse} are the set of *elephant flow requests* and the set of *mouse flow requests*, respectively. Let SC_i represent the original SFC of r_i that is to be deployed. The flow of a user request is split into one or multiple subflows based on the size of b_i. The set of subflows is represented by $b_i^{SUB}=\{b_{i,j}, 1 \le j \le |SUB_i|\}$, where $|SUB_i|$ is the number of subflows. At the same time, the original SFC SC_i is replicated a number of times equal to the number of subflows; let the set of those replicated SFCs be denoted by $Rep_i=\{SC_{i,j}, 1 \le j \le |SUB_i|\}$, where $SC_{i,j} =< F_{i,j}, L_{i,j} >$ is defined as the jth replicated SFC of r_i. $F_{i,j}=\{f_{i,j}^k, 1 \le k \le |F_{i,j}|\}$ denotes the set of required VNFs in the desired order. $L_{i,j}=\{l_{i,j}^k, 1 \le k \le |L_{i,j}|\}$ denotes the virtual link set for $SC_{i,j}$. $f_{i,j}^k$ is the k$^{\mathrm{th}}$ VNF of $SC_{i,j}$, and $l_{i,j}^k$ is the k$^{\mathrm{th}}$ virtual link.

Thus, after splitting, each subflow $b_{i,j}$ is allocated to a corresponding replicated SFC $SC_{i,j}$. Let $r_i^{SUB}=\{r_{i,j}, 1 \le j \le |SUB_i|\}$ denote the *sub-user request* set, where $r_{i,j} =< s_i, t_i, SC_{i,j}, b_{i,j} >$. For each virtual link $l_{i,j}^k$, $Res(l_{i,j}^k)$ is the required amount of link bandwidth resources, and we consider a situation in which the traffic for req_i does not change

after processing by VNFs: $Res\left(l_{i,j}^{k}\right) = b_{i,j}$. For each VNF, it is assumed that the computing resources allocated to its instance are proportional to the traffic capacity to be processed [48]: $Res\left(f_{i,j}^{k}\right) = b_{i,j} \cdot r_{i,j}^{k,h}$, where $r_{i,j}^{k,h}$ denotes the computing resource demand for VNF instance $f_{i,j}^{k}$ to process one unit of traffic rate on server $h \in N_h$. We assume that no VNF instance can be shared by any two sub-SFCs.

3) QUEUEING DELAY MODEL

We assume that the number of user requests arriving is subject to a Poisson distribution, with λ representing the mean arrival rate of user requests. After a user traffic is processed at a physical node $n \in N$, its packets will wait to be transmitted onto a link, incurring a queueing delay; we model the physical nodes in the DCN as M/M/1 queues. Let the size of all packets be denoted by L bits for simplicity. For a sequence of packets, when a subflow of this size passes through physical link $e \in E$, the average queueing delay can be defined as follows.

$$D_e = \frac{L}{c\left(e\right) - ar\left(e\right)} - \frac{L}{c\left(e\right)} \tag{1}$$

where $ar(e)$ represents the current arrival traffic rate (in bits/second) at e and $c\left(e\right)$ represents the maximum transmission rate (in bits/second) or bandwidth of link e.

We denote the set of all paths in the network by $P = \{p^k, 1 \le k \le |P|\}$, $p^k =< N^k, E^k >$, where N^k and E^k are the physical node and link sets, respectively. Let $S_{p^k} = \{s_{p^k}^m, 1 \le m \le |S_{p^k}|\}$ denote the set of server nodes on path p^k. Considering that after the traffic of req_i has been split into subflows b_i^{SUB}, those subflows need to be steered through the required VNF instances, we define a variable $X_{i,j}^{p} \in (0, 1)$, $p \in P$, to indicate whether the subflow $b_i^j \in b_i^{SUB}$ is using the path p that starts from s_i, passes through the physical nodes that are running the VNF instances of $SC_{i,j}$, and finally ends at t_i. This can be expressed as follows:

$$X_{i,j}^{p} = \begin{cases} 1, & if\ sub\ flow\ b_{i,j}\ is\ using\ path\ p, \\ 0, & otherwise. \end{cases} \tag{2}$$

We also define a variable v_p^e, which is set to 1 if path $p \in P$ passes through the physical link $e \in E$; otherwise, $v_p^e = 0$. According to Eq. (1), the queueing delay $D_{i,j}$ of subflow $b_{i,j}$ passing through a physical path p can be calculated as

$$D_{i,j} = \sum_{p \in P} \sum_{e \in E} X_{i,j}^{p} \cdot v_p^e \cdot \mathrm{d}D_e \tag{3}$$

Consequently, for a request r_i, its queueing delay D_i is formulated as

$$D_i = \max_{b_{i,j} \in b_i^{SUB}} D_{i,j} \tag{4}$$

C. ONLINE PARALLELIZED SFC ORCHESTRATION

In this paper, we aim to minimize the total queueing delay D_{total} of all accepted SFC requests.

Objective:

$$\min D_{total} = \min \sum_{r_i \in R} D_i \tag{5}$$

1) RESOURCE CONSTRAINTS

We define a binary variable $N_{i,j}^{k,h}$, which determines whether VNF $f_{i,j}^{k}$ is placed on server $h \in N_h (N_{i,j}^{k,h} = 1)$ or not $(N_{i,j}^{k,h} = 0)$. To account for the limited amounts of server computing resources and physical link bandwidth resources, we impose the following constraints:

$$\sum_{r_i \in R} \sum_{b_{i,j} \in b_i^{SUB}} \sum_{f_{i,j}^{k} \in F_{i,j}} N_{i,j}^{k,h} \cdot Res\left(f_{i,j}^{k}\right) \le c\left(h\right), \ \forall h \in N_h \tag{6}$$

$$\sum_{r_i \in R} \sum_{b_{i,j} \in b_i^{SUB}} \sum_{p \in P} X_{i,j}^{p} \cdot v_p^e \cdot \mathrm{d}b_{i,j} \le c\left(e\right), \ \forall e \in E \tag{7}$$

2) SFC DEPLOYMENT CONSTRAINTS

First, we should ensure that every subflow $b_{i,j}$ of each user request is steered onto one path p and that every VNF of the corresponding replicated SFC $SC_{i,j}$ is placed on only one server.

$$\sum_{p \in P} X_{i,j}^{p} = 1, \quad \forall r_i \in R, \ \forall b_{i,j} \in b_i^{SUB} \tag{8}$$

$$\sum_{h \in N_h} N_{i,j}^{k,h} \le 1, \quad \forall r_i \in R, \ \forall b_{i,j} \in b_i^{SUB}, \forall f_{i,j}^{k} \in F_{i,j} \tag{9}$$

Then, the following Eq. (10) ensures that the servers that host the instances of $f_{i,j}^{k} \in F_{i,j}$ are on the path p, where $X_{i,j}^{p} = 1$.

$$X_{i,j}^{p} \le \sum_{h \in S_p} N_{i,j}^{k,h}, \quad \forall r_i \in R, \ \forall b_{i,j} \in b_i^{SUB}, \ \forall f_{i,j}^{k} \in F_{i,j} \tag{10}$$

Finally, Eq. (11) ensures that every VNF $f_{i,j}^{k}$ is deployed in the desired order. Before VNF $f_{i,j}^{k}$ is assigned to server h_p^m, it is necessary to ensure that the previous VNF $f_{i,j}^{k-1}$ has been deployed on a server that precedes server h_p^m on the path p, where $X_{i,j}^{p} = 1$.

$$\left(\sum_{n=1}^{m} N_{i,j}^{k-1,h_p^n}\right) - N_{i,j}^{k,h_p^m} \ge X_{i,j}^{p} - 1,$$
$$\forall r_i \in R, \quad \forall b_{i,j} \in b_i^{SUB}, \ \forall f_{i,j}^{k} \in F_{i,j}, \ \forall h_p^m \in S_p \tag{11}$$

IV. ALGORITHM DESIGN

According to Refs. [1] and [49], the SFC orchestration problem has been proven to be an NP-hard problem, thus, we cannot obtain the optimal solution of the ILP model mentioned in Sections III within a polynomial time. In this section, a heuristic algorithm for online parallelized SFC orchestration in a single DCN is proposed to solve not only the problem of how to map the VNFs and virtual links to corresponding servers and physical links but also the problem of how to split flows into subflows to realize network load balancing. Our proposed online parallelized SFC orchestration (ONP_SFO) algorithm is introduced as follows.

A. ONP_SFO ALGORITHM

As stated above, it is assumed that the arrivals of the user SFC requests follow a Poisson distribution. All user SFC requests are stored in a queue *ArrivalSFC* in order of their arrival time T^{arr}. Once user requests' expiration time T^{exp} arrive, they are pushed into another queue named *FinishedSFC*. In addition, we use SFC_{blo} to denote the set of SFC requests that are blocked due to network resource limitations.

We define Map_{slt}= $\{DS_i, 1 \le i \le |R|$ as the set of mapping solutions for all user requests R, where $DS_i = \{DS_{i,j}, 1 \le j \le |SUB_i|\}$ is the set of mapping solutions for the subflows b_i^{SUB} of r_i. $DS_{i,j} = (DS_{i,j}^f, DS_{i,j}^l, p_{i,j})$ is the mapping solution for subflow $b_{i,j} \in b_i^{SUB}$, where $DS_{i,j}^f = \{DS\left(f_{i,j}^k\right), 1 \le k \le |F_{i,j}|\}$ represents the set of VNF deployment solutions and $DS_{i,j}^l = \{DS\left(l_{i,j}^k\right), 1 \le k \le |L_{i,j}|\}$ represents the set of virtual link deployment solutions for replicated SFC $SC_{i,j}$. $DS\left(f_{i,j}^k\right)$ is the server that hosts VNF $f_{i,j}^k$, and $DS\left(l_{i,j}^k\right)$ is the physical path to which virtual link $l_{i,j}^k$ is mapped. Let $p_{i,j} = \sum_k DS\left(l_{i,j}^k\right)$ denote the path on which subflow $b_{i,j}$ travels.

For each user request $r_i =< s_i, t_i, SC_i, b_i >$, ONP_SFO calls ***Procedure* 1** to split it into a set r_i^{SUB} of sub-user requests based on the size of b_i. For each $r_{i,j} \in r_i^{SUB}$, $r_{i,j} =< s_i, t_i, SC_{i,j}, b_{i,j} >$, ***Procedure* 2** is called to deploy it on the network and obtain its mapping solution $DS_{i,j}$. If the deployment of any $r_{i,j} \in r_i^{SUB}$ fails in ***Procedure* 2**, user request r_i will be rejected, and thus, its mapping solution will be $DS_i = \emptyset$; otherwise, the mapping solution set Map_{slt} will be updated.

In ***Procedure* 1**, we set a parameter k_{sub} to represent the maximum size of the subflows. For $r_i \in R$, we try to split the flow of r_i into subflows of size k_{sub}. When the bandwidth demand of a user request is no greater than k_{sub}, there is only one subflow, whose bandwidth demand is equal to b_i. Consequently, the number of subflows after splitting can be calculated using the following equation:

$$|SUB_i| = \begin{cases} 1, & b_i \le k_{sub} \\ \left\lceil \dfrac{b_i}{k_{sub}} \right\rceil, & b_i > k_{sub} \end{cases} \tag{12}$$

When $b_i \le k_{sub}$, the size of the single subflow is equal to b_i, and there is no need to replicate SC_i. When $b_i > k_{sub}$, the sizes of all subflows can be computed based on the following equation:

$$b_{i,j} = \begin{cases} k_{sub}, j = 1, 2, \ldots, |SUB_i| - 1 \\ b_i - (|SUB_i| - 1) \cdot k_{sub}, j = |SUB_i| \end{cases} \tag{13}$$

At the same time, the original SFC SC_i is replicated $|SUB_i|$ times. Finally, the user request r_i is split into a set of sub-user requests, denoted by r_i^{SUB}.

B. DEPLOYMENT OF SUB-USER REQUESTS

***Procedure* 2** deploys the sub-user requests $r_{i,j}$, which are generated by splitting the original user request flow b_i and replicating the original SFC SC_i, in the fat-tree DCN. The main idea is to select the links with the most remaining bandwidth resources based on a worst-fit strategy and attempt

Algorithm 1 Online Parallelized SFC Orchestration (ONP_SFO)

Input: (1) The DCN fat-tree topology G= (N, E) and the resource capacity constraints RC = (C_N, C_E);
(2) The user SFC request queue *ArrivedSFC*.
Output: The mapping solution set Map_{slt} and the blocked SFC request set SFC_{blo}.
Initialization: $Map_{slt} = \emptyset$, $SFC_{blo} = \emptyset$;
while *ArrivedSFC* $\neq \emptyset$, **do**
  Release the resources occupied by expired SFC requests according to *FinishedSFC*;
  Call ***Procedure 1*** to split the flow of the first user request $r_1 =< s_1, t_1, SC_1, b_1 >$ in *ArrivedSFC* into subflows b_1^{SUB} and replicate the original SFC SC_1 into replicated SFCs Rep_1, which reduces to the sub-user request set r_1^{SUB};
  for $r_{1,j} \in r_1^{SUB}$, $where r_{1,j} =< s_1, t_1, SC_{1,j}, b_{1,j} >$ **do**
    Call ***Procedure 2*** to generate mapping solution $DS_{i,j}$;
    if $DS_{i,j} \neq \emptyset$, **do**
      $DS_i = DS_i \cup DS_{i,j}$;
    else
      $DS_i = \emptyset$;
    end if
  end for
  if $DS_i \neq \emptyset$, **do**
    $Map_{slt} = Map_{slt} \cup DS_i$;
  else
    $SFC_{blo} = SFC_{blo} \cup \{r_1\}$;
  end if
  $ArrivalSFC$=$ArrivalSFC \setminus \{r_1\}$;
end while
return Map_{slt} and SFC_{blo}.

Algorithm 2 Downoperation Algorithm: Deployment of $f_{i,j}^k$

Input: *tmpLocation* and $f_{i,j}^k$.
Output: A server node n_s, a physical path p_{down}.
Initialization: $n_s = \emptyset$, $p_{down} = \emptyset$;
for $n_{tmpLocation} \notin N_h$, **do**
  Find the node $n_m \in Candidate(n_{tmpLocation})$, ensuring that the link $l_{worstfit}$ consisting of $n_{tmpLocation}$ and n_m meets the bandwidth requirement and has the most remaining bandwidth;
  if n_m exists, **do**
    $tmpLocation = m$, $p_{down} = p_{down} + l_{worstfit}$;
  else
    return $n_s = \emptyset$, $p_{down} = \emptyset$;
$s = tmpLoaction$;
return n_s, p_{down}.

Algorithm 3 Upoperation Algorithm: Find an Appropriate Switch

Input: $tmpLocation$ and $f_{i,j}^{k}$.
Output: A switch node n_{sw}, a physical path p_{up}.
1: Initialization: $n_{sw} = \emptyset$, $p_{up} = \emptyset$;
2: **for** $layer(n_{tmpLocation}) \neq 4$, **do**
3: Find the node $n_m \in Candidate(n_{tmpLocation})$, ensuring that the link $l_{worstfit}$ consisting of $n_{tmpLocation}$ and n_m meets the bandwidth requirement and has the most remaining bandwidth;
4: **if** n_m exists, **do**
5: $tmpLocation = m$, $p_{up} = p_{up} + l_{worstfit}$;
6: $sw = tmpLoaction$;
7: **return**n_{sw}, p_{up};
8: **else**
9: Find the node $n_{m'} \in Adj_{up}\left(n_{tmpLocation}\right)$, ensuring that the link $l_{worstfit'}$ consisting of $n_{tmpLocation}$ and n_m meets the bandwidth requirement and has the most remaining bandwidth;
10: **if** n_m exists, **do**
11: $tmpLocation = m^{'}$, $p_{up} = p_{up} + l_{worstfit'}$;
12: **else**
13: break;
14: **return** $n_{sw} = \emptyset$, $p_{up} = \emptyset$.

Procedure 1 Split an Original User Request

Input: User request $r_1 =< s_1, t_1, SC_1, b_1 >, k_{sub}$.
Output: Sub-user request set r_1^{SUB}.
1: Initialization: the number of subflows $sub\_num = 0$, $r_1^{SUB} = \emptyset$;
2: Compute sub_num according to Eq. (12);
3: **while** $sub\_num > 1$, **do**
4: $r_1^{SUB} = r_1^{SUB} \cup \{< s_1, t_1, SC_1, k_{sub} >\}$;
5: sub_num –;
6: $r_1^{SUB} = r_1^{SUB} \cup \{< s_1, t_1, SC_1, b_i - (|SUB_i| - 1) \cdot k_{sub} >\}$;
7: **return** r_1^{SUB}.

Procedure 2 Deploy a Sub-User Request

Input: (1) The DCN fat-tree topology $G = (N, E)$ and the resource capacity constraints $RC = (C_N, C_E)$;
(2) Sub-user request $r_{i,j} =< s_i, t_i, SC_{i,j}, b_{i,j} >$.
Output: The mapping solution $DS_{i,j} = (DS_{i,j}^{f}, DS_{i,j}^{l}, p_{i,j})$ for $r_{i,j}$.
1: Initialization: $DS_{i,j} = \emptyset$, $tmpServer = 0$ and $tmpLocation = b$ (b is the number of s_i);
2: At the source node s_i, call the *downOperation algorithm* to find a server n_s to host the first VNF $f_{i,j}^{1}$ and the path p_{down} (from s_i to n_s);
3: **if** n_s and p_{down} exist, **do**
4: Update $DS\left(f_{i,j}^{k}\right) = n_s$, $DS\left(l_{i,j}^{1}\right) = p_{down}$, $MaxCap_m$, $tmpServer = tmpLocation = s$;
5: **else**
6: **return** $DS_{i,j} = \emptyset$;
7: **for** each VNF $f_{i,j}^{k}$, $2 \leq k \leq |F_{i,j}|$, **do**
8: **if** server $n_{tempServer}$ has sufficient resources to host $f_{i,j}^{k}$, **do**
9: Update $DS\left(f_{i,j}^{k}\right) = n_s$, $DS\left(l_{i,j}^{k}\right) = \emptyset$, $MaxCap_m$;
10: **else**
11: At the server n_s, call the *upOperation algorithm* to find an appropriate switch n_{sw} on the path p_{up} (from n_s to n_{sw});
12: **if** n_{sw} and p_{up} exist, **do**
13: $tmpLocation = sw$;
14: **else**
15: **return** $DS_{i,j} = \emptyset$;
16: At the switch n_{sw}, call the *downOperation algorithm* to find another server n_{as} to host the VNF $f_{i,j}^{k}$ and the path p_{down} (from n_{sw} to n_{as});
17: **if** n_s and p_{down} exist, **do**
18: Update $DS\left(f_{i,j}^{k}\right) = n_{as}$, $DS\left(l_{i,j}^{k}\right) = p_{up} + p_{down}$, $MaxCap_m$, $tmpServer = tmpLocation = as$;
19: **else**
20 **return** $DS_{i,j} = \emptyset$;
21: **return** $DS_{i,j}$.

to deploy all VNFs of the replicated SFC $SC_{i,j}$ on the same server to reduce the network communication cost while realizing network load balancing.

For each physical node $n_m \in N$ in the network, a variable $MaxCap_m$ is maintained. For each server node $n_m \in N_h$, $MaxCap_m$ is equal to its remaining node computing resources, denoted by $AvailCap_m$. For each switch $n_m \in N_{sw}$, we denote all physical nodes that are adjacent to it and in lower layers of the network topology by $Adj_{low}(n_m) = \{n_j\}$, where $n_j \in Adj(n_m)$ and $layer\left(n_j\right) < layer(n_m)$. Then, when $n_m \in N_{sw}$, $MaxCap_m$ is equal to the maximum $MaxCap_j$ among the $n_j \in Adj_{low}(n_m)$. As a result, for any $n_m \in N$, $MaxCap_m$ can be computed using the following equation:

$$MaxCap_m = \begin{cases} AvailCap_m, n_m \in N_h \\ \max\left\{MaxCap_j, n_j \in Adj_{low}(n_m)\right\}, n_m \in N_{sw} \end{cases} \quad (14)$$

Similarly, $Adj_{high}(n_m) = \{n_j\}$, where $n_j \in Adj(n_m)$ and $layer\left(n_j\right) > layer(n_m)$.

In ***Procedure* 2**, when deploying a user request $r_{i,j}$, the *downOperation algorithm* is first called at the source node of the request to find a server n_s on which to deploy the first VNF, starting from the core layer of the network topology and proceeding towards the server layer. For each subsequent VNF, an attempt is made to deploy it on the same server that is hosting the previous VNF. If that server cannot accommodate the current VNF due to resource constraints, the *upOperation algorithm* is called to find an appropriate switch, starting from the server layer and proceeding to higher layers, and the *downOperation algorithm* is then called again to find another server on which to instantiate the current VNF. Finally, the deployment solution $DS_{i,j}$ for $r_{i,j}$ is returned.

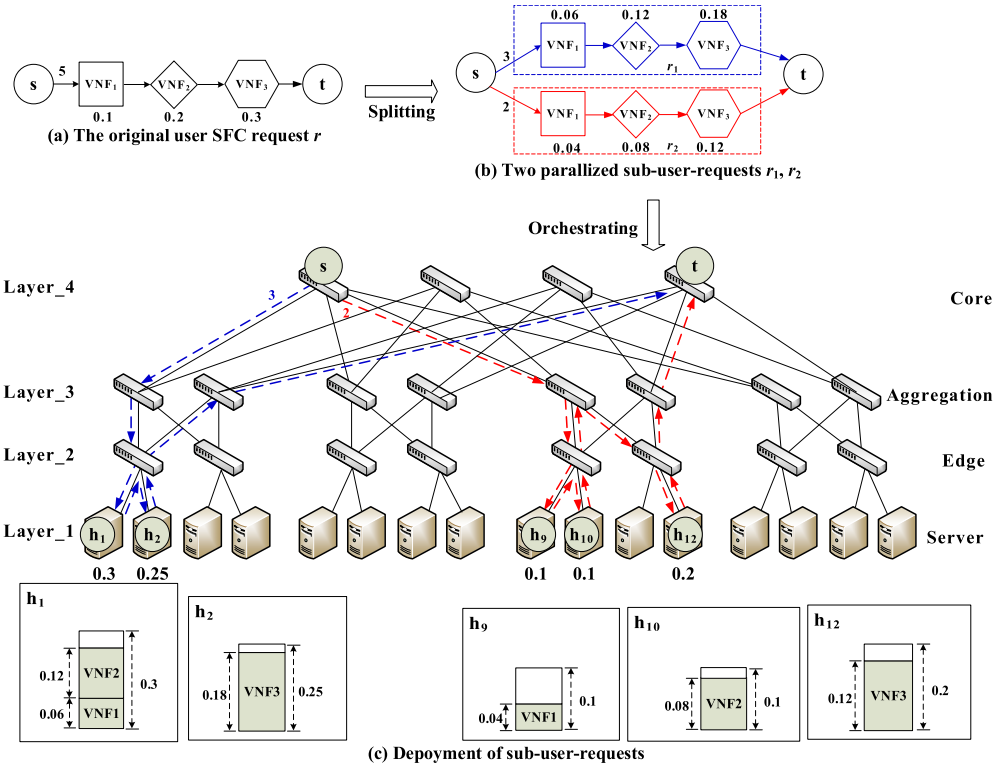


FIGURE 3. An example of the *ONP_SFO* algorithm.

C. TWO BASIC OPERATIONS FOR SFC DEPLOYMENT

As described in ***Procedure* 2,** the *downOperation algorithm* is called when deploying the first VNF or when finding another server because the current server does not have sufficient resources to support the VNF to be deployed. As one of the basic operations of our algorithm, the *downOperation algorithm* selects the physical links with the most remaining bandwidth resources layer by layer, from higher layers of the network topology to the server layer. In addition, to avoid finding a server whose node resources cannot satisfy the demand based on the worst-fit strategy, the physical nodes on the selected links where VNF $f_{i,j}^k$ is being deployed must satisfy $MaxCap \geq Res\left(f_{i,j}^k\right)$. At the same time, the whole operation is subject to the link bandwidth resource constraints. The set of nodes that satisfy $MaxCap \geq Res\left(f_{i,j}^k\right)$ among $Adj_{low}\left(n_{tmpLocation}\right)$ is denoted by $Candidate(n_{tmpLocation})$, where $Adj_{low}\left(n_{tmpLocation}\right)$ is the set of lower layer nodes that are adjacent to $n_{tmpLocation}$.

As the other basic operation, when the server that is hosting the previous VNF does not have sufficient resources to instantiate the next VNF, the *upOperation algorithm* selects the links with the most available bandwidth resources layer by layer, from the service layer to the higher layers, until an appropriate switch is found that satisfies $MaxCap \geq Res\left(f_{i,j}^k\right)$ and is on a selected link. Similarly, constraints on bandwidth resources are satisfied during this operation.

The set of nodes that satisfy $MaxCap \geq Res\left(f_{i,j}^k\right)$ among $Adj_{up}\left(n_{tmpLocation}\right)$ is denoted by $Candidate(n_{tmpLocation})$, where $Adj_{up}\left(n_{tmpLocation}\right)$ is the set of nodes that are adjacent to $n_{tmpLocation}$ and in a higher layer.

D. EXAMPLE OF THE ONP_SFO ALGORITHM

As shown in Fig. 3 (a), consider an original user request r with 5 units of bandwidth demand that is composed of the source core switch s, the destination core switch t and three VNFs with individual computing resource demands of 0.1, 0.2 and 0.3, and suppose that the computing resource capacity of each server is 1. As shown in Fig. 3 (b), when k_{sub}= 3, r is split into two parallelized sub-user requests: r_1, with 3 units of bandwidth demand, and r_2, with 2 units. As assumed in section III, the node resource demand of each VNF is proportional to the traffic capacity to be processed. For example, for VNF1, the computing resource demand of r_1 is 0.06, and that of r_2 is 0.04.

Next, r_1 and r_2 are deployed as depicted by the blue dotted lines and red dotted lines, respectively, in Fig. 3 (c). The number below each server node indicates the available node resources of that server. When r_1 is deployed, first, the *downOperation algorithm* is called to deploy VNF1 on h_1 based on the worst-fit strategy. Then, VNF2 is also deployed on h_1 because the remaining node resources of h_1 are sufficient: $0.3 - 0.06 = 0.24 > 0.12$. However, when VNF3 is deployed, it is necessary to find another server to host it.

By calling the *upOperation algorithm,* an appropriate switch sw_9 is found, the *MaxCap* value of which satifies $MaxCap = \max(0.12, 0.25) > 0.18$. Then, the *downOperation algorithm* is called again, and VNF3 is deployed on h_2. Similarly, for sub-user request r_2, VNF1 is deployed on h_9, VNF2 is deployed on h_{10}, and VNF3 is deployed on h_{12}, as shown by the red dotted lines.

E. ALGORITHM COMPLEXITY ANALYSIS

This section presents a brief complexity analysis of the ONP_SFO algorithm. Suppose that the number of user requests is R, the number of VNFs for each SFC is N, and the physical network topology is a k-ary fat-tree topology. Because the time complexity of finding the link with the most remaining bandwidth is $O(klogk)$, the time complexities of the *downOperation algorithm* and the *upOperation algorithm* are both $O(klogk)$. Consequently, the complexity of deploying a sub-user request of length N is $O(Nklogk)$. Finally, under the assumption that the proportion of elephant flow requests among the total user requests and the number of sub-user requests corresponding to each elephant flow request are constant values, the time complexity of ONP_SFO is $O(|R|Nklogk)$.

V. SIMULATION RESULTS AND ANALYSIS

A. EXPERIMENTAL ENVIRONMENT

1) PHYSICAL NETWORK

We conduct the simulation experiments in a datacenter which is attached to the US-wide NSF network topology as shown in Figure. 4. In the datacenter, a 4-ary fat-tree topology [50] like Figure. 2 is used, which contains 36 nodes (including 16 servers and 20 switches) and 48 links. The bandwidth capacity of every link is set to 1000 units, and the computing capacity of each server is set to 3600 units. Each unit of bandwidth resources or computing resources costs 1 US dollar/second.

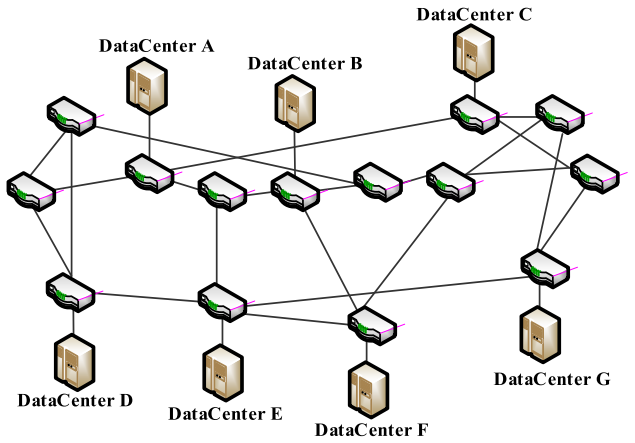


FIGURE 4. The US-wide NSF network.

2) USER SFC REQUESTS

The source and destination nodes of each request are randomly selected from among the core switches in the fat-tree network topology. The bandwidth demand is set between 2 kbps and 1 Mbps for mouse flow requests and between 1 Mbps and 50 Mbps for elephant flow requests, in accordance with the reports in [18], [19], [51] on the characteristics of flows in DCNs. In addition, the durations of the mouse flows vary from 0 to 1 seconds, while elephant flows last from 10 to 25 seconds. The number of requests arriving is subject to a Poisson distribution with a mean value λ (requests/second) that can be adjusted to adjust the load on the network. In general, the length of an SFC is uniformly randomly chosen from the range of [1], [5]. In this paper, the computing resource demand for any VNF instance at any server node to process one unit of traffic rate is $r_{i,j}^{k,h} = 1$.

3) PARAMETERS

By adjusting the parameter k_{sub}, which specifies the maximum size of the subflows, we can observe the effects of the number and size of the subflows on the performance of our ONP_SFO algorithm. We report the results for multiple versions of the algorithm, with k_{sub} set to 50, 40, 30, 20, 10, and 5 Mbps. In particular, when $k_{sub} = 50$ Mbps, no flows are split; this version of the algorithm is referred to as the ON_SFO algorithm, whereas the others are called ONP_SFO algorithms. We generate 20000 user SFC requests in every experiment, among which elephant flows are generated with a probability denoted by $p_{elephant}$; unless stated otherwise, $p_{elephant} = 0.2$. In addition, the size of each packet, L, is set to 1500 bytes [16].

4) EVALUATION METHOD

The performance of our designed algorithms is compared with that of the algorithm proposed in [44], called NF-LGT, in which the VNF instances are randomly allocated to servers and whose goal is to achieve load balance for the whole network. The run time of each algorithm is obtained based on a machine with a 3.7 GHz Intel Core CPU and 8 GB of RAM.

B. PERFORMANCE METRICS

1) ACCEPTANCE RATIO

Due to network resource constraints, with a heavier network load, more user SFC requests may be blocked. The acceptance ratio for all requests, $AcptRt_{total}$, can be computed using the following equation:

$$AcptRt_{total} = \frac{|ArrivedReq_{total}| - |BlockedReq_{total}|}{|ArrivedReq_{total}|} \quad (15)$$

where $|ArrivedReq_{total}|$ represents the total number of arrived requests and $|BlockedReq_{total}|$ represents the total number of blocked requests. Similarly, the acceptance ratio for elephant flow requests, $AcptRt_{elep}$, is calculated using the following formula:

$$AcptRt_{elep} = \frac{|ArrivedReq_{elep}| - |BlockedReq_{elep}|}{|ArrivedReq_{elep}|} \quad (16)$$

where $|ArrivedReq_{elep}|$ represents the number of arrived elephant flow requests and $|BlockedReq_{elep}|$ represents the number of blocked elephant flow requests.

2) AVERAGE MAPPING COST

In this paper, the mapping cost of a request includes the VNF mapping cost incurred due to computing resource consumption and the virtual link mapping cost incurred due to bandwidth resource consumption. The average mapping cost, $Cost_{total}^{average}$, represents the cost per unit of traffic rate, including the average VNF mapping cost, $Cost_{node}^{average}$, and the average virtual link mapping cost, $Cost_{link}^{average}$.

$$\begin{aligned} Cost_{total}^{average} &= Cost_{node}^{average} + Cost_{link}^{average} \\ &= \frac{1}{\sum_i b_i} \cdot \sum_{r_i \in R} Cost_{node}^{i} + \frac{1}{\sum_i b_i} \cdot \sum_{r_i \in R} Cost_{link}^{i} \end{aligned} \tag{17}$$

where $Cost_{node}^{i}$ and $Cost_{link}^{i}$ are the VNF mapping cost and virtual link mapping cost, respectively of request r_i.

$$Cost_{node}^{i} = \sum_{b_{i,j} \in b_i^{SUB}} \sum_{f_{i,j}^k \in F_{i,j}} Res\left(f_{i,j}^k\right) \cdot uCost\left(DS\left(f_{i,j}^k\right)\right) \cdot T_i \tag{18}$$

$$Cost_{link}^{i} = \sum_{b_{i,j} \in b_i^{SUB}} \sum_{p \in P} X_{i,j}^{p} \cdot \sum_{l \in p} b_{i,j} \cdot uCost\,(l) \cdot T_i \tag{19}$$

where $DS\left(f_{i,j}^k\right)$ is the server that hosts VNF $f_{i,j}^k$, $T_i = T_i^{exp} - T_i^{arr}$, denotes the duration time of r_i.

3) QUEUEING DELAY

This is an important performance metric in our paper, especially for mouse flow requests, which are sensitive to an "end-to-end" delay that also includes the propagation delay, transmission delay and processing delay, although these delays are fixed or negligible in our experimental environment. The average queueing delay for mouse flow requests, $D_{mouse}^{average}$, is defined as

$$D_{mouse}^{average} = \frac{1}{|ArrivedReq_{mouse}|} \cdot \sum_{r_i \in R_{mouse}} D_i \tag{20}$$

where $|ArrivedReq_{mouse}|$ represents the number of arrived mouse flow requests. The queueing delay D_i experienced by request r_i is calculated in accordance with Eq. (4).

4) LINK UTILIZATION

One of our goals is to achieve network load balancing. Thus, it is important to observe the load rate on the physical links. The average link utilization, $U_{link}^{average}$, is calculated as

$$U_{link}^{average} = \frac{1}{|E|} \cdot \sum_{e_i \in E} u\,(e_i) \tag{21}$$

where $u(e_i)$ represents the utilization of physical link e_i and is computed via Eq. (22):

$$u\,(e_i) = \frac{1}{c\,(e_i)} \cdot \sum_{r_i \in R} \sum_{b_{i,j} \in b_i^{SUB}} \sum_{p \in P} X_{i,j}^{p} \cdot v_p^{e} \cdot b_{i,j} \tag{22}$$

C. SIMULATION RESULTS

1) THE INFLUENCE OF THE NETWORK LOAD

First, we evaluate the performance of our ON_SFO and ONP_SFO algorithms against that of the NF-LGT algorithm with respect to the network load. From Fig. 5 and Fig. 6, we can see that the acceptance ratios for all requests and elephant flow requests both decrease as the network load increases due to the limited amount of physical resources available. Compared to the NF-LGT algorithm, all of our algorithms yield higher values of both acceptance ratios. This occurs because in the NF-LGT algorithm, consecutive VNF instances are randomly allocated to different servers in the network topology, leading to more bandwidth consumption when an SFC request is deployed, whereas in our algorithms, an attempt is made to deploy all VNFs of any sub-user request on the same server or on servers between which the hops are minimal. This phenomenon is more obvious for the deployment of elephant flow requests, whose bandwidth demands are large.

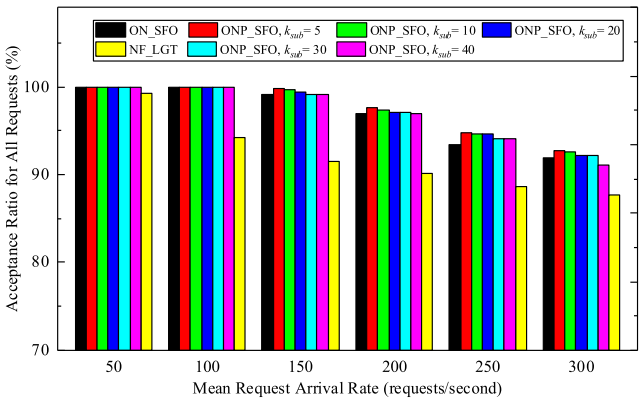


FIGURE 5. The acceptance ratio for all requests.

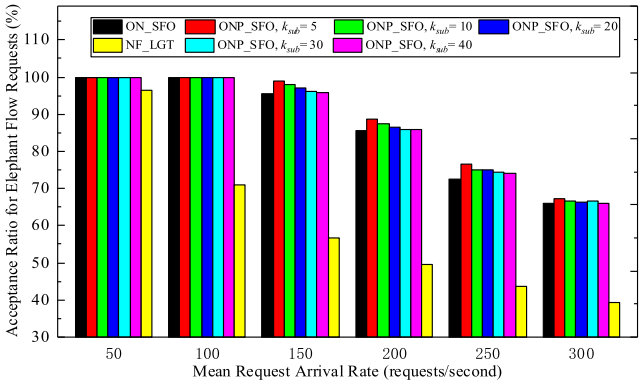
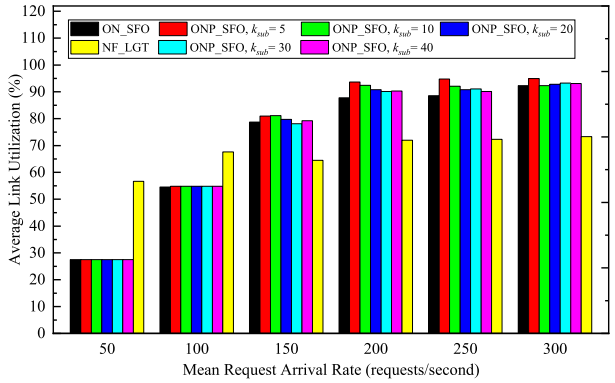
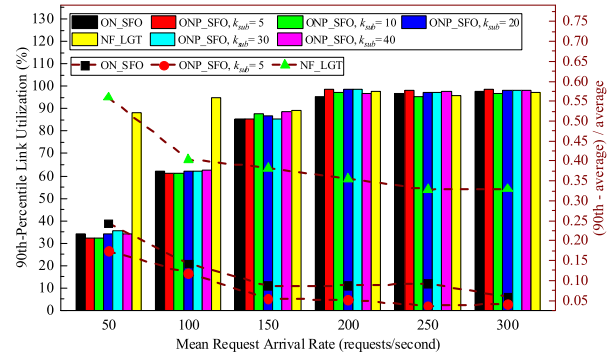


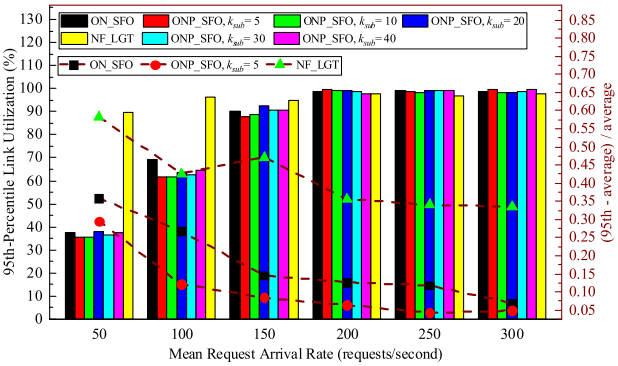
FIGURE 6. The acceptance ratio for elephant flow requests.



(a) Average link utilization

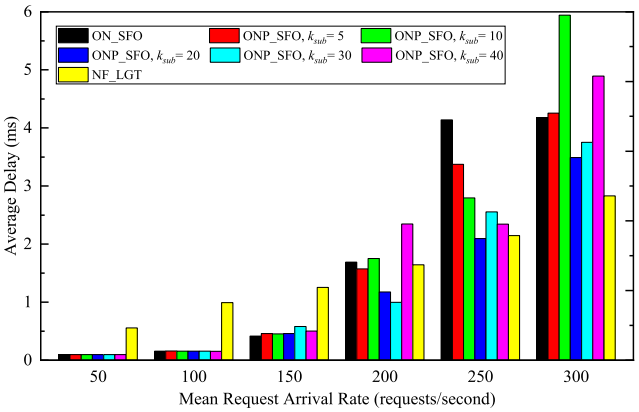


(b) The 90th-percentile link utilization and the link utilization ratio between the 90th percentile minus the average and the average

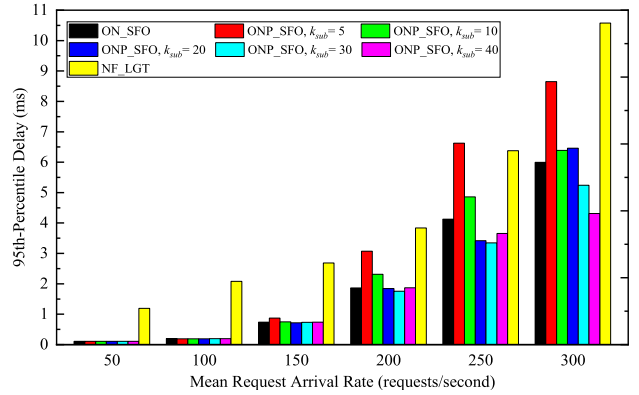


(c) The 95th-percentile link utilization and the link utilization ratio between the 95th percentile minus the average and the average

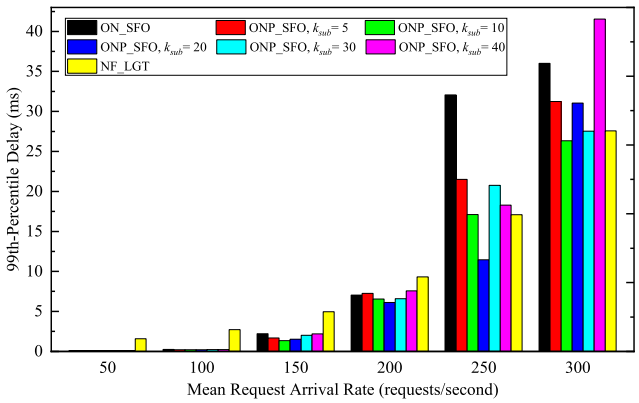
FIGURE 7. Link utilization as a function of network load.



(a) Average queueing delay for mouse flow requests



(b) The 95th-percentile queueing delay for mouse flow requests



(c) The 99th-percentile queueing delay for mouse flow requests

FIGURE 8. Queueing delay for mouse flows *vs.* network load.

At the same time, Fig. 5 and Fig. 6 show that among our algorithms, the acceptance ratios of the ONP_SFO algorithms are higher than that of ON_SFO. In addition, it can be observed that when the value of k_{sub} is smaller, the acceptance ratio is higher. These findings reveal that a smaller subflow size is beneficial for improving the acceptance ratio for elephant flow requests because more physical paths are selected to carry the whole flow, thus reducing the occurrence of link bottlenecks.

Now, we evaluate the link utilization performance of our algorithms with various values of k_{sub} and that of the NF-LGT algorithm, including the average link utilization, the 90th-percentile link utilization and the 95th-percentile link utilization, where the 90th- or 95th-percentile link utilization refers to the link utilization value that is greater than the utilizations of 90% or 95%, respectively, of all links in the network. As shown in Fig. 7 (a), as the mean user request arrival rate λ increases from 50 to 300 requests/second, the average link utilization increases and tends towards a fixed value for almost all algorithms. For example, after λ reaches 100 requests/second, the average link utilization of NF-LGT slowly grows to approximately 73%, while the average link utilization of our algorithms is still growing

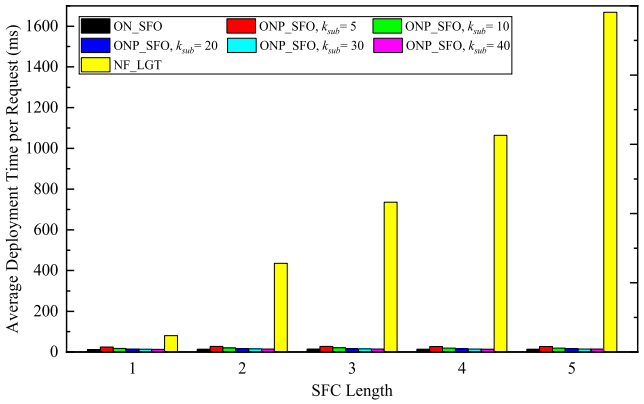


FIGURE 9. Average deployment time per request as a function of the length of the SFCs when the mean arrival rate is 100 requests/second.

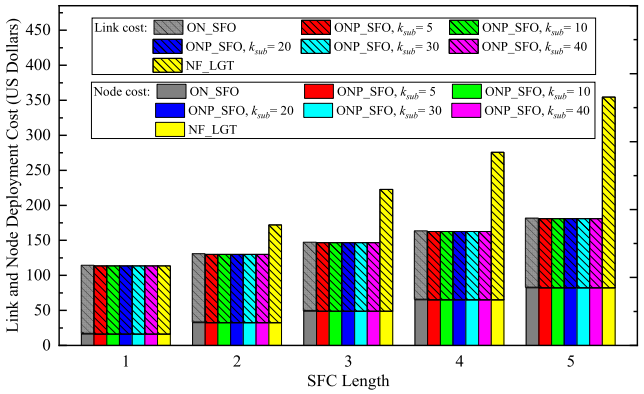
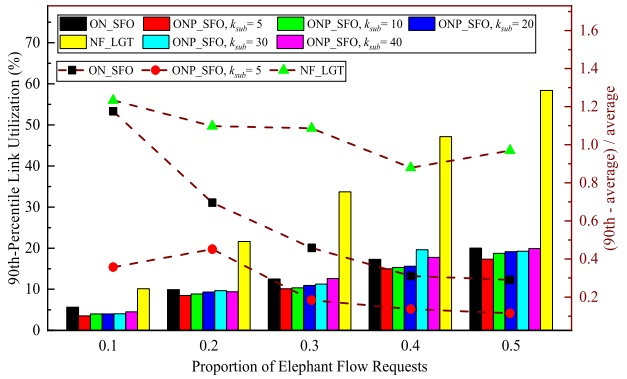
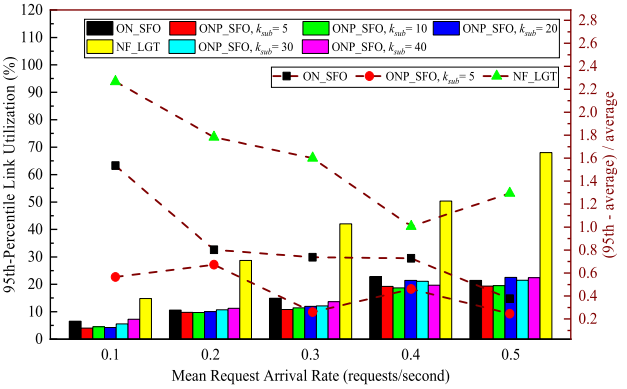


FIGURE 10. Average mapping cost as a function of the length of the SFCs.



(a) The 90th-percentile link utilization and the link utilization ratio between the 90th percentile minus the average and the average



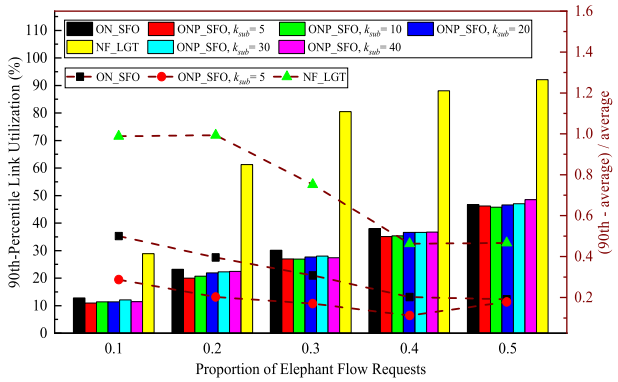
(b) The 95th-percentile link utilization and the link utilization ratio between the 95th percentile minus the average and the average

FIGURE 11. Mean arrival rate: 10 requests/second.

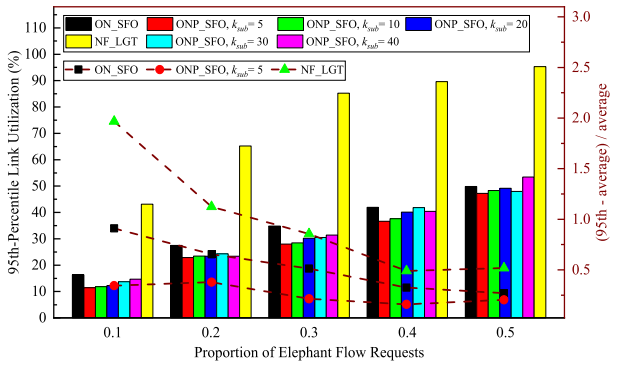
rapidly. When λ reaches 200 requests/second, the average link utilization of our algorithms tends to grow slowly to approximately 94%. Thus, we can see that the link resource utilization of our algorithms is nearly 30% higher than that of the NF-LGT algorithm. Among our algorithms, the link resource utilization performance of the algorithms that consider parallelized SFC deployment is higher than that of the version in which only the original SFCs are deployed.

In addition, as seen in Fig. 7 (b) and Fig. 7 (c), the two ratios $\frac{90th-average}{90th}$ and $\frac{95th-average}{95th}$ are both greater than 0.25 for the NF-LGT algorithm, illustrating that the link utilizations considerably differ among different physical links; most of the links have light loads, while a few have heavy loads, indicating that this algorithm does not achieve effective network load balancing. In contrast, these ratios are mostly approximately 0.1 for ON_SFO. Moreover, compared with that for ON_SFO, the ratio $\frac{95th-average}{95th}$ for ONP_SFO with $k_{sub} = 5$ is reduced by up to 50% (from 0.216 to 0.108) when $\lambda = 100$ requests/second in Fig. 7 (c). These findings reveal that our algorithms are able to achieve better network load balancing than the NF-LGT algorithm is and that considering the splitting of flows is helpful.

In Fig. 8, we focus on the performance in terms of the queueing delay for mouse flows, which can greatly affect the end-to-end delay and user experience, especially in a DCN, under the assumption that other types of delays, such as the propagation delay, transmission delay and processing delay, are fixed or can be ignored. As Fig. 8 (a) shows, once λ reaches 200 requests/second, the average queueing delays experienced by mouse flow requests are shorter with NF-LGT than with our algorithms. This occurs because NF-LGT reduces the average link utilization, as shown in Fig. 7 (a), at the expense of the acceptance ratio for elephant flow requests, as shown in Fig. 6. Fig. 8 (b) and Fig. 8 (c) show the 95th- and 99th-percentile queueing delays, which are defined similarly to the 90th- and 95th-percentile link utilizations and represent the maximum queueing delays experienced by 95% and 99%, respectively, of the mouse flow requests. As seen in Fig. 8 (b), the 95th-percentile queueing delay of NF-LGT is always greater than that of ON_SFO, while Fig. 8 (c) shows that the 99th-percentile queueing delay of NF-LGT is smaller when the network load is heavy ($\lambda \geq 250$requests/second). The former finding is consistent with the conclusion drawn from Fig. 7 that in NF-LGT, the network load is not balanced. The latter finding can be explained by the fact that, as shown in Fig. 7 (b) and Fig. 7 (c), the 90th- and 95th-percentile upper limits on the link utilization for the NF-LGT algorithm are lower than those for our algorithms.

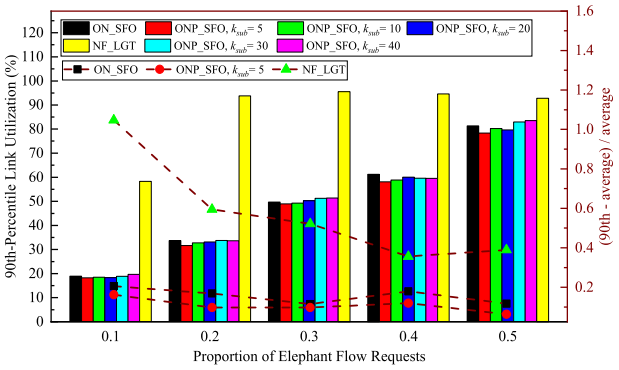


(a) The 90th-percentile link utilization and the link utilization ratio between the 90th percentile minus the average and the average

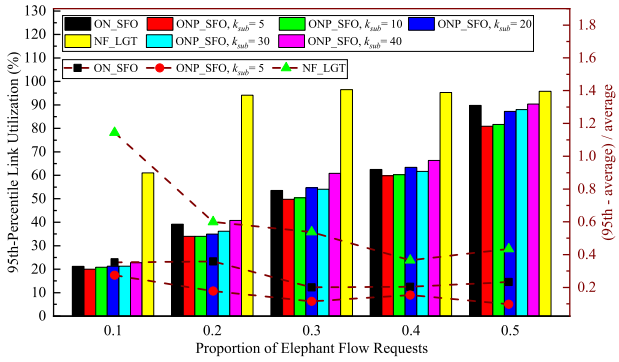


(b) The 95th-percentile link utilization and the link utilization ratio between the 95th percentile minus the average and the average

FIGURE 12. Mean arrival rate: 30 requests/second.



(a) The 90th-percentile link utilization and the link utilization ratio between the 90th percentile minus the average and the average



(b) The 95th-percentile link utilization and the link utilization ratio between the 95th percentile minus the average and the average

FIGURE 13. Mean arrival rate: 50 requests/second.

On the other hand, among our algorithms, as seen in Fig. 8, the queueing delay of ON_SFO is roughly equal to those of the ONP_SFO algorithms when the network load is light, while when the network load is heavy, the performance of the former degrades due to the inherent characteristics of the queueing delay. According to Eq. (1), when the traffic intensity $L \cdot ar(e)/C(e)$ is close to zero, the queueing delay D_e is also close to zero, while when the traffic intensity is close to 1, the queueing delay considerably increases. Therefore, when the network load is light, the gap between the queueing delays of our algorithms is small, whereas when the network load is heavy, the differences between the link utilization characteristics of the different algorithms will greatly impact the queueing delays experienced by mouse flow requests.

2) THE INFLUENCE OF THE LENGTH OF THE SFCS

Fig. 9 depicts the performance of the algorithms in terms of the average deployment time per request when the length of the SFCs varies from 1 to 5. For the NF-LGT algorithm, a large amount of time is required on average to deploy each SFC; this finding can be explained by the fact that this algorithm needs to traverse all available paths to find the one with the shortest delay when deploying each VNF of each chain. This also explains why the average deployment time of the NF-LGT algorithm increases significantly as the number of VNFs per SFC increases.

In Fig. 10, we compare the average mapping costs of our algorithms with that of the NF-LGT algorithm with respect to the length of the SFCs when link bandwidth resources and server computing resources are unlimited. At that situation, all user requests are accepted in all algorithms and node mapping cost only changes with the numbers of SFCs according to Formula (18), which is the reason why the average node costs for all algorithms are equal when the SFC's length is fixed. In addition, this figure shows the average link mapping costs of our algorithms are equal and reduced by up to 63% compared to NF-LGT. This result can also be explained that our algorithms try to deploy the VNFs of the same sub-SFC into a server whereas in the compared algorithms the VNFs are randomly deployed into the physical network.

3) THE INFLUENCE OF THE PROPORTION OF ELEPHANT FLOW REQUESTS

Fig. 11, Fig. 12 and Fig. 13 show how the link utilization, including the 90th- and 95th-percentile link utilizations, of our algorithms and the NF-LGT algorithm varies with the proportion of elephant flow requests when the mean arrival rate is 10, 30 and 50 requests/second, respectively. For NF-LGT, as the proportion of elephant flow requests

increases from 0.1 to 0.5, the link utilization increases and finally tends towards a certain value. This occurs because when the proportion of elephant flow requests among all requests is greater and the mean arrival rate is higher, the network load is also heavier. Once the network load increases past a certain threshold, an increasing number of requests are rejected due to the limited resources available, causing the growth rate of link utilization to decrease and finally approach 0. By contrast, for our algorithms, the link utilization is continuously increasing and is much lower than that of NF-LGT because of better resource utilization.

In addition, we compare the performance of ON_SFO, ONP_SFO with $k_{sub} = 5$, and NF-LGT in terms of network load balancing as measured by the two ratios $\frac{90th-average}{90th}$ and $\frac{95th-average}{95th}$. As shown in the figures, under different arrival rates, both ratios for ONP_SFO with $k_{sub} = 5$ are always lower than the ratios for ON_SFO, which, in turn, are lower than those for NF_LGT, as the proportion of elephant flow requests increases. These findings show that the performance of our algorithms is not sensitive to different proportions of elephant flow requests.

VI. CONCLUSION

In this paper, we consider the heterogeneous needs of elephant flow requests and mouse flow requests in a DCN. By splitting original user requests with large bandwidth demands into sub-user requests, our approach is expected to alleviate the hash collision of elephant flows and avoid mouse flows queueing behind elephant flows, which can cause the mouse flows to experience long queueing delays. Based on parallelized SFC mapping in the DCN, we first propose a model with the objective of minimizing the total queueing delay for all user requests subject to the physical resource constraints and the SFC requirements. Then, an online heuristic algorithm called ON_SFO is proposed, which not only determines the method of splitting the original user requests into sub-user requests but also effectively deploys each sub-user request by calling the *downOperation* and *upOperation* algorithms. These two basic operations select servers or switches based on a worst-fit strategy to improve the load balancing of the network. Moreover, the ON_SFO algorithm tries to deploy all VNFs of the same SFC on servers within the fewest number of hops to save bandwidth resources. Finally, based on a large number of simulations conducted to compare ON_SFO with ONP_SFO and NF-LGT, our results show that ON_SFO realizes better network load balancing compared with NF-LGT, for example, as shown in Fig. 7 (b) and Fig. 7 (c), the ratios $\frac{90th-average}{90th}$ and $\frac{95th-average}{95th}$ of ONP_SFO when $k_{sub} = 5$ are both less than 0.25 than compared algorithms. Furthermore, the acceptance ratio for all user requests is improved, especially for elephant flow requests at least by 72% when the network link utilization tends to be stable. In addition, regardless of whether flow splitting is considered, our algorithms significantly outperform NF-LGT in terms of the deployment time and link mapping cost.

Our future work will focus on deploying SFC across distributed datacenters and propose efficient algorithms to make a tradeoff between datacenter resources and the heterogeneous demand of service requests (latency-aware, throughput-aware, especially safety-aware requests, etc.), taking into account the flow splitting.

REFERENCES

[1] D. Zhao, J. Ren, R. Lin, S. Xu, and V. Chang, "On orchestrating service function chains in 5G mobile network," *IEEE Access*, vol. 7, pp. 39402–39416, 2019.
[2] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, "Service function chaining in next generation networks: State of the art and research challenges," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 216–223, Feb. 2017.
[3] M. Mechtri, C. Ghribi, O. Soualah, and D. Zeghlache, "NFV orchestration framework addressing SFC challenges," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 16–23, Jun. 2017.
[4] G. Sun, Y. Li, D. Liao, and V. Chang, "Service function chain orchestration across multiple domains: A full mesh aggregation approach," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 3, pp. 1175–1191, Sep. 2018.
[5] G. Sun, Y. Li, H. Yu, A. V. Vasilakos, X. Du, and M. Guizani, "Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks," *Future Gener. Comput. Syst.*, vol. 91, pp. 347–360, Feb. 2019.
[6] B. Yi, X. Wang, S. K. Das, K. Li, and M. Huang, "A comprehensive survey of network function virtualization," *Comput. Netw.*, vol. 133, pp. 212–262, Mar. 2018.
[7] Y. Xiao, X. Du, J. Zhang, and S. Guizani, "Internet protocol television (IPTV): The killer application for the next generation Internet," *IEEE Commun. Mag.*, vol.45, no. 11, pp. 126–134, Nov. 2007.
[8] H. Hawilo, M. Jammal, and A. Shami, "Exploring microservices as the architecture of choice for network function virtualization platforms," *IEEE Netw.*, vol. 33, no. 3, pp. 202–210, Mar./Apr. 2019.
[9] G. Sun, Z. Xu, H. Yu, V. Chang, X. Du, and M. Guizani, "Toward SLAs guaranteed scalable VDC provisioning in cloud data centers," *IEEE Access*, vol. 7, no. 3, pp. 80219–80232, 2019.
[10] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira, "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 80–87, May 2017.
[11] V. G. Nguyen, A. Brunstrom, K.-J. Grinnemo, and J. Taheri, "SDN/NFV-based mobile packet core network architectures: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1567–1602, 3rd Quart., 2017.
[12] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis, "Efficient NFV-enabled multicasting in SDNs," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 2052–2070, Nov. 2019.
[13] H. Cao, H. Zhu, and L. Yang, "Dynamic embedding and scheduling of service function chains for future SDN/NFV-enabled networks," *IEEE Access*, vol. 7, pp. 39721–39730, 2019.
[14] P. Dong, X. Du, H. Zhang, and T. Xu, "A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows," in *Proc. IEEE ICC*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.
[15] G. Sun, Y. Li, Y. Li, D. Liao, and V. Chang, "Low-latency orchestration for workflow-oriented service function chain in edge computing," *Future Gener. Comput. Syst.*, vol. 85, pp. 116–128, Aug. 2018.
[16] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella, "Presto: Edge-based load balancing for fast datacenter networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 465–478, 2015.
[17] P. Wang, G. Trimponias, H. Xu, H. Liu, and Y. Geng, "Luopan: Sampling-based load balancing in data center networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 1, pp. 133–145, Jan. 2019.
[18] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 63–74, Oct. 2011.
[19] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proc. ACM SIGCOMM Conf. Internet Meas.*, 2009, pp. 202–208.
[20] H. Xu and B. Li, "TinyFlow: Breaking elephants down into mice in data center networks," in *Proc. IEEE Int. Workshop Local Metrop. Area Netw. (LANMAN)*, May 2014, pp. 1–6.

[21] S. Liu, H. Xu, L. Liu, W. Bai, K. Chen, and Z. Cai, "RepNet: Cutting latency with flow replication in data center networks," *IEEE Trans. Services Comput.*, to be published.

[22] C.-H. Hsieh, J.-W. Chang, C. Chen, and S.-H. Lu, "Network-aware service function chaining placement in a data center," in *Proc. IEEE Asia–Pacific Netw. Oper. Manage. Symp.*, Oct. 2016, pp. 1–6.

[23] Y. Li, L. T. X. Phan, and B. T. Loo, "Network functions virtualization with soft real-time guarantees," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

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

[25] T.-M. Pham, T.-T.-L. Nguyen, S. Fdida, and H. T. T. Binh, "Online load balancing for network functions virtualization," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[26] P.-C. Lin, Y.-D. Lin, C.-Y. Wu, Y.-C. Lai, and Y.-C. Kao, "Balanced service chaining in software-defined networks with network function virtualization," *Computer*, vol. 49, no. 11, pp. 68–76, Nov. 2016,

[27] M.-T. Thai, Y.-D. Lin, P.-C. Lin, and Y.-C. Lai, "Towards load-balanced service chaining by hash-based traffic steering on softswitches," *J. Netw. Comput. Appl.*, vol. 109, pp. 1–10, May 2018.

[28] A. Engelmann and A. Jukan, "A reliability study of parallelized VNF chaining," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[29] G. Sun, D. Liao, S. Bu, H. Yu, Z. Sun, and V. Chang, "The efficient framework and algorithm for provisioning evolving VDC in federated data centers," *Future Gener. Comput. Syst.*, vol. 73, pp. 79–89, Aug. 2017.

[30] T. Chen, J. Liu, Q. Tang, T. Huang, and R. Huo, "Virtual network embedding algorithm for location-based identifier allocation," *IEEE Access*, vol. 7, pp. 31159–31169, 2019.

[31] G. Sun, H. Yu, L. Li, V. Anand, and H. Di, "The framework and algorithms for the survivable mapping of virtual network onto a substrate network," *IETE Tech. Rev.*, vol. 28, no. 3, pp. 381–391, 2011.

[32] H. Di, L. Li, V. Anand, H. Yu, and G. Sun, "Cost efficient virtual infrastructure mapping using subgraph isomorphism," in *Proc. Asia Commun. Photon. Conf. Exhib. (ACP)*, 2010, pp. 533–534.

[33] H. Di, H. Yu, V. Anand, L. Li, G. Sun, and B. Dong, "Efficient online virtual network mapping using resource evaluation," *J. Netw. Syst. Manage.*, vol. 20, no. 3, pp. 468–488, 2012.

[34] S. Su, Z. Zhang, A. X. Liu, X. Cheng, Y. Wang, and X. Zhao, "Energy-aware virtual network embedding," *IEEE/ACM Trans. Netw.*, vol. 22, no. 3, pp. 1607–1620, Oct. 2014.

[35] G. Sun, V. Anand, H.-F. Yu, D. Liao, and L. Li, "Optimal provisioning for elastic service oriented virtual network request in cloud computing," in *Proc. IEEE GLOBECOM*, Anaheim, CA, USA, Dec. 2012, pp. 2517–2522.

[36] H. Cao, Y. Zhu, G. Zheng, and L. Yang, "A novel optimal mapping algorithm with less computational complexity for virtual network embedding," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 1, pp. 356–371, Mar. 2018.

[37] D. Dietrich, A. Rizk, and P. Papadimitriou, "Multi-provider virtual network embedding with limited information disclosure," *IEEE Trans. Netw. Service Manag.*, vol. 12, no. 2, pp. 188–201, Jun. 2015.

[38] M. Alaluna, L. Ferrolho, J. R. Figueira, N. Neves, and F. M. V. Ramos, "Secure virtual network embedding in a multi-cloud environment," 2017, *arXiv:1703.01313v1*. [Online]. Available: https://arxiv.org/abs/1703.01313v1

[39] G. Sun, D. Liao, D. Zhao, Z. Sun, and V. Chang, "Towards provisioning hybrid virtual networks in federated cloud data centers," *Future Gener. Comput. Syst.*, vol. 87, pp. 457–469, Oct. 2018.

[40] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for NFV chaining in packet/optical datacenters," *J. Lightw. Technol.*, vol. 33, no. 8, pp. 1565–1570, Apr. 15, 2015.

[41] D. Li, P. Hong, K. Xue, and J. Pei, "Virtual network function placement considering resource optimization and SFC requests in cloud datacenter," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 7, pp. 1664–1677, Jul. 2018.

[42] J. Fan, M. Jiang, O. Rottenstreich, Y. Zhao, T. Guan, R. Ramesh, S. Das, and C. Qiao, "A framework for provisioning availability of NFV in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 2246–2259, Oct. 2018.

[43] Y. Jia, C. Wu, Z. Li, F. Le, and A. Liu, "Online scaling of NFV service chains across geo-distributed datacenters," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 699–710, Apr. 2018.

[44] M.-T. Thai, Y.-D. Lin, and Y.-C. Lai, "A joint network and server load balancing algorithm for chaining virtualized network functions," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[45] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba, "Distributed service function chaining," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2479–2489, Nov. 2017.

[46] F. Carpio, S. Dhahri, and A. Jukan, "VNF placement with replication for Loac balancing in NFV networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[47] Y. Sang, B. Ji, G. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *Proc. IEEE INFOCOM*, Atlanta, Ga, USA, May 2017, pp. 1–9.

[48] F. B. Jemaa, G. Pujolle, and M. Pariente, "Qos-aware VNF placement optimization in edge-central carrier cloud architecture," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–7.

[49] G. Sun, G. Zhu, D. Liao, H. Yu, X. Du, and M. Guizani, "Cost-efficient service function chain orchestration for low-latency applications in NFV networks," *IEEE Syst. J.*, to be published. doi: 10.1109/JSYST.2018.2879883.

[50] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

[51] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 267–280.