This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2880992, IEEE Transactions on Parallel and Distributed Systems

Efficiently Embedding Service Function Chains with Dynamic Virtual Network Function Placement in Geo-distributed Cloud System

Jianing Pei, Peilin Hong, Kaiping Xue, *Senior Member, IEEE* and Defang Li, *Student Member, IEEE*

***Abstract*—Network Function Virtualization (NFV) and Software-Defined Networks (SDN) enable Internet Service Providers (ISPs) to place Virtual Network Functions (VNFs) to achieve the performance and security benefit without incurring high Operating Expenses (OPEX) and Capital Expenses (CAPEX). In NFV environment, Service Function Chains (SFCs) always need to steer the traffic through a series of VNF instances in predefined orders. Moreover, the required number and placement of VNF instances should be optimized to adapt to dynamic network load. Therefore, it is considerable for ISPs to conduct an optimal SFC embedding strategy to improve the network performance and revenue. In the paper, we study the SFC embedding problem (SFC-EP) with dynamic VNF placement in geo-distributed cloud system. We formulate this problem as a Binary Integer Programming (BIP) model aiming to embed SFC requests with the minimum embedding cost. Furthermore, the novel SFC eMbedding APproach (SFC-MAP) and VNF Dynamic Release Algorithm (VNF-DRA) have been proposed to efficiently embed SFC requests and optimize the number of placed VNF instances. Performance evaluation results show that the proposed algorithms can provide higher performance in terms of SFC request acceptance rate, network throughput and mean VNF utilization rate and efficiently reduce the total VNF running time compared with the algorithms in existing literatures.**

***Index Terms*—Service Function Chain, Virtual Network Function, Dynamic VNF Placement**

I. Introduction

IT is ubiquitous to place middleboxes in today's network to offer varieties of network services to customers. Traditional middleboxes are implemented by dedicated hardware appliances which lead to high infrastructure and management costs [1]. Since the advent of Network Function Virtualization (NFV) and Software-Defined Networks (SDN), the Virtual Network Functions (VNFs) are implemented in software and placed on commercial-off-the-shelf devices. Because of great management, flexibility and cost-efficiency, VNF has great potential to replace traditional middleboxes and provide performance and security enhancements in the network [2], [3]. Moreover, combined with geo-distributed cloud system, ISPs can provide network services with better reliability and lower latency by placing VNFs in Micro-DataCenters (MDCs) which are closer to end users [4], [5].

J. Pei, P. Hong, K. Xue and D. Li are all with the Key Laboratory of Wireless-Optical Communications, Chinese Academy of Sciences, School of Information Science and Technology, University of Science and Technology of China, Hefei, 230027, China (e-mail: jianingp@mail.ustc.edu.cn; plhong@ustc.edu.cn; kpxue@ustc.edu.cn; ldf911@mail.ustc.edu.cn).

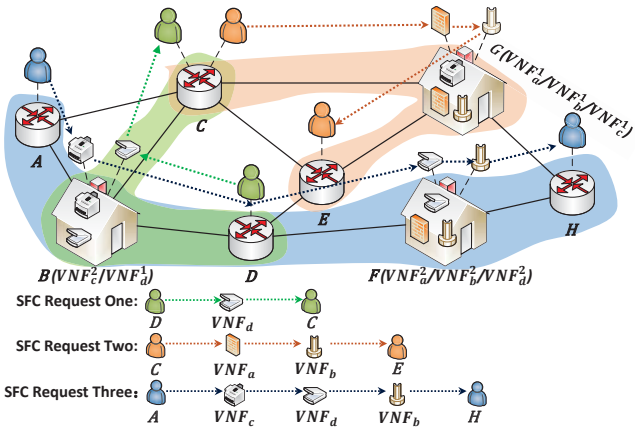


Fig. 1. Embedding SFC requests in geo-distributed cloud system.

When dealing with the requests from customers, it is often needed to steer their traffic to traverse the VNFs concatenated in a specified order to comply with security and performance policies, which is defined as Service Function Chain (SFC) [4], [6]. In NFV environment, each type of VNF is usually multi-instance and these VNF instances can be dynamically placed at various network locations. Therefore, it is a challenge to embed SFC requests with the optimal VNF selection and placement [6], [7]. Besides, due to finite physical resources (e.g. bandwidth, memory, CPU and so on) in the network, the optimal path selection for traffic steering also should be focused on to achieve load balancing and reduce resource bottlenecks [1], [4], [8].

An example of this problem is shown in Fig. 1. In the geo-distributed cloud system, there are five switch nodes and three MDC nodes with four types of VNFs placed on them, and each type of VNF is multi-instance. VNF_a^1 and VNF_a^2 represent the first and second instances of VNF_a, respectively. As for SFC request two, the traffic needs to sequentially traverse the ingress node C, the instances of VNF_a and VNF_b before reaching the egress node E. However, it is noted that both of MDC node F and G have been placed with the instances of VNF_a and VNF_b, and MDC node B can place new instances for these two types of VNFs as well. Therefore, the optimal SFC embedding strategy should be made to select or place the instances of VNF_a and VNF_b and steer the traffic for this SFC request.

Furthermore, due to highly dynamic nature of resource usage, start-up and lifetime, SFC requests are dynamic and result

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2880992, IEEE Transactions on Parallel and Distributed Systems

in the variation of network load [9], [10]. In the network, the placement of VNF instances needs the consumption of system resources [11], [12]. The management, monitoring and maintenance for VNF instances all incur the increasement of energy consumption, Operating Expenses (OPEX) and Capital Expenses (CAPEX) [13], [14]. Therefore, it is beneficial to enhance the revenue of ISPs by optimizing the required number and placement of VNF instances and reducing the total running time of VNFs according to the variation of network load. When network load is heavy, it is better to increase the number of VNF instances to keep the network with high performance. On the contrary, with light network load, it is helpful to reduce the total VNF running time by decreasing the number of VNF instances.

Given finite physical resources and multiple VNF instances environment in geo-distributed cloud system, the next two problems should be focused on: (i) how to optimize the number and placement of VNF instances; (ii) how to select and concatenate VNF instances with the minimum embedding cost for an SFC request. In the paper, the embedding cost includes resource cost and VNF placement cost. The resource cost is related to the remaining resources on links, nodes and VNF instances, which is used to achieve load balancing. The VNF placement cost results from the costs of computing power, license fees and network utilization [12], [15]. In our work, these problems above are defined as SFC Embedding Problem (SFC-EP) with dynamic VNF placement.

In order to solve the SFC-EP with dynamic VNF placement, first, we formulate it as a Binary Integer Programming (BIP) model and the objective is to minimize the embedding cost for each SFC request. Then, we propose two novel algorithms named SFC eMbedding APproach (SFC-MAP) and VNF Dynamic Release Algorithm (VNF-DRA). SFC-MAP ingeniously obtains the selection, placement and concatenation solutions of VNF instances by running the shortest path algorithm (*e.g.,* Dijkstra) in multi-layer graph which is a transformation of original network topology based on the order constraints of SFC requests. VNF-DRA is in charge of the optimization of placed VNF instances to reduce their running time. In VNF-DRA, we periodically check the placed VNF instances and release the ones with their utilization rates being lower than the threshold which can be dynamically adjusted according to the variation of network load. The contributions are listed as follows:

- Taking the bandwidth, memory, CPU, end-to-end delay and VNF placement cost into account, we formulate the SFC-EP with dynamic VNF placement as a BIP model aiming to minimize the embedding cost for each SFC request.
- We propose the novel SFC-MAP algorithm to place, select and concatenate VNF instances for SFC requests. And we also propose the VNF-DRA algorithm so as to reduce the running time of VNF instances by releasing redundant VNF instances according to the variation of network load.
- We conduct a detailed analysis to our proposed algorithms. The performance evaluation results show that our proposed algorithms can obtain higher network performance in terms of SFC request acceptance rate, network throughput and mean VNF utilization rate and efficiently reduce the total VNF running time compared with the algorithms in existing literatures [12], [16].

The rest of the paper is organized as follows: the related work is presented in Section II. We explain the system model in Section III. In Section IV, the SFC-EP with dynamic VNF placement is defined and formulated as a BIP model. We propose the SFC-MAP and VNF-DRA algorithms in Section V and Section VI, respectively. The solutions are evaluated in Section VII. Finally, we conclude this paper in Section VIII.

II. Related Work

Recently, addressing the problem of VNF placement and chaining has become a hot issue, and many solutions have been proposed. Li *et al.* [17] and Moens *et al.* [18] respectively studied the VNF placement problem in the NFV environment and formulated this problem as integer linear programming models. Lin *et al.* [19] and Addis *et al.* [20] studied the VNF placement and routing optimization problem respectively and formulated it as mixed integer programming models. However, the integer linear programming model and mixed linear programming model can only be solved offline, because of high complexity.

Pham *et al.* [21] studied the problem of VNF placement for SFC with a sampling-based markov approximation approach and they also proposed a matching algorithm based on markov approximation to solve this problem with short convergence time. Ma *et al.* [22] formulated the traffic-aware middlebox placement problem as a graph optimization problem and proposed a two-step algorithm to develop results. Zeng *et al.* [23] considered to optimize the VNF placement multicast routing and spectrum assignment with tree-type VNF Forwarding Graphs (VNF-FG) and three heuristic algorithms were proposed to solve this problem. In most of the above solutions, the VNF placement and chaining problem is solved in two steps where the first step is to decide the number and location of VNF instances and the second step is to concatenate the VNF instances for SFC requests. In the paper, we can jointly embed SFC requests and place VNF instances, where load balancing is also considered to reduce the resource bottlenecks.

In order to solve the VNF orchestration problem, Bari *et al.* [12] formulated this problem as an integer linear programming model and proposed a heuristic named ProvisionTraffic to embed SFC requests by executing the Viterbi algorithm [24] in multi-stage graph. The problem of VNF placement and chaining for VNF-FG is studied in [16] and the authors designed a heuristic based on eigendecomposition which could match the extended adjacency matrix of a VNF-FG with the adjacency matrix of physical network according to Umeyama's eigendecomposition approach [25]. The joint optimization of service graph decomposing and its embedding problem is studied in [26] and an ILP-based algorithm and a mapping algorithm are proposed to minimize the resource consumption referring to virtual machines, bandwidth, I/O and hardware. In addition, some works [22], [27], [28], [29], [30] study how to place VNF instances and embed SFC requests with the optimal resource utilization. Nevertheless, most of the related works

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2880992, IEEE Transactions on Parallel and Distributed Systems

do not consider to release VNF instances when the network load goes down, which leads to high OPEX/CAPEX because of high VNF running time.

Considering the energy cost, Eramo *et al.* [11] proposed to route SFC request and place VNF instances with discrete time markov decision process, and they reduced the energy cost at low traffic period by migrating VNF instances and shutting down empty servers. Eramo *et al.* [31] and Ghaznavi *et al.* [14] also proposed to optimize resource utilization and reduce energy cost by the migration of VNF instances in NFV architecture. Moreover, the works [13], [32] and [33] present the VNF orchestration architectures to install, monitor, migrate and release VNF instances.

Different from the literatures mentioned above, we focus on the SFC-EP with dynamic VNF placement in geo-distributed cloud system. We jointly place VNF instances and embed SFC requests considering load balancing based on multi-layer graph. And we also pay attention to release redundant VNF instances for the reduction of the total VNF running time, when the network load becomes light.

III. System Model

A. Physical Network

The physical network is represented as an undirected graph $G = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ and $\mathcal{L}$ denote the sets of nodes and links, respectively. In the physical network, we use $u, v \in \mathcal{V}$ to indicate two nodes and $uv \in \mathcal{L}$ to represent the link connecting node u and v. In this paper, we consider the SFC-EP with dynamic VNF placement in geo-distributed cloud system. Then we suppose that there exist two kinds of nodes in the network. One kind is switch node that is responsible to forward data to neighbor nodes, and the other kind is MDC node that is not only in charge of data forwarding but also holding VNF instances to process the traffic of SFC requests. We denote $\mathcal{V}_{sn} \subseteq \mathcal{V}$ as the set of switch nodes, and $\mathcal{V}_{mdc} \subseteq \mathcal{V}$ as the set of MDC nodes.

In the network, the bandwidth, memory and CPU are considered on links, nodes, and VNF instances. For link $uv \in \mathcal{L}$, the bandwidth capacity is denoted as C_{uv}^{bw}. The bandwidth remaining rate, when embedding SFC request i, is symbolized as $r_{i,uv}^{bw}$. The parameter C_u^{mem}, $u \in \mathcal{V}_{sn}$, stands for the memory capacity of switch node u. We use $r_{i,u}^{mem}$, $u \in \mathcal{V}_{sn}$ to represent the memory remaining rate of switch node u, when embedding SFC request i. It is noted that, compared with general switch nodes, the memory capacity of MDC nodes $C_u^{mem}, u \in \mathcal{V}_{mdc}$ is regarded as infinite. Due to the fact that the computation resource of an MDC node is finite, we define $n_u, u \in \mathcal{V}_{mdc}$ as the maximum number of VNF instances permitted to be placed in MDC node u. We denote $\mathcal{M}$ as the set of all the VNF instances permitted to be placed in MDC nodes, and $|\mathcal{M}| = \sum_{u \in \mathcal{V}_{mdc}} n_u$. We use $m \in \mathcal{M}$ to represent the VNF instance m. The parameter C_m^{cpu} represents the CPU capacity that VNF instance $m \in \mathcal{M}$ can apply from corresponding MDC node. The CPU remaining rate of $m \in \mathcal{M}$, when embedding SFC request i, is denoted as $r_{i,m}^{cpu}$. And we define $\mathcal{K}$ as the set of VNF types, and $VNF_k, k \in \mathcal{K}$ stands for the VNF type k.

B. Service Function Chain Requests

In this paper, a 7-tuple, $\{S_i, T_i, \mathcal{Q}_i, \Psi_i^{bw}, \Psi_i^{mem}, \Psi_i^{cpu}, \Psi_i^{td}\}$ is used to represent SFC request i, where S_i and T_i represent the ingress node and egress node, respectively. The set of VNFs requested by SFC request i is denoted by $\mathcal{Q}_i = \{\mathcal{Q}_i^1, \mathcal{Q}_i^2, ..., \mathcal{Q}_i^l\}$, $l = |\mathcal{Q}_i|$, where $\mathcal{Q}_i^1, \mathcal{Q}_i^2, ..., \mathcal{Q}_i^l$ represent the $1^{st}, 2^{nd}, ..., l^{th}$ VNF requests in $\mathcal{Q}_i$, respectively. The parameters $\Psi_i^{bw}, \Psi_i^{mem}$ and Ψ_i^{cpu} represent the demands of bandwidth, memory and CPU on links, nodes and VNF instances, respectively. Ψ_i^{td} means the maximum tolerated delay of SFC request i.

We use $G_i = (\mathcal{V}_i, \mathcal{L}_i)$ to denote the service function graph of SFC request i. Service function graph is a directed graph, and the directions of links satisfy the order constraint of VNF requests. In service function graph, the parameters $\bar{u}, \bar{v} \in \mathcal{V}_i$ represent two VNF request nodes, and $\bar{u}\bar{v} \in \mathcal{L}_i$ is the link connecting node $\bar{u}$ and $\bar{v}$ in G_i. For example, as for the service function graph of SFC request two in Fig. 1, it starts at the ingress node C and ends at the egress node E traversing the instances of VNF_a and VNF_b in sequence.

IV. Problem Statement

A. Problem Description

In a geo-distributed cloud system with NFV environment, ISPs should make optimal plan to embed SFC requests with dynamic VNF placement. Since VNFs are multi-instance and can be flexibly placed at various network locations, as for new arrival SFC requests, it is important for ISPs to decide whether to select the placed VNF instances or place extra VNF instances to serve them. The traffic of SFC requests should traverse a series of VNF instances in predefined orders and the path selection can influence the resource consumption on links, nodes and VNF instances. Then, how to concatenate VNF instances for SFC requests should be considered to achieve load balancing and reduce resource bottlenecks. As stated before, the network load varies over time. Therefore, how to optimize the number of VNF instances should also be focused on by ISPs to reduce the OPEX/CAPEX and improve their revenue.

B. Problem Formulation

In this section, we formulate the SFC-EP with dynamic VNF placement as a BIP model. All the symbols and variables used in this part are listed in TABLE 1.

For SFC request i, the consumptions of bandwidth, memory and CPU cannot exceed the available resources on links, nodes and VNF instances, respectively, which are ensured as:

$$\sum_{\bar{u}\bar{v} \in \mathcal{L}_i} \Psi_i^{bw} z_{i,uv}^{\bar{u}\bar{v}} \leq C_{uv}^{bw} r_{i,uv}^{bw}, \forall uv \in \mathcal{L}, \tag{1}$$

$$\sum_{\bar{u}\bar{v} \in \mathcal{L}_i} \Psi_i^{mem} z_{i,u}^{\bar{u}\bar{v}} \leq C_u^{mem} r_{i,u}^{mem}, \forall u \in \mathcal{V}, \tag{2}$$

$$\sum_{\bar{u} \in \mathcal{V}_i} \Psi_i^{cpu} x_{i,m}^{\bar{u}} \leq C_m^{cpu} r_{i,m}^{cpu}, \forall m \in \mathcal{M}. \tag{3}$$

Here, we use the binary variables $z_{i,u}^{\bar{u}\bar{v}}$ and $z_{i,uv}^{\bar{u}\bar{v}}$ to indicate whether $\bar{u}\bar{v} \in \mathcal{L}_i$ traverses the node $u \in \mathcal{V}$ and link $uv \in \mathcal{L}$, respectively. $z_{i,u}^{\bar{u}\bar{v}}$ and $z_{i,uv}^{\bar{u}\bar{v}}$ equal 1, if $\bar{u}\bar{v} \in \mathcal{L}_i$ traverses the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2880992, IEEE Transactions on Parallel and Distributed Systems

TABLE 1
SYMBOLS AND VARIABLES

Symbols and Variables	Description
Physical Network	
$G = (\mathcal{V}, \mathcal{L})$	Physical network G with the sets of nodes $\mathcal{V}$ and links $\mathcal{L}$, $u, v \in \mathcal{V}$, $uv \in \mathcal{L}$.
$\mathcal{V}_{mdc}$, $\mathcal{V}_{sn}$	Sets of MDC nodes and switch nodes, $\mathcal{V} = \mathcal{V}_{mdc} \cup \mathcal{V}_{sn}$.
C_{uv}^{bw} ,C_{u}^{mem}, C_{m}^{cpu}	Capacity of bandwidth, memory and CPU.
$r_{i,uv}^{bw}$, $r_{i,u}^{mem}$, $r_{i,m}^{cpu}$	Remaining rates of bandwidth, memory and CPU, when embedding SFC request i.
$d_{i,uv}$, $d_{i,u}$, $d_{i,m}$	Delay on link $uv \in \mathcal{L}$, node $u \in \mathcal{V}$ and VNF instance $m \in \mathcal{M}$, when embedding SFC request i.
$c_{i,uv}^{bw}$, $c_{i,u}^{mem}$, $c_{i,m}^{cpu}$	Resource costs on link $uv \in \mathcal{L}$, node $u \in \mathcal{V}$, and VNF instance $m \in \mathcal{M}$, when embedding SFC request i.
n_u	The maximum number of VNF instances permitted to be placed in MDC node $u \in \mathcal{V}_{mdc}$.
VNF_k	VNF type $k \in \mathcal{K}$.
c_k^{vnf}	Placement cost of VNF_k, $k \in \mathcal{K}$.
$\mathcal{M}$	Set of all the VNF instances permitted to be placed in MDC nodes, $m \in \mathcal{M}$.
Ω_m	Set of VNF instances with the same VNF type of m, $\Omega_m \subset \mathcal{M}$.
Service Function Graph	
$G_i = (\mathcal{V}_i, \mathcal{L}_i)$	Service function graph G_i with the sets of nodes $\mathcal{V}_i$ and links $\mathcal{L}_i$, $\bar{u}, \bar{v} \in \mathcal{V}_i$, $\bar{u}\bar{v} \in \mathcal{L}_i$.
S_i, T_i, $\mathcal{Q}_i$	The ingress node, egress node and set of necessary VNF requests of SFC request i; $\mathcal{Q}_i = \{\mathcal{Q}_i^1, \mathcal{Q}_i^2, ..., \mathcal{Q}_i^l\}$, $l = \lvert\mathcal{Q}_i\rvert$.
Ψ_i^{bw}, Ψ_i^{mem}, Ψ_i^{cpu}, Ψ_i^{td}	The demands of bandwidth, memory, CPU and the maximum tolerated delay of SFC request i.
Binary Variables	
q_k^m	Whether VNF instance $m \in \mathcal{M}$ belongs to VNF_k, $k \in \mathcal{K}$.
$\widehat{w}_{i,m}$, $w_{i,m}$	Whether VNF instance $m \in \mathcal{M}$ is placed before and after embedding SFC request i, respectively.
$x_{i,m}^{\bar{u}}$	Whether VNF instance $m \in \mathcal{M}$ is used to serve VNF request $\bar{u} \in \mathcal{V}_i$, when embedding SFC request i.
y_u^m	Whether VNF instance $m \in \mathcal{M}$ can be placed in MDC node $u \in \mathcal{V}_{mdc}$.
$z_{i,uv}^{\bar{u}\bar{v}}$	Whether link $\bar{u}\bar{v} \in \mathcal{L}_i$ traverses link $uv \in \mathcal{L}$, when embedding SFC request i.
$z_{i,u}^{\bar{u}\bar{v}}$	Whether link $\bar{u}\bar{v} \in \mathcal{L}_i$ traverses node $u \in \mathcal{V}$ when embedding SFC request i.

node $u \in \mathcal{V}$ and link $uv \in \mathcal{L}$, and 0 otherwise. The binary variable $x_{i,m}^{\bar{u}}$ is used to indicate whether $\bar{u} \in \mathcal{V}_i$ is served by VNF instance $m \in \mathcal{M}$. $x_{i,m}^{\bar{u}}$ equals 1, if $\bar{u} \in \mathcal{V}_i$ is served by VNF instance $m \in \mathcal{M}$, and 0 otherwise.

As all the placed VNF instances in an MDC node cannot exceed the maximum number of VNF instances permitted to be placed after embedding SFC request i, the next constraint must be satisfied:

$$\sum_{m \in \mathcal{M}} y_u^m w_{i,m} \leq n_u, \forall u \in \mathcal{V}_{mdc}. \tag{4}$$

In the model, the binary variable y_u^m indicates whether the VNF instance $m \in \mathcal{M}$ can be placed in MDC node $u \in \mathcal{V}_{mdc}$. y_u^m equals 1, if $m \in \mathcal{M}$ can be placed in $u \in \mathcal{V}_{mdc}$, and 0 otherwise. In addition, we use the binary variable $w_{i,m}$ to indicate whether a VNF instance is placed after embedding SFC request i. And $w_{i,m}$ equals 1, if m is placed after embedding SFC request i, and 0 otherwise.

For SFC request i, the end-to-end delay of the path to embed it must meet the constraint of the maximum tolerated delay as:

$$\sum_{uv \in \mathcal{L}} \sum_{\bar{u}\bar{v} \in \mathcal{L}_i} d_{i,uv} z_{i,uv}^{\bar{u}\bar{v}} + \sum_{u \in \mathcal{V}} \sum_{\bar{u}\bar{v} \in \mathcal{L}_i} d_{i,u} z_{i,u}^{\bar{u}\bar{v}} + \sum_{m \in \mathcal{M}} \sum_{\bar{u} \in \mathcal{V}_i} d_{i,m} x_{i,m}^{\bar{u}} \leq \Psi_i^{td}. \tag{5}$$

In Eq. (5), the end-to-end delay consists of three parts where the first part represents the delay on links, and the second and third parts represent the delay on nodes and VNF instances, respectively.

If $uv \in \mathcal{L}$ is traversed by $\bar{u}\bar{v} \in \mathcal{L}_i$, $u, v \in \mathcal{V}$ must be traversed as well. Then the constraint must be ensured as:

$$z_{i,u}^{\bar{u}\bar{v}} z_{i,v}^{\bar{u}\bar{v}} = 1 \text{ if } z_{i,uv}^{\bar{u}\bar{v}} = 1. \tag{6}$$

We must guarantee that the links on the path to embed SFC request i are connected head-to-tail as:

$$\sum_{v \in \mathcal{V}} \sum_{\bar{u}\bar{v} \in \mathcal{L}_i} (z_{i,uv}^{\bar{u}\bar{v}} - z_{i,vu}^{\bar{u}\bar{v}}) = \begin{cases} 1, & u = S_i, \\ -1, & u = T_i, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

For SFC request i, the VNF instance $m \in \mathcal{M}$ may not be placed. And if necessary, we can serve this SFC request by placing VNF instance m in the corresponding MDC node as:

$$\sum_{u \in \mathcal{V}_{mdc}} \sum_{m \in \mathcal{M}} x_{i,m}^{\bar{u}} y_u^m w_{i,m} = 1, \forall \bar{u} \in \mathcal{V}_i \setminus \{S_i, T_i\}. \tag{8}$$

If VNF instance $m \in \mathcal{M}$ is selected to serve SFC request i, the MDC node holding m must ensure to be traversed as:

$$x_{i,m}^{\bar{u}} y_u^m \leq z_{i,u}^{\bar{u}\bar{v}}, \forall u \in \mathcal{V}_{mdc}, \forall \bar{u} \in \mathcal{V}_i, \forall \bar{u}\bar{v} \in \mathcal{L}_i. \tag{9}$$

Each VNF request of an SFC can be served by only one VNF instance as:

$$\sum_{m \in \mathcal{M}} x_{i,m}^{\bar{u}} = 1, \forall \bar{u} \in \mathcal{V}_i \setminus \{S_i, T_i\}. \tag{10}$$

In order to ensure that each VNF instance can be only placed in one MDC node, Eq. (11) must be satisfied as:

$$\sum_{u \in \mathcal{V}_{mdc}} y_u^m = 1, \forall m \in \mathcal{M}. \tag{11}$$

For the fact that a VNF instance can only belong to one type of VNF, the next constraint must be satisfied:

$$\sum_{k \in \mathcal{K}} q_k^m = 1, \forall m \in \mathcal{M}. \tag{12}$$

In Eq. (12), the binary variable q_k^m represents whether the VNF instance $m \in \mathcal{M}$ belongs to VNF_k, $k \in \mathcal{K}$. q_k^m equals 1, if $m \in \mathcal{M}$ belongs to VNF_k, $k \in \mathcal{K}$, and 0 otherwise.

In our work, the resource costs are considered to balance network load and reduce resource bottlenecks. We use $c_{i,uv}^{bw}$, $c_{i,u}^{mem}$ and $c_{i,m}^{cpu}$ to represent the costs of bandwidth, memory and CPU on link uv, switch node u and VNF instance m, when

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2880992, IEEE Transactions on Parallel and Distributed Systems

embedding SFC request i, respectively. And they are defined as:

$$c_{i,uv}^{bw} = \frac{\underset{uv\in\mathcal{L}}{max}\, C_{uv}^{bw}}{C_{uv}^{bw} r_{i,uv}^{bw}}, \tag{13}$$

$$c_{i,u}^{mem} = \frac{\underset{u\in\mathcal{V}_{sn}}{max}\, C_{u}^{mem}}{C_{u}^{mem} r_{i,u}^{mem}}, \tag{14}$$

$$c_{i,m}^{cpu} = \frac{\underset{m\in\Omega_m}{max}\, C_{m}^{cpu}}{C_{m}^{cpu} r_{i,m}^{cpu}}, \Omega_m \subset \mathcal{M}, \tag{15}$$

where $\Omega_m \subset \mathcal{M}$ in Eq. (15) represents a set that includes all the VNF instances with the same VNF type of m.

The resource costs defined in Eqs. (13)-(15) have reciprocal relationships to the remaining bandwidth, memory and CPU. For example, in Eq. (13), the numerator denotes the maximum bandwidth capacity in the network, and the denominator represents the remaining bandwidth on link uv. Eqs. (14)-(15) are calculated similarly as Eq. (13), and their value ranges are uniform between $(1, +\infty)$. It is obvious that the resource costs can be equally transformed to increasing and convex functions by replacing resource remaining rates with resource utilization rates. The benefit of adopting increasing and convex functions to set resource costs has been discussed in literatures [34], [35]. In Eqs. (13)-(15), $c_{i,uv}^{bw}, c_{i,u}^{mem}$ and $c_{i,m}^{cpu}$ increase slowly, if the network load is low. And they increase quickly, if the resource consumptions are approximate to their resource capacities. Therefore, $c_{i,uv}^{bw}, c_{i,u}^{mem}$ and $c_{i,m}^{cpu}$ can be used to indicate the resource bottlenecks in the network. It is noted that, since the memory capacities of MDC nodes are regarded as infinite, we define $c_{i,u}^{mem} \equiv 0, r_{i,u}^{mem} \equiv 1, \forall u \in \mathcal{V}_{mdc}$.

Since the resource cost functions defined in Eqs. (13)-(15) can reflect the load status on links, nodes and VNF instances, we use the sum cost to indicate the load status of a path. The cost of a path to embed SFC request i is defined as:

$$\begin{aligned}\mathbb{R} = \sum_{uv\in\mathcal{L}} \sum_{\bar{u}\bar{v}\in\mathcal{L}_i} c_{i,uv}^{bw} z_{i,uv}^{\bar{u}\bar{v}} + \sum_{u\in\mathcal{V}} \sum_{\bar{u}\bar{v}\in\mathcal{L}_i} c_{i,u}^{mem} z_{i,u}^{\bar{u}\bar{v}} \\ + \sum_{m\in\mathcal{M}} \sum_{\bar{u}\in\mathcal{V}_i} c_{i,m}^{cpu} x_{i,m}^{\bar{u}}.\end{aligned} \tag{16}$$

According to Eq. (16), if the cost of a path is small, we can infer that there exist no bottleneck nodes, links or VNF instances on this path. On the contrary, if the cost of the path is very big, we can infer that there exist some bottleneck nodes, links or VNF instances, and we have to find another path with smaller cost to steer the traffic of SFC requests.

Furthermore, the VNF placement cost to embed SFC request i is computed as:

$$\mathbb{D} = \sum_{m\in\mathcal{M}} \sum_{k\in\mathcal{K}} c_k^{vnf} q_k^m max\{w_{i,m} - \widehat{w}_{i,m}, 0\}. \tag{17}$$

We use c_k^{vnf} to represent the placement cost of VNF_k. The binary variable $\widehat{w}_{i,m}$ indicates whether the VNF instance $m \in \mathcal{M}$ is placed before embedding SFC request i. $\widehat{w}_{i,m}$ equals 1, if $m \in \mathcal{M}$ is placed before embedding SFC request i, and 0 otherwise. Additionally, in Eq. (17), $max\{w_{i,m} - \widehat{w}_{i,m}, 0\}$ indicates whether VNF instance m needs to be placed after embedding SFC request i. If m needs to be placed after embedding SFC request i, $max\{w_{i,m} - \widehat{w}_{i,m}, 0\}$ equals 1, and 0 otherwise.

In the paper, the embedding cost is defined as the sum of resource cost and VNF placement cost. If there are sufficient resources in the network, the resource costs are much smaller than the VNF placement cost. Then, SFC requests are embedded by reusing the placed VNF instances. If the network is short of resources, the resource cost will rise rapidly. And once the sum of resource cost is much bigger than the VNF placement cost, placing new VNF instances can get more benefit than reusing the placed ones. Then many new VNF instances can be placed to balance network load and reduce resource bottlenecks, which can keep the network with high performance. Therefore, our objective is to embed an SFC request with the minimum embedding cost as:

$$\textit{Minimize } (\mathbb{D} + \mathbb{R}). \tag{18}$$

In the model, with the increasement of network load, there are more and more VNF instances placed in the network to provide sufficient resources and achieve load balancing. When the network load decreases, placing overmany VNF instances will incur high OPEX/CAPEX due to long VNF running time and high VNF placement cost. Therefore, when embedding SFC requests, how to efficiently place and release VNF instances should be focused on. Instead of finding exact numerical solutions by analytical method which suffers from combinatorial complexity and is extremely time-consuming, we propose a novel SFC-MAP algorithm which can optimize the placement of VNF instances and minimize the embedding cost for SFC requests. Moreover, in order to adapt to the variation of network load, we also propose the VNF-DRA algorithm to release redundant VNF instances for the reduction of the total VNF running time.

V. SFC-MAP Algorithm

SFC-MAP is designed to obtain the optimal chaining solution of Eq. (18). First, we construct a multi-layer graph including the VNF placement cost and resource costs of links, nodes and VNF instances. Then, based on these costs defined in multi-layer graph, we iterate the shortest path algorithm to obtain the optimal chaining solution for an SFC request. Details on SFC-MAP are provided in the next subsections.

A. Constructing Multi-layer Graph

A multi-layer graph consists of several copies of physical graph and the adjacent layers are connected by inter-layer links and inter-layer nodes. For SFC request i shown in Fig. 2, the multi-layer graph includes $l + 1$ $(l = |\mathcal{Q}_i|)$ copies of original physical graph. The subscript of each node indicates the layer number. All the links in the same layer are named as intra-layer links. The links and nodes, that are used to connect the MDC nodes in adjacent layers, are named as inter-layer links and inter-layer nodes. In the multi-layer graph, the inter-layer nodes are chosen and arranged according to the order constraint of VNF instances that an SFC request needs to concatenate. For example, between the j^{th} and $(j+1)^{th}$ layers of the multi-layer

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2880992, IEEE Transactions on Parallel and Distributed Systems

graph for SFC request i, all the inter-layer nodes indicate the VNF instances with the same VNF type of $\mathcal{Q}_i^j$, $j = 1, 2, ..., l$. The costs of the intra-layer links, switch nodes and MDC nodes in the multi-layer graph equal their costs in physical graph which are computed based on Eqs. (13)-(14). The costs of inter-layer nodes include the VNF placement cost $c_k^{vnf}, k \in \mathcal{K}$ and the CPU cost calculated in Eq. (15). In addition, the costs of inter-layer links that connect the inter-layer nodes and MDC nodes in the multi-layer graph are set as zero.

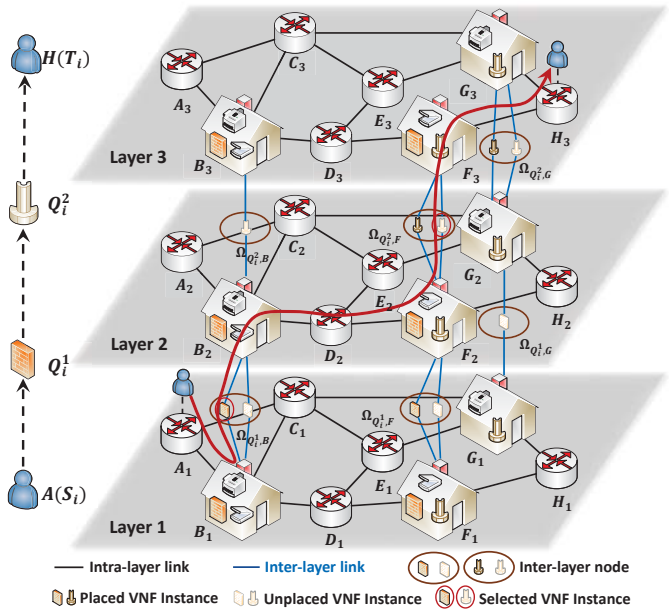


Fig. 2. Embedding an SFC request in multi-layer graph.

In the paper, we define $\Omega_{\mathcal{Q}_i^j,u} \subset \mathcal{M}$, $j = 1, 2, ..., l$ as a set of VNF instances that are chosen from MDC node $u \in \mathcal{V}_{mdc}$ and belong to the same VNF type of $\mathcal{Q}_i^j$. In $\Omega_{\mathcal{Q}_i^j,u}$, there exist at most two elements. One element is a placed VNF instance with the minimum CPU cost in MDC node $u \in \mathcal{V}_{mdc}$ and the other one is an unplaced VNF instance. Both of the elements in $\Omega_{\mathcal{Q}_i^j,u}$ belong to the same VNF type of $\mathcal{Q}_i^j$. We use a dark-colored image to denote the placed VNF instance and a light-colored image to present the unplaced VNF instance. All these VNF instances in $\Omega_{\mathcal{Q}_i^j,u}$, $u \in \mathcal{V}_{mdc}$, $j = 1, 2, ..., l$ are inter-layer nodes. It is noted that, if there is no placed VNF instances in MDC node $u \in \mathcal{V}_{mdc}$, only an unplaced VNF instance is included in $\Omega_{\mathcal{Q}_i^j,u}$. And according to Eq. (4), if the number of placed VNF instances in MDC node u is maximized, there is no available resource to place a new VNF instance. Then, only a placed VNF instance, that belongs to the same type of $\mathcal{Q}_i^j$ and has the minimum CPU cost in MDC node u, is included in $\Omega_{\mathcal{Q}_i^j,u}$.

In the multi-layer graph, the MDC nodes between two adjacent layers which represent the same MDC node in physical graph are connected with inter-layer links and inter-layer nodes. For SFC request i in Fig. 2, as $\mathcal{Q}_i^1$ is the first requested VNF ($\mathcal{Q}_i^1$ = VNF_a) and the VNF instances with the same type of $\mathcal{Q}_i^1$ can be placed in MDC nodes B, F and G, we compute the $\Omega_{\mathcal{Q}_i^1,B}$, $\Omega_{\mathcal{Q}_i^1,F}$ and $\Omega_{\mathcal{Q}_i^1,G}$ for these three MDC nodes. Then, in the 1^{st} and 2^{nd} layers of the multi-layer graph, B_1 and B_2 are connected with each node in $\Omega_{\mathcal{Q}_i^1,B}$. We also connect F_1 and F_2 and G_1 and G_2 with each node in $\Omega_{\mathcal{Q}_i^1,F}$ and $\Omega_{\mathcal{Q}_i^1,G}$, respectively. Similarly, for the second requested VNF ($\mathcal{Q}_i^2$ = VNF_b), we do the same operation for MDC nodes B, F and G between the 2^{nd} and 3^{rd} layers.

In the multi-layer graph, the ingress node is set in the 1^{st} layer and the egress node is set in the $(l+1)^{th}$ layer. Because the adjacent layers in the multi-layer graph are connected with the VNF instances arranged in predefined order, each path from the ingress node to the egress node can concatenate VNF instances to satisfy the order constraint of this SFC request. As stated before, each layer of the multi-layer graph is consistent with the original physical graph, and the costs in the multi-layer graph equal the costs in physical network. Therefore, we can equivalently solve Eq. (18) by solving the optimal chaining solution in multi-layer graph.

For example, in Fig. 2, SFC request i starts from the ingress node A and ends at the egress node H traversing VNF_a and VNF_b in sequence. In multi-layer graph, the ingress and egress nodes of SFC request i are A_1 and H_3. Supposing that the optimal chaining solution to embed SFC request i in multi-layer graph is: $A_1 \rightarrow B_1 \rightarrow \Omega^1_{\mathcal{Q}_i^1,B} \rightarrow B_2 \rightarrow D_2 \rightarrow F_2 \rightarrow \Omega^2_{\mathcal{Q}_i^2,F} \rightarrow F_3 \rightarrow H_3$, where $\Omega^1_{\mathcal{Q}_i^1,B}$ represents the first VNF instance of $\Omega_{\mathcal{Q}_i^1,B}$ and $\Omega^2_{\mathcal{Q}_i^2,F}$ stands for the second VNF instance of $\Omega_{\mathcal{Q}_i^2,F}$. Observing that $\Omega^1_{\mathcal{Q}_i^1,B}$ is placed, but $\Omega^2_{\mathcal{Q}_i^2,F}$ is not placed. Then, this SFC request is served by placing a new VNF instance with the same VNF type of $\mathcal{Q}_i^2$ in MDC node F. Therefore, the optimal chaining solution in original physical graph is $A \rightarrow B \rightarrow \Omega^1_{\mathcal{Q}_i^1,B} \rightarrow B \rightarrow D \rightarrow F \rightarrow \Omega^2_{\mathcal{Q}_i^2,F} \rightarrow F \rightarrow H$.

B. Embedding SFC Requests in Multi-layer Graph

SFC-MAP is executed based on multi-layer graph. In multi-layer graph, we define the ingress node in the 1^{st} layer and the egress node in the $(l+1)^{th}$ layer. Then, pruning the nodes, links and VNF instances with insufficient resources, we use the shortest path algorithm to solve the chaining solution with the minimum embedding cost in multi-layer graph. According to the relationship between the multi-layer graph and the physical graph, we transform the optimal solution in multi-layer graph to original physical graph to obtain the final chaining solution of Eq. (18).

As the chaining solution obtained in the multi-layer graph could violate the resource constraints in Eqs. (1)-(5), in our work, a penalty factor λ is introduced in to solve this problem. In multi-layer graph, though the sum cost of a chaining solution is the minimum, the resource constraints in Eqs. (1)-(5) can be violated. Given the fact that the chaining solution is solved based on costs, we can avoid unfeasible solutions by changing costs. We use a penalty factor λ, which is greater than 1, to perform this process. If a chaining solution derived from the multi-layer graph is unfeasible, the resource costs and VNF placement costs on the corresponding links, nodes and VNF instances which violate constraints will be multiplied by λ. Then, the costs on these links, nodes and VNF instances

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2880992, IEEE Transactions on Parallel and Distributed Systems

increase, which decreases the possibilities of being selected in the next computation. It is noted that, since the introduction of the penalty factor λ changes costs, SFC-MAP can only ensure to find optimal solutions, if they can be obtained before λ taking effect, otherwise they could be missed.

The pseudocode of SFC-MAP is shown in **Algorithm 1**. The iteration times γ is initialized in line 1. Line 2 is to prune the links, nodes and VNF instances that cannot be used to serve the SFC request. In lines 3-7, we calculate the costs of bandwidth, memory and CPU. Line 8 is to find all the placed VNF instances in inter-layer nodes. The multi-layer graph is constructed in line 9. Lines 10-27 solve the chaining solution iteratively with the minimum embedding cost for this SFC request in multi-layer graph. The shortest path algorithm is executed to find the chaining solution in multi-layer graph in line 11. In lines 12-14, if there exists candidate chaining solution in S_{multi}, we transform this solution to physical graph and check it with the constraints in Eqs. (1)-(12). If S_{phy} satisfies all the constraints, the final chaining solution is obtained and we place the VNF instances in the corresponding MDC nodes according to S_{phy} in lines 15-17. Then, the network status is updated after the SFC request being embedded in lines 18-20. Otherwise, all the links, nodes, and VNF instances that violate constraints in Δ are punished with the penalty factor λ in lines 21-23. Then the iteration times γ is updated and checked whether it exceeds the maximum iteration times Γ in lines 24 and 10, respectively. We go on iterating SFC-MAP to search for the chaining solution with the minimum embedding cost, until the feasible chaining solution is found or γ exceeds the maximum iteration times Γ.

Algorithm 1: SFC-MAP

1: Initialize $\gamma = 1$;
2: $\{\mathcal{V}, \mathcal{L}, \mathcal{M}\} \leftarrow$ Pruning the nodes, links and VNF instances with less resources than the demand of SFC request i;
3: **for** *each* $u \in \mathcal{V}, uv \in \mathcal{L}, m \in \mathcal{M}$ **do**
4: $\quad c_{i,uv}^{bw} \leftarrow$ Calculate the bandwidth cost of uv;
5: $\quad c_{i,u}^{mem} \leftarrow$ Calculate the memory cost of u;
6: $\quad c_{i,m}^{cpu} \leftarrow$ Calculate the CPU cost of m;
7: **end**
8: $\Lambda \leftarrow$ Find the placed VNF instances with the minimum CPU costs for all the VNF types in each MDC node $u \in \mathcal{V}_{mdc}$;
9: $G_{multi} \leftarrow$ Construct multi-layer graph according to $c_{i,uv}^{bw}$, $c_{i,u}^{mem}$, $c_{i,m}^{cpu}$ and c_k^{vnf}, $\forall uv \in \mathcal{L}, \forall u \in \mathcal{V}, \forall m \in \Lambda, \forall k \in \mathcal{K}$;
10: **while** $\gamma \leq \Gamma$ **do**
11: $\quad S_{multi} \leftarrow$ Execute the shortest path algorithm in G_{multi};
12: $\quad$ **if** $S_{multi} \sim= \phi$ **then**
13: $\quad\quad S_{phy} \leftarrow$ Map S_{multi} from G_{multi} to G;
14: $\quad\quad Bool \leftarrow$ Check whether S_{phy} satisfies all the constraints in Eqs.(1)-(12);
15: $\quad\quad$ **if** *Bool* $==$ *true* **then**
16: $\quad\quad\quad \Theta \leftarrow$ Get all the VNF instances waiting to be placed according to S_{phy};
17: $\quad\quad\quad$ Place all the VNF instances of Θ in the corresponding MDC nodes according to S_{phy};
18: $\quad\quad\quad$ Embed the SFC request with S_{phy};
19: $\quad\quad\quad NewNetworkStatus \leftarrow$ Update the network;
20: $\quad\quad\quad$ **return** $NewNetworkStatus$;
21: $\quad\quad$ **else**
22: $\quad\quad\quad \Delta \leftarrow$ Get all the links, nodes and VNF instances that violate constraints from S_{phy};
23: $\quad\quad\quad$ Update the costs for each element of Δ with a penalty factor λ;
24: $\quad\quad\quad \gamma = \gamma + 1$;
25: $\quad\quad$ **end**
26: $\quad$ **end**
27: **end**
28: **return** Failed;

C. Complexity Analysis

In SFC-MAP, the complexity of calculating the costs of links, nodes, and VNF instances is no more than $O(|\mathcal{V}| + |\mathcal{L}| + |\mathcal{M}|)$. The construction of a multi-layer graph needs to copy original physical graph and set inter-layer links and inter-layer nodes, which results in at most $O(l(|\mathcal{V}| + |\mathcal{L}|) + |\mathcal{M}|)$ computations. As for multi-layer graph, because it consists of $(l+1)$ copies of physical graph, the numbers of nodes and links in multi-layer graph are no more than $(l+1)|\mathcal{V}| + 2l|\mathcal{V}_{mdc}|$ and $(l+1)|\mathcal{L}| + 4l|\mathcal{V}_{mdc}|$. Given the fact that $\mathcal{V}_{mdc} \subset \mathcal{V}$ and executing the shortest path algorithm (*e.g.,* Dijkstra) in physical graph $G = (\mathcal{V}, \mathcal{L})$ runs in $O(|\mathcal{L}| + |\mathcal{V}|log|\mathcal{V}|)$, the total complexity of SFC-MAP with the maximum iteration times Γ is $O(|\mathcal{M}| + \Gamma l(|\mathcal{L}| + |\mathcal{V}|logl|\mathcal{V}|))$.

VI. Releasing VNF Instances According to the Variation of Network Load

In the network, the number of placed VNF instances needs to be optimized to adapt to the variation of network load. The SFC-MAP solves the placement of VNF instances and can concatenate them with the order constraints for SFC requests, however, it does not consider how to release overmany placed VNF instances, and this defect will lead to low VNF utilization rates and high OPEX/CAPEX when network load goes down. In order to tackle this issue, we propose the VNF-DRA algorithm to efficiently release redundant VNF instances according to the variation of network load.

A. VNF-DRA Algorithm

With the variation of network load, VNF-DRA is proposed to reduce the total VNF running time by releasing redundant VNF instances with low utilization rates. VNF-DRA runs every period T. We define $r_m^{cpu}(t)$ as the CPU remaining rate of $m \in \mathcal{M}$ at time t, and $f(t)$ represents the threshold of VNF utilization rate at time t. When executing VNF-DRA, first, $r_m^{cpu}(t)$ should be calculated for each VNF instance $m \in \mathcal{M}$. Then, VNF-DRA redirects the SFC requests in the VNF instances with the utilization rate $(1 - r_m^{cpu}(t)) \leq f(t), m \in \mathcal{M}$. Next, the empty VNF instances are released for the reduction of running time.

Since the SFC requests with short lifetime will come and go as they run, it is reasonable to redirect the SFC requests with long lifetime during their serving windows or by ample warning and prior planning [36]. And we can infer that an SFC request is a long lifetime one, if it has run for a long time. In

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2880992, IEEE Transactions on Parallel and Distributed Systems

the paper, we use $\varphi_i(t)$ to represent the remaining lifetime of SFC request i at time t. The lifetime threshold of SFC request is defined as ρ, which is used to differentiate the long lifetime and short lifetime SFC requests. All the notations used in this section are listed in TABLE 2.

TABLE 2
NOTATIONS IN VNF-DRA

Notations	Description
T	Execution period.
ρ	Lifetime threshold of SFC request.
ε	Fluctuation threshold of network throughput.
$f(t)$	Threshold of VNF utilization rate at time t.
f_b, f_s	Values of $f(t)$.
$r_m^{cpu}(t)$	CPU remaining rate of $m \in \mathcal{M}$ at time t.
$\phi(t)$	Network throughput at time t.
$\Phi(t)$	Mean network throughput during $(t-T, t]$.
$\varphi_i(t)$	Remaining lifetime of SFC request i at time t.
$\sigma(t)$	Fluctuation of network throughput during $(t-T, t]$.

In VNF-DRA, all the VNF instances with the utilization rate $(1 - r_m^{cpu}(t)) \leq f(t), m \in \mathcal{M}$ are selected and sorted in ascending order. After that, for each of these selected VNF instances, all the SFC requests served in it with the remaining lifetime $\varphi_i(t) > \rho$ are recognized as long lifetime ones and redirected to the VNF instances with the utilization rate $(1 - r_m^{cpu}(t)) > f(t), m \in \mathcal{M}$ using the SFC-MAP algorithm. It is noted that, when redirecting SFC requests with SFC-MAP algorithm, only the VNF instances with the utilization rate $(1 - r_m^{cpu}(t)) > f(t), m \in \mathcal{M}$ are considered as the inter-layer nodes in multi-layer graph. For short lifetime SFC requests served in these selected VNF instances, we do no operations and only wait them to expire. Moreover, new arrival SFC requests are avoided to be steered through the VNF instances waiting to be released (do not set the VNF instances waiting to be released as inter-layer nodes, when executing SFC-MAP). We will release a VNF instance, if there are no SFC requests served in it.

As the network load changes over time and $f(t)$ decides the number of VNF instances that will be checked and released, the value of $f(t)$ should be set according to the variation of network load. For example, when network load increases, more VNF instances should be placed. Therefore, it is applicable to set $f(t)$ as a small value, which can keep the network with sufficient resources to cope with the increasing number of SFC requests. On the contrary, with decreasing network load, there are many VNF instances with low utilization rate, which leads to high OPEX/CAPEX due to long VNF running time. Thus, it is beneficial to set $f(t)$ as a big value, which can timely release the redundant VNF instances and reduce the total VNF running time. Additionally, when the network load keeps stable, the placed VNF instances can already provide enough resource to cope with SFC requests. Then, decreasing the value of $f(t)$ can maintain the network stable and reduce the OPEX/CAPEX resulting from the adjustment of VNF instances.

Before presenting the definition of $f(t)$, we define some symbols to describe the variation of network load. In the paper, $\phi(t)$ denotes the network throughput at time t. The parameter $\Phi(t)$ stands for the mean network throughput during $(t-T, t]$, and it is calculated below:

$$\Phi(t) = \frac{1}{T} \int_{t-T}^{t} \phi(t) dt. \tag{19}$$

In addition, we define $\sigma(t)$ as the fluctuation of network throughput during $(t-T, t]$, and ε as the fluctuation threshold of network throughput. Both of the parameters are used to indicate the variation of network load. Then, $\sigma(t)$ can be computed as:

$$\sigma(t) = \frac{1}{T} \int_{t-T}^{t} |\phi(t) - \Phi(t)| dt, \tag{20}$$

where $|\phi(t) - \Phi(t)|$ represents the fluctuation around the mean network throughput $\Phi(t)$ at time t. In geo-distributed cloud system, the network throughput could not change sharply during a small time interval. Therefore, when network load keeps stable, the network throughput during T period will fluctuate around the mean value, which leads to $\sigma(t) \leq \varepsilon$. And when network load rapidly increases or decreases during T period, the network throughput will deviate the mean value, which leads to $\sigma(t) > \varepsilon$.

Given the analysis above, $f(t)$ is defined as a piecewise function which is calculated according to the variation of network load as:

$$f(t) = \begin{cases} f_b, & \Phi(t-T) > \Phi(t),\ \sigma(t) > \varepsilon, \\ f_s, & \text{otherwise.} \end{cases} \tag{21}$$

In Eq. (21), the network load is indicated to be decreasing, if $\Phi(t-T) > \Phi(t)$ and $\sigma(t) > \varepsilon$. Then, we increase the number of VNF instances that will be checked by setting $f(t)$ with a big value f_b, which is helpful to timely release redundant VNF instances for the reduction of the total VNF running time. Otherwise, the network load is indicated to be increasing or stable, and we decrease the number of VNF instances that will be checked by setting $f(t)$ with a small value f_s, which is beneficial to provide sufficient resources for SFC requests or keep the network status stable.

The pseudocode of VNF-DRA and the release of VNF instances are described in **Algorithm 2** and **Algorithm 3**, respectively. In **Algorithm 2**, line 1 is to calculate $\Phi(t)$ and $\sigma(t)$ according to Eqs. (19)-(20). We update $f(t)$ based on Eq. (21) in line 2 of **Algorithm 2**. We calculate the VNF utilization rate for each VNF instance in lines 3-5 of **Algorithm 2**. In line 6 of **Algorithm 2**, all the VNF instances with the utilization rate $(1 - r_m^{cpu}(t)) \leq f(t), m \in \mathcal{M}$ are sorted and put into Π. Lines 7-14 of **Algorithm 2** aim to redirect the SFC requests with $\varphi_i(t) > \rho$ to the VNF instances with utilization rate $(1 - r_m^{cpu}(t)) > f(t), m \in \mathcal{M}$. In **Algorithm 3**, a VNF instance will be released, if it is empty in line 3.

B. Complexity Analysis

In VNF-DRA, searching for the VNF instances with the utilization rate $(1 - r_m^{cpu}(t)) \leq f(t), m \in \mathcal{M}$ runs in at most $O(|\mathcal{M}|)$ computations. The parameter $\Pi \subset \mathcal{M}$ represents the set of all these selected VNF instances with the utilization rate $(1 - r_m^{cpu}(t)) \leq f(t), m \in \mathcal{M}$. The number of long lifetime SFC requests in a VNF instance is assumed at most I. As VNF-DRA needs to redirect each long lifetime SFC request

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2880992, IEEE Transactions on Parallel and Distributed Systems

Algorithm 2: VNF-DRA

Calculate $\Phi(t)$ and $\sigma(t)$;
Update $f(t)$;
for *each VNF instance* $m \in \mathcal{M}$ **do**
  $(1 - r_m^{cpu}(t)) \leftarrow$ Calculate the VNF utilization rate of m;
end
$\Pi \leftarrow$ Find all the VNF instances with the utilization rate $(1 - r_m^{cpu}(t)) \leq f(t), m \in \mathcal{M}$ and sort them in ascending order;
while $\Pi(1) \sim= \phi$ **do**
  $Bool \leftarrow$ Check whether all the SFC requests with $\varphi_i(t) > \rho$ can be redirected;
  if $Bool == true$ **then**
    Redirect the SFC requests with $\varphi_i(t) > \rho$ to the VNF instances with the utilization rate $(1 - r_m^{cpu}(t)) > f(t), m \in \mathcal{M} \leftarrow$ **SFC-MAP**;
    $NewNetworkStatus \leftarrow$ Update the network;
  end
  $\Pi \leftarrow \Pi \setminus \Pi(1)$;
end
return $NewNetworkStatus$;

Algorithm 3: Release VNF Instances

for *each VNF instance* $m \in \mathcal{M}$ *waiting to be released* **do**
  if *there are no SFC requests served in VNF instance* m **then**
    Release VNF instance m;
  end
end

with SFC-MAP algorithm, the computation complexity to run VNF-DRA is $O(I|\Pi|[|\mathcal{M}| + \Gamma l(|\mathcal{L}| + |\mathcal{V}|logl|\mathcal{V}|)])$.

VII. Performance Evaluation

In this section, we demonstrate the performance evaluation of our proposed algorithms. First, we discuss the simulation setup used to evaluate the algorithms in our work. Then, based on this simulation, we compare our algorithms with other ones in existing literatures and evaluate their performance in different scenarios.

A. Simulation Setup

The simulation is implemented on Matlab, a wildly used software in modeling and analysis, and conducted on a computer with Intel(R) Core(TM) i7-4790 CPU 3.60 GHz and 32 GB RAM. The reason why we make this choice is because SDN technology can achieve centralized management in cloud computing. In SDN, the controller is mainly in charge of monitoring the resource of network and VNF instances, analyzing collected information and making flexible solutions and holistic management in large scale complex networks [1], [6]. Therefore, it is reasonable to run algorithms on SDN controllers to optimize network performance based on collected information without considering the operations and signaling interactions in real network. Further, with reasonable parameter settings, the numerical simulation can approximate the emulation in real SDN/NFV-enabled networks.

Fig. 3. CORONET CONUS Topology (75 nodes and 99 links).

The network graph we use is a US carrier network topology named CORONET CONUS Topology [37] which consists of 75 nodes and 99 links, and it is shown in Fig. 3. In the topology, we select MDC nodes according to node degree. We sort all the nodes based on their node degrees in descending order, then select the first 12 nodes as MDC nodes. There are 20 types of VNFs that can be placed in MDC nodes. Considering the location constraints for the placement of VNF instances and the number of licenses that operators own for VNFs [38], we assume that each MDC node can only place 10 different types of VNFs. The maximum number of VNF instances permitted to be placed per MDC node is set as 20, and the VNF placement cost $c_k^{vnf}, \forall k \in \mathcal{K}$ is set as 50. The bandwidth capacity per link is 1000 Mbps. The memory capacity per switch node and CPU capacity per VNF instance are set as 1000 MB and 100 MIPS, respectively. All the propagation delay, transmission delay, queuing delay and processing delay are considered in the simulation and computed based on Eqs. (22)-(24) [39] as:

$$d_{i,uv} = d_{uv}^{prop} + d_{uv}^{tx} + \frac{1 - r_{i,uv}^{bw}}{r_{i,uv}^{bw}} d_{uv}^{tx}, \forall uv \in \mathcal{L}, \tag{22}$$

$$d_{i,u} = \frac{1 - r_{i,u}^{mem}}{r_{i,u}^{mem}} t_u^{proc}, \forall u \in \mathcal{V}, \tag{23}$$

$$d_{i,m} = \frac{1 - r_{i,m}^{cpu}}{r_{i,m}^{cpu}} t_m^{proc}, \forall m \in \mathcal{M}. \tag{24}$$

In Eq. (22), the first part d_{uv}^{prop} represents the propagation delay which is computed by the ratio of the length of link uv to the propagation speed of signals in that medium. The second part d_{uv}^{tx} denotes the transmission delay and it is computed by dividing the bandwidth capacity of link uv with the packet size. The third part means the queuing delay, and it is related to the load and transmission delay. Eqs. (23)-(24) computes the processing delay on node u and VNF instance m. The parameters t_u^{proc} and t_m^{proc} indicate the per-packet processing delay on node u and VNF instance m, and they are set as 10 μs and 1 ms, respectively [40].

In the simulation, the ingress and egress nodes and the requested VNFs of SFC requests are all set randomly. The arrival rate of SFC requests abides by the rule shown in Fig. 4. For each SFC request, the bandwidth, memory and CPU demands are set as numbers distributed randomly between (0,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2880992, IEEE Transactions on Parallel and Distributed Systems

TABLE 3
SIMULATION PARAMETER SETTINGS

Description		Value
Network topology		CORONET CONUS Topology
Number of MDC nodes		12
Number of VNF types		20
Number of permitted VNF types per MDC node		10
Number of permitted VNF instances per MDC node		20
Number of VNF requests per SFC request		3
Lifetime of each SFC request		$X \sim E(\frac{1}{1000})$
Parameters	**Description**	**Value**
C_{uv}^{bw}	Bandwidth capacity of link uv	1000 Mbps
C_{u}^{mem}	Memory capacity of switch node u	1000 MB
C_{m}^{cpu}	CPU capacity of VNF instance m	100 MIPS
Ψ_{i}^{bw}	Bandwidth demand	(0, 10] Mbps
Ψ_{i}^{mem}	Memory demand	(0, 10] MB
Ψ_{i}^{cpu}	CPU demand	(0, 10] MIPS
Ψ_{i}^{td}	Maximum tolerated delay	[50, 100] ms
d_{uv}^{tx}	Transmission delay on link uv	1.5 μs
d_{uv}^{prop}	Propagation delay on link uv	defined by topology
t_{m}^{proc}	Per-packet processing delay on instance m	1 ms
t_{u}^{proc}	Per-packet processing delay on node u	10 μs
c_{k}^{vnf}	Placement cost of *VNF*$_k$	50
λ	Penalty factor	1.5
T	Execution period of VNF-DRA	500 time units
f_b, f_s	Values of $f(t)$	50%, 20%
ε	Fluctuation threshold of network throughput	50 Mbps
ρ	Lifetime threshold of SFC request	100 time units

10]. The number of VNF requests per SFC request is set as 3 [41]. The maximum tolerated delay for each SFC request is between 50 and 100 ms [42]. During the embedding process, all the constraints in Eqs. (1)-(12) must be satisfied, otherwise an SFC request will fail to be served. Furthermore, in order to simulate dynamic load environment, we set a lifetime for each SFC request. The lifetime of each SFC request obeys the exponential distribution with an average of 1000 time units. Within the lifetime, the system needs to ensure that there are enough resources for the served SFC requests. And an SFC request will release the occupied resources when its lifetime is expired.

As for SFC-MAP, the penalty factor λ is set as 1.5. In VNF-DRA, the execution period T is set as 500 time units. The lifetime threshold of SFC request ρ, which is used to differentiate long and short lifetime SFC requests, is set as 100 time units. We judge whether the network load is stable, increasing or decreasing based on the fluctuation threshold of network throughput ε, and it is set as 50 Mbps. The values of f_b and f_s in the piecewise function of $f(t)$, which is used to check and release VNF instances, are set as 50% and 20%, respectively. Additionally, each group of results is tested 20 times, and we evaluate the performance within 15000 time units. All the parameter settings in this part are shown in TABLE 3.

B. Introduction of Compared Algorithms

The ProvisionTraffic algorithm [12] and Eigendecomposition [16] algorithm have been used as the compared algorithms in the simulations. Before introducing the evaluation results, we give a brief description to these compared algorithms.

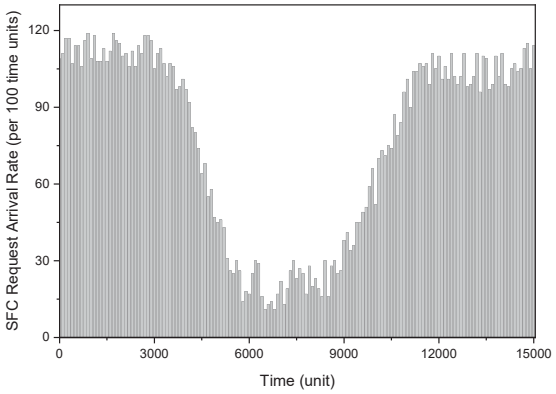


Fig. 4. SFC request arrival rate over time.

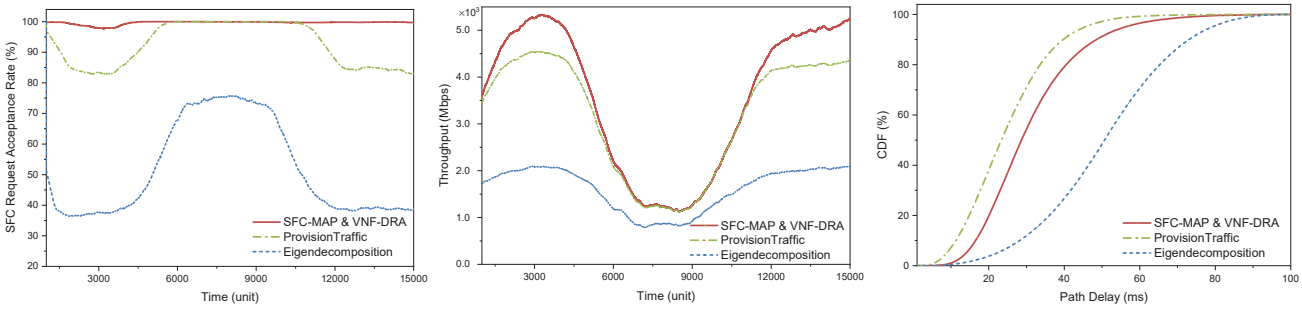
- ProvisionTraffic: ProvisionTraffic places VNF instances and steer the traffic of SFC requests based on a multi-stage graph. For an SFC request, all the necessary placed VNF instances or necessary pseudo-VNF instances are included in multi-stage graph, and they are arranged and connected with the order constraint of this SFC request. The link cost in the multi-stage graph is computed based on VNF deployment cost, energy cost of servers, cost of forwarding traffic, penalty for SLO violation and resource fragmentation. Then, ProvisionTraffic runs the Viterbi algorithm [24] to solve the embedding solutions with the minimum OPEX/CAPEX.
- Eigendecomposition: Eigendecomposition extends and adapts Umeyama's eigendecomposition approach [25] to map VNF-FG to physical graph with the optimal matching. First, Eigendecomposition produces an adjacent matrix for the network, and the weight for each element is computed by running the widest-shortest path routing algorithm in network topology. And an adjacent matrix is also produced for each SFC request based on its resource demand. Then, the adjacent matrix of SFC request is extended to be with the same size of the network's. Next, Eigendecomposition computes the eigenvector matrixes for both of the adjacent matrixes. After that, the conjugate matrixes of the two eigenvector matrixes are computed and multiplied together. Finally, the locations with the maximum value in each row of the product are used as the solution to place VNF instances and steer the traffic of the SFC request.

C. Simulation Results

1) ***Evaluation of SFC-MAP & VNF-DRA with Compared Algorithms****:* In Fig. 5-6, we evaluate the performances of SFC-MAP & VNF-DRA, ProvisionTraffic and Eigendecomposition with the SFC request arrival rate shown in Fig. 4.

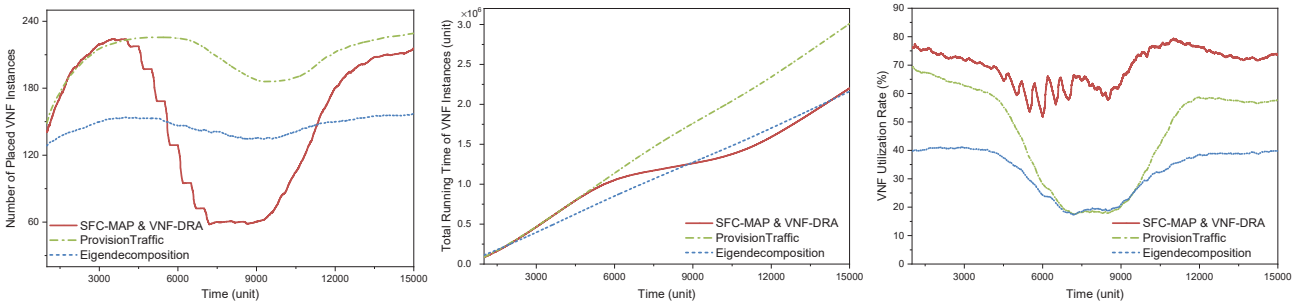
Fig. 5(a) shows the evaluations of SFC request acceptance rate of these three algorithms. When network load is heavy between 1500-3500 and 12000-15000 time units, our algorithms can approximately obtain 100% SFC request acceptance rate comparing with other algorithms. This is because, in SFC-MAP, the remaining resources are considered to set the costs on nodes, links and VNF instances, which is used to achieve load balancing. Therefore, SFC-MAP can reduce

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2880992, IEEE Transactions on Parallel and Distributed Systems



(a) SFC request acceptance rate over time. (b) Network throughput over time. (c) CDF of end-to-end delay.

Fig. 5. The comparison of SFC request acceptance rate, network throughput and the CDF of end-to-end delay.



(a) The number of VNF instances over time. (b) Total running time of VNF instances over time. (c) Mean VNF utilization rate over time.

Fig. 6. The comparison of the number of placed VNF instances, the total running time of VNF instances and mean VNF utilization rate over time.

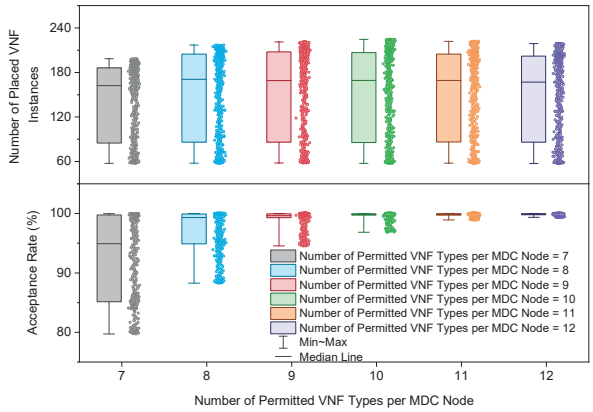
resource bottlenecks and enhance SFC request acceptance rate. Compared with our algorithms, ProvisionTraffic minimizes a joint costs including VNF placement cost, traffic forwarding, energy consumption, SLA penalty and resource fragmentation. However, as ProvisionTraffic pays no attention to the remaining resources in the network, the bottleneck cannot be avoided when embedding SFC requests. In addition, though it is beneficial to take full use of network resources by minimizing the resource fragmentation, the bottleneck and congestion are further aggravated because of the resource exhaustion. As for Eigendecomposition, it cannot ensure to obtain the optimal mapping of an SFC request, and the widest-shortest path routing algorithm results in the chaining solutions with longer paths. Hence, Eigendecomposition consumes more resources, which leads to the worst performance in the simulation.

In Fig. 5(b), we evaluate the network throughput of the three algorithms. Since SFC acceptance rate of SFC-MAP & VNF-DRA shown in Fig. 5(a) is the highest, they obtains the best performance of network throughput as well. With heavy network load between 1500-3500 and 12000-15000 time units, the network throughput of SFC-MAP & VNF-DRA is 1000 Mbps higher than the ProvisionTraffic's and 3500 Mbps higher than the Eigendecomposition's. With light network load between 6000-9000 time units, both the SFC-MAP & VNF-DRA and ProvisionTraffic obtain the highest network throughput, which is about 400 Mbps higher than the Eigendecomposition's.

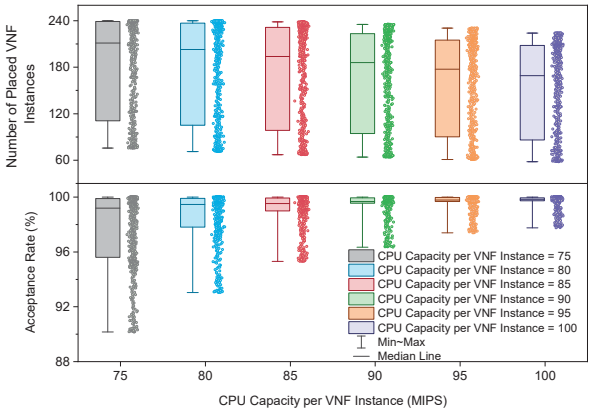
Fig. 5(c) shows the distribution of end-to-end delay with these three algorithms. ProvisionTraffic gets the best result in this simulation. This is because the end-to-end delay is considered in its optimization objective, when solving chaining solutions for SFC requests. In SFC-MAP & VNF-DRA, we pay no attention to minimize the end-to-end delay. Moreover, we also have to make a tradeoff between the path selection and load balancing, which aims to avoid bottlenecks and reduce network congestion. Therefore, SFC-MAP & VNF-DRA leads to longer end-to-end delay than the ProvisionTraffic's. In Eigendecomposition, the end-to-end delay is not considered as well, and the widest-shortest path routing algorithm leads to longer paths than other two algorithms', so it performs the worst in this simulation.

Fig. 6(a) describes the number of placed VNF instances resulting from three algorithms. In this figure, it is obvious that our proposed algorithms SFC-MAP & VNF-DRA can optimize the placed VNF instances more efficiently than ProvisionTraffic and Eigendecomposition. When network load is declining between 3500-6000 time units, VNF-DRA observes this phenomenon and adjusts the threshold of VNF utilization rate $f(t)$ to be a big value f_b based on Eq. (21). After that, more VNF instances with the utilization rate lower than f_b can be released, which results in the reduction of the total VNF running time. However, ProvisionTraffic and Eigendecomposition do not optimize the number of VNF instances with the variation of network load. Therefore, when network load decreases, the VNF instances cannot be released timely. Moreover, when network load increases between 9000-12000 time units, more SFC requests are embedded in the network. At the beginning of that time, as there is only small number of placed VNF instances in MDC nodes, the resources around the MDC nodes are fast consumed, which aggravates network congestion

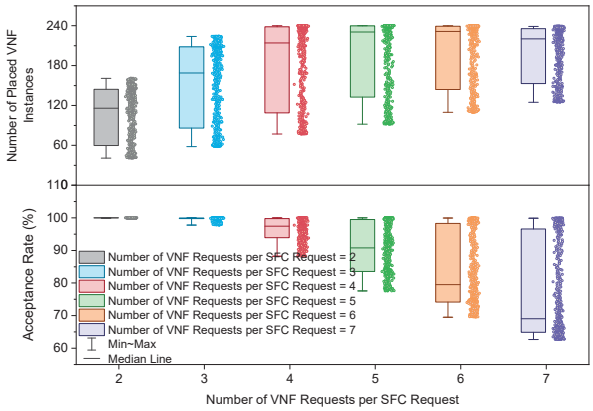
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2880992, IEEE Transactions on Parallel and Distributed Systems



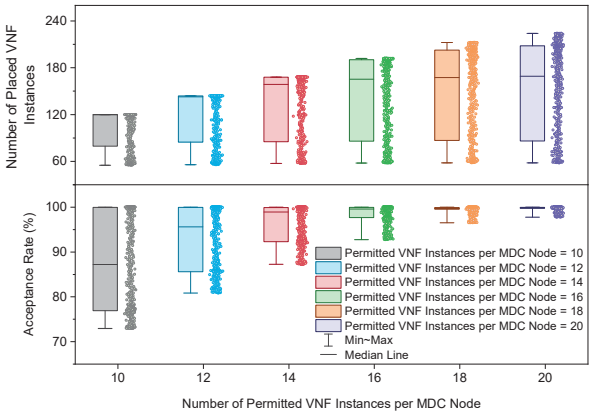
(a) The distributions of SFC request acceptance rate and the number of placed VNF instances with different number of permitted VNF types per MDC node.



(b) The distributions of SFC request acceptance rate and the number of placed VNF instances with different CPU capacity per VNF instance.



(c) The distributions of SFC request acceptance rate and the number of placed VNF instances with different number of VNF requests per SFC request.



(d) The distributions of SFC request acceptance rate and the number of placed VNF instances with different number of permitted VNF instances per MDC node.

Fig. 7. The comparison of SFC request acceptance rate and the number of placed VNF instances with different scenarios.

and load imbalance. Then, in SFC-MAP, the resource costs defined in Eqs. (13)-(15) start to take effect. With resource consumption, the resource costs of links, nodes, and VNF instances increase quickly, and it becomes more beneficial to place new VNF instances than reuse the placed ones according to Eq. (18). Next, many new VNF instances are placed in MDC nodes to provide more choices of path selection for SFC requests. Hence, the load turns to be balanced. Though ProvisionTraffic and Eigendecomposition enhance the network performance by placing more VNF instances to cope with increasing network load, the neglect of load balancing leads to lower SFC request acceptance rate. Furthermore, due to low SFC request acceptance rate, the number of VNF instances with Eigendecomposition is smaller than the ProvisionTraffic's. Additionally, because VNF-DRA executes periodically, the number of placed VNF instances decreases step by step during 3500-7000 time units in this figure.

Fig. 6(b) presents the total running time of VNF instances of these three algorithms. According to this figure, SFC-MAP & VNF-DRA can get better performance in this simulation. Nevertheless, ProvisionTraffic and Eigendecomposition almost cannot reduce the total running time of VNF instances. Since the network load becomes light during 3500-10000 time units, the SFC-MAP & VNF-DRA can reduce the running time of VNF instances by releasing the redundant VNF instances with low utilization rates. And SFC-MAP & VNF-DRA can get about 25% reduction of total running time of VNF instances than ProvisionTraffic's.

In Fig. 6(c), we evaluate the mean VNF utilization rate of these three algorithms. SFC-MAP & VNF-DRA obtains more than 75% mean VNF utilization rate under heavy network load during 1500-3500 and 12000-15000 time units. VNF-DRA runs periodically to release the VNF instances with low utilization rate, so our algorithms can keep the mean VNF utilization rate around 60%, when network load becomes light during 3500-9000 time units. In ProvisionTraffic, because of lower SFC request acceptance rate and larger number of VNF instances, the mean VNF utilization rate is about 10%-30% lower than SFC-MAP & VNF-DRA's during 1500-3500 and 10000-15000 time units. Without considering to release overmany placed VNF instances with decreasing network load, the mean VNF utilization rate of ProvisionTraffic drops sharply to about 20%, during 6000-9000 time units. Due to poor SFC request acceptance rate and overmany placed VNF instances,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2880992, IEEE Transactions on Parallel and Distributed Systems

the mean VNF utilization rate with Eigendecomposition is only about 40% during heavy network work load and about 20% during light network load.

*2) **Evaluation of SFC-MAP & VNF-DRA with Different Scenarios**:* In Fig. 7, we evaluate the performance of SFC-MAP & VNF-DRA with different scenarios. In these figures, box plots are used to show the distributions of SFC request acceptance rate and the number of placed VNF instances with the SFC request arrival rate shown in Fig. 4. In these box plots, the maximum, median and minimum of the simulation results are presented. As for the box plots which depict the distributions of SFC request acceptance rate, the heavier (lighter) the network load is, the lower (higher) SFC request acceptance rate will be. Nevertheless, as for the box plots which depict the distributions of the number of placed VNF instances, the rule is contrary. This is because the heavier (lighter) the network load is, the larger (smaller) number of VNF instances should be placed.

In Fig. 7(a), the performance of SFC-MAP & VNF-DRA is evaluated with different number of permitted VNF types per MDC node. With the number of permitted VNF types per MDC node increasing from 7 to 10, the SFC request acceptance rate increases from the range about 80%-100% to the range about 96%-100%, and the number of placed VNF instances increases from the range about 60-225 to the range about 60-235. This is because, if MDC nodes can place more types of VNF instances, SFC requests are more likely to be served by one MDC node. Therefore, it is beneficial to shorten the paths in chaining solutions by placing more types of VNF instances in MDC nodes, which can save resources to serve more SFC requests. In addition, when the number of permitted VNF types increases from 10 to 12, the maximum, median and minimum, which depict the distributions of the number of placed VNF instances, decrease in the box plots. This phenomenon presents that it is helpful to balance network load and improve resource utilization by placing more types of VNF instances in MDC nodes.

Fig. 7(b) presents the influence of the CPU capacity per VNF instance on the performance of SFC-MAP & VNF-DRA. As shown in the figure, if the CPU capacity per VNF instance is increased from 75 MIPS to 100 MIPS, the SFC acceptance rate becomes higher and the number of placed VNF instances decreases. This is because increasing the CPU capacity per VNF instance allows a VNF instance to serve more SFC requests. Furthermore, when the CPU capacity per VNF instance changes from 90 MIPS to 100 MIPS, though there are still many VNF instances that can be placed to serve SFC requests, the improvement of SFC request acceptance rate is small. This phenomenon indicates that, the bandwidth on links and memory on switch nodes are the bottlenecks which lead to the failure of SFC embedding under this circumstance.

The performance of SFC-MAP & VNF-DRA is evaluated with different numbers of VNF requests per SFC request in Fig. 7(c). With the increasing number of VNF requests per SFC request, the number of placed VNF instances increases rapidly from the range about 40-160 to the range about 140-240. And the minimum of the SFC request acceptance rate also drops from about 99% to about 63%. The reason is that, when serving the SFC requests with more VNF requests, there are more VNF instances needed to be concatenated with order constraints, which leads to longer end-to-end delay and more resource consumption.

We evaluate the performance of SFC-MAP & VNF-DRA with different number of permitted VNF instances per MDC node in Fig. 7(d). In the figure, the more number of permitted VNF instances per MDC node, the more resource an MDC node can provide. Hence, with the number of permitted VNF instances per MDC node increasing from 10 to 18, the SFC request acceptance rate continuously increases from the range about 73%-100% to about 97%-100%. To keep high SFC request acceptance rate, the number of placed VNF instances also increases from the range about 60-120 to about 60-220. And when we go on increasing the number of permitted VNF instances per MDC node from 18 to 20, the benefits on the SFC request acceptance rate and the number of placed VNF instances become smaller. This phenomenon reflects that MDC nodes have provided enough resource to deal with the SFC requests under this circumstance. In addition, combined with Fig. 7(b), we get that it is effective to improve the network performance by increasing the resource capacities of VNF instances and MDC nodes.

VIII. Conclusion

In the paper, we study the SFC-EP with dynamic VNF placement in geo-distributed cloud system aiming to minimize the embedding costs for SFC requests and optimize the number of VNF instances for the reduction of the total VNF running time. Then, SFC-MAP and VNF-DRA are proposed to solve this problem. SFC-MAP places VNF instances and embeds SFC requests by executing the shortest path algorithm in a multi-layer graph. The VNF-DRA with a piecewise threshold related to the variation of network load is proposed to efficiently release redundant VNF instances and reduce the total VNF running time. Performance evaluation results show that, SFC-MAP & VNF-DRA obtains more than 20% improvements in the terms of SFC request acceptance rate, network throughput and mean VNF utilization rate, and about 25% reduction of the total VNF running time compared with algorithms in existing literatures.

As a future work, we plan to extend our work in a number of ways. We plan to explore the influences of power consumption, VNF placement delay and ping-pong effect which leads to a VNF instance being frequently placed and released during dynamic VNF placement. We plan to study the proactive resource management for VNF instances in date center networks. And we also plan to design efficient VNF placement algorithm combined with load prediction mechanism.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 61671420, 61672484, Youth Innovation Promotion Association CAS under Grant No. 2016394, and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network Function Virtualization: State-of-the-art and Research Challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[2] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra, "Exploiting Congestion Games to Achieve Distributed Service Chaining in NFV Networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 2, pp. 407–420, 2017.

[3] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 13–24, 2012.

[4] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *Journal of Network and Computer Applications*, vol. 75, pp. 138–155, 2016.

[5] W. Xiao, W. Bao, X. Zhu, and L. Liu, "Cost-Aware Big Data Processing across Geo-distributed Datacenters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 11, pp. 3114–3127, 2017.

[6] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network Function Virtualization: Challenges and Opportunities for Innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[7] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, "Research Directions in Network Service Chaining," in *Proc. IEEE SDN for Future Networks and Services (SDN4FNS)*, pp. 1–7, 2013.

[8] K. Xu, M. Shen, H. Liu, J. Liu, F. Li, and T. Li, "Achieving Optimal Traffic Engineering Using a Generalized Routing Framework," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 1, pp. 51–65, 2016.

[9] R. Mijumbi, J. Serrat, J.-L. Gorricho, and R. Boutaba, "A Path Generation Approach to Embedding of Virtual Networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 3, pp. 334–348, 2015.

[10] S. Zhang, Z. Qian, J. Wu, S. Lu, and L. Epstein, "Virtual Network Embedding with Opportunistic Resource Sharing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 816–827, 2014.

[11] V. Eramo, E. Miucci, M. Ammar, and F. G. Lavacca, "An Approach for Service Function Chain Routing and Virtual Function Network Instance Migration in Network Function Virtualization Architectures," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2008–2025, 2017.

[12] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating Virtualized Network Functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 725–739, 2016.

[13] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The Dynamic Placement of Virtual Network Functions," in *Proc. IEEE Network Operations and Management Symposium (NOMS)*, pp. 1–9, 2014.

[14] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba, "Elastic Virtual Network Function Placement," in *Proc. IEEE International Conference on Cloud Networking (CloudNet)*, pp. 255–260, 2015.

[15] M. Bouet, J. Leguay, T. Combe, and V. Conan, "Cost-based placement of vDPI functions in NFV infrastructures," *International Journal of Network Management*, vol. 25, no. 6, pp. 490–506, 2015.

[16] M. Mechtri, C. Ghribi, and D. Zeghlache, "A Scalable Algorithm for the Placement of Service Function Chains," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 533–546, 2016.

[17] X. Li and C. Qian, "The Virtual Network Function Placement Problem," in *Proc. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 69–70, IEEE, 2015.

[18] H. Moens and F. De Turck, "VNF-P: A Model for Efficient Placement of Virtualized Network Functions," in *Proc. IEEE International Conference on Network and Service Management (CNSM)*, pp. 418–423, 2014.

[19] T. Lin, Z. Zhou, M. Tornatore, and B. Mukherjee, "Optimal Network Function Virtualization Realizing End-to-End Requests," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2015.

[20] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual Network Functions Placement and Routing Optimization," in *Proc. IEEE International Conference on Cloud Networking (CloudNet)*, pp. 171–177, 2015.

[21] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and Energy-efficient vNF Placement for Service Chaining: Joint Sampling and Matching Approach," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[22] W. Ma, C. Medina, and D. Pan, "Traffic-Aware Placement of NFV Middleboxes," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2015.

[23] M. Zeng, W. Fang, and Z. Zhu, "Orchestrating Tree-Type VNF Forwarding Graphs in Inter-DC Elastic Optical Networks," *Journal of Lightwave Technology*, vol. 34, no. 14, pp. 3330–3341, 2016.

[24] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[25] S. Umeyama, "An Eigendecomposition Approach to Weighted Graph Matching Problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 5, pp. 695–703, 1988.

[26] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester, "Network service chaining with optimized network function embedding supporting service decompositions," *Computer Networks*, vol. 93, pp. 492–505, 2015.

[27] D. Dietrich, A. Abujoda, and P. Papadimitriou, "Network Service Embedding Across Multiple Providers with Nestor," in *Proc. IEEE IFIP Networking Conference (IFIP Networking)*, pp. 1–9, 2015.

[28] A. Mohammadkhan, S. Ghapani, G. Liu, *et al.*, "Virtual Function Placement and Traffic Steering in Flexible and Dynamic Software Defined Networks," in *Proc. IEEE International Workshop on Local and Metropolitan Area Networks (LANMAN)*, pp. 1–6, 2015.

[29] Z. Cao, M. Kodialam, and T. Lakshman, "Traffic Steering in Software Defined Networks: Planning and Online Routing," in *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 65–70, 2014.

[30] 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," in *Proc. IEEE International Conference on Computer Communications*, pp. 1–9, 2016.

[31] V. Eramo, M. Ammar, and F. G. Lavacca, "Migration Energy Aware Reconfigurations of Virtual Network Function Instances in NFV Architectures," *IEEE Access*, vol. 5, pp. 4927–4938, 2017.

[32] A. Csoma, B. Sonkoly, L. Csikor, F. Németh, A. Gulyas, W. Tavernier, and S. Sahhaf, "ESCAPE: Extensible Service Chain Prototyping Environment Using Mininet, Click, Netconf and Pox," in *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 125–126, ACM, 2014.

[33] R. Riggio, J. Schulz-Zander, and A. Bradai, "Virtual Network Function Orchestration with Scylla," in *ACM SIGCOMM Computer Communication Review*, vol. 45, pp. 375–376, ACM, 2015.

[34] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing ospf weights," in *INFOCOM 2000. Nineteenth annual joint conference of the IEEE computer and communications societies. Proceedings. IEEE*, vol. 2, pp. 519–528, IEEE, 2000.

[35] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional ip routing protocols," *IEEE communications Magazine*, vol. 40, no. 10, pp. 118–124, 2002.

[36] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008.

[37] "Optical Network Design and Planning." [Online]. Available: http://www.monarchna.com/topology.html.

[38] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and Placing Chains of Virtual Network Functions," in *Proc. IEEE International Conference on Cloud Networking (CloudNet)*, pp. 7–13, 2014.

[39] A. Dwaraki and T. Wolf, "Adaptive Service-Chain Routing for Virtual Network Functions in Software-Defined Networks," in *Proc. ACM workshop on Hot topics in Middleboxes and Network Function Virtualization*, pp. 32–37, 2016.

[40] R. Ramaswamy, N. Weng, and T. Wolf, "Characterizing Network Processing Delay," in *Proc. IEEE Global Telecommunications Conference*, vol. 3, pp. 1629–1634, 2004.

[41] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C. Lau, "Online Stochastic Buy-Sell Mechanism for VNF Chains in the NFV Market," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 2, pp. 392–406, 2017.

[42] L. Pantel and L. C. Wolf, "On the Impact of Delay on Real-Time Multiplayer Games," in *Proc. ACM international workshop on Network and operating systems support for digital audio and video*, pp. 23–29, 2002.