Embedding Service Function Tree with Minimum Cost for NFV Enabled Multicast

Bangbang Ren, *Student Member, IEEE,* Deke Guo, *Senior Member, IEEE,* Yulong Shen, *Member, IEEE,* Guoming Tang, *Member, IEEE,* Xu Lin, *Student Member, IEEE*

***Abstract*—Usually, a data flow needs to traverse a series of network functions, which is called a service function chain (SFC), before reaching its destination. The emergence of network function virtualization (NFV) makes the embedding solution of SFC flexible as far as the deployment location is concerned. When providers embed the SFC into a substrate network, they will hope to minimize the setup cost of the SFC and link connection cost towards clients. For unicast, since there is one path connecting the source node to the destination node, all functions of the SFC are just needed to be deployed along the path. However, when embedding the SFC for a multicast task, the topology of the SFC may change because the function deployment locations have impacts on the traffic delivery cost. Thus, a service function tree (SFT) may be a better choice. Given the huge space of SFT embedding solutions, however, it is extremely hard to find the optimal one such that the total traffic delivery cost is minimized. In this paper, we tackle the optimal SFT embedding problem in NFV enabled multicast task. Specifically, we formally define the problem and formulate it with an integer linear programming (ILP), which turns out to be NP-hard. Then, a two-stage method is proposed to deal with the problem with an approximation ratio of $1+\rho$, where ρ is the best approximation ratio of Steiner tree and can be as small as 1.39. With extensive experimental evaluations, we demonstrate that by applying our SFT embedding solution, the delivery cost of multicast traffic can be reduced by 22.05% at most against three benchmarks.**

***Index Terms*—network function virtualization, service function chain, minimum cost, multicast**

I. Introduction

Multicast is designed to deliver the same content from a single source node to a group of destination nodes [2]. Compared with unicast, multicast can significantly save the bandwidth consumption and relieve the load of the source server as it avoids duplicated transmissions among independent unicast paths, especially by multiplexing a shared multicast tree. However, the traditional distributed IP network cannot support centralized optimal multicast routing (e.g., the Steiner tree [3]) well. The emergence of software defined network (SDN) changes this situation. SDN separates the data plane and control plane of network [4], [5]. This architecture simplifies traffic engineering by using the controller to manage the routing of flows directly. In SDN, the centralized control plane can calculate the optimal multicast routing and then install relevant routing tables into network devices [6].

Usually, a request flow from the source node to the destination node may be processed by various network functions in a particular order, which forms the service function chain (SFC) [7], [8]. For example, in the email service, the data flow will go through virus detection, spam identification, and phishing detection in turn [9]. These network functions usually are proprietary hardware appliances. Though SDN can help to steer traffic to go through these boxes, bringing new services into today's network is becoming increasingly difficult due to the proprietary nature of existing hardware appliances. With the growth of various new network function needs, hardware-based appliances rapidly reach the end of life, requiring much of the design-integrate-deploy cycle to be repeated with little or no revenue benefit. Network function virtualization (NFV) is proposed in 2012 to solve the above problems along with SDN [10], [11]. The emergence of NFV enables network operators to virtualize their network services as virtualized network functions (VNFs). These VNFs substitute dedicated hardware middleboxes and bring high flexibility and elasticity. These VNFs not only reduce the cost and time of deploying new network services but also bring new challenges to traffic engineering. Many works investigate NFV enabled traffic engineering for both unicast and multicast (see the related work in Section II).

A. Motivation: from SFC to SFT

For NFV enabled unicast, there are only one source and one destination. The SFC is straightforward to deploy, e.g., by sequentially choosing embedding nodes and deploying VNFs along the unicast routing path [12]. However, embedding the SFC for multicast becomes tricky, especially when the costs of deployment and connection are sensitive to the function location. For example, in the video streaming service, ISPs strategically deploy network functions (e.g., intrusion detection, load balance, and format transcoding) among the

Manuscript received May 5, 2018; revised January 30, 2019, accepted March 13, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant No.61802421, No.61772544, U1536202, U1736216, the National Key R&D Program of China No.2017YFB1400700, the Hunan Provincial Natural Science Fund for Distinguished Young Scholars under Grant No.2016JJ1002 and the NUDT Research Plan under Grant No.ZK17-03-50.

Corresponding authors: Deke Guo, Yulong Shen

B. Ren, D. Guo and G. Tang are with the Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha Hunan 410073, China. D. Guo is also with the College of Intelligence and Computing, Tianjin University, Tianjin, 300350, China. Email: {renbangbang11,dekeguo, gmtang.}@nudt.edu.cn.

Y. Shen and X. Lin are with the School of Computer Science and Technology, Xidian University, Shaanxi, China. Email: {ylshen@mail.xidian.edu.cn,xulin_xidian@163.com.}

[*] A preliminary version of this paper is accepted in ICDCS 2018 [1]. In this version, we modified and added some contents into abstract, introduction, problem formulation and experiment analyses.

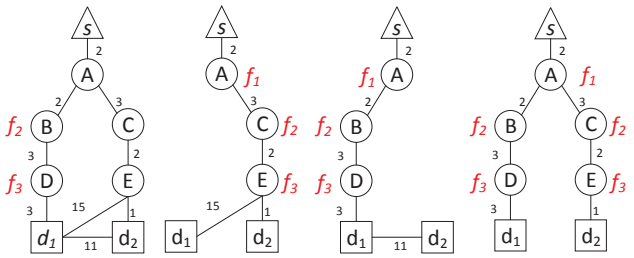


Fig. 1. Three SFT embedding strategies for the same multicast task in a target network, where s is the source, d_1 and d_2 are the destinations, and $A \sim E$ are server nodes. The link connection cost is labeled beside each edge and the VNF setup cost equals one.

network nodes, as the costs of connecting the source server to geographically distributed users (traversing all the network functions) can be much different [13]–[15]. Considering the NFV enabled multicast for such tasks, we need to embed the SFC for the request flows carefully, which leads to embedding a *service function tree* (SFT) in the multicast tree.

Furthermore, with VNFs that have been deployed in a target network, there are multiple choices to deploy the new VNFs and construct the SFT for a particular multicast task. Nevertheless, it is not easy to find the optimal SFT embedding solution with the least cost under this situation. Fig. 1 provides a simple example for SFT embedding over a network with deployed VNFs. Fig. 1(a) shows i) the target network with eight nodes and two deployed VNFs f_2 and f_3, ii) the multicast task with source node s and two destination nodes $\{d_1, d_2\}$, and iii) link connection costs labeled on the links. Assuming that the SFC requirement for the multicast task is $(f_1 \rightarrow f_2 \rightarrow f_3)$ and the setup cost of each VNF equals to one, Fig. 1(b)$\sim$Fig. 1(d) give three SFT embedding solutions with traffic delivery costs[1] of 26, 22 and 19, respectively. We can see that different SFT embedding strategies can result in diverse traffic delivery cost, and our task is to find the optimal one (as shown by Fig. 1(d) in this example). The details of this example will be further explained in Section III-A.

B. Challenges & Contributions

The example given in Fig. 1 is simple and may be easy to solve. In reality, however, the network topology and size of multicast tasks can be much larger and more complex, which leads to a huge space of feasible SFT embedding solutions. Hence, for a specific NFV enabled multicast task, it is a challenging problem to find the most efficient multicast tree where the embedded SFT is proved optimal, considering the link connection cost, VNF setup cost, and nodes capacities. Indeed, even though we assume that the required SFC can be deployed on one node to free the VNF order constraint, the problem is still NP-hard [16].

In this paper, aiming to solve the optimal SFT embedding problem for NFV enabled multicast, we make the following contributions:

- We formally define the problem of optimal SFT embedding in NFV enabled multicast task and formulate the problem by an integer linear programming (ILP), aiming to minimize the traffic delivery cost. We prove that the optimal SFT embedding problem is NP-hard.
- We propose a two-stage algorithm to tackle the NP-hard problem. In the first stage, an initial feasible solution to the ILP problem is produced by embedding an SFC together with a Steiner tree. In the second stage, the initial solution is optimized by adding new VNF instances to construct an SFT. With sufficient node capacities, we prove that the approximation ratio of our algorithm is $1+\rho$, where ρ is the best approximation ratio of Steiner tree and can be as small as 1.39 [17].
- We extend our algorithm to be compatible to the common situation where a certain number of (virtual) network functions have been deployed, like some public clouds handle base load by physical hardware and spillover load by virtual service instances [18], and our algorithm can be aware and reuse these deployed functions in solving the SFT embedding problem.
- We investigate the performance of our algorithm under different parameter settings. With extensive experiments, we demonstrate that our SFT embedding method can reduce the total traffic delivery cost by 22.05% at most compared to three benchmarks.

[1]The traffic delivery cost of a multicast task is computed by the sum of all VNFs' setup cost and link connection cost over the target network [16].

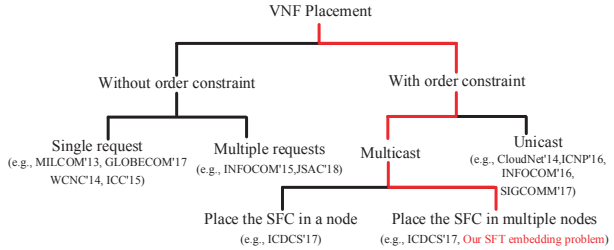


Fig. 2. Classification of the related work for VNF placement.

The remainder of this paper is organized as follows. In Section II, we review the related work. We formulate the optimal SFT embedding problem by an integer linear programming in Section III. We design a two-stage algorithm to tackle the posed problem and provide the algorithm analysis in Section IV. Section V presents performance evaluations on the proposed algorithm and sensitivity analysis to the parameter settings. Section VI concludes the paper.

II. Related Work

A. VNF placement

Researches on NFV enabled multicast and relevant to this work can be roughly grouped into two categories: VNF placement with or without order constraints, as shown in Fig. 2.

VNF Placement without order constraint. This type of works focus on placing independent VNFs. Literatures [19] and [20] studied how to deploy VNFs for single request case. Bouet et al. proposed a heuristic algorithm to optimize the cost of VNF deployment which usually concerns license fees and resource consumption [19]. Miloud et al. studied the optimal VNF placement with the consideration of the QoS requirements [20]. As for multiple cases, Rami et al. concentrated on satisfying multiple client requests that need various

VNFs [21]. In [21], the authors investigated the problem of finding the optimal placement of multiple independent VNFs within a physical network, by minimizing the sum of the traffic delivery distance and the VNF setup cost. Specially, [22] uses conformal mapping to give an universal placement solution considering demands imbalance. Literatures [23]–[25] investigate VNF placement problem and network slicing in the 5G access network. We list the above representative works to give a direct comparison to our work, and more related works are surveyed in [8].

VNF Placement with order constraint. The requirement of traversing VNFs in order (i.e., SFC) makes VNF placement problem more complicated. In [7], the SFC was embedded into the target network for the unicast task with multiple objectives such as maximizing remaining data rate, minimizing the number of applied nodes and minimizing traffic delivery latency. Kuo et al. [26] studied the joint optimization problem of VNF placement and path selection with the consideration of link and server usage. Lin et al. [27] devoted to minimizing the embedding cost of SFC with parallel VNFs. The dynamic SFC embedding problem was studied in [12], [28], where the sequence of functions could change during the life of a session. All the above researches essentially focus on SFC embedding in the unicast request. However, embedding an SFC in a multicast request is different. Xu et al. [16] devoted to finding a minimum cost pseudo-multicast with an assumption that the SFC is placed on one node (to neglect the order constraint). The assumption is not practical, especially under the multi-cloud service function chaining architecture [29]. Kuo et al. investigated to embedding multiple SFC instances for the many-to-many flow request, and a service overlay forest was introduced in [30]. Yi et al. [31] separated traffic forwarding graph and function delivery graph, which cannot exploit the benefits of multicast multiplexing.

As the most relevant work to ours, Cheng et al. [32] formulated the SFC embedding problem for multicast. They first construct a Steiner tree that covering all destinations and find a minimum cost path connecting the source to the Steiner tree. Then they embed the SFC along the minimum cost path. However, they embed an SFC rather than an SFT. In this work, we embed an optimal SFT in the shared multicast tree that helps to deliver the desired traffic flows from the source to multiple destinations with the minimum cost.

B. Multicast Routing under SDN.

Ratnasamy et al. [33], [34] revisited the multicast design and gave examples of multicast applications, including multiplayer games, IPTV and file sharing. Li and Freedman [34] leveraged the unique topological structure of datacenter networks to build multicast trees. Both works above indicate that the size of a multicast tree can be significantly large, while the emergence of SDN technology helps alleviate this problem with convenient ways to deal with multicast routing. Multicast traffic engineering has been widely investigated under the SDN diagram. Shen et al. [6] looked into the packet loss recovery problem for reliable multicast routing and applied an approximation algorithm to minimize both tree and recovery costs. Iyer et al. [35] applied SDN aided multicast strategy in data centers and presented an SDN based multicast system named Avalanche that was used to minimize the size of the routing tree.

III. Optimal SFT Embedding Problem

In this section, we first give an illustrative example of the SFT embedding problem for the NFV enabled multicast task. Then we formally define and formulate the optimal SFT embedding problem, which is the focus of this work.

A. Problem Illustration

As mentioned in Section I, Fig. 1(a) depicts a network model with link connection cost labeled on each edge. Five server nodes ($A \sim E$) are in the network upon which VNFs can be installed, and two types of VNFs (f_2 and f_3) have been already deployed as illustrated in the network. For simplicity, we do not show the switch nodes between server nodes. Indeed, the number of switch nodes and other factors (e.g., link bandwidth) can be modeled in the link cost. Fig. 1(a), without loss of generality, we assume that the setup costs of VNF instances upon different nodes are the same and set the value as one [30].

Here we denote the multicast task as $\delta=\{s,\hat{D},\ell\}$, in which s is source, $\hat{D}$ is the destination set with $\hat{D}=\{d_1,d_2\}$ and ℓ represents the SFC requirement $\ell=(f_1 \rightarrow f_2 \rightarrow f_3)$. For this specific multicast request, Fig. 1(b), Fig. 1(c) and Fig. 1(d) provide three different solutions, respectively. In Fig. 1(b), we deploy new instances of f_1, f_2, f_3 on A,C,E, respectively, so the traffic delivery cost is $2+3+2+15+1+3\times 1=26$. In Fig. 1(c), we steer flow from s to destinations through B and D. Since f_2 and f_3 have already been deployed on B and D, their setup cost is zero, and then the traffic delivery cost is $2+2+3+3+11+1+0\times 2=22$. Particularly, the solution in Fig. 1(d) builds a service function tree by i) reusing the deployed VNFs (on the $A-B-D$ branch) and ii) establishing a new flow path (on the $A-C-E$ branch). With the newly builted SFT, the multicast flow avoids expensive links e_{Ed_1} and $e_{d_1d_2}$, which further cuts down the traffic delivery cost and results in the minimum traffic delivery cost, i.e., $2+2+3+3+2+3+1+3\times 1=19$.

With the example shown in Fig. 1(d), we formally define the service function tree (SFT) as follow:

Definition 1. ***Service Function Tree.*** *Given a target network with or without deployed VNFs and a multicast task, we need to install some new VNF instances among the network nodes to fulfill the SFC requirement. Particularly, the goal of minimizing the total traffic delivery cost will enforce a tree structure of service functions embedded in the shared multicast tree, which we name service function tree (SFT).*

Note that the number of VNFs is certain for an SFC, while the number of VNFs for an SFT is not deterministic given the target network and a multicast task, i.e., multiple versions of SFTs can fulfill the multicast task (e.g., Fig. 1 gives three different SFTs). Under this situation, considering the VNF setup cost, link connection cost and node capacity

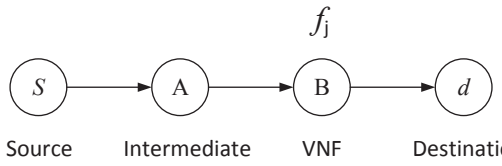


Fig. 3. Four locations of node u in constraint (4).

for VNF deployment, it is meaningful yet challenging to find the optimal SFT embedding solution that leads to the least traffic delivery cost for the multicast task.

B. Problem Definition

Target Network. We consider a network G=(V,E) with V=$V_M \bigcup V_S$, in which V_M and V_S denote the sets of server nodes and switch nodes, respectively. For a server node $v \in V_M$, $cap(v)$ represents its capacity for VNF deployment and is measured by its resources. This capacity can be evaluated by the bottleneck resource, usually is CPU. For an edge e_{uv}, c_{uv} denotes the link connection cost of transferring the flow. It is important to emphasize that c_{uv} is the input data of our problem and can be scaled down/up according to the flow size.

VNF Deployment Constraints. Any VNF will have a setup cost when it is deployed upon the server node in V_M. A binary number $\pi_{f_j u}$ is used to denote whether f_j has been deployed on node u; a real number $\gamma_{f_j u}$ is used to denote the setup cost of a new VNF instance f_j upon node u. For those VNF instances that have already been deployed, we treat their setup costs as zero; a real number μ_{f_j} is applied to represent the resource amount that deploying f_j is needed. For simplicity, we assume that each instance of VNFs can serve traffic flows with any sizes by scaling vertically so that the same VNF instance will not be deployed repeatedly due to flow splitting [36].

Definition 2. ***Multicast Task.*** *Given the target network G with or without deployed VNFs, a multicast task can be represented by a three-element tuple δ=$(s,\hat{D},\ell)$, where $s,\hat{D},\ell$ are the source node, destination nodes, and the SFC requirement for the multicast flow, respectively. Specifically, the SFC is denoted by ℓ=$(f_1,f_2,...,f_n)$.*

Definition 3. ***Optimal SFT Embedding Problem.*** *For a multicast task δ given by Definition 2, under the VNF deployment constraints, the optimal SFT embedding problem is to build an embedded service function tree upon the target network such that i) the multicast request is satisfied and ii) the traffic delivery cost is minimized.*

C. Problem Formulation

For ease of reference, we list all the notations in Table I.

To describe the optimal SFT embedding problem easily, we can treat s as f_0 and $\forall d \in D$ as f_{n+1}. Then the SFC requirement ℓ can be expanded as $\hat{\ell} = (f_0,f_1,...,f_n,f_{n+1})$. Though the multicast can save resource by multiplexing the shared tree, the flow entering into each destination must traverse a complete SFC. In Fig. 1(d), the flow entering into d_1 traverses $(A \to B \to C)$ to get the services of $(f_1 \to f_2 \to f_3)$. For each destination node d, the flow entering it must go through all VNFs of $\hat{\ell}$ only once. This characteristic means that the VNF will never process the flow and the physical server node just forward the flow when the flow have been processed by the same VNF deployed on the other server nodes visits the VNF again. We use $\varphi_{du}^{f_j}$ to represent whether the flow entering to d gets the service of f_j at the node u. Then, we have

$$\sum_{u \in V_M} \varphi_{du}^{f_j}=1, \forall f_j \in \hat{\ell}, d \in D \tag{1}$$

Remark 1. *When we say that each destination needs to be served by a SFC means that all VNFs of the SFC process the flow in order before it reaches to the destination. Some destinations may get service by the same VNF instance due to the multiplexing characteristic of the multicast.*

Though the source can be seen as f_0, each destination cannot get service of f_0 except from source s. Thus, we have

$$\varphi_{ds}^{f_0}=1, \forall d \in D \tag{2}$$

Moreover, we need to ensure that the new deployed VNF instances never violate the capacity constraints. We use a binary variable $\omega_{f_j u}$ to denote whether f_j is deployed in u. Then

$$\sum_{f_j \in \hat{\ell}} (\pi_{f_j u}+\omega_{f_j u})\mu_{f_j} \leqslant cap(u), \forall u \in V_M \tag{3}$$

The most challenging part in our problem formulation is how to model the link connection cost. The requirement of

TABLE I
SUMMARY OF NOTATIONS

Notations	Description
SFT	service function tree
MOD	the mutlilevel overlay directed network
TSA	our two-stage algorithm
STB	the Steiner tree based algorithm
SCA	the minimum set cover based algorithm
RSA	the randomly selecting based algorithm
G	graph of network topology
V	the set of all nodes in G
V_M	the set of all server nodes in G
V_S	the set of all switch nodes in G
E	the set of all edges (links) in G
u,v	a node in G
N_u	the neighbor nodes of u
e_{uv}	an edge of G
$cap(v)$	the capacity of node v
δ	the multicast task
s	the multicast source
$\hat{D}$	the set of all multicast destinations
d_i	the i^{th} destination of multicast
ℓ	the SFC requirement
$\hat{\ell}$	the extended SFC requirement
f_i	the i^{th} VNF of ℓ
$\gamma_{f_j u}$	the setup cost when f_j is deployed in node u
c_{uv}	the link cost of edge c_{uv}
$\pi_{f_j u}$	denote whether f_n has been deployed in node u
μ_{f_j}	the amount of resource needed to deploy f_j
$\omega_{f_j u}$	indicates whether or not a new VNF instance f_j is deployedon node u
$\tau_{d,f_j,u,v}$	denotes if the edge e_{uv} is located between f_j and f_{j+1} for the traffic flow heading for destination d
$\varphi_{du}^{f_j}$	represents whether the flow heading for destination d is served by VNF f_j on node u
$\psi_{f_j,u,v}$	denotes if edge e_{uv} is located between f_j and f_{j+1}

traversing the SFC in correct order may enforce some edges be visited multiple times. However, the contents of packages in this flow are different at each time visiting the same edge. The reason behind this characteristic is that each VNF does effects towards the packages. For a destination $d \in D$, there is a walk $Walk(s,d)$ connecting it to s in the multicast routing. This path can be divided into $n+1$ segments by $\hat{\ell}$. Take Fig. 1(d) for example, the path from s to d_2 can be divided into four parts, i.e., $\{s(f_0) \to A(f_1) \to C(f_2) \to E(f_3) \to d_2(f_4)\}$. For the edge e_{uv}, we use a binary variable $\tau_{d,f_j,u,v}$ to denote whether it is included in the final multicast routing. If e_{uv} lies in the segment between f_j and f_{j+1}, then $\tau_{d,f_j,u,v} = 1$. It needs to emphasize that $\tau_{d,f_j,u,v}$ is different from $\tau_{d,f_j,v,u}$ since the flow has direction from s to d.

Embedding an SFT is not only to decide on which nodes VNF instances should be deployed, it should also find the proper routing to ensure that the flow received by the destination completely goes through all the required VNFs of $\hat{\ell}$ in order. There are four types node (i.e, source node, intermediate node, VNF node and destination node) in $Walk(s,d)$ as shown in Fig. 3. We use N_u to denote the neighbor nodes of u. Considering both the conservation of flows and the SFC ordering request, we have

$$\begin{aligned} &\sum_{v \in N_u} \tau_{d,f_j,u,v} - \sum_{v \in N_u} \tau_{d,f_j,v,u} \geqslant \varphi_{du}^{f_j} - \varphi_{du}^{f_{j+1}}, \\ &\forall f_j \in \hat{\ell}, u \in V, d \in D \end{aligned} \tag{4}$$

Remark 2. *A node u has four kinds of possible positions: source node, intermediate node, VNF node and destination node. Fig. 3 depicts these four cases. For case 1, f_0 denotes s. Given f_j=f_0,u=s,v=A, if s has other VNFs, like f_1, then $\sum_{v \in N_u} \tau_{d,f_0,u,v}$=0 as the flow leaving from s is processed by f_1. $\sum_{v \in N_u} \tau_{d,f_0,v,u}$=0, $\varphi_{du}^{f_0}$=1 and $\varphi_{du}^{f_1}$=1. Thus, we have $0-0 \geq 1-1$. If s has no VNFs, then $\tau_{d,f_0,u,v}$=1,$\sum_{v \in N_u} \tau_{d,f_0,v,u}$=0, $\varphi_{du}^{f_0}$=1 and $\varphi_{du}^{f_1}$=0, then we have $1-0 \geq 1-0$. Similarly, The situations for case 2, case 3 and case 4 can also be proved to satisfy constraint (4).*

The contents of multicast flow will be changed after being processed by different VNFs. Thus, the contents of this flow in different segments of $Walk(s,d)$ are different. Multicast avoids duplicated transmissions and only copy itself at the branch node. For different destinations (e.g. $\{d_1,d_2\}$), the case with $\tau_{d_1,f_j,u,v} = \tau_{d_2,f_j,u,v} = 1$ means that there is only one flow copy that is processed by f_j on edge e_{uv}. This means that we can merge variables $\tau_{d,f_j,u,v}$ for different destinations. We use a binary variable $\psi_{f_j,u,v}$ to denote whether edge e_{uv} is used to transfer flow that has been processed by f_j but not f_{j+1}. For instance, in Fig. 1(d), $\tau_{d_1,f_0,s,A}$=1 and $\tau_{d_2,f_0,s,A}$=1, and we just need to transfer one flow copy, since the flow content from s is the same. This multicast flow constraint can be modeled as

$$\begin{aligned} &\psi_{f_j,u,v} \geqslant \tau_{d,f_j,u,v} \\ &\forall f_j \in \hat{\ell}, u \in V, v \in V, u \neq v, d \in D \end{aligned} \tag{5}$$

Constraint (5) indicates that $\psi_{f_j,u,v}$=1 if $\tau_{d,f_j,u,v}$=1 for any $d \in D$. With the definitions of $\omega_{f_j u}$ and $\psi_{f_j,u,v}$, we can easily depict the total cost of transferring such NFV enabled multicast as follows.

$$\sum_{f_j \in \hat{\ell}} \sum_{u \in V_M} \omega_{f_j u} \gamma_{f_j u} + \sum_{f_j \in \hat{\ell}} \sum_{e_{uv} \in E} \psi_{f_j,u,v} c_{uv} \tag{6}$$

The goal of embedding optimal SFT is to minimize the cost of embedding SFT, i.e.,

$$\text{Min} \quad (6) \tag{7a}$$

$$s.t. \quad (1),(2),(3),(4) and (5) \tag{7b}$$

Theorem 1. *The optimal SFT embedding problem formulated in (7) is NP-hard.*

Proof: We prove the theorem by a polynomial-time reduction from the Steiner tree problem. Assume there is a graph G=(V,E) where V and E are the node set and edge set, respectively. For each edge $e \in E$, it has an edge cost of c_e. Given a subset of the node set $D \subseteq V$, Steiner tree problem is to find a minimum cost tree OPT_G spanning all nodes in D. Then, we construct an instance of embedding SFT as follows.

We first replicate G into G', D into D'. Besides G', consider another connected graph with the node set P=$\{p_0, p_1, ...p_n\}$ and each edge in the graph has a cost. Then we connect $d_j \in D'$ in G' to $p_i \in P$ and assign the edge a random cost. Also, p_0 can act as the source node, and other nodes in P can act as servers where VNFs can be deployed. All nodes in G' can only act as intermediate nodes or destination nodes. We assume that some VNFs have already been deployed in P. Each node in P has a capacity constraint and deploying new VNF instances in P has a cost.

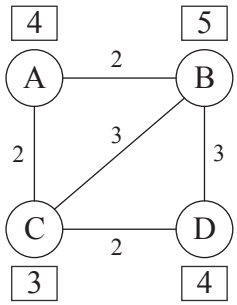
Consider a multicast task δ=$(s$=p_0, D=$D', \ell)$ in $G' \bigcup P$. Assume that we can find the optimal solution $OPT_{G'}$ for SFT embedding. Since all VNFs can be deployed only in P, then we can delete P and the edge between P and G'. The remainder subgraph of $OPT_{G'}$ in G' is the OPT_G for G. Otherwise, $OPT_{G'}$ is not the optimal solution. This means that we can find the optimal solution for the Steiner tree problem. Nevertheless, the Steiner tree problem is NP-hard [3]. Thus, the assumption is not true, and Theorem 1 is proved. ■

IV. Methodology: A Two-Stage Algorithm

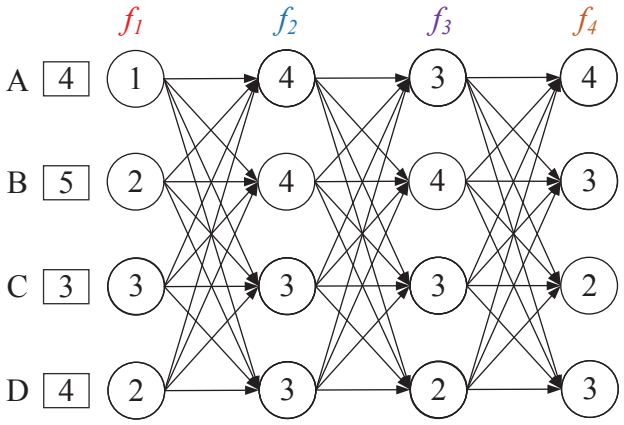
In this section, we propose a two-stage approximation algorithm to solve the optimal SFT embedding problem. As the preparation of the algorithm, we first construct a multilevel overlay directed (MOD) network which includes all information of the original network.

A. MOD Network Construction

As the preliminary, we transform the target network to a multilevel overlay directed (MOD) network, which contains all information of the original network including the link connection cost and VNF setup cost. Fig. 4(a) shows an original network with four nodes. The weight attached to each edge denotes the link connection cost, and the number near the node represents the node capacity. The deployment costs



(a) Original network G with link cost and node capacity.



(b) Multilevel overlay directed network G'

Fig. 4. An example of MOD network with ℓ=($f_1 \rightarrow f_2 \rightarrow f_3 \rightarrow f_4$).

of these four functions on the four nodes are different, and a matrix can be used to denote them as shown in Equation (8).

$$deployment\_cost = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} f_1 & f_2 & f_3 & f_4 \end{array} \\ \begin{pmatrix} 1 & 4 & 3 & 4 \\ 2 & 4 & 4 & 3 \\ 3 & 3 & 3 & 2 \\ 2 & 3 & 2 & 3 \end{pmatrix} \end{array} \tag{8}$$

Fig. 4(b) illustrates a MOD network, which is transformed from the target network consisting of 4×4 nodes as shown in Fig. 4(a). The MOD network can be partitioned by columns and rows, in which each row denotes one node and each column denotes one VNF. Specifically, the order of column is consistent with the order of the SFC. The weight attached to each node represents the deployment cost of the corresponding VNF on the node. For instance, the weight attached to the node lies in the first column, and the first row represents that the cost of deploying f_1 on A is 1. All nodes in the left column connect to all nodes in its right neighbor column with directed edges, and the costs of these edges are the same with the total link costs of the corresponding shortest paths in the original network [37]. There are three steps to get a MOD network as follows.

- Step 1: Replicate all nodes of the network k times, where k denotes the length of the SFC. Place these nodes in a matrix form, in which columns denote VNFs and rows denote physical server nodes.

Algorithm 1 The construction of multilevel overlay directed network

Input A network G=(V,E,$c(E)$) and a service function chain ℓ=$\{f_1, f_2, ..., f_k\}$.
Output Multilevel overlay directed network G'.

```
Calculate all shortest paths between each pair nodes in G
Replicate all |V|=n nodes k times and place these n × k nodes in a form of n × k matrix. Each node can be denoted by v_nk
for i=1; i++; i ≤ k − 1 do
    Connect all nodes in the i^th column to all nodes in the i+1^th column with directed edges
    Set the edge cost between two nodes as the corresponding shortest path cost in G
Set the node weight equal to the corresponding VNF deployment cost in G.
```

- Step 2: For each node in the i^{th} column, connect it to all the nodes in the $i+1^{th}$ column with directed edges.
- Step 3: Set the node weights equal to the corresponding VNF setup costs. Set the edge weights equal to the total link connection cost of the corresponding shortest paths in G.

The above three steps can transform any target network with a certain SFC requirement into a MOD network. The procedure for constructing a MOD network ensures that the original network is a subgraph of the overlay network, which guarantees that no information will be missed. Algorithm 1 depicts the procedure in detail.

B. Stage One: Find a Feasible Solution

In this stage, we first design an algorithm for the problem under the condition that each node has sufficient resources, and then generalize it to common situations. To find a feasible solution to the optimal SFT embedding problem, we first embed the SFC into the transformed MOD network and then connect the last node of the SFC to all the destinations.

Fig. 5 shows an expanded MOD network based on the network in Fig. 4(b). As shown in the figure, all nodes in G' are split into two identical nodes connected by a virtual link with connection cost equal to the VNF setup cost. Based on the expanded MOD network, the detailed operations to find a feasible solution are as follow.

- Step 1: Add the source node s into the expanded MOD network; connect it to all nodes in the first column; set each link connection cost as that of the corresponding shortest path in original network G.
- Step 2: From the source node s to one node in the last column (where the last VNF is deployed), find a path to embed the required SFC, using the shortest (weighted) path search algorithm (e.g., Dijkstra algorithm). We will prove that such a path has the least traffic delivery cost.
- Step 3: For the node deployed the last VNF, find a Steiner tree connecting it to all destination nodes. Thus, the Steiner tree along with the path resulting from Step 2 yields a feasible solution for the SFT embedding problem.

Theorem 2. *In Step 2, for the selected node in the last column of the expanded MOD network, the (weighted) shortest path is optimal for SFC embedding.*

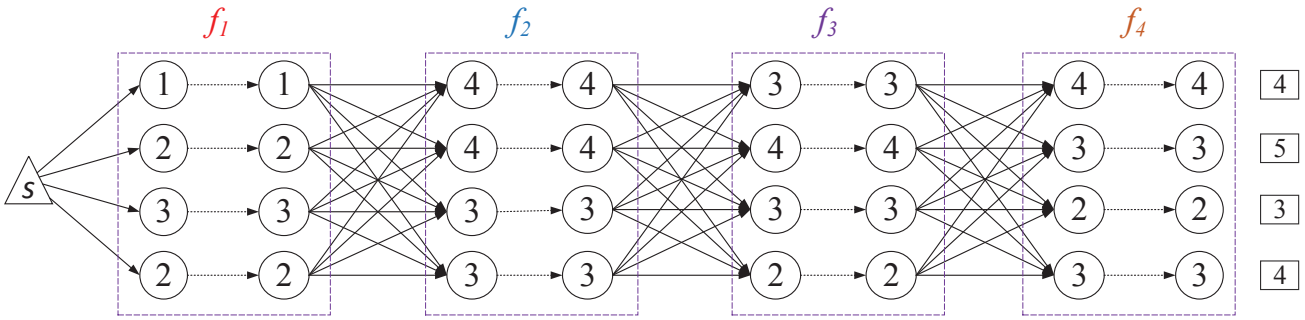


Fig. 5. The expanded MOD network.

Proof. We assume that the selected node deploying the last VNF is v_t. Indeed, any one solution of embedding the SFC in original network G can be mapped into a path in the expanded MOD network. The cost of embedding an SFC can be treated as the sum of two parts, i.e., the VNF setup cost and the link connection cost. The VNF setup cost can be mapped to the virtual edge weights, and the link connection cost between neighbor VNFs can be mapped to the real edge weights between neighbor columns. The condition that nodes have sufficient resources ensures that any path from s to v_t can be used to embed the SFC, and the shortest path computed by our algorithm in the expanded MOD network provides the optimal SFC embedding. Thus, Theorem 2 is proved. □

Nevertheless, if the assumption that each node has sufficient resources is not held, the path generated by Step 2 with the least traffic delivery cost may be infeasible in practice. The reason is that some node along the path may be overloaded. Therefore, for the obtained solution, we further check if there exists the overloading problem. If so, corresponding node adjustment will be performed as follow.

We check all the VNFs of the SFC in order. If one VNF has been deployed in an overloaded node, then we should find another node to deploy it. Without loss of generality, we assume VNF f_i has been deployed on node v_j that is overloaded, and its neighbor VNFs f_{i-1} and f_{i+1} have been deployed on nodes v_k and v_m, respectively. To find a new node to deploy VNF f_i, we check all the other nodes in the same column which v_j lies. For each node with enough capacity, we compute the sum of i) its link connection cost to v_k, ii) its link connection cost to v_m, and iii) its VNF setup cost. Thus, we choose the node with the minimum cost as the new deployment place for f_i.

In Step 3, the node with the last VNF makes a big impact on the final solution, as the different locations of the node will lead to a different Steiner tree. Therefore, we consider all the cases for the choices of the node with the last VNF and select the solution with the minimum cost as the output of the first stage.

Theorem 3. *The solution resulting from stage one is a feasible solution to the SFT embedding problem.*

Proof. In the problem of embedding SFT for NFV enabled multicast $\delta=(s,D,\ell)$, the feasible solution requires that the flow traverses all VNFs of ℓ in order before reaching destinations. In stage one, we first embed the SFC and then connect the node deploying the last VNF to all destinations. This step ensures that flow is steered from the first VNF to the last VNF in order and then goes to the destination. Thus, each destination receives the flow processed by the entire ℓ, theorem 3 is proved. □

Algorithm 2 The first stage of our two-stage algorithm

Input An overlay network G', a multicast task $\delta=(s,\hat{D},\ell)$ and VNF deployment costs on all nodes.
Output A solution with an SFC and a Steiner tree.
1: Add the source node s into G'.
2: Connect s to all nodes of the first column in G' and set the cost as corresponding shortest path.
3: Divide each nodes in G' into two parts, and connect these two parts with cost $\gamma_{f_k u}$.
4: **for** each node v of the last column in G' **do**
5: Find the shortest path Υ from s to v in G' and get $[\omega_{f_n u}]$.
6: Build a Steiner tree to cover v and all destinations.
7: **for** j=0;$j++$;$j \leq n$ **do**
8: Check whether f_j is deployed in an overloaded node, if so, find a new node with the minimum sum of setup cost and connection cost to deploy f_j.
9: Update $[\omega_{f_n u}]$, Υ and get the total cost of the solution.
10: Output the solution with the minimum cost.

Overall, the first stage of our two-stage algorithm is shown in Algorithm 2.

C. Stage Two: Optimize the Feasible Solution

For the feasible solution in stage one, it only contains an embedded SFC. As we have mentioned in Section III-A, embedding an SFT for the multicast task can outperform embedding an SFC. Therefore, in the second stage, we transform the obtained SFC in the first stage to an SFT by adding new VNF instances and eliminating links with expensive cost.

Fig. 6 depicts an example of SFT. In such an SFT, we call f_i the predecessor VNF of f_j if f_i takes effect before f_j, and correspondingly, f_j is the successor VNF of f_i. In Fig. 6, it can be observed that the number of predecessor VNFs is smaller than that of successor VNFs, and we can prove that this is necessary for an SFT with Theorem 4.

Theorem 4. *In an SFT, the number of predecessor VNFs is always smaller than that of successor VNFs.*

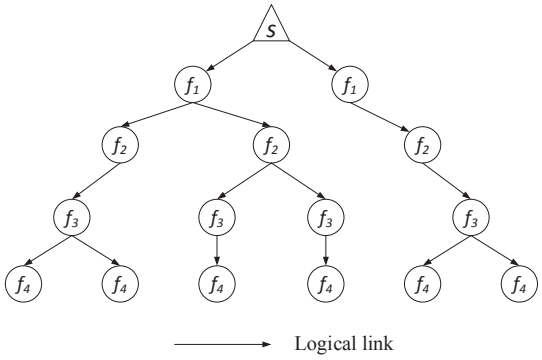


Fig. 6. An example of SFT with the SFC requirement $(f_1 \to f_2 \to f_3 \to f_4)$. Note that the links connecting VNFs are logical.

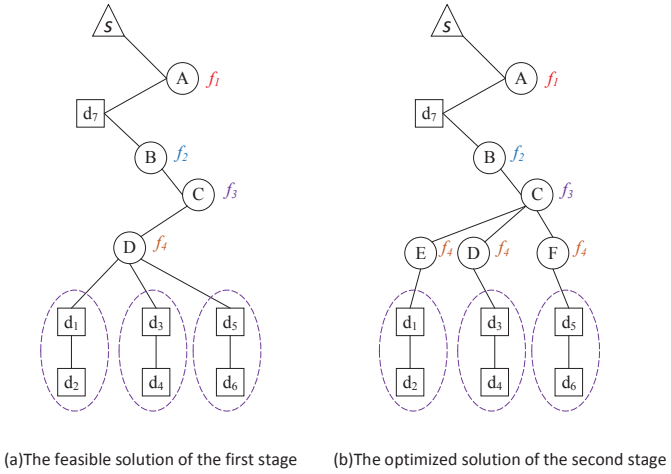


Fig. 7. An example of the second stage where all links denote the corresponding shortest paths.

Proof. In the SFT, predecessor VNFs are parent nodes of successor VNFs. All leaf nodes in the SFT must be the instances of last VNF. This property ensures that the predecessor VNFs in the SFT must have children nodes. Otherwise, these instances will not be contained by the SFT due to the minimum cost constraint. Since each parent node has at least one children node, predecessor VNFs have fewer instances than successor VNFs. Thus, Theorem 4 is proved. □

To describe stage two clearly, we provide an example to demonstrate the procedure. Suppose that there is a multicast task $\delta=\{s,\hat{D},(f_1 \to f_2 \to f_3 \to f_4)\}$, and $D = \{d_1,d_2,d_3,d_4,d_5,d_6,d_7\}$. Fig. 7(a) shows a feasible solution produced in stage one. In stage two, we optimize the feasible solution additionally by transforming the SFC into an SFT.

In the last step of the first stage, a Steiner tree is found to connect all destinations with the last VNF. As shown in Fig. 7(a), the Steiner tree connect the destinations $\{d_1,d_2,d_3,d_4,d_5,d_6,d_7\}$ and the last VNF f_4. From node D that deploys the last VNF, there are four paths for the multicast flow to all destinations (i.e., leaves of the Steiner tree $\{d_2,d_4,d_6,d_7\}$). These four paths can be divided into two categories based on whether they have common edges with the embedded SFC. In Fig. 7(a), the path $(D \to d_7)$ overlaps with the SFC, and we call such a path **dependent path**. Otherwise, like paths $(D \to d_2)$, $D \to d_4$ and $(D \to d_6)$, which have no common edges with the SFC, we call them **independent paths**. For an independent path, it may contain multiple destinations and we define the one nearest to the source as its **connection node**. In Fig. 7(b), the connection nodes are d_1, d_3 and d_5.

After identifying the dependent/independent paths as well as connection nodes, we can further optimize the solution. Since Theorem 4 shows that predecessor VNFs cannot have more instances than successor VNFs, we increase the number of VNF instances in an inverted order. Thus, as illustrated by Fig. 7(b), we further deploy new instances of f_4 in different nodes.

Rule of Adding New VNF Instances: In Fig. 7(a), each independent path gets service of f_3 and f_4 from C and D. While in Fig. 7(b), f_4 can be deployed on $\{D,E,F\}$. Denote the cost of the shortest path between two nodes v_i and v_j as $c_{v_iv_j}$. If we could find a node E such that $c_{d_1E}+c_{EC}+\gamma_{f_4E}<c_{d_1D}$, we can deploy a new instance of f_4 on E. Similarly, if $c_{d_5F}+c_{FC}+\gamma_{f_4F}<c_{d_1D}$, we can deploy a new instance of f_4 on F. After adding instances of f_4, all nodes with f_4 become the new connection nodes of each dependent path. Repeat the above procedures until one VNF cannot be deployed on multiple nodes. Theorem 4 ensures that this terminate condition is valid. The details of the second stage can be formally depicted by Algorithm 3. Overall, our two-stage algorithm (TSA) is summarized in Algorithm 4.

Algorithm 3 The Second stage of our two-stage algorithm

Input A feasible solution X_0 produced by Algorithm 2, service function chain $\ell=\{f_1,f_2,...,f_k\}$, network G, all nodes capacity and VNFs deployment cost γ_{f_ju}.
Output Optimized solution X_{opt}.
1: Find all connection nodes in X_0.
2: **for** $j=k;j--;j>0$ **do**
3: **if** f_j can be deployed in other nodes according to the rule in Section IV-C **then**
4: Deploy new instances of f_j in corresponding node, update connection nodes and $[\omega_{f_nu}],\Upsilon$
5: **else**
6: break
7: return $X_{opt}=\{[\omega_{f_nu}],\Upsilon\}$.

Algorithm 4 The overview of our two-stage algorithm (TSA).

Input A network $G=(V,E,c(E))$, a multicast task $\delta=(s,\hat{D},\ell)$ and VNF deployment costs γ_{f_ju} on all nodes.
Output A minimum cost SFT embedding solution to deliver the multicast task δ.
1: Construct the MOD G' according to Algorithm 1.
2: Embed the SFC to get the feasible solution X_0 according to Algorithm 2.
3: Find the optimal solution X_{opt} by transforming the SFC into the SFT according to Algorithm 3.

D. Algorithm for Network with Deployed VNFs

For a target network, some VNFs may have already been deployed. We modify Algorithm 1 to tackle the SFT embedding problem in such a network.

Indeed, when constructing the MOD network, those VNFs that have already been deployed in the network can be divided into two categories. For those VNFs in the SFC ℓ, we treat their setup cost as zero, since there is no need to deploy them again for a single multicast task. For those VNFs not in the SFC ℓ, they will not occupy a column in the MOD network according to Algorithm 1. Thus, we only need to modify the capacities of those nodes where VNFs have already been deployed. Then Algorithm 2 and Algorithm 3 can be applied to the modified MOD network to get the solution.

E. Algorithm Analysis

Theorem 5. *The computational complexity of the two-stage algorithm is $O((k^2+|D|)\times|V|^3)$, in which $|V|$ is the number of nodes in the target network, $|D|$ is the number of destinations and k is the length of the required SFC for each traffic flow in the multicast task.*

Proof. In the procedures of constructing the MOD network, we first need to find the shortest path between each pair of nodes in the original network. Floyd algorithm can find all shortest paths in $O(|V|^3)$ [37]. The overlay network has $k\times|V|^2$ edges and $k\ll|V|$, thus the construction of overlay network needs $O(|V|^3)+O(k\times|V|^2)=O(|V|^3)$ steps.

In the first stage, we first need to find the shortest path between S and T, and this path can be found by Dijkstra algorithm in $O(k^2\times|V|^2)$ [38]. As for the node that violates the capacity constraint, corresponding adjustment procedures need $O(k\times|V|)$ comparisons. The time complexity of finding the Steiner tree in the original network is $O(|D|\times|V|^2)$ [3]. One iteration in the first stage needs $O(k^2\times|V|^2)+O(k\times|V|)+O(|D|\times|V|^2)=O((k^2+|D|)\times|V|^2)$. Thus, the computational complexity of the first stage is $O((k^2+|D|)\times|V|^3)$.

In the second stage, the computational complexity of the feasible solution optimizing depends on the number of dependent paths and the length of SFC, which needs $O(k\times|D|)$ comparisons. Therefore, the computational complexity of all the above operations is $O(|V|^3)+O((k^2+|D|)\times|V|^3)+O(k\times|D|)=O((k^2+|D|)\times|V|^3)$. Thus, Theorem 5 is proved. □

Theorem 6. *With sufficient resources on each node, the two-stage algorithm guarantees an approximation ratio smaller than $(1+\rho)$, where ρ is the best approximation ratio of Steiner tree and can be as small as 1.39 [17].*

Proof. X_{opt} represents the optimal solution of delivering the multicast task and there will be an SFT in X_{opt}. X_{alg} is the solution generated by our algorithm, and it also has an SFT. Indeed, X_{alg} is generated after executing the second stage algorithm. In the first stage of our algorithm, a feasible solution X'_{alg} has been generated which has embed an SFC. Since X_{alg} is the solution optimized on the basis of X'_{alg}, then $c(X_{alg})<c(X'_{alg})$. X'_{alg} is composed of an SFC P_{alg} and a Steiner tree T_{alg}. X_{opt} has an SFT, and we can find an SFC P_{opt} in the SFT and a tree T_{opt} connecting all destinations. Assume the last node of P_{opt} is W, and the optimal Steiner tree covering all nodes in $W\cup D$ is $T_{W\cup D}$. There is no doubt that $c(P_{opt})<c(X_{opt})$ and $c(T_{W\cup D})<c(T_{opt})<c(X_{opt})$. In Algorithm 2, node W has been considered, and the solution can be denoted as the SFC P_{W-alg} add the $T_{W\cup D}$. According to Algorithm 2, $c(P_{alg})+c(T_{alg})<c(P_{W-alg})+c(T_{W\cup D})$.

Theorem 2 has proved that P is the optimal SFC when all nodes have sufficient resources, thus $c(P_{W-alg})<c(P_{opt})<c(X_{opt})$. The best approximation ratio of the Steiner tree can be as small as 1.39 [17]. However, in this proof, we do not concentrate the concrete value, and we use ρ denotes the approximation ratio. And we can get, $c(T_{W\cup D})<\rho c(T_{opt})<\rho c(X_{opt})$, since T_{opt} is a feasible tree to cover all destinations and W. Then $c(P_{W-alg})+c(T_{W\cup D})<c(X_{opt})+\rho c(X_{opt})$, $c(X_{alg})<c(X'_{alg})=c(P_{alg})+c(T_{alg})<(1+\rho)c(X_{opt})$. Thus, Theorem 6 is proved. □

TABLE II
PARAMETER SETTINGS.

	Parameter	Setting
Target Network	Network size	[50, 250]
	Deployed VNF	Deploy randomly
	Node Capacity	[1, 5]
	Link Connection Cost	The Euclidean distance
	VNF Deployment Cost	Relate to the connection cost
Multicast Task	S	Select randomly
	D	Select randomly
	SFC length	[5, 25]

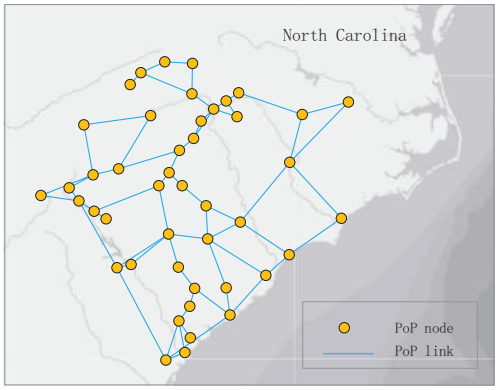


Fig. 8. The Palmetto network [39].

V. Experimental Evaluation

In this section, we first introduce the evaluation environment and methodology and then evaluate the performance of the algorithm. Our evaluations focus on quantifying the benefits of our method at delivery cost and running time reductions.

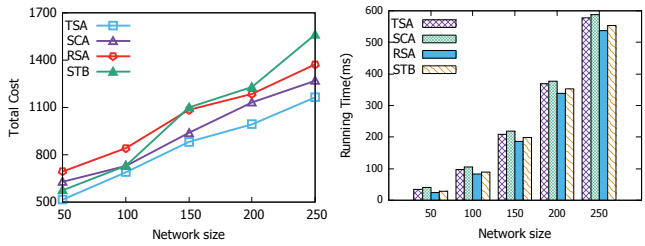
A. Experiment Design

Experiment Configurations. To evaluate our method, we first generate synthetic target networks using the ER random graphs [40]. Then, we make use of the real-world network of PalmettoNet shown in Fig. 8, which is a backbone network with 45 nodes in the U.S. [39]. With the synthetic and real-world networks, we evaluate the impacts of the multicast size (i.e., the number of destinations), the average VNF setup cost and the length of SFC (the number of VNFs). It is worth to mention that the link connection cost and VNF setup cost are prior knowledge and can be set according to the network dynamics. For example, if one link is congested or

one server node cannot support VNF deployment, we can considerably increase their costs such that the embedding of an SFT would never select such kinds of links. We always report the average results of 50 rounds in the same parameter setting. Table II summarizes the parameter settings with the following considerations:

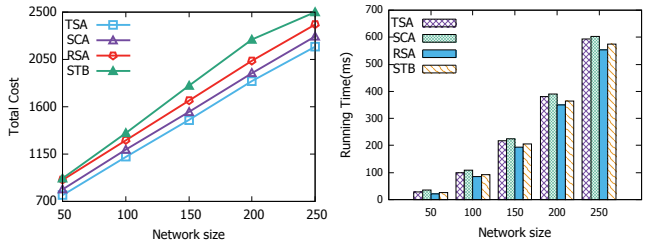
- The network size: Following the previous work [16], we set the network size in the range of $[50, 250]$, which covers small networks and large ones.
- The deployed VNFs: There are many value-added (VNF) services as the NFV market is developing [41]. We arbitrarily select thirty different VNFs and deploy some of them randomly among the server nodes (as the existed VNFs).
- The node capacity: To ease the evaluation, we set the node capacity randomly within $[1,5]$, which means at most $1 \sim 5$ VNFs can be deployed on the node.
- The link connection cost: To ease the evaluation, we set the link connection cost of any pair of nodes as the Euclidean distance between them.
- The VNF deployment cost: we set the VNF deployment cost upon any server node following the Normal distribution $N(\mu l_G, \sigma^2)$, where $\mu \in \{1,3\}$, $\sigma = l_G/4$, and l_G is the average shortest path cost of the network (to balance the costs of link connection and VNF deployment). It is vital to emphasize that the VNF deployment cost can follow any distribution.
- The size of a multicast task: For each simulated multicast task, the source node and the destination nodes are selected randomly from the target network. Furthermore, we set the value of $|D|/|V|$ as 0.1 or 0.3 to evaluate the impact of multicast size [16].
- The SFC length: As the NFV market develops, the SFC length would be increased. Thus, we set the SFC length from 5 to 25, which is twice larger than it in real deployments [42].

Benchmark Algorithms. To the best of our knowledge, this is the first work tackling the SFT embedding problem (refer to the related work). In [32], Cheng et al. have proposed a method which we call STB (Steiner tree-based). STB can solve NFV enabled multicast, but it embeds an SFC rather than an SFT. In detail, the STB method first constructs a Steiner tree covering all destinations of the multicast and find the minimum cost path to connect the source to this tree. Then STB embeds the required SFC along the minimum cost path. We treat this STB method as one of the benchmarks that evaluate the cost of transferring NFV enabled multicast. Besides, we compare our algorithm (i.e., TSA) with another two benchmarks we designed that can embed SFT: the minimum set cover algorithm (SCA) and the randomly selecting algorithm (RSA). These three algorithms are different in generating the feasible solution at the first stage, while the optimization procedure at the second stage is the same. SCA tries to occupy as few nodes as possible when embedding the SFC in the first stage. It chooses the minimum number of nodes to cover as many VNFs as possible. If some VNF has no existed instance in the network, SCA will deploy a new instance upon the nearest node to the predecessor VNF. As for RSA, it randomly selects VNFs that have been deployed. While for those VNFs that have not been deployed, RSA randomly selects nodes with sufficient capacities to deploy them. After all requested VNFs having been deployed, RSA connects them in order with the shortest paths. With various parameter settings, we compare our algorithm with these three benchmarks concerning the traffic delivery cost and the method running time. Furthermore, to evaluate the accuracy of approximated solutions, we perform experiments over the Palmetto network, as its network size is suitable to obtain the optimal solutions.



(a) The traffic delivery cost (b) The running time

Fig. 9. The changing trends of the two metrics when $|D|/|V|$=0.1



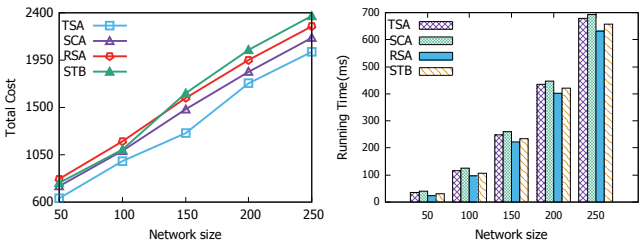
(a) The traffic delivery cost (b) The running time

Fig. 10. The changing trends of the two metrics when $|D|/|V|$=0.3

B. Evaluation with Synthetic Networks

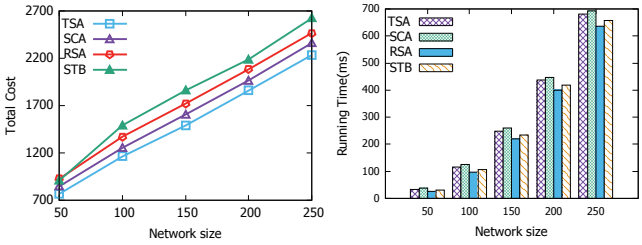
1) The impact of destination ratio: We define the destination ratio as the number of destinations in the network to the total number of network nodes, i.e., $|D|/|V|$. To concentrate on the impact of destination ratio under different network size, we fix the SFC length with 5 and the average VNF deployment cost with $\mu = 2$. Fig. 9 and Fig. 10 show the variations of traffic delivery cost and running time as the growth of network size under different destination ratios. In average, the traffic delivery costs resulting from TSA are 11.01% and 16.62% lower than those from STB shown in Fig. 9(a) and Fig. 10(a), respectively. We can see that the traffic delivery cost of the solution increases as the growth of network size. The reason for this phenomenon is that the final solution includes more links (incurring more link connection cost) as the network size increases. Also, the number of destinations increases as the network size increases, which leads to the increases of both connection links and VNF instances. Thus, comparing Fig. 9(a) with Fig. 10(a), we can find that the solutions in the latter figure possess higher traffic delivery costs.

The curves that represent the method running time are illustrated in Fig. 9(b) and Fig. 10(b). The running times of TSA can be larger than them of STB by 7.33% and 7.49%,



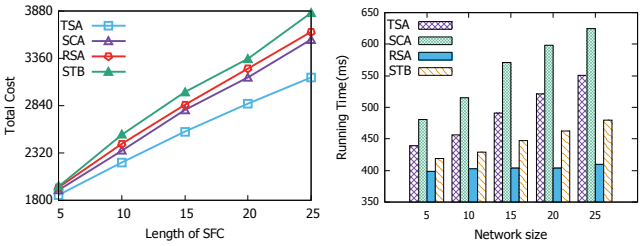
(a) The traffic delivery cost (b) The running time

Fig. 11. The changing trends of the two metrics when the average setup cost is 1× the average shortest path cost.



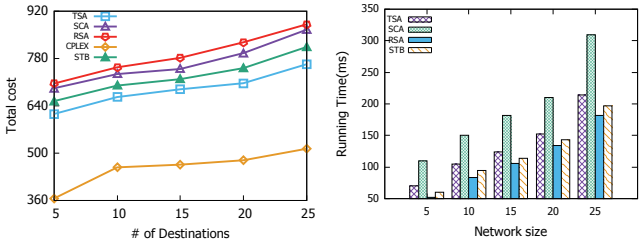
(a) The traffic delivery cost (b) The running time

Fig. 12. The changing trends of the two metrics when the average setup cost is 3× the average shortest path cost



(a) The traffic delivery cost (b) The running time

Fig. 13. The changing trends of the two metrics with different SFC lengths.



(a) The traffic delivery cost (b) The running time

Fig. 14. The changing trends of the two metrics in PalmettoNet when the numbers of destinations are different.

respectively. The extra time consumption results from the multiple comparisons in the stage one of TSA. These two figures see the same change trend, both of which increase gradually. As the number of destinations and network size increase, the time consumption increases nonlinearly. The reason behind this phenomenon is that the computation complexity in constructing the Steiner tree is nonlinear.

2) The impact of setup cost: To evaluate the impact of VNF setup cost under different network size, we restrict the value of destination ratio to 0.2 (i.e., $|D|/|V|$=0.2) and the length of SFC to 5. The curve change trends shown in these two figures are almost the same as those in Fig. 9 and Fig. 10. Compared with STB, TSA reduces the traffic delivery cost by 16.37% and 17.49% (under different VNF setup costs) in Fig. 11(a) and Fig. 12(a), respectively. Particularly, the cost reduction is up to 22.05% in Fig. 12(a). We can see that the traffic delivery cost in Fig. 12(a) is higher than that in Fig. 11(a), as higher VNF setup cost contributes more to the traffic delivery cost.

Fig. 11(b) and Fig. 12(b) are almost same, which illustrates that the average VNF setup cost does no effect in the method running time. This phenomenon agrees with what is indicated in Theorem 5.

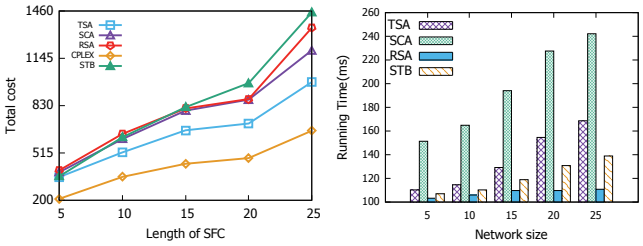
3) The impact of SFC length: To evaluate the impact of SFC length, we restrict the value of destination ratio to $|D|/|V|$=0.2, the average deployment cost to μ=3, and the network size to $|V|$=200. The results are shown in Fig. 13(a). We can see that the solutions of these four methods possess different traffic delivery costs. The performance gap between the curves of STB and TSA increases as the SFC length grow up, and the average cost reduction can be up to 12.91%. This is because longer SFC provides more opportunities to inherit deployed VNFs and more optimization space to embed SFT (rather than SFC) with our TSA.

Fig. 13(b) depicts the curves of the running time. Specifically, the curves of TSA, SCA and STB rise along with the growth of SFC length, while the curve of RSA is stable. This is because RSA randomly selects server nodes for VNF deployment without comparison, while the other three methods perform more comparisons between server nodes when deciding the VNF locations due to higher SFC length.

C. Evaluation with Real-world Network

In this experiment, we apply the real-world network topology of Palmetto and evaluate our method by changing the number of destinations and the SFC length. Furthermore, with this small-scale network, we can obtain the optimal solution using CPLEX [43], and then compare it with the designed methods.

Considering the size of this real network, we set $|D| \in [5,25]$ to evaluate the impact of the multicast size. We restrict the SFC length to 10 and set the average deployment cost with $\mu = 2$. Fig. 14 illustrates the impact of the number of destinations. In Fig. 14(a), the traffic delivery cost grows up as the number of destinations increases. Our TSA can reduce the traffic delivery cost by 5.41% and 12.86% in average against STB and RSA, respectively. Compared with the optimal solutions (illustrated by the yellow curve in Fig. 14(a)), our STA method can achieve an approximation ratio of 1.51 as far as the average algorithm performance concerned in this experiment, which is larger than the given theoretical result in Theorem 6. This gap results from the SFT embedding solution and the approximate Steiner tree. We have used the fast algorithm in [3] to construct the Steiner tree. According to Fig. 14(b), the running times of all four methods rise as the number of destinations increases. Note that the running times of CPLEX are not able to be illustrated in the figure since they are much (>30 times) higher than that of the other four methods.



(a) The traffic delivery cost (b) The running time

Fig. 15. The changing trends of the two metrics in PalmettoNet with different SFC lengths.

Fig. 15 evaluates the impact of the SFC length. In this experiment, we restrict the number of the destinations to 15 (i.e., $|D|$=15) and set the average deployment cost with $\mu = 2$. Similar to Fig. 13, the curves in Fig. 15 show the same trend. Compared with RSA and STB, our TSA can reduce the traffic delivery cost by 18.69% and 19.62% on average, respectively. The curves in Fig. 15(b) also show the same trend as Fig. 14(b), i.e., the running times of all the four methods rise as the SFC length increases.

VI. Conclusion

In NFV enabled multicast, as the costs of VNF deployment and link connection are sensitive to the VNF location, embedding a service function tree (SFT) is proved to be a better choice for cost saving than a service function chain (SFC). In this paper, we study the optimal SFT embedding problem for NFV enabled multicast. Firstly, we formally defined and formulated the optimal SFT embedding problem with an integer linear programming. Secondly, to solve the problem efficiently, we designed a two-stage algorithm by i) producing an initial feasible solution with embedding an SFC and ii) optimize the initial solution by adding new VNF instances and constructing an SFT. Then we proved that the approximation ratio of our algorithm is $1+\rho$ where ρ can be as small as 1.39, under the condition that the server nodes have sufficient resources. Finally, with extensive experimental evaluations, we disclosed the impacts of different parameters on embedding an SFT and showed that our SFT embedding method could reduce the traffic delivery cost by 22.05% at most against other three benchmarks.

References

[1] B. Ren, D. Guo, G. Tang, Y. Qin, and X. Lin, "Optimal service function tree embedding for nfv enabled multicast," in *Proc.of ICDCS*, 2018.

[2] H. L. Hao, H. H. Chun, S. S. Hsiang, Y. D. Nian, and C. W. Tsuen, "Multicast traffic engineering for software-defined networks," in *Proc.of IEEE INFOCOM*, 2016.

[3] K. L., M. G., and B. L., "A fast algorithm for steiner trees," *Acta Informatica*, vol. 15, no. 2, pp. 141–145, 1981.

[4] D. Kreutz, F. M. V. Ramos, P. J. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[5] A. Ksentini, M. Bagaa, T. Taleb, and I. Balasingham, "On using bargaining game for optimal placement of SDN controllers," in *Proc. of IEEE ICC*, 2016.

[6] S. Shan-Hsiang, H. Liang-Hao, Y. De-Nian, and C. Wen-Tsuen, "Reliable multicast routing for software-defined networks," in *Proc.of IEEE INFOCOM*, 2015.

[7] M. Sevil, K. Matthias, and K. Holger, "Specifying and placing chains of virtual network functions," in *Proc. of IEEE CloudNet*, 2014.

[8] B. Deval, J. Raj, S. Mohammed, and E. Aiman, "A survey on service function chaining," *Journal of Network and Computer Applications*, vol. 75, no. C, pp. 138–155, 2016.

[9] H. Nicolas, J. Brigitte, and G. Frédéric, "Optimization of network service chain provisioning," in *Proc.of IEEE ICC*, 2017.

[10] H. Bo, G. Vijay, J. Lusheng, and L. Seungjoon, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[11] M. Rashid, S. Joan, G. Juan-Luis, B. Niels, D. T. Filip, and B. Raouf, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[12] Z. Pamela, F. R. A, Z. X. Kelvin, M. Masaharu, and R. Jennifer, "Dynamic service chaining with dysco," in *Proc.of ACM SIGCOMM*, 2017.

[13] M. Ahmed and A. Mostafa, "Analysis of adaptive streaming for hybrid cdn/p2p live video systems," in *Proc. of IEEE ICNP*, 2011.

[14] H. Wang, G. Tang, K. Wu, and J. Fan, "Speeding up multi-cdn content delivery via traffic demand reshaping," in *Proc. of IEEE ICDCS*, 2018, pp. 422–433.

[15] G. Tang, H. Wang, K. Wu, and D. Guo, "Tapping the knowledge of dynamic traffic demands for optimal CDN design," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 98–111, 2019.

[16] X. Zichuan, L. Weifa, H. Meitian, J. Mike, G. Song, and G. Alex, "Approximation and online algorithms for nfv-enabled multicasting in sdns," in *Proc.of IEEE ICDCS*, 2017.

[17] B. Jaroslaw and G. Fabrizio, "An improved lp-based approximation for steiner tree," in *ACM Symposium on Theory of Computing*, 2010.

[18] M. Hendrik and D. T. Filip, "Vnf-p: A model for efficient placement of virtualized network functions," in *Proc.of CNSM*, 2014.

[19] B. Mathieu, L. Jeremie, and C. Vania, "Cost-based placement of virtualized deep packet inspection functions in sdn," in *Proc.of IEEE MILCOM*, 2013.

[20] M. Bagaa, T. Taleb, and A. Ksentini, "Service-aware network function placement for efficient traffic handling in carrier cloud," in *Proc. of IEEE WCNC*, 2014.

[21] C. Rami, L.-E. Liane, N. J. Seffi, and R. Danny, "Near optimal placement of virtual network functions," in *Proc.of IEEE INFOCOM*, 2015.

[22] A. Laghrissi, T. Taleb, and M. Bagaa, "Conformal mapping for optimal network slice planning based on canonical domains," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 519–528, 2018.

[23] M. Bagaa, T. Taleb, A. Laghrissi, A. Ksentini, and H. Flinck, "Coalitional game for the creation of efficient virtual core network slices in 5g mobile systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 469–484, 2018.

[24] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck, "Towards edge slicing: VNF placement algorithms for a dynamic & realistic edge cloud environment," in *Proc. of IEEE GLOBECOM*, 2017.

[25] T. Taleb, M. Bagaa, and A. Ksentini, "User mobility-aware virtual network function placement for virtual 5g network infrastructure," in *Proc. of IEEE ICC*, 2015, pp. 3879–3884.

[26] K. T. Wei, L. B. Heng, L. C. Ju, and T. M. Jer, "Deploying chains of virtual network functions: On the relation between link and server usage," in *Proc.of IEEE INFOCOM*, 2016.

[27] X. Lin, D. Guo, Y. Shen, G. Tang, and B. Ren, "DAG-SFC: minimize the embedding cost of SFC with parallel vnfs," in *Proc. of ICPP*, 2018.

[28] G. Linqi, P. John, and W. Anwar, "Dynamic service function chaining in sdn-enabled networks with middleboxes," in *Proc.of IEEE ICNP*, 2016.

[29] B. Deval, S. Mohammed, E. Aiman, J. Raj, G. Lav, and C. H. Anthony, "Optimal virtual network function placement in multi-cloud service function chaining architecture," *Computer Communications*, vol. 102, pp. 1–16, 2017.

[30] K. Jian-Jhih, S. Shan-Hsiang, Y. Ming-Hong, Y. De-Nian, T. Ming-Jer, and C. Wen-Tsuen, "Service overlay forest embedding for software-defined cloud networks," in *Proc.of IEEE ICDCS*, 2017.

[31] B. Yi, X. Wang, M. Huang, and A. Dong, "A multi-stage solution for nfv-enabled multicast over the hybrid infrastructure," *IEEE Communications Letters*, vol. 21, no. 9, pp. 2061–2064, 2017.

[32] Y. Cheng and L. Yang, "VNF deployment and routing for nfv-enabled multicast: A steiner tree-based approach," in *9th International Conference on Wireless Communications and Signal Processing*, 2017.

[33] S. Ratnasamy, A. Ermolinskiy, and S. Shenker, "Revisiting ip multicast," in *Proc. of ACM SIGCOMM*, 2006.

[34] X. Li and M. J. Freedman, "Scaling ip multicast on datacenter topologies," in *Proc. of CoNEXT*, 2013.

[35] I. Aakash, K. Praveen, and M. Vijay, "Avalanche: Data center multicast using software defined networking," in *Proc.of IEEE COMSNETS*, 2014.
[36] 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 Transaction on Networking.*, vol. 25, no. 4, pp. 2008–2025, 2017.
[37] F. R. W, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.
[38] D. E. W, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
[39] "The internet topology," http://topology-zoo.org/maps/.
[40] P. ErdóS and A. Rényi, "On random graphs i," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.
[41] G. Brown, S. Analyst, and H. Reading, "White paper: Service chaining in carrier networks," http://www.qosmos.com/, 2015.
[42] M. C. Luizelli, D. Raz, and Y. Sa'ar, "Optimizing NFV chain deployment through minimizing the cost of virtual switching," in *Proc. of INFOCOM*, 2018.
[43] "IBM ILOG CPLEX," https://www.ibm.com/products/ilog-cplex-optimization-studio/.

bangbang BangBang Ren received the B.S.degree and M.S. degree in management science and engineering from National University of Defense Technology in 2015 and 2017, respectively. He is currently working toward Ph.D in College of Systems Engineering, National University of Defense Technology, Changsha, China. His research interests include software-defined network, network function virtualization, approximation algorithm.

Deke Guo Deke Guo received the B.S. degree in industry engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001, and the Ph.D. degree in management science and engineering from the National University of Defense Technology, Changsha, China, in 2008. He is currently a Professor with the College of System Engineering, National University of Defense Technology, and is also with the College of Intelligence and Computing, Tianjin University. His research interests include distributed systems, software-defined networking, data center networking, wireless and mobile systems, and interconnection networks. He is a senior member of the IEEE and a member of the ACM.

Yulong Shen Yulong Shen received the B.S. and M.S. degrees in computer science and the Ph.D. degree in cryptography from Xidian University, Xian, China, in 2002, 2005, and 2008, respectively. He is currently a Professor with the School of Computer Science and Technology, Xidian University, and also an Associate Director of the Shaanxi Key Laboratory of Network and System Security. He has also served on the technical program committees of several international conferences, including the NANA the ICEBE, the INCoS, the CIS, and the SOWN. His research interests include wireless network security and cloud computing security.

Guoming Tang (S'12-M'17) is an Assistant Professor at the Key Laboratory of Science and Technology on Information System Engineering, National University of Defense Technology (NUDT), China. He received the Ph.D. degree in Computer Science from the University of Victoria, Canada, in 2017, and both the Bachelor's and Master's degrees from NUDT in 2010 and 2012, respectively. Aided by machine learning and optimization techniques, his research focuses on computational sustainability in distributed and networking systems.

Xu Lin Xu Lin received the B.S. degrees in computer science from Xidian University, Xian, China, in 2016. He is currently a Professor with the School of Computer Science and Technology, Xidian University. He is currently working toward Ph.D in the School of Computer Science and Technology, National University of Defense Technology, Changsha, China. His research interests include software-defined network, network function virtualization, network security.