QoS-aware Management Reconfiguration of vNF Service Trees with Heterogeneous NFV Platforms

Tingyu Li and Zuqing Zhu, *Senior Member, IEEE*

***Abstract*—The rapid development of in-network computing motivates people to consider deploying virtual network functions (vNFs) on heterogeneous platforms that include both software systems like virtual machines and docker containers and hardware systems like programmable data plane switches. Meanwhile, with the emergence of multi-client network services, service providers need to build vNF service trees (vNF-STs) in their substrate networks (SNTs). In this work, we study how to optimize the management reconfiguration of vNF-STs in an SNT equipped with heterogeneous platforms. An integer linear programming (ILP) model is first formulated to consider three common reasons for vNF-ST reconfiguration and reduce both the total resource usage after reconfiguration and the overall vNF migration cost during reconfiguration. Then, we design a two-step algorithm to reduce the time complexity of problem-solving. Specifically, the algorithm first checks all the active vNF-STs to select the vNF-STs that should be reconfigured, and then leverages an approach based on layered auxiliary graphs (LAGs) to reconfigure the selected vNF-STs. Extensive simulations verify the effectiveness of our algorithms on optimizing the management reconfiguration, and demonstrate that they can outperform existing benchmarks.**

***Index Terms*—Network function virtualization (NFV), Heterogeneous NFV platforms, Service function trees.**

I. Introduction

THESE days, to adapt to the explosive growth of data, users, and applications [1, 2] in the Internet, networking technologies have made tremendous advances, especially on physical-layer [3–6] and virtualization technologies [7–9]. This has promoted the idea of network function virtualization (NFV) [10], which was developed to address drawbacks of the traditional way of deploying network services with special-purpose middle-boxes (*e.g.*, high cost, complicated maintenance, and long time-to-market). Specifically, NFV abstracts network services as virtual network functions (vNFs) and realizes them on general-purpose software/hardware platforms, such as virtual machines (VMs), docker containers (Dockers), and programmable data plane switches (PDP-SWs) [11, 12].

To satisfy the service demands of clients, service providers (SPs) can organize vNFs in various topologies (*e.g.*, chains [13], trees [14], and generic forwarding graphs [15]) and steer application traffic through them. In a vNF service chain (vNF-SC), the vNFs are arranged in a line to process application traffic in sequence. In addition, the emergence of multi-client network services, such as webcasts, online multiplayer games, and the Metaverse, has motivated SPs to arrange vNFs in a tree-type forwarding graph (*i.e.*, vNF service tree (vNF-ST)), which can process traffic with multicast characteristics [14]. Specifically, a vNF-ST is set up to fulfill the demands for point-to-multiple-point communications, one vNF in it can process the application traffic to multiple clients, and each branch of the vNF-ST is actually a vNF-SC, corresponding to the network service of a client.

Previously, there have been a few studies on the provisioning schemes of vNF-STs [14, 16–22] in various networks with different optimization objectives. Although the algorithm designs in them are interesting, none of these studies have considered to deploy vNFs on heterogeneous NFV platforms [23]. Note that, vNFs were traditionally instantiated on software platforms such as VMs and Dockers, but with the advent of in-network computing [24], SPs can handle computing tasks in packet processing pipelines, which is equivalent to realize vNFs on hardware platforms (*e.g.*, PDP-SWs).

Software platforms are usually runtime programmable, have advantages in cost, memory space and resiliency, and thus can support computing-intensive vNFs effectively [25]. However, the major inherent drawback of them is that their throughput and processing latency might not be suitable for bandwidth-intensive vNFs [23]. On the other hand, PDP-SWs have excellent traffic processing performance and thus can guarantee high throughput and low processing latency [11], but they are more expensive and less flexible than the software platforms. Therefore, if one combines the hardware (PDP-SWs) and software (VMs and Dockers) platforms to form a network system that consists of heterogeneous NFV platforms, the benefits of both platforms can be unified to give SPs more flexibility to fulfill various quality-of-service (QoS) requirements cost-efficiently.

Note that, heterogeneous NFV platforms can actually bring more advantages to the provisioning of vNF-STs than that of VNF-SCs. This is because a vNF in one vNF-ST can be shared by multiple branches, and so can its benefits. Moreover, if we consider the practical network environment where management reconfiguration can be triggered occasionally, the advantages of heterogeneous NFV platforms can be further explored [26]. Specifically, management reconfiguration can be invoked for various reasons. First, when the capacities of the NFV platforms in an SP's network cannot catch up with the increasing service demands from clients, the SP will need to install new NFV platforms and reconfigure certain active vNFs onto them for re-satisfying QoS requirements [27]. Second, NFV platforms can have exceptions or even become unusable, and these issues actually happen more frequently than we expected in commercial networks [28]. Hence, an SP needs to reconfigure/migrate vNFs to restore the affected network services. Finally, management reconfiguration can be triggered

T. Li and Z. Zhu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, P. R. China (email: zqzhu@ieee.org).

Manuscript received on July 31, 2022.

to address the changes made by clients, *e.g.*, changes on access points, required vNFs, and QoS demands [29].

To this end, it is relevant to study QoS-aware management reconfiguration of vNF-STs in a network equipped with heterogeneous NFV platforms. Nevertheless, to the best of our knowledge, this problem has not been considered in the literature, except for our preliminary work in [30]. In [30], we briefly described the problem and designed a greedy-based heuristic to solve it. However, we did not formulate a solid optimization model to tackle the problem, the heuristic was not optimized to avoid being trapped in a local optimum, and the tradeoff between the complexity and performance of QoS-aware management reconfiguration was not analyzed.

The aforementioned issues motivated us to significantly expand our study in this work for a much more comprehensive investigation. We first jointly consider the cases of management reconfiguration to formulate an integer linear programming (ILP) model, which can provide the exact solution of the problem. Then, to solve the problem more time-efficiently, we propose a two-step algorithm that runs in polynomial time. The algorithm first checks the active vNF-STs and the target of a management reconfiguration to select the vNF-STs that should be reconfigured, and then it leverages an approach based on layered auxiliary graphs (LAGs) to get the reconfiguration schemes of the selected vNF-STs. Extensive simulations verify the effectiveness of our proposed algorithms, and analyze the tradeoff between the complexity and performance of the QoS-aware management reconfiguration in depth.

The rest of this article is organized as follows. We survey the related work in Section II. In Section III, we define the problem of QoS-aware management reconfiguration of vNF-STs with heterogeneous NFV platforms. The algorithm design is described in Section IV, including both the ILP and the two-step algorithm. Section V discusses the simulation results. Finally, we summarize the paper in Section VI.

II. Related Work

Nowadays, NFV has attracted many interests from academia and industry, and its framework, requirements, and typical use cases have been elaborated in a few standardization documents [10, 31]. In order to serve one vNF-ST in a substrate network (SNT), the SP needs to first instantiate the required vNFs with the NFV platforms on substrate nodes (SNs) and then route application traffic among the deployed vNFs over substrate links (SLs), such that each client of the vNF-ST can get the required network service [14]. We hope to point out that the provisioning of vNF-STs is fundamentally different from the famous virtual network embedding (VNE) [32, 33]. This is because an SP can deploy multiple vNFs of a vNF-ST on one SN, which does not follow the principle of one-to-one node mapping in VNE and complicates the algorithm design [14].

Previously, there have been a few studies on the provisioning schemes of vNF-STs [14, 16–22], which considered various networks and tackled the problem with different optimization objectives. The study in [16] addressed the problem in a packet network based on software-defined networking (SDN) and proposed an approximation algorithm and several heuristics. Zeng *et al.* [14] assumed that the SNT is a flexible-grid elastic optical network (EON) [34], and thus the traffic routing of vNF-STs needs to consider the well-known routing and spectrum assignment (RSA) problem and minimize the spectrum fragmentation on fiber links [35]. The authors of [17] addressed the network environment where NFV platforms and special-purpose middle-boxes coexist, and designed a multi-stage algorithm to provision vNF-STs in it. The approach for jointly optimizing the vNF placement and multicast routing in a 5G core network was studied in [18]. Ren *et al.* [19] proposed a two-stage approximation algorithm based on Steiner trees to solve the provisioning of vNF-STs in a packet network. The authors of [20] discussed how to save IT resources by merging the vNF-STs whose sources are the same. The studies in [21] and [22] addressed the provisioning schemes of vNF-STs in mobile edge clouds and wireless networks, respectively. However, all of these studies did not consider management reconfiguration or heterogeneous NFV platforms.

Besides one-time service provisioning, the algorithms for reconfiguring vNF-STs dynamically should also be studied. Nevertheless, although the reconfiguration schemes of vNF-SCs have been addressed in a number of studies [29, 36–38] and system frameworks such as FAST [39] and ShareOn [40] have been developed to implement the decisions made by vNF-SC reconfiguration algorithms, how to reconfigure vNF-STs for various reasons was seldom covered in the literature due to its complexity. On the other hand, the existing studies that considered heterogeneous NFV platforms (*i.e.*, VMs, Dockers, and PDP-SWs) were also few. Here, the heterogeneous NFV platforms are different from the hybrid platforms that include NFV platforms and special-purpose middle-boxes [17]. Specifically, although both middle-boxes and PDP-SWs are hardware-based, each middle-box usually can only support one type of vNFs while a PDP-SW can carry many types.

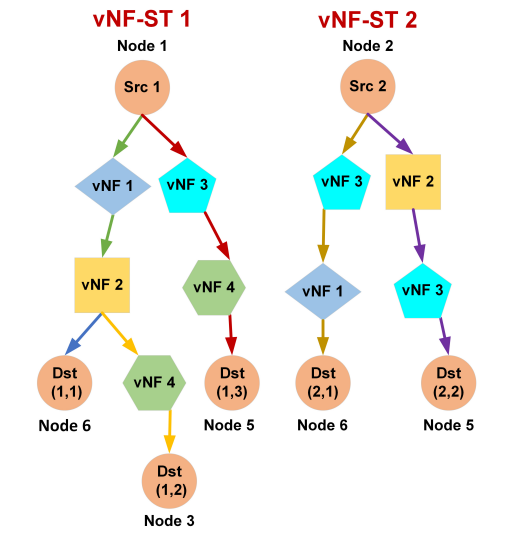
Sun *et al.* [41] designed HYPER for the service provisioning with heterogeneous NFV platforms, but the study was focused on system implementation and did not address management reconfiguration of vNF-STs. In [42], we discussed how to provision vNF-SCs with heterogeneous NFV platforms, formulated an ILP model, and proposed an approximation algorithm based on randomized rounding. The authors of [43] tackled a similar problem in data-center networks and designed a greedy-based heuristic. In [26], we extended the work in [42] to study the service upgrade of vNF-SCs for better QoS satisfaction. However, none of the studies above considered vNF-STs. Hence, to the best of our knowledge, the problem of management reconfiguration of vNF-STs with heterogeneous platforms has only been tackled preliminarily in [30].

III. Problem Description

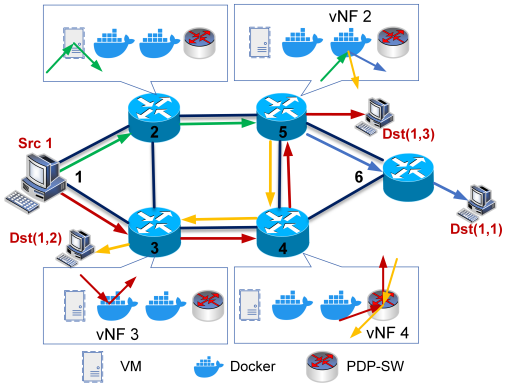
This section first describes the network model for the deployment and reconfiguration of vNF-STs with heterogeneous NFV platforms, and then explains the major reasons for reconfiguring vNF-STs in management reconfiguration.

A. Network Model

We model the topology of the SNT as an undirected graph $G(V, E)$, where V and E are the sets of SNs and SLs,



(a) Topologies of vNF-STs



(b) Provisioning scheme of *vNF-ST* 1

Fig. 1. Example on deploying vNF-STs on heterogeneous NFV platforms.

respectively. Each SN $v \in V$ contains p_v heterogeneous NFV platforms. This work considers three types of heterogeneous NFV platforms, which are VMs, Dockers and PDP-SWs, and their numbers on an SN v are p_v^V, p_v^D, and p_v^T, respectively. Apparently, we have $p_v = p_v^V + p_v^D + p_v^T$. We assume that t_{total} types of vNFs can be deployed on the NFV platforms, and each NFV platform can only carry one type of vNFs in runtime. The deployment of vNFs on NFV platforms mainly consumes two types of resources: memory space and bandwidth. We define the memory sizes of the three platforms on an SN v as C_v^V, C_v^D, and C_v^T, respectively. As for each vNF of type $t \in t_{\text{total}}$, its memory usages on the platforms are γ_t^V, γ_t^D, and γ_t^T, respectively. Similarly, if we deploy a vNF of type t on the platforms, the bandwidth capacities of the platforms respectively become B_t^V, B_t^D, and B_t^T, and the processing latencies on them are ρ_t^V, ρ_t^D, and ρ_t^T, respectively.

Each vNF-ST is defined as $ST_i(s_i, D_i, R_i, b_i)$, where i is its unique index, s_i denotes the source, D_i stands for the destination set, R_i is the required vNF-ST, and b_i represents its bandwidth demand. Then, for the j-th destination $d_{i,j} \in D_i$, its branch is essentially a vNF-SC, which can be denoted as $r_{i,j}(s_i, d_{i,j}, F_{i,j}, \tau_{i,j})$. Here, $F_{i,j} = \{f_{i,j,1}, \cdots, f_{i,j,N_{i,j}}\}$ is the branch's vNF-SC, where $f_{i,j,l}$ denotes the l-th vNF in the vNF-SC and $N_{i,j}$ is the length of the vNF-SC, and $\tau_{i,j}$ is the end-to-end (E2E) latency tolerated by the vNF-SC.

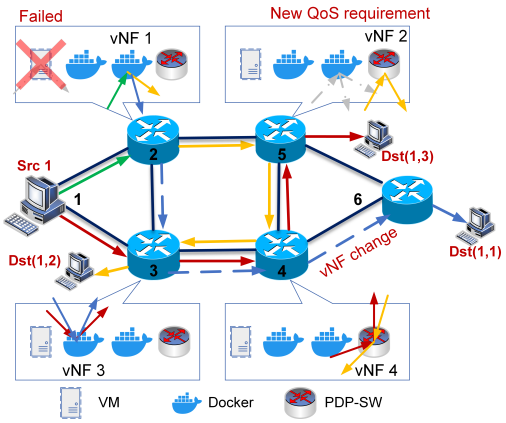


Fig. 2. Management reconfiguration of vNF-STs for various reasons.

Fig. 1 shows an example on the deployment of vNF-STs in an SNT with heterogeneous NFV platforms. There are two vNF-STs in Fig. 1(a), which respectively consist of 3 and 2 branches. Each branch represents a vNF-SC between one source-destination pair, and we refer to it as a *request* in this work. Different requests in a vNF-ST can share vNFs, *e.g.*, the segment of *Node* 1→*vNF* 1→*vNF* 2 is shared by the requests to *Nodes* 6 and 3 in *vNF-SC* 1. If a segment of one vNF-ST is shared by different requests, we only need to allocate one copy of resources (*i.e.*, memory space and bandwidth) on it for the requests. When the application traffic exits the shared segment, it will be replicated by the last vNF of the segment and multicasted to different branches. Moreover, when the related resource capacities allow, multiple vNF-STs can also share a same vNF. For instance, the request to *Node* 5 in *vNF-ST* 1 and that to *Node* 6 in *vNF-ST* 2 can share a *vNF* 3. Fig. 1(b) plots the provisioning scheme of *vNF-ST* 1 in the six-node SNT, where we specify how to deploy the required vNFs on the heterogeneous NFV platforms on SNs and how to steer application traffic through the vNFs.

B. Management Reconfiguration of vNF-STs

In this work, we assume that the management reconfiguration of vNF-SCs can be triggered for three reasons: 1) a request changes its vNF-SC, 2) a request changes its QoS requirement on E2E latency, and 3) an NFV platform fails. Based on the provisioning scheme of *vNF-ST* 1 in Fig. 1(b), Fig. 2 provides illustrative examples to explain the three scenarios. To show the first scenario, the request to *Node* 6 in *vNF-ST* 1 changes its vNF-SC from *Node* 1→*vNF* 1→*vNF* 2→*Node* 6 to *Node* 1→*vNF* 1→*vNF* 3→*Node* 6. Hence, the SP changes the routing path of its vNF-SC from 1→2→5→6 to 1→2→3→4→6, where the numbers are for the nodes in the SNT. As for the second scenario, the request to *Node* 3 tightens its QoS requirement on E2E latency, and thus the SP needs to deploy a new *vNF* 2 on one PDP-SW on *Node* 5 and reconfigure the request to use it. Finally, the third scenario assumes that the *vNF* 1, which was deployed on a VM on *Node* 2, is down. Therefore, a new *vNF* 1 should be instantiated

there (*e.g.*, on a Docker) for service recovery. We can see that the management reconfiguration, which jointly considers these three reasons, can be complicated, and thus it will be challenging to find the overall optimal reconfiguration scheme.

IV. Algorithm Design

In this section, we design the algorithms for optimizing the management reconfiguration of vNF-STs. Specifically, we first describe the overall procedure, then formulate an ILP model to solve the problem exactly, and finally design a two-step heuristic to improve the time efficiency of problem-solving.

A. Overall Procedure

Algorithm 1 shows the overall procedure of the management reconfiguration of vNF-STs in a dynamic SNT. *Line* 1 initializes the set of requests that need to be reconfigured ($\mathbf{R}'$) and the system timer ($\mathcal{T}$). Then, when the SNT is operational, its SP collects the requests that need to be reconfigured for the three reasons mentioned in Section III-B (*Lines* 2-3). If the SP finds such a request r, it first inserts the request into set $\mathbf{R}'$ (*Line* 4). Then, the SP checks the reason of the reconfiguration for r, and if the timer $\mathcal{T}$ reaches the preset time interval T_0 or r needs to be reconfigure to address a platform failure, it invokes a management reconfiguration to at least reconfigure all the requests in $\mathbf{R}'$ (*Lines* 5-6). *Line* 7 resets $\mathbf{R}'$ and $\mathcal{T}$.

Algorithm 1: Procedure of Management Reconfiguration

```
1  R' = ∅, T = 0;
2  while the SNT is operational do
3      if a request r needs to be reconfigured then
4          insert r into R';
5          if r is due to platform failure or T ≥ T_0 then
6              invoke management reconfiguration with R';
7              R' = ∅, T = 0;
8          end
9      end
10 end
```

B. ILP Model

The optimization for each management reconfiguration can be described with an ILP model. In order to make the ILP model as compact as possible, we extend our approach based on layered auxiliary graphs (LAGs), which was proposed in [42]. Specifically, we generalize the LAG-based approach for modeling vNF-SCs in [42] to make sure that it can model vNF-STs. As a vNF-ST is modeled as $ST_i(s_i, D_i, R_i, b_i)$, we denote the request of the j-th destination $d_{i,j} \in D_i$ as $r_{i,j}(s_i, d_{i,j}, F_{i,j}, \tau_{i,j})$, where $F_{i,j} = \{f_{i,j,1}, \cdots, f_{i,j,N_{i,j}}\}$ explains how its vNF-SC is composed, *i.e.*, $f_{i,j,l}$ denotes the l-th vNF in it and $N_{i,j}$ is the length of the vNF-SC. Then, we use several LAGs that are derived from the SNT's topology $G(V, E)$ to get an integrated view during modeling [42].

The method for obtaining the LAGs is as follows. First, for each request $r_{i,j}(s_i, d_{i,j}, F_{i,j}, \tau_{i,j})$ in the vNF-ST, we divide its vNF-SC $F_{i,j} = \{f_{i,j,1}, \cdots, f_{i,j,N_{i,j}}\}$ into $(N_{i,j}+1)$ segments, each of which is for the connection between two adjacent vNFs. Therefore, the segments are between $\{s_i, f_{i,j,1}\}$, $\cdots$, $\{f_{i,j,l-1}, f_{i,j,l}\}$, $\cdots$, $\{f_{i,j,N_{i,j}}, d_i\}$. Second, we duplicate the SNT's topology for N^* times to obtain N^* LAGs.

$$N^* = \max_i (N_{i,j} + 1). \tag{1}$$

The LAGs are denoted as $\{G_0, \cdots, G_n, \cdots, G_{N^*-1}\}$. Then, the provisioning of each vNF-ST $ST_i(s_i, D_i, R_i, b_i)$ is accomplished with $(N_{i,j}+1)$ LAGs. Specifically, for each request $r_{i,j}$ in the vNF-ST, we serve its first segment ($\{s_i, f_{i,j,1}\}$) in G_0 and so on, so forth, *i.e.*, the segment for $\{f_{i,j,l-1}, f_{i,j,l}\}$ is provisioned in G_{l-1}. Then, the edges between two adjacent LAGs G_{l-1} and G_l represent the l-th vNFs in all the requests in the vNF-ST, which are $\{f_{i,j,l} : \forall j \in [1, |D_i|]\}$.

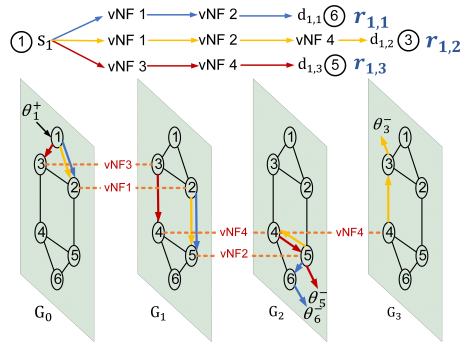


Fig. 3. Example on leveraging LAGs to provision a vNF-ST.

Fig. 3 shows an example on LAGs. As the longest vNF-SC in the vNF-ST (*i.e.*, $r_{1,2}$) contains 3 vNFs, we duplicate the six-node topology of the SNT to get 4 LAGs. Here, G_0 is for the first segments in the requests, and since the source of the vNF-ST is *Node* 1, we place a dummy platform θ_1^+ to point to *Node* 1, and *vNFs* 1 and 3 are deployed on *Nodes* 2 and 3, respectively, to connect G_0 with G_1. This procedure is repeated for each LAG until G_3, where *vNF* 4 is deployed on *Node* 4 to connect G_2 with G_3 and the provisioning of $r_{1,2}$ ends at its destination (*Node* 3) with a dummy platform θ_3^-.

The ILP model is formulated as follows, where its notations and variables are listed in Tables I and II, respectively.

Objective:

We can see that the cost of the management reconfiguration of vNF-STs mainly comes from two parts: 1) the cost of total resource usage after reconfiguration, and 2) the overall reconfiguration cost. We define the optimization objective as

$$\text{Minimize} \qquad \alpha \cdot (\mathcal{B} + \mathcal{F}) + (1 - \alpha) \cdot \mathcal{M}, \tag{2}$$

where $\mathcal{B}$ and $\mathcal{F}$ respectively denote the total usages of bandwidth and IT resources, $\mathcal{M}$ is the overall reconfiguration cost due to migrating vNFs, and $\alpha \in (0, 1)$ is the weight coefficient to balance the importance of the two aspects. Note that, when reconfiguring a vNF-ST, we might need to replace a vNF with a new one, and to ensure that such a reconfiguration is hitless to the service of the vNF-ST, the state information of the old vNF needs to be transferred to the new one. This actually generates additional operational cost, which is the reconfiguration cost considered in this work. Apparently, the

TABLE I
PARAMETERS OF ILP

Context	Notation	Description
Network Topology	$G(V,E)$	the topology of the SNT.
	$ST_i(s_i, D_i, R_i, b_i)$	the i-th vNF-ST in the SNT.
	$r_{i,j}(s_i, d_{i,j}, F_{i,j}, \tau_{i,j})$	the j-th request in ST_i.
	G_n	the n-th LAG obtained from the SNT.
vNF Management Reconfiguration	t_{total}	the set of vNF types that are supported in the SNT.
	$\eta^t_{i,j,l}$	the boolean parameter that equals 1 if the l-th vNF in $r_{i,j}$ is one of type t, and 0 otherwise.
	$\mu_{v,k}/\zeta_{v,k}/\xi_{v,k}$	the boolean parameter that equals 1 if the k-th platform of $v \in V$ is a VM/Docker/PDP-SW, respectively, and 0 otherwise.
	$C^V_v/C^D_v/C^T_v$	the memory size of VMs/Dockers/PDPSWs of SN v, respectively.
	$\hat{c}^V_t/\hat{c}^D_t/\hat{c}^T_t$	the memory space used by one vNF of type t on a VM/Docker/PDP-SW, respectively.
	$B^V_t/B^D_t/B^T_t$	the bandwidth capacity of a VM/Docker/ PDP-SW when carrying vNFs of type t.
	$\rho^V_t/\rho^D_t/\rho^T_t$	the processing latency of a VM/Docker/PDPSW when carrying vNFs of type t.
	$\tilde{x}^{v,k}_{i,j,l}$	the boolean parameter that equals 1 if the l-th vNF of $r_{i,j}$ is deployed on the k-th platform of SN v before the management reconfiguration, and 0 otherwise.
	$\gamma^V_t/\gamma^D_t/\gamma^T_t$	the cost of deploying a vNF of type t on a VM/Docker/PDP-SW, respectively.
Links	m^{v,k_2}_{u,k_1}	the cost of migrating a vNF from the k_1-th platform of SN u to the k_2-th platform of SN v, and we assume that it is irrelevant to the type of the vNF.
	θ^+_v/θ^-_v	the dummy platform to/from SN v if it is the source/destination of a request.
	β	the unit cost of the bandwidth usage on an SL.
	$\lambda_{(u,v)}$	the transmission delay of link $(u,v) \in E$.

TABLE II
VARIABLES OF ILP

Variable	Description
$x^{v,k}_{i,j,l}$	the boolean variable that equals 1 if the l-th vNF of $r_{i,j}$ is on the k-th platform of SN v, and 0 otherwise.
$h_{v,k,t}$	the boolean variable that equals 1 if a type-t vNF is deployed on the k-th platform of SN v, and 0 otherwise.
$g^{u,k_1,v,k_2}_{i,j,n}$	the boolean variable that equals 1 if the traffic of request $r_{i,j}$ is routed over the link between the k_1-th platform of SN u and the k_2-th platform of SN v in LAG G_n, and 0 otherwise.
$\delta^{u,k_1,v,k_2}_{i,j,l}$	the boolean variable that equals 1 if the l-th vNF of request $r_{i,j}$ is migrated from k_1-th platform of SN u to the k_2-th platform of SN v, and 0 otherwise.

reconfiguration cost increases with the number of requests that were using the old vNF. Hence, we assume that if the old vNF was deployed on the k_1-th platform of SN u and was serving n requests, the reconfiguration cost is $n \cdot m^{v,k_2}_{u,k_1}$ if the new vNF is deployed on the k_2-th platform of SN v. To this end, the $\mathcal{B}$, $\mathcal{F}$, and $\mathcal{M}$ in Eq. (2) can be calculated as

$$\begin{cases} \mathcal{B} = \sum\limits_{i,j}\sum\limits_{n}\sum\limits_{(u,v)\in E,k_1,k_2} \beta \cdot b_i \cdot g^{u,k_1,v,k_2}_{i,j,n}, \\ \mathcal{F} = \sum\limits_{t}\sum\limits_{v,k} h_{v,k,t} \cdot \left(\gamma^V_t \mu_{v,k} + \gamma^D_t \cdot \zeta_{v,k} + \gamma^T_t \cdot \xi_{v,k}\right), \\ \mathcal{M} = \sum\limits_{i,j,l}\sum\limits_{u,v,k_1,k_2} \delta^{u,k_1,v,k_2}_{i,j,l} \cdot m^{v,k_2}_{u,k_1}. \end{cases} \quad (3)$$

Constraints:

1) Constraints on vNF Placements after Reconfiguration:

$$\sum_{v,k} x^{v,k}_{i,j,l} = 1, \quad \forall i,j,l. \quad (4)$$

Eq. (4) ensures that each vNF in the vNF-STs is deployed on one and only one platform after management reconfiguration.

$$\sum_{t} h_{v,k,t} \le 1, \quad \forall v,k. \quad (5)$$

Eq. (5) ensures that at most one type of vNFs can be deployed on one NFV platform of each SN.

$$\begin{cases} \sum\limits_{i,j,l} \eta^t_{i,j,l} \cdot x^{v,k}_{i,j,l} > (h_{v,k,t} - 1) \cdot \left(1 + \sum\limits_{i,j,l} \eta^t_{i,j,l}\right), \\ \sum\limits_{i,j,l} \eta^t_{i,j,l} \cdot x^{v,k}_{i,j,l} \le h_{v,k,t} \cdot \sum\limits_{i,j,l} \eta^t_{i,j,l} \end{cases}, \quad \forall v,k,t, \quad (6)$$

$$h_{v,k,t} \ge \eta^t_{i,j,l} \cdot x^{v,k}_{i,j,l}, \quad \forall v,k,t,i,j,l. \quad (7)$$

Eqs. (6)-(7) ensure that the values of variables $\{x^{v,k}_{i,j,l}\}$ and $\{h_{v,k,t}\}$ are set correctly based on the relations among them.

$$\begin{cases} \delta_{i,j,l}^{u,k_1,v,k_2} = 0, & \text{if } u = v \text{ and } k_1 = k_2, \\ \delta_{i,j,l}^{u,k_1,v,k_2} \geq x_{i,j,l}^{v,k_2} + \tilde{x}_{i,j,l}^{u,k_1} - 1, & \text{otherwise}, \end{cases} \quad \forall i, j, l, u, v, k_1, k_2. \tag{8}$$

Eq. (8) ensures that the values of variables $\{\delta_{i,j,l}^{u,k_1,v,k_2}\}$ are calculated correctly to describe the vNF reconfigurations.

2) Constraints on Resource Allocation:

$$\begin{cases} \sum_{t,k} \hat{c}_t^V \cdot h_{v,k,t} \cdot \mu_{v,k} \leq C_v^V \\ \sum_{t,k} \hat{c}_t^D \cdot h_{v,k,t} \cdot \zeta_{v,k} \leq C_v^D \\ \sum_{t,k} \hat{c}_t^T \cdot h_{v,k,t} \cdot \xi_{v,k} \leq C_v^T \end{cases}, \quad \forall v. \tag{9}$$

Eq. (9) ensures that the memory space used by the vNFs deployed on NFV platforms of each SN does not exceed the platforms' memory size.

$$\begin{cases} \sum_{i,j} bd_i \cdot \sum_l x_{i,j,l}^{v,k} \cdot \eta_{i,j,l}^t \cdot \mu_{v,k} \leq B_t^U \\ \sum_{i,j} bd_i \cdot \sum_l x_{i,j,l}^{v,k} \cdot \eta_{i,j,l}^t \cdot \zeta_{v,k} \leq B_t^D \\ \sum_{i,j} bd_i \cdot \sum_l x_{i,j,l}^{v,k} \cdot \eta_{i,j,l}^t \cdot \xi_{v,k} \leq B_t^T \end{cases}, \quad \forall v, k, t. \tag{10}$$

Eq. (10) ensures that the bandwidth used by the vNFs deployed on one NFV platform of each SN does not exceed the platform's bandwidth capacity.

3) Constraints on Traffic Routing:

$$\begin{aligned} &\sum_{t,l} \eta_{i,j,l}^t \cdot \sum_{v,k} \left(\rho_t^V \cdot \mu_{v,k} + \rho_t^D \cdot \zeta_{v,k} + \rho_t^T \cdot \xi_{v,k} \right) \\ &+ \sum_n \sum_{(u,v) \in E, k_1, k_2} g_{i,j,n}^{u,k_1,v,k_2} \cdot \lambda_{(u,v)} \leq \tau_{i,j}, \quad \forall i, j. \end{aligned} \tag{11}$$

Eq. (11) ensures that the E2E latency of each request $r_{i,j}$ in the vNF-STs does not exceed its QoS requirement $\tau_{i,j}$.

$$\sum_{v,k_2} g_{i,j,n}^{u,k_1,v,k_2} - \sum_{v,k_2} g_{i,j,n}^{v,k_2,u,k_1} = x_{i,j,n}^{u,k_1} - x_{i,j,n+1}^{u,k_1}, \quad \forall i, j, n \in [1, N_{i,j} - 1], u, k_1. \tag{12}$$

Eq. (12) ensures the flow conservation condition of the routing of each request $r_{i,j}$ in the vNF-STs, *i.e.*, except for the first and last LAGs of $r_{i,j}$, the inflow and outflow of each NFV platform that carries its vNF are equal.

$$\begin{aligned} &\sum_{v,k_2} g_{i,j,0}^{u,k_1,v,k_2} - \sum_{v,k_2} g_{i,j,0}^{v,k_2,u,k_1} \\ &= \begin{cases} 1, & \text{if } k_1 = \theta_u^+ \text{ and } u = s_{i,j}, \\ -x_{i,j,1}^{u,k_1}, & \text{otherwise}, \end{cases} \quad \forall i, j. \end{aligned} \tag{13}$$

Eq. (13) ensures the flow conservation condition of the routing of each request $r_{i,j}$ in its first LAG.

$$\begin{aligned} &\sum_{v,k_2} g_{i,j,N_{i,j}}^{u,k_1,v,k_2} - \sum_{v,k_2} g_{i,j,N_{i,j}}^{v,k_2,u,k_1} \\ &= \begin{cases} -1, & \text{if } k_1 = \theta_u^- \text{ and } u = d_{i,j}, \\ x_{i,j,N_{i,j}}^{u,k_1}, & \text{otherwise}, \end{cases} \quad \forall i, j. \end{aligned} \tag{14}$$

Eq. (14) ensures the flow conservation condition of the routing of each request $r_{i,j}$ in its last LAG.

$$x_{i,j,l}^{v,\theta_v^-} = x_{i,j,l}^{v,\theta_v^+} = 0, \quad \forall i, j, l, v. \tag{15}$$

Eq. (15) ensures that the dummy platforms to/from each SN v cannot be used for real vNF deployment.

$$\sum_{(u,v) \in E, k_1, k_2} g_{i,j,n}^{u,k_1,v,k_2} \geq 1, \quad \forall i, j, n \in [0, N_{i,j}]. \tag{16}$$

Eq. (16) ensures that the routing path of request $r_{i,j}$ uses at least one link in each of its LAGs.

$$g_{i,j,n_1}^{u,k_1,v,k_2} + g_{i,j,n_2}^{u,k_1,v,k_2} \leq 1, \forall i, j, \ (u,v) \in E, k_1, k_2, \ n_1, n_2 \in [0, N_{i,j}]. \tag{17}$$

Eq. (17) ensures that each vNF-ST is provisioned in the SNT without routing loops.

$$\begin{cases} \sum_{v,k_2} g_{i,j,n}^{u,k_1,v,k_2} \geq x_{i,j,n}^{u,k_1}, \\ \sum_{v,k_2} g_{i,j,n-1}^{v,k_2,u,k_1} \geq x_{i,j,n}^{u,k_1}, \end{cases} \quad \forall i, j, u, k_1, n \in [1, N_{i,j}]. \tag{18}$$

Eq. (18) ensures the correct relation between the routing path and vNF deployment of each request $r_{i,j}$ in the vNF-STs.

C. Heuristic Algorithm Design

Although ILP can provide the exact solution to the problem of QoS-aware management reconfiguration of vNF-STs, it becomes intractable for large-scale problems. Meanwhile, the management reconfiguration of vNF-STs includes the reprovisioning of a set of vNF-SCs (*i.e.*, requests), which is known to be $\mathcal{NP}$-hard [29]. Hence, the problem of QoS-aware management reconfiguration of vNF-STs is $\mathcal{NP}$-hard too. In this section, we will design a two-step heuristic to solve this problem in polynomial time. Specifically, the heuristic first selects the requests to be reconfigured, including those that need to be reconfigured and some other ones that would make the optimization result better, and then reconfigures the selected requests to minimize the objective in Eq. (2).

*1) **Selecting Requests for Management Reconfiguration**:* The first step of the heuristic is to select requests for management reconfiguration based on those in $\mathbf{R}'$ and put them in set $\mathbf{R}''$. Here, we denote the set of all the requests in active vNF-STs as $\mathbf{R}$. Note that, before each management reconfiguration, the deployment of active requests is correlated, *i.e.*, different requests can share vNFs deployed on NFV platforms of SNs. Therefore, if the management reconfiguration only considers the requests in $\mathbf{R}'$, the solution might not be good enough. This can be seen the examples in Fig. 4. The left subplot shows the deployment scheme of a vNF-ST before the reconfiguration, and we need to reconfigure *Request* 1 because its QoS demand on E2E latency has been tightened. The middle subplot shows the reconfiguration scheme that only considers *Request* 1, and it needs to instantiate two new vNFs on PDP-SWs, using more-than-necessary NFV platforms. The right subplot describes a better reconfiguration scheme, which considers *Request* 2

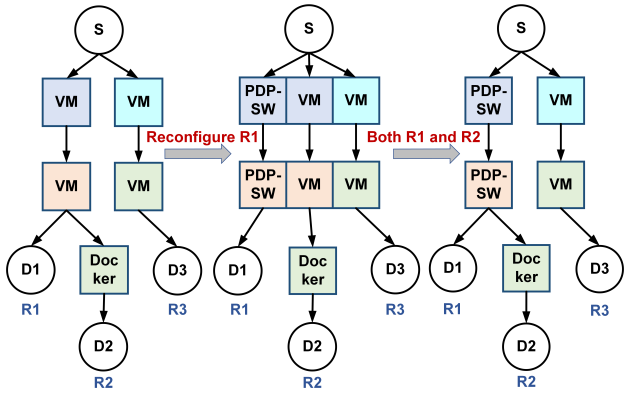


Fig. 4. Examples on selecting correlated requests to reconfigure.

Algorithm 2: Selecting Requests for Reconfiguration

1 $\mathbf{R}'' = \mathbf{R}'$, $\mathbf{R}_1 = \mathbf{R} \setminus \mathbf{R}'$, $\mathbf{R}_2 = \emptyset$, $\Phi = \emptyset$, $b_{f^*}^{\max} = 0$;
2 **for** *each request* $r \in \mathbf{R}'$ **do**
3 **if** r *is due to latency requirement change* **then**
4 insert all the requests in $\mathbf{R}_1$ that share vNF(s) with r into $\mathbf{R}_2$;
5 **end**
6 **end**
7 insert the vNF types used by requests in $\mathbf{R}_2$ into Φ;
8 sort vNF types in Φ to obtain combinations;
9 **for** *each combination* $f \in \Phi$ **do**
10 calculate potential cost reduction b_f with Eq. (19);
11 **if** $b_f > b_{f^*}^{\max}$ **then**
12 $b_{f^*}^{\max} = b_f$;
13 $f^* = f$;
14 **end**
15 **end**
16 **for** *each request* $r \in \mathbf{R}_2$ **do**
17 **if** *at least one vNF type used by* r *is in* f^* **then**
18 insert r into $\mathbf{R}''$;
19 **end**
20 **end**

together with *Request* 1. The scheme not only saves two NFV platforms but also reduce the E2E latency of *Request* 2.

Algorithm 2 shows our heuristic for selecting the requests to reconfigure. *Line* 1 is for the initialization, where $\mathbf{R}_1$ stores all the active requests that do not need to be reconfigured at this moment. In *Lines* 2-6, we check each request $r \in \mathbf{R}'$ (*i.e.*, those need to be reconfigured for the three reasons in Section III-B) and find the requests in $\mathbf{R}_1$ that share vNF(s) with r. These requests are those that can be potentially reconfigured to save vNFs in the SNT, and we put them in set $\mathbf{R}_2$ (*Line* 4). *Line* 7 collects the vNF types used by the requests in $\mathbf{R}_2$ and put them in set Φ. Then, we sort the vNF types in Φ to get combinations of vNF types (*Line* 8). Specifically, we first sort the vNF types in descending order of their usages by the requests in $\mathbf{R}_2$, then obtain the combinations based on the sorted results. For instance, if we assume that the sorted results in $\mathbf{R}_2$ are $\{t_1, t_2, t_3, t_4\}$ ($t_1, t_2, t_3, t_4 \in t_{\text{total}}$), the combinations in Φ will be $\{< t_1 >, < t_1, t_2 >, < t_1, t_2, t_3 >, < t_1, t_2, t_3, t_4 >\}$. The rationale behind this operation is to find the combinations of vNFs that are shared the most by the requests in $\mathbf{R}_2$, for minimizing the number of vNFs after the reconfiguration.

The for-loop that covers *Lines* 9-15 checks each combination $f \in \Phi$ to find the one (f^*) whose potential cost reduction b_f is the largest, which is calculated as follows

$$b_f = \sum_{t \in f} \sum_{r_{i,j} \in \mathbf{R}_2} \sum_{l} \eta_{i,j,l}^{t} \cdot (\gamma_t - \bar{m}), \tag{19}$$

where γ_t denotes the current deployment cost of a type-t vNF in request $r_{i,j} \in \mathbf{R}_2$, and $\bar{m}$ is the average migration cost obtained based on $\{m_{u,k_1}^{v,k_2}\}$. Finally, *Lines* 16-20 select requests in $\mathbf{R}_2$ to add in $\mathbf{R}''$ (*i.e.*, the set of requests to be reconfigured) according to the vNF types in f^*.

Complexity analysis: The complexity of *Lines* 2-6 is $O(|\mathbf{R}'| \cdot |\mathbf{R}_1| \cdot \hat{N}^2)$, where we have $\hat{N} = \max_{i,j}(N_{i,j})$ as the maximum number of vNFs in a request. The complexity of *Lines* 7-8 is $O(|\mathbf{R}_1| \cdot \hat{N} + |t_{\text{total}}| \cdot \log(|t_{\text{total}}|))$, and that of *Lines* 9-15 is $O(|t_{\text{total}}| \cdot |\mathbf{R}_1| \cdot \hat{N})$. Finally, the complexity of *Lines* 16-20 is also $O(|t_{\text{total}}| \cdot |\mathbf{R}_1| \cdot \hat{N})$, and the maximum number of elements in f does not exceed T. In summary, the time complexity of *Algorithm* 2 is $O(|\mathbf{R}'| \cdot |\mathbf{R}_1| \cdot \hat{N}^2 + |\mathbf{R}_1| \cdot \hat{N} + |t_{\text{total}}| \cdot \log(|t_{\text{total}}|) + 2 \cdot |t_{\text{total}}| \cdot |\mathbf{R}_1| \cdot \hat{N})$.

*2) **Reconfiguring Selected Requests**:* The second step of our heuristic is to reconfigure the requests that were selected by *Algorithm* 2 (*i.e.*, those in set $\mathbf{R}''$), which can be accomplished by leveraging the LAGs described in Section IV-B. The basic idea of the LAG-based algorithm is to 1) build N^* LAGs for all the requests in $\mathbf{R}''$, where

$$N^* = \max_{r_{i,j} \in \mathbf{R}''} (N_{i,j} + 1), \tag{20}$$

2) figure out the new vNF deployment schemes of the requests in $\mathbf{R}''$ by grouping their required vNFs according to the LAGs, and 3) connect the deployed vNFs to finish the reconfiguration. The detailed procedure is shown in *Algorithm* 3.

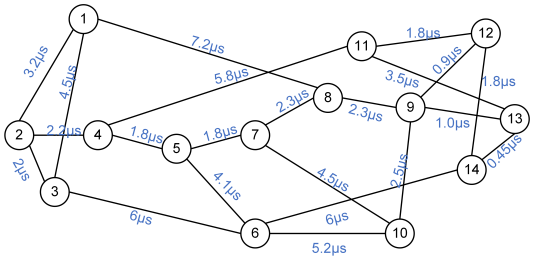
We first remove the failed NFV platforms from the SNT (*Line* 1). Then, the for-loop that covers *Lines* 2-27 determines the reprovisioning scheme of each request in each LAG in sequence. Specifically, the procedure of the service provisioning in each LAG G_n is as follows. *Lines* 3-9 check each request r that uses G_n, and if the request needs to be reconfigured due to the change of its QoS requirement on E2E latency, we remove the NFV platforms that cannot satisfy its new latency requirement from LAG G_n to get a restricted LAG G_n'. Then, we organize the requests' vNFs that need to be deployed in the current LAG according to their types, and denote them as tuples, each of which includes a type of the vNFs (t_i), the requests that use the vNF type in the LAG ($\mathbf{R}_i$), the number of the requests (c_i), and their total bandwidth demand (B_i) (*Lines* 10-13). The for-loop covering *Lines* 14-19 checks the requests in each tuple of $\{\mathbf{R}_j, B_j, c_j, t_j\}$. If their total bandwidth demand is larger than the minimum capacity of an NFV platform, it divides the requests in $\mathbf{R}_j$ into groups to make sure that the total bandwidth demand of each group does not exceed the minimum capacity of an NFV platform, and generates a tuple for each group (*Lines* 16-17). Then, for each tuple of $\{\mathbf{R}_j, B_j, c_j, t_j\}$ in descending order of c_j, *Lines*

Algorithm 3: Reconfiguring Selected Requests

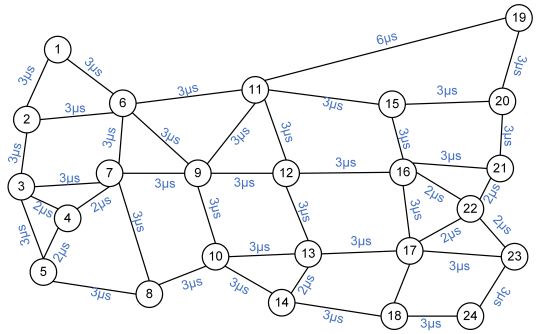
remove failed NFV platforms from SNT;
for *each LAG* G_n *with* $n \in [0, N^* - 1]$ **do**
  for *each* r *that uses* G_n *and has latency change* **do**
    for *each type of NFV platform* h **do**
      if h *cannot satisfy latency demand of* r **then**
        remove all the NFV platforms of type h in G_n to get G'_n;
      end
    end
  end
  get all the vNFs that need to be deployed in G_n and store their types in t_{G_n}, and set $i = 1$;
  for *each vNF type* $t \in t_{G_n}$ **do**
    store the requests that use it in set $\mathbf{R}_i$, their number in c_i, their total bandwidth demand in B_i, and t in t_i, and set $i = i + 1$;
  end
  for *each* $j \in [1, i-1]$ **do**
    if $B_j > \min\left(B_t^V, B_t^D, B_t^T\right)$ **then**
      divide requests in $\mathbf{R}_j$ into groups to ensure the total bandwidth demand of each group not exceed $\min\left(B_t^V, B_t^D, B_t^T\right)$;
      generate a tuple to include the vNF type, requests, number of requests (c_i), and total bandwidth demand of each group;
    end
  end
  sort tuples $\{\mathbf{R}_j, B_j, c_j, t_j\}$ in descending order of c_j;
  for *each tuple* $\{\mathbf{R}_j, B_j, c_j, t_j\}$ *in sorted order* **do**
    if *requests in* $\mathbf{R}_j$ *have latency changes* **then**
      $G_n = G'_n$;
    end
    find an NFV platform in G_n that can carry the type t_j vNF required by requests in $\mathbf{R}_j$ and the resulting cost of resource usages is the smallest;
  end
end
for *each request* $r \in \mathbf{R}''$ **do**
  connect its vNFs with the shortest routing paths;
end

20-26 find a proper NFV platform to deploy/reuse a vNF of type t_j for all the requests in $\mathbf{R}_j$. Specifically, in LAG G_n, we find the NFV platform on which deploying/reusing a vNF of type t_j results in the smallest cost of resource usages (*i.e.*, both IT and bandwidth resource usages). Finally, we set up paths to connect the vNFs of each request in $\mathbf{R}''$ in sequence, and finish the management reconfiguration (*Lines* 28-30).

Complexity analysis: For each iteration of *Lines* 2-27, the complexity of *Lines* 3-9 is $O(|\mathbf{R}''| \cdot |V| \cdot K)$, where K is the number of NFV platforms on each SN, that of *Lines* 10-20 is $O(|\mathbf{R}''| + |t_{\text{total}}| \cdot \log(|t_{\text{total}}|))$, and that of *Lines* 21-26 is $O(|\mathbf{R}''| \cdot |V| \cdot K)$. The complexity of *Lines* 28-30 is $O(|\mathbf{R}''| \cdot N^* \cdot |V|^3)$. Hence, the overall time complexity of *Algorithm* 3 is $O(N^* \cdot (|\mathbf{R}''| + |t_{\text{total}}| \cdot \log(|t_{\text{total}}|) + |\mathbf{R}''| \cdot |V| \cdot K + |\mathbf{R}''| \cdot |V|^3))$.



(a) NSFNET topology



(b) US backbone topology

Fig. 5. SNT topologies marked with transmission delays of links.

V. Performance Evaluation

In this section we perform extensive numerical simulations to evaluate the performance of our proposed algorithms.

A. Simulation Setup

Our simulations consider two SNT topologies, which are the 14-node NSFNET topology in Fig. 5(a) and the 24-node US backbone topology in Fig. 5(b), and the transmission delay of each link is also marked in Fig. 5. We assume that each SNT can support $|t_{\text{total}}| = 4$ types of vNFs. To ensure that the simulations are representative of real-world network environments, we set simulation parameters according to the parameters of practical NFV platforms (*e.g.*, the PDP-SWs in [11]) and our own experimental results in [23].

We set the available memory space of heterogeneous NFV platforms in the SNT before the management reconfiguration (C_v^V/C_v^D/C_v^T) within $[30\%, 100\%]$. We assume that deploying a vNF of any type on a PDP-SW will consume all of its memory space, *i.e.*, $\{\hat{c}_t^T = 100\%, \forall t \in t_{\text{total}}\}$. As for VMs and Dockers, the memory usages of a vNF on them are set as $\hat{c}_t^V \in [3\%, 4\%]$ and $\hat{c}_t^D \in [0.002\%, 0.01\%]$, respectively. For vNFs deployed on VMs/Dockers/PDP-SWs, their bandwidth capabilities B_t^V/B_t^D/B_t^T are 1.5 Gbps, 1.3 Gbps, and 100 Gbps, respectively, while their processing latencies are $\rho_t^V \in [170, 260]$ μs, $\rho_t^D \in [150, 160]$ μs, and $\rho_t^T \in [10, 20]$ μs, respectively. The costs of deploying a vNF on VMs/Dockers/PDP-SWs are $\gamma_t^V = 1$, $\gamma_t^D = 1.6$, and $\gamma_t^T = 1.76$, respectively. The vNF migration cost m_{u,k_1}^{v,k_2} is set within $[0.02, 0.06]$, and we assume that the cost of migrating a vNF to/from a PDP-SW is higher than those of other migration cases. This is because we need to reconfigure packet processing pipelines and update registers to migrate state information when migrating a vNF to/from a PDP-SW.

TABLE III
SIMULATION RESULTS WITH NSFNET TOPOLOGY

# of Requests	Scenario	Objective				Total Resource Cost ($\mathcal{B}+\mathcal{F}$)				Total Reconfiguration Cost ($\mathcal{M}$)			
		ILP	LAG	LAA	Sort	ILP	LAG	LAA	Sort	ILP	LAG	LAA	Sort
3	*Mix*	11.958	13.214	13.993	14.029	16.76	18.74	19.92	19.98	0.76	0.33	0.17	0.15
5	*Mix*	12.848	14.519	15.239	15.954	17.96	20.40	21.60	22.67	0.92	0.79	0.42	0.28
7	*Mix*	12.605	14.010	15.134	16.494	17.65	19.66	21.41	23.40	0.83	0.84	0.48	0.38
9	*Mix*	12.745	14.227	15.178	16.354	17.84	19.91	21.31	23.20	0.92	0.97	0.75	0.39
11	*Mix*	12.891	14.668	15.684	16.872	17.98	20.53	22.02	23.89	1.01	1.00	0.90	0.50
7	*QoS*	13.079	15.161	16.654	19.715	18.22	21.10	23.49	27.87	1.10	1.34	0.71	0.69
9	*QoS*	13.138	15.166	17.098	21.315	18.28	21.08	24.03	29.06	1.14	1.37	0.93	0.90
11	*QoS*	13.138	14.834	17.272	20.808	18.28	20.59	24.22	29.28	1.14	1.40	1.07	1.04
7	*vNF*	11.662	11.916	11.916	11.916	16.43	16.96	16.96	16.96	0.33	0.15	0.15	0.15
9	*vNF*	11.599	11.954	11.954	11.954	16.40	17.00	17.00	17.00	0.40	0.18	0.18	0.18
11	*vNF*	11.738	12.510	12.510	12.510	16.60	17.76	17.76	17.76	0.39	0.18	0.18	0.18

For the requests in vNF-STs, we assume that each request contains $[1, 4]$ vNFs. The management reconfiguration considers two types of requests, *i.e.*, latency-sensitive and latency-tolerable ones. We set the E2E latency requirement $\tau_{i,j}$ of a latency-sensitive request $r_{i,j}$ within $[0.06, 0.15] \cdot N_{i,j}$ msec, while that of a latency-tolerable request $r_{i,j}$ is assumed to be within $[0.25, 0.3] \cdot N_{i,j}$ msec. The bandwidth demand of each vNF-ST is set as $b_i = 0.1$ Gbps. To calculate the cost of bandwidth usage in Eq. (3), we set the normalized unit cost as $\beta = 2$ per Gbps. As for the weight coefficient in Eq. (2), we first have $\alpha = 0.7$, and then change its value to observe its impact on the optimization of management reconfiguration.

We will compare four algorithms for the QoS-aware management reconfiguration, which are the ILP model in Section IV-B (ILP), the LAG-based heuristic that combines *Algorithms* 2 and 3 (LAG), the greedy-based heuristic that we developed in [30] (LAA), and a simple sorting-based algorithm (Sort). Specifically, Sort first sorts the requests to be reconfigured according to their reasons, and then reconfigures the sorted requests one by one. To the best of our knowledge, the problem of how to optimize the management reconfiguration of vNF-STs with heterogeneous NFV platforms still has not been fully explored yet. Therefore, it is difficult for us to find benchmarks, which are more sophisticated than LAA and Sort, in the literature. Meanwhile, certain recent studies on vNF-SC reconfiguration also used greedy-based procedures (like those in LAA and Sort) in their algorithm design (*e.g.*, in [38]).

For simplicity, we abbreviate the reasons for reconfiguration as 1) *vNF*: requests change their vNF-SCs, 2) *QoS*: requests change their QoS demands on E2E latency, and 3) *Failure*: NFV platforms fail. We run the simulations on a computer with 3.0 GHz Intel Core i5-9500 CPU and 16 GB memory, and the software environment is Python 3.7 with Gurobi v9.1. To maintain sufficient statistical accuracy, the simulations average the results from 30 independent runs to get each data point. To show the algorithms' stability in the simulations, we also mark the range of the 95% confidence interval in Figs. 6-8.

B. Small-Scale Simulations

We first consider the NSFNET topology to run small-scale simulations in it, where each SN has 4 heterogeneous NFV platforms as one VM, two Dockers and one PDP-SW. To obtain the network state before management reconfiguration, we use ILP to deploy 5 vNF-STs that includes 13 requests in the SNT. Then, we simulate the scenarios in which the management reconfiguration needs to reconfigure $\{3, 5, 7, 9, 11\}$ requests. For each scenario, we consider cases with different reasons for reconfiguration. Specifically, we address three cases in each scenario: 1) *Mix*: the ratios of *vNF* and *QoS* are set within $[10\%, 60\%]$ and $[20\%, 40\%]$, while the remaining is for *Failure*, 2) *vNF*: all the requests need to be reconfigured due to *vNF*, and 3) *QoS*: the ratio of *QoS* is 100%.

TABLE IV
AVERAGE RUNNING TIME OF SIMULATIONS WITH NSFNET (SECONDS)

# of Requests	Scenario	ILP	LAG	LAA	Sort
3	*Mix*	9.27	0.027	0.017	0.028
5	*Mix*	11.83	0.038	0.020	0.038
7	*Mix*	11.37	0.045	0.023	0.039
9	*Mix*	15.12	0.045	0.027	0.040
11	*Mix*	17.08	0.046	0.029	0.042
7	*QoS*	11.53	0.053	0.038	0.22
9	*QoS*	10.56	0.061	0.044	0.33
11	*QoS*	10.74	0.062	0.046	0.40
7	*vNF*	13.04	0.023	0.023	0.028
9	*vNF*	14.43	0.027	0.027	0.030
11	*vNF*	15.34	0.029	0.029	0.031

Table III shows the simulation results of different scenarios. We can see that objectives provided by LAG is always the closest to those from ILP, which justifies the advantage of LAG over LAA and Sort. Specifically, LAG can adaptively select requests to reconfigure and save the total resource cost after management reconfiguration. This can be confirmed by comparing the total reconfiguration costs from the algorithms, because the results from ILP and LAG are larger than those from LAA and Sort. LAA outperforms Sort in most of the scenarios, but it can perform worse when the number of requests to be reconfigured is relatively small. We also notice that the advantage of LAG becomes more significant in *Mix* and *QoS* scenarios. The average running time of the algorithms are listed in Table IV. We can see that the three heuristics run

(a) $\alpha = 0.5$ (b) $\alpha = 0.7$ (c) $\alpha = 0.9$

Fig. 6. Results of simulations with NSFNET topology and *Mix*.

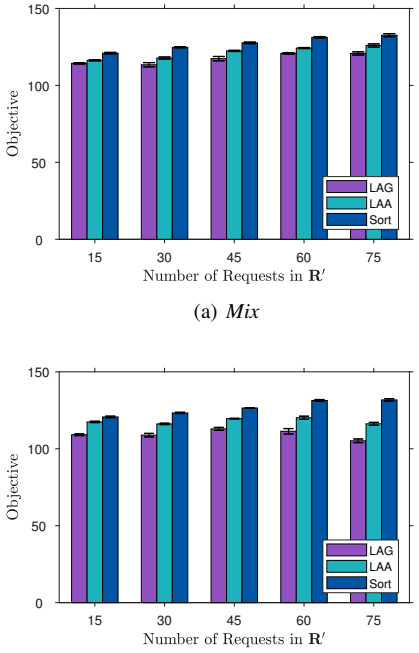
several orders of magnitude faster than ILP, and the running time of LAG is similar to that of LAA and they both run faster than Sort. This is because LAG and LAA reprovision requests in groups but do not process them one by one as in Sort.

Next, we fix the reconfiguration scenario as *Mix* and change the value of α in the optimization objective in Eq. (2), to investigate its effect on the algorithms' performance. Fig. 6 shows the results on objective for the cases of $\alpha \in \{0.5, 0.7, 0.9\}$. We observe that the objectives from all the algorithms increase with α, and for all the cases, the objectives from LAG are always the closest to those from ILP. This suggests that the advantage of LAG will not be affected by the setting of α.

C. Large-Scale Simulations

We then evaluate the algorithms with the large-scale US backbone topology. This time, we assume that there are 20 heterogeneous NFV platforms on each SN, which are 7 VMs, 10 Dockers, and 3 PDP-SWs. To obtain the network state before management reconfiguration, we deploy 31 vNF-STs that contains a total of 100 requests with LAA. Then, the management reconfiguration needs to reconfigure $\{15, 30, 45, 60, 75\}$ requests in the simulations. Due to its high time complexity, we do not consider ILP. The reconfiguration scenario is fixed as *Mix*, we have $\alpha = 0.5$ and other simulation parameters are the same as those used in the previous subsection. Meanwhile, to further study the heuristics' performance in an SNT where the QoS demands of vNF-STs are stringent, we add a new scenario in which the E2E latency requirement $\tau_{i,j}$ of a latency-sensitive or latency-tolerable request $r_{i,j}$ is within $[0.01, 0.09] \cdot N_{i,j}$ or $[0.06, 0.15] \cdot N_{i,j}$ msec, respectively.

Fig. 7 shows the simulation results on objective, which indicate that LAG still performs the best among the three algorithms. Moreover, by comparing Figs. 7(a) and 7(b), we can see that the performance gaps between LAG and LAA/Sort actually become larger in *Mix* with tightened latency requirements, especially for the cases in which there are a relatively large number of requests to be reconfigured. These results further verify the advantage of LAG in saving NFV platforms, which can be seen more clearly in Fig. 8. Specifically, Fig. 8 shows the usages of NFV platforms after reconfiguration for the case where there are 60 requests to be reconfigured ($|\mathbf{R}'| = 60$). In Fig. 8(a), we observe that LAG deploys more vNFs on Dockers than LAA and Sort, while Sort uses the most NFV platforms due to its poor performance on



(a) *Mix*

(b) *Mix* with tightened latency requirements

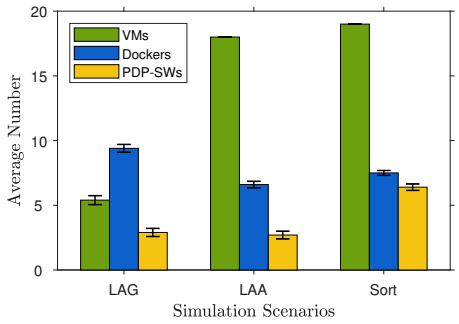
Fig. 7. Results of simulations with US backbone topology.

vNF aggregation. On the other hand, Fig. 8(b) indicates that when the QoS requirements of requests are tightened, LAG uses more PDP-SWs to satisfy the stringent demands on E2E latency and its usage of Dockers gets reduced significantly. This confirm that LAG can realize QoS-aware management reconfiguration. The average running time of the algorithms is listed in Table V. We can see that LAG still runs as fast as LAA, and Sort still takes much longer time than them to find the reconfiguration scheme of each request.

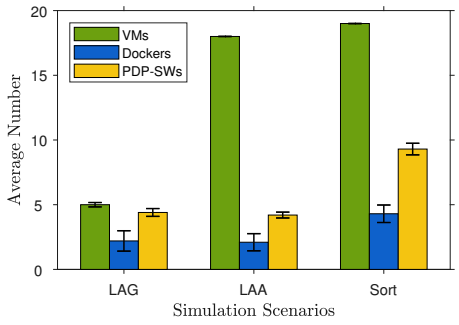
VI. Conclusion

In this paper, we studied how to optimize the management reconfiguration of vNF-STs in an SNT with heterogeneous NFV platforms, for three common reasons. We first formulated an ILP model to reduce both the total resource usage after reconfiguration and the overall vNF migration cost during reconfiguration. Then, we proposed a two-step algorithm based on LAGs to solve the problem more time-efficiently. Extensive simulations confirmed the effectiveness of our algorithms on

optimizing the management reconfiguration, and demonstrated that they could outperform existing benchmarks.



(a) *Mix* with $|\mathbf{R}'| = 60$



(b) *Mix* with tightened latency requirements and $|\mathbf{R}'| = 60$

Fig. 8. Results on usages of NFV platforms after reconfiguration.

TABLE V
AVERAGE RUNNING TIME PER REQUEST WITH US BACKBONE (SECONDS)

# of Requests	LAG	LAA	Sort
15	0.074	0.081	1.44
30	0.074	0.079	0.92
45	0.075	0.077	0.92
60	0.074	0.062	0.84
75	0.075	0.064	0.95

ACKNOWLEDGMENTS

This work was supported by NSFC project 61871357 and Fundamental Fund for Central Universities (WK3500000006).

REFERENCES

[1] "Cisco Annual Internet Report (2018-2023)," *Online White Report*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[2] P. Lu *et al.*, "Highly-efficient data migration and backup for Big Data applications in elastic optical inter-datacenter networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.

[3] P. Marsch *et al.*, "5G radio access network architecture: Design guidelines and key considerations," *IEEE Commun. Mag.*, vol. 54, pp. 24–32, Nov. 2016.

[4] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *J. Lightw. Technol.*, vol. 31, pp. 15–22, Jan. 2013.

[5] L. Gong *et al.*, "Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. 836–847, Aug. 2013.

[6] Y. Yin *et al.*, "Spectral and spatial 2D fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. A100–A106, Oct. 2013.

[7] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.

[8] W. Fang *et al.*, "Joint spectrum and IT resource allocation for efficient vNF service chaining in inter-datacenter elastic optical networks," *IEEE Commun. Lett.*, vol. 20, pp. 1539–1542, Aug. 2016.

[9] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. of INFOCOM 2014*, pp. 1–9, Apr. 2014.

[10] M. Chiosi *et al.*, "Network functions virtualization (NFV)," Oct. 2014. [Online]. Available: https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf.

[11] Tofino switch. [Online]. Available: https://www.barefootnetworks.com/products/brief-tofino/.

[12] S. Li *et al.*, "Protocol oblivious forwarding (POF): Software-defined networking with enhanced programmability," *IEEE Netw.*, vol. 31, pp. 58–66, Mar./Apr. 2017.

[13] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted NFV service chain deployment based on affiliation-aware vNF placement," in *Proc. of GLOBECOM 2016*, pp. 1–6, Dec. 2016.

[14] M. Zeng, W. Fang, and Z. Zhu, "Orchestrating tree-type VNF forwarding graphs in inter-DC elastic optical networks," *J. Lightw. Technol.*, vol. 34, pp. 3330–3341, Jul. 2016.

[15] Y. Wang, P. Lu, W. Lu, and Z. Zhu, "Cost-efficient virtual network function graph (vNFG) provisioning in multidomain elastic optical networks," *J. Lightw. Technol.*, vol. 35, pp. 2712–2723, Jul. 2017.

[16] S. Zhang, Q. Zhang, H. Bannazadeh, and A. Leon-Garcia, "Routing algorithms for network function virtualization enabled multicast topology on SDN," *IEEE Trans. Netw. Serv. Manag.*, vol. 12, pp. 580–594, Dec. 2015.

[17] B. Yi, X. Wang, M. Huang, and A. Dong, "A multi-stage solution for NFV-enabled multicast over the hybrid infrastructure," *IEEE Commun. Lett.*, vol. 21, pp. 2061–2064, Jun. 2017.

[18] O. Alhussein *et al.*, "A virtual network customization framework for multicast services in NFV-enabled core networks," *IEEE J. Sel. Areas Commun.*, vol. 38, pp. 1025–1039, Apr. 2020.

[19] B. Ren *et al.*, "Embedding service function tree with minimum cost for NFV-enabled multicast," *IEEE J. Sel. Areas Commun.*, vol. 37, pp. 1085–1097, Mar. 2019.

[20] N. Kiji, T. Sato, R. Shinkuma, and E. Oki, "Virtual network function placement and routing for multicast service chaining using merged paths," *Opt. Switch. Netw.*, vol. 36, p. 100554, Feb. 2020.

[21] Y. Ma, W. Liang, J. Wu, and Z. Xu, "Throughput maximization of NFV-enabled multicasting in mobile edge cloud networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, pp. 393–407, Feb. 2020.

[22] G. Mirjalily, M. Asgarian, and Z. Luo, "Interference-aware NFV-enabled multicast service in resource-constrained wireless mesh networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, pp. 424–436, May 2021.

[23] L. Dong *et al.*, "On application-aware and on-demand service composition in heterogenous NFV environments," in *Proc. of GLOBECOM 2019*, pp. 1–6, Dec. 2019.

[24] N. Zilberman. (2019, Apr.) In-network computing. [Online]. Available: https://www.sigarch.org/in-network-computing-draft/.

[25] K. Han *et al.*, "Application-driven end-to-end slicing: When wireless network virtualization orchestrates with NFV-based mobile edge computing," *IEEE Access*, vol. 6, pp. 26 567–26 577, 2018.

[26] Y. Xue and Z. Zhu, "On the upgrade of service function chains with heterogeneous NFV platforms," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, pp. 4311–4323, Dec. 2021.

[27] M. Carvalho, W. Cirne, F. Brasileiro, and J. Wilkes, "Long-term SLOs for reclaimed cloud computing resources," in *Proc. of SOCC 2014*, pp. 1–13, Nov. 2014.

[28] R. Govindan *et al.*, "Evolve or die: High-availability design principles drawn from Google's network infrastructure," in *Proc. of ACM SIGCOMM 2016*, pp. 58–72, Aug. 2016.

[29] J. Liu *et al.*, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, pp. 543–553, Sept. 2017.

[30] T. Li and Z. Zhu, "Leveraging heterogeneous NFV platforms for QoS-aware reconfiguration of vNF service trees," in *Proc. of ICOCN 2022*, pp. 1–3, Aug. 2022.

[31] "Network functions virtualisation (NFV): use cases," Tech. Rep., Oct. 2013. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/nfv/001_099/001/01.01.0160/gs_nfv001v010101p.pdf.

[32] H. Jiang, Y. Wang, L. Gong, and Z. Zhu, "Availability-aware survivable virtual network embedding (A-SVNE) in optical datacenter networks," *J. Opt. Commun. Netw.*, vol. 7, pp. 1160–1171, Dec. 2015.

[33] L. Gong, H. Jiang, Y. Wang, and Z. Zhu, "Novel location-constrained virtual network embedding (LC-VNE) algorithms towards integrated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 24, pp. 3648–3661, Dec. 2016.

[34] W. Shi, Z. Zhu, M. Zhang, and N. Ansari, "On the effect of bandwidth fragmentation on blocking probability in elastic optical networks," *IEEE Trans. Commun.*, vol. 61, pp. 2970–2978, Jul. 2013.

[35] M. Zhang, C. You, H. Jiang, and Z. Zhu, "Dynamic and adaptive bandwidth defragmentation in spectrum-sliced elastic optical networks with time-varying traffic," *J. Lightw. Technol.*, vol. 32, pp. 1014–1023, Mar. 2014.

[36] T. Wen, H. Yu, G. Sun, and L. Liu, "Network function consolidation in service function chaining orchestration," in *Proc. of ICC 2016*, pp. 1–6, May 2016.

[37] M. Eramo, E. Miucci, M. Ammar, and F. Lavacca, "An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures," *IEEE/ACM Trans. Netw.*, vol. 25, pp. 2008–2025, Mar. 2017.

[38] B. Li *et al.*, "Joint resource optimization and delay-aware virtual network function migration in data center networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, pp. 2960–2974, Mar. 2021.

[39] T. Doan, G. Nguyen, M. Reisslein, and F. Fitzek, "FAST: Flexible and low-latency state transfer in mobile edge computing," *IEEE Access*, vol. 9, pp. 115 315–115 334, 2021.

[40] S. Choudhury, S. Maheshwari, I. Seskar, and D. Raychaudhuri, "Share-On: Shared resource dynamic container migration framework for real-time support in mobile edge clouds," *IEEE Access*, vol. 10, pp. 66 045–66 060, 2022.

[41] C. Sun, J. Bi, Z. Zheng, and H. Hu, "Hyper: A hybrid highperformance framework for network function virtualization," *IEEE J. Sel. Areas Commun.*, vol. 35, pp. 2490–2500, Nov. 2017.

[42] L. Dong, N. da Fonseca, and Z. Zhu, "Application-driven provisioning of service function chains over heterogeneous NFV platforms," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, pp. 3037–3048, Sept. 2021.

[43] L. Cui *et al.*, "Enabling heterogeneous network function chaining," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, pp. 842–854, Sept. 2019.