On Application-aware and On-demand Service Composition in Heterogenous NFV Environments

Lu Dong, Kai Han, Lipei Liang, Bin Niu, Sicheng Zhao, Zuqing Zhu†
School of Information Science and Technology, University of Science and Technology of China, Hefei, China
†Email: {zqzhu}@ieee.org

***Abstract*—In this work, we try to further enhance the flexibility and cost-effectiveness of network function virtualization (NFV) by considering virtual network function service chaining (vNF-SC) in a heterogeneous NFV environment that can instantiate vNFs on virtual machines (VMs), docker containers, and SmartNICs. Specifically, we first lay out the network model, build a real network testbed that supports vNF deployment on kernel-based VMs, docker containers, and commercial SmartNICs, and then conduct experiments to measure the throughput and latency of traffic processing and memory usage of four types of vNFs implemented on them. Next, based on the measurement results, we formulate an integer linear programming (ILP) model to optimize the application-aware vNF-SC provisioning in the heterogenous NFV environment. Finally, we design and perform two experiments to demonstrate that our heterogeneous NFV environment can combine the advantages of VM/docker container/SmartNIC to provide enhanced flexibility for realizing on-demand and application-aware vNF-SC composition.**

***Index Terms*—Network function virtualization (NFV), Heterogeneous NFV environment, Service chaining.**

I. Introduction

The fast development of the Internet has stimulated huge volumes of new services [1]. However, due to the increasing equipment cost and maintenance complexity of special-purpose middleboxes, it becomes more and more challenging to deploy these services in traditional networks, and moreover, short time-to-market can hardly be achieved. Therefore, network function virtualization (NFV) [2] has been introduced to realize network services with virtual network functions (vNFs) that can be implemented on general-purpose hardware/software. Then, by steering traffic through a sequence of vNFs, a service provider can quickly realize various service compositions with vNF service chaining (vNF-SC) [3, 4].

Due to the apparent advantages of vNF-SC, people have conducted intensive researches on it, including both the theoretical approaches on how to deploy requested vNFs on proper locations and connect them in sequence [5-8], and the system implementations to orchestrate IT and bandwidth resources for realizing vNF-SC [9, 10]. However, all these studies overlooked the intrinsic difference among the substrate network elements (NEs) and assumed that vNFs were provisioned in a homogeneous NFV environment. This assumption becomes impractical for the following two reasons. Firstly, in addition to utilizing virtual machines (VMs) deployed on commodity servers, vNFs have been realized on various software/hardware platforms, *e.g.*, docker containers [11] and programmable network interface cards (*i.e.*, SmartNICs) [12]. Secondly, network services might have various quality-of-service (QoS) requirements, but some stringent ones can hardly be satisfied if the vNFs are deployed on certain platforms. For instance, the vNFs based on VMs provide relatively long latency and small throughput for traffic processing [10], while SmartNICs cannot achieve capacity elasticity even though they have superior performance on traffic processing.

The aforementioned dilemma motivates us to study application aware vNF-SC composition in a heterogeneous NFV environment that can instantiate vNFs on VMs, docker containers, and SmartNICs. In this work, we first lay out the network model, build a real network testbed that supports vNF implementation on kernel-based VMs, docker containers, and commercial SmartNICs, and then conduct experiments to measure the throughput and latency of traffic processing and memory usage of four types of vNFs implemented on them. Next, based on the measurement results, we formulate an integer linear programming (ILP) model to optimize the application-aware vNF-SC provisioning in such a heterogenous NFV environment, so that the resource cost of vNF-SC deployment is minimized while all the QoS requirements (*i.e.*, on bandwidth and latency) are satisfied as well. Finally, we consider two use-cases of our proposal, and design and perform experiments in the real network testbed to demonstrate the advantages of application-aware and on-demand vNF-SC composition in the heterogeneous NFV environment.

The rest of this paper is organized as follows. Section II provides a brief survey on the related work. The network model of application-aware vNF-SC provisioning in heterogenous NFV environments is described in Section III, and we formulate the ILP model to solve the problem exactly in Section IV. The experimental demonstrations of two use-cases are discussed in Section V. Finally, Section VI summarizes the paper.

II. Related Work

Since its inception, NFV has attracted intensive research interests, and people have proposed numerous algorithms to optimize vNF-SC deployments in various networks [3-8]. Note that, as explained in [2], the problem of vNF-SC provisioning is fundamentally different from the well-known virtual network embedding (VNE) problem [13], and thus it deserves further investigations. The authors of [5, 8] formulated theoretical models to optimize the vNF-SC provisioning in packet networks. Also for packet networks, Sun *et al.* [7] proposed a forecast-assisted vNF-SC provisioning

algorithm, while the vNF-SC reconfiguration to adapt to dynamic network environments was studied in [3]. Considering flexible-grid elastic optical networks [14-16] as the underlying infrastructure, Fang *et al.* [6] investigated how to optimize the spectrum and IT resources jointly for vNF-SC provisioning, and later on, the authors of [4] designed a deep learning assisted framework to facilitate proactive optimization.

However, none of the studies mentioned above has considered application-aware vNF-SC provisioning in heterogeneous NFV environments. More specifically, the heterogeneous NFV environment here consists of general-purpose NEs whose performance on traffic processing and memory usage can be different, when supporting the same type of vNFs. Note that, it is different from the hybrid environment discussed in [17], which refers to the one that combines special-purpose middleboxes and homogeneous general-purpose NEs. In other words, all the NEs in our heterogeneous NFV environment support different types of vNFs, while the middleboxes in the hybrid environment are dedicated to single network services. Hence, the vNF-SC provisioning in our case is more complex.

From the perspective of system implementation, vNF implementations have been realized on docker containers [9-11], SmartNICs [12], and field programmable gate arrays (FPGAs) [18]. Their experimental results suggested that each platform has its pros and cons to support vNF-SC provisioning. Therefore, we expect a combination of them to have more flexibility and better cost-effectiveness, when various QoS requirements have to be satisfied simultaneously. This actually motivates us to conduct the study in this work.

III. Problem Description

This section introduces the heterogenous NFV environment and defines the problem of vNF-SC provisioning in it.

A. Heterogeneous NFV Environment

Fig. 1 shows an example of our heterogenous NFV environment. Here, we consider three types of platforms for vNF implementation, *i.e.*, VM, docker container, and SmartNIC. As illustrated in Fig. 1, each node in the substrate network (SNT) can include an arbitrary combination of the three types of platforms. For example, *Node* 2 includes all the three platforms, while *Node* 3 only supports the vNF implementation based on VMs. Then, the application-aware vNF-SC provisioning in such a heterogenous NFV environment should minimize the resource cost of vNF-SC deployment and ensure all the QoS requirements from the vNF-SCs are satisfied.

For the vNFs, we consider four types of them[1], denoted as $\{vNF\ 1, vNF\ 2, vNF\ 3, vNF\ 4\}$. Here, *vNF* 1 is for packet forwarding, *vNF* 2 realizes a firewall for IP address screening, *vNF* 3 performs network address translation (NAT) for traffic between two different subnets, and *vNF* 4 is a load-balancer

[1]Due to the limited budget and human resources, we can only implement four types of vNFs in our experimental testbed for performance measurements. However, this would not restrict the applicability of our work since the ILP model in Section IV is generic. In other words, if more vNF types needs to be considered, one only need to implement them in the platforms, measure their performance metrics, and input the results in the ILP model.

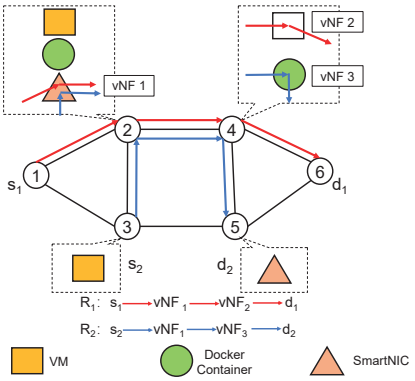


Fig. 1. Example of vNF-SC provisioning in heterogenous NFV environment.

that hashes the 5-tuple of each packet to map it to an output port. Then, we build a real experimental testbed to include the platforms. Specifically, for fair comparisons, we test the three platforms by attaching them to Linux servers with the same software/hardware configuration. The experiments measure the throughput and latency of traffic processing and memory usage for each vNF type, and Table I lists the results. Since SmartNIC realizes the vNFs in hardware, it performs the best on traffic processing. Nevertheless, for the same reason, its memory usage is also the most, which suggests that the elasticity of its capacity is the worst.

TABLE I
Experimental Results on vNF Performance Measurements

		vNF 1	*vNF* 2	*vNF* 3	*vNF* 4
Throughput (Gbps)	VM (b_m^U)	1.60	1.55	1.57	1.42
	Docker (b_m^D)	1.32	1.26	1.22	1.7
	SmartNIC (b_m^S)	10	10	10	10
Processing Latency (μs)	VM (r_m^U)	177	190	224	262
	Docker (r_m^D)	154.5	154.6	155.7	197
	SmartNIC (r_m^S)	110.2	110.7	110.9	111.7
Memory Usage (%)	VM ($\hat{c}_m^U$)	3.7	3.7	3.5	3.7
	Docker ($\hat{c}_m^D$)	0.003	0.003	0.002	0.01
	SmartNIC ($\hat{c}_m^S$)	26.25	22.75	26.25	23.83

B. Network Model and Problem Definition

We model the topology of the SNT as $G(V, E)$, where V and E are the sets of nodes and network links, respectively. Each node $v \in V$ is a commodity server whose capacities on VMs, docker containers, and SmartNICs are denoted as h_U^v, h_D^v and h_S^v, respectively. For example, $h_U^v = 10$, $h_D^v = 100$ and $h_S^v = 1$ indicate that node v can accommodate 10 VMs, 100 docker containers, and one SmartNIC. The memory space of a VM/container/SmartNIC attached on node v is $C_v^U/C_v^D/C_v^S$, respectively. The heterogeneous NFV environment can support M vNF types, while for each type $m \in M$, a vNF consumes $\hat{c}_m^U/\hat{c}_m^D/\hat{c}_m^S$ memory space, can achieve a maximum throughput of $b_m^U/b_m^D/b_m^S$, and takes a processing latency of $r_m^U/r_m^D/r_m^S$, if it is deployed on a VM/container/SmartNIC, respectively.

A vNF-SC request is modeled as $R_i(s_i, d_i, SC_i, b_i, t_i)$, where s_i and d_i are the source and destination nodes, respectively, $SC_i = (f_{i,1}, \cdots, f_{i,l}, \cdots, f_{i,N_i})$ denotes its vNF sequence (*i.e.*, $f_{i,l}$ is the type of the l-th vNF, and N_i is the number of vNFs in the vNF-SC), b_i represents its bandwidth demand, and t_i is the longest latency that it can tolerate. All the pending vNF-SC requests are stored in set $\mathcal{R}$. To improve resource utilization and save the cost of vNF deployment, we allow different vNF-SC requests to share vNFs if necessary. For instance, vNF-SCs R_1 and R_2 in Fig. 1 share the *vNF* 1 implemented on a SmartNIC on *Node* 2.

IV. ILP Formulation

A. ILP Model

For vNF-SC provisioning in the heterogeneous NFV environment, we need to instantiate the vNFs on proper platforms in nodes in V and steer the traffic of each vNF-SC R_i from s_i to d_i through the vNFs in SC_i in sequence. The following ILP model is formulated to optimize the vNF-SC provisioning.

Parameters:

- $G(V, E)$: the SNT's physical topology.
- $R_i(s_i, d_i, SC_i, b_i, t_i)$: a vNF-SC request, $R_i \in \mathcal{R}$.
- $\mathcal{P}$: the set of pre-calculated paths, each of which is among the K shortest paths between a node pair.
- $P_{u,v}$: the set of pre-calculated paths between u and v.
- L_p: the hop-count of a path $p \in P$.
- β: the unit cost of 1 Gbps bandwidth usage per hop.
- M: the set of available vNF types.
- $f_{i,l}^m$: the boolean flag that equals 1 if the l-th vNF in SC_i is a type-m vNF ($m \in M$), and 0 otherwise.
- $h_U^v/h_D^v/h_S^v$: the numbers of VMs/containers/SmartNICs that can be used on node v, respectively.
- $C_v^U/C_v^D/C_v^S$: the memory space of a VM/container /SmartNIC on node v, respectively.
- $b_m^U/b_m^D/b_m^S$: the bandwidth throughput of a VM/container /SmartNIC that carries a type-m vNF, respectively.
- $r_m^U/r_m^D/r_m^S$: the traffic processing latency of a VM/ container/SmarterNIC that carries a type-m vNF.
- γ_v^p: the boolean flag that equals 1 if node v is on path p, and 0 otherwise.
- η_v^p: the sequence number of node v on path p.
- $\alpha_m^U/\alpha_m^D/\alpha_m^S$: the cost of memory usage for deploying a type-m vNF on a VM/container/SmartNIC.

Variables:

- $u_{i,l}^{v,k}/d_{i,l}^{v,k}/s_{i,l}^{v,k}$: the boolean variable that equals 1 if the l-th vNF in SC_i is deployed on the k-th VM/container /SmarterNIC on node v, respectively, and 0 otherwise.
- $y_{i,l}^v$: the boolean variable that equals 1 if the l-th vNF in SC_i is deployed on node v, and 0 otherwise.
- $\phi_m^{v,k}/\varphi_m^{v,k}/\psi_m^{v,k}$: the boolean variable that equals 1 if the k-th VM/container/SmartNIC on node v carries a type-m vNF, respectively, and 0 otherwise.
- z_p^i: the boolean variable that equals 1 if SC_i uses path $p \in P$, and 0 otherwise.

Objective:

The optimization objective is to minimize the total cost of vNF-SC deployment, which includes both the cost of vNF deployment (T_v) and that of bandwidth usage (T_b). Here, the two costs can be calculated as

$$T_v = \sum_{m,v} \left(\sum_{k=1}^{h_U^v} \alpha_m^U \cdot \phi_m^{v,k} + \sum_{k=1}^{h_D^v} \alpha_m^D \cdot \varphi_m^{v,k} + \sum_{k=1}^{h_S^v} \alpha_m^S \cdot \psi_m^{v,k} \right),$$

$$T_b = \sum_{i,p} \beta \cdot b_i \cdot z_p^i \cdot L_p.$$

Hence, the objective is formulated as

$$\text{Minimize} \quad (T_v + T_b). \tag{1}$$

Constraints:

$$\begin{cases} \sum\limits_{m,i,l} \sum\limits_{k=1}^{h_U^v} \hat{c}_m^U \cdot \phi_m^{v,k} \cdot f_{i,l}^m \le C_v^U \\ \sum\limits_{m,i,l} \sum\limits_{k=1}^{h_D^v} \hat{c}_m^D \cdot \varphi_m^{v,k} \cdot f_{i,l}^m \le C_v^D, \quad \forall v \in V. \\ \sum\limits_{m,i,l} \sum\limits_{k=1}^{h_S^v} \hat{c}_m^S \cdot \psi_m^{v,k} \cdot f_{i,l}^m \le C_v^U \end{cases} \tag{2}$$

Eq. (2) ensures that the vNF deployment on each node will not exceed its memory capacity.

$$\begin{cases} \sum\limits_i b_i \cdot \sum\limits_l f_{i,l}^m \cdot u_{i,l}^{v,k} \le b_m^U \\ \sum\limits_i b_i \cdot \sum\limits_l f_{i,l}^m \cdot d_{i,l}^{v,k} \le b_m^D, \quad \forall m, v, k. \\ \sum\limits_i b_i \cdot \sum\limits_l f_{i,l}^m \cdot s_{i,l}^{v,k} \le b_m^S \end{cases} \tag{3}$$

Eq. (3) ensures that each vNF's traffic processing throughput will not be exceeded.

$$\sum_{l,m} f_{i,l}^m \cdot \sum_v \left(\sum_{k=1}^{h_U^v} u_{i,l}^{v,k} \cdot r_m^U + \sum_{k=1}^{h_D^v} d_{i,l}^{v,k} \cdot r_m^D + \sum_{k=1}^{h_S^v} s_{i,l}^{v,k} \cdot r_m^D \right) \le t_i, \quad \forall i. \tag{4}$$

Eq. (4) ensures that for each vNF-SC request, its QoS requirement on processing latency is satisfied[2].

$$\sum_m f_{i,l}^{m,k} \le \sum_v \left(\sum_{k=1}^{h_U^v} u_{i,l}^{v,k} + \sum_{k=1}^{h_D^v} d_{i,l}^{v,k} + \sum_{k=1}^{h_S^v} s_{i,l}^{v,k} \right), \quad \forall i, l. \tag{5}$$

Eq. (5) ensures that the vNFs in SC_i of each vNF-SC request has been deployed.

$$\begin{cases} \sum\limits_m \phi_m^{v,k} \le 1 \\ \sum\limits_m \varphi_m^{v,k} \le 1, \quad \forall v, k. \\ \sum\limits_m \psi_m^{v,k} \le 1 \end{cases} \tag{6}$$

[2]In our experiments, we find that the traffic processing latency of a vNF is normally much longer than the transmission delay of a network link between two nodes. Hence, we ignore the transmission delay in the ILP.

Eq. (6) ensures that each platform of VM/container/SmartNIC can only be used to realize at most one vNF.

$$\begin{cases} \sum_{i,l} f_{i,l}^m \cdot u_{i,l}^{v,k} > (\phi_m^{v,k} - 1) \cdot \left(1 + \sum_{i,l} f_{i,l}^m\right), \\ \sum_{i,l} f_{i,l}^m \cdot u_{i,l}^{v,k} \le \phi_m^{v,k} \cdot \sum_{i,l} f_{i,l}^m \end{cases} \quad \forall m, v, k, \quad (7)$$

$$\begin{cases} \sum_{i,l} f_{i,l}^m \cdot d_{i,l}^{v,k} > (\varphi_m^{v,k} - 1) \cdot \left(1 + \sum_{i,l} f_{i,l}^m\right), \\ \sum_{i,l} f_{i,l}^m \cdot d_{i,l}^{v,k} \le \varphi_m^{v,k_D} \cdot \sum_{i,l} f_{i,l}^m \end{cases} \quad \forall m, v, k, \quad (8)$$

$$\begin{cases} \sum_{i,l} f_{i,l}^m \cdot s_{i,l}^{v,k} > (\psi_m^{v,k} - 1) \cdot \left(1 + \sum_{i,l} f_{i,l}^m\right), \\ \sum_{i,l} f_{i,l}^m \cdot s_{i,l}^{v,k} \le \psi_m^{v,k} \cdot \sum_{i,l} f_{i,l}^m \end{cases} \quad \forall m, v, k. \quad (9)$$

Eq. (7)-(9) ensure that the variables' values are set correctly according to their inherent relations. For instance, Eq. (7) makes sure that if $\left(\sum_{i,l} f_{i,l}^m \cdot u_{i,l}^{v,k}\right) > 0$, we have $\phi_m^{v,k_U} = 1$, and ϕ_m^{v,k_U} should be set as zero otherwise.

$$\sum_v \left(\sum_{k=1}^{h_U^v} u_{i,l}^{v,k} + \sum_{k=1}^{h_D^v} d_{i,l}^{v,k} + \sum_{k=1}^{h_S^v} s_{i,l}^{v,k}\right) \le 1, \quad \forall i, l. \quad (10)$$

Eq. (10) ensures that the l-th vNF in SC_i can only be deployed in one platform.

$$\sum_{p \in P_{s_i,d_i}} z_p^i = 1, \quad \forall i, \quad (11)$$

$$\sum_{p \in P_{s_i,d_i}} z_p^i \cdot \gamma_v^p \ge y_{i,l}^v, \quad \forall i, l, v. \quad (12)$$

Eqs. (11) and (12) ensure that one and only one path in P_{s_i,d_i} is used to provision request R_i for $s_i \to d_i$, and each vNF can only be deployed along the chosen path.

$$\begin{aligned} &\sum_v (y_{i,l_1}^v \cdot \eta_v^p) - \sum_v (y_{i,l_2}^v \cdot \eta_v^p) \ge (z_p^i - 1) \cdot L_p, \\ &\forall i,\ p \in P_{s_i,d_i},\ \{l_1, l_2 : l_1 > l_2,\ l_1, l_2 \in SC_i\}. \end{aligned} \quad (13)$$

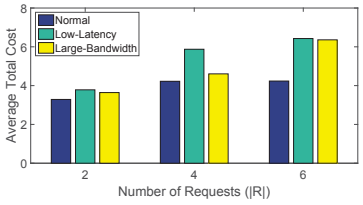
Eq. (13) ensures that the vNFs of each vNF-SC request are connected in the right sequence along the chosen path.

$$y_{i,l}^v = \sum_{k=1}^{h_U^v} u_{i,l}^{v,k} + \sum_{k=1}^{h_D^v} d_{i,l}^{v,k} + \sum_{k=1}^{h_S^v} s_{i,l}^{v,k}, \quad \forall i, l, v. \quad (14)$$

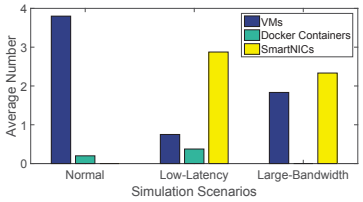
Eq. (14) explains how to determine the value of $y_{i,l}^v$.

B. ILP Results

We use the six-node topology in Fig. 1 to evaluate the ILP. We still consider $|M| = 4$ types of vNFs, and the parameters regarding their deployments in VMs/containers/SmartNICs use the values in Table I. The simulations consider three scenarios: normal, large-bandwidth and low-latency. We assume that for each R_i, SC_i consists of $[1, 4]$ vNFs, its bandwidth demand b_i is randomly selected within $[0.1, 0.3]$ Gbps for the normal and low-latency scenarios and within $[0.5, 0.8]$ Gbps for the large-bandwidth one, while its latency requirement is set as $t_i \in [0.8, 5]$ msec for the normal and large-bandwidth scenarios and as $t_i \in [0.4, 0.6]$ msec for the low-latency one. The cost coefficients are $\beta = 0.05$, $\alpha_m^U = 1$, $\alpha_m^D = 1.6$, and $\alpha_m^S = 1.76$ units according to the latest realistic data [19, 20].



(a) vNF-SC deployment cost



(b) Device number for $|R| = 6$

Fig. 2. ILP simulation results.

Fig. 2 shows the simulation results, where each data point is obtained by averaging the results from 10 independent runs. The results in Fig. 2(a) indicate that for same numbers of requests, the low-latency scenario has the highest total cost for vNF-SC deployment while that from the normal scenario is the lowest. The reason behind this trend can be explained with the results in Fig. 2(b), which suggest that in the normal scenario, the vNF-SCs can be supported without any SmartNICs, while the low-latency scenario requires the most SmartNICs among the three scenarios.

V. Experimental Demonstrations

We conduct experiments in a real network testbed to demonstrate the advantages of application-aware and on-demand vNF-SC composition in heterogeneous NFV environment. Fig. 3 shows the network orchestration system that includes four layers. The service layer is for various applications to register in the system to describe their service and QoS demands. The orchestration layer determines the vNF-SC composition schemes for the applications and makes corresponding decisions on vNF deployment, removal, migration, and chaining. The mediator layer translates the instructions from the orchestration layer into policies and sends them to the manager for heterogeneous vNFs, which controls the life cycle of vNFs on VMs/containers/SmartNICs, and to the forwarding controller, which is the SDN controller to steer application traffic through required vNF instances. Finally, the infrastructure layer consists of substrate network elements in the heterogeneous NFV environment.

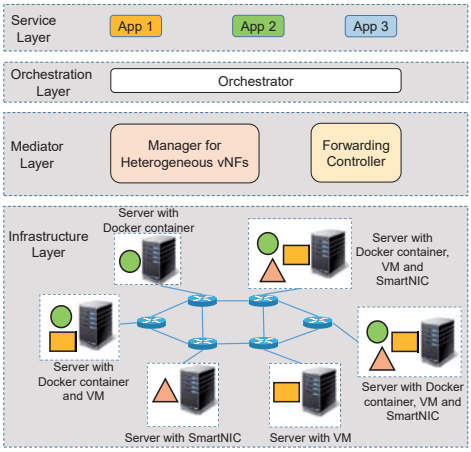
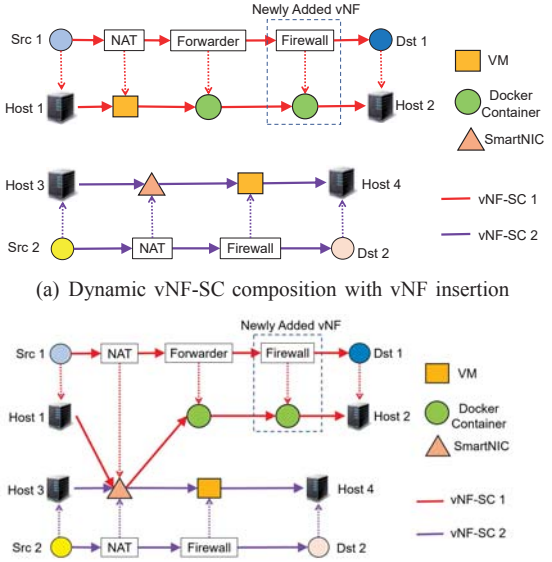


Fig. 3. Network orchestration system for heterogeneous NFV environment.



(a) Dynamic vNF-SC composition with vNF insertion

(b) vNF migration to ensure low-latency requirement

Fig. 4. Scenario of the first experiment.

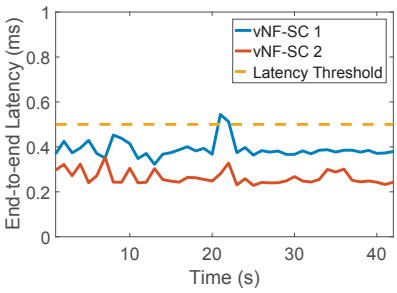


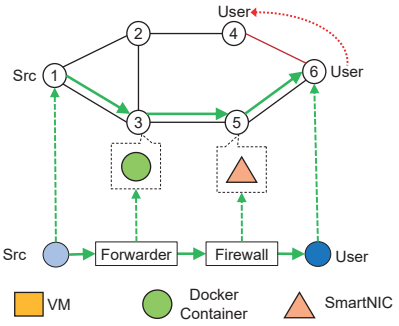
Fig. 5. End-to-end latency of two vNF-SCs in the first experiment.

We first conduct an experiment to demonstrate the benefit of having SmartNICs in the heterogeneous NFV environment, and its scenario is shown in Fig. 4. Initially, we have two vNF-SCs in the NFV environment, each of which processes 100 Mbps UDP traffic constantly, and their provisioning schemes are shown in Fig. 4(a). Then, at $t = 20$ seconds, *vNF-SC* 1 needs to append a firewall vNF at the end of its service chain. To avoid long setup delay, we decide to instantiate the new vNF on a docker container, as shown in Fig. 4(a). Meanwhile, the QoS demand of *vNF-SC* 1 states that its end-to-end latency should not exceed 0.5 msec. However, we find that the new end-to-end latency of *vNF-SC* 1 increases to around 0.54 msec after the vNF insertion, and this violates its QoS requirement. Hence, at $t = 22$ seconds, we migrate the NAT vNF on *vNF-SC* 1 and make *vNF-SCs* 1 and 2 share the NAT vNF deployed on a SmartNIC (as shown in Fig. 4(b)).

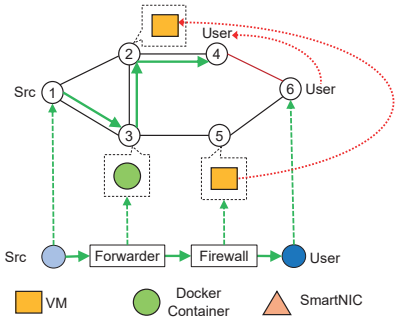
Fig. 5 shows the results on end-to-end latency, which indicates that after the NAT vNF in *vNF-SC* 1 having been migrated at $t = 22$ seconds, the latency of *vNF-SC* 1 decreases to around 0.38 msec and re-satisfies its QoS requirement. Meanwhile, during the whole process, the latency of *vNF-SC* 2 does not experience dramatic changes. This suggests that due to the relatively large processing capacity of the SmartNIC, the sharing of the NAT vNF deployed on it by *vNF-SCs* 1 and 2 would not affect its service to *vNF-SC* 2. Therefore, the heterogeneous NFV environment can properly address the intrinsic drawbacks of software-based NFV platforms (*i.e.*, docker containers and VMs) by leveraging the benefits provided by SmartNICs, and thus application-aware and on-demand vNF-SC composition can be realized.

Next, we conduct another experiment to demonstrate the advantage of having software-based platforms in the heterogeneous NFV environment. The experimental scenario is illustrated in Fig. 6, which considers the NFV-SC provisioning for mobile users. Initially, the user of the vNF-SC in Fig. 6(a) attaches to *Node* 6, and the vNF-SC consists of two vNFs that are deployed on a docker container and a SmartNIC, respectively. The traffic going through the vNF-SC is live video streaming with a constant throughput of 5 Mbps, and we assume that the user also has a QoS demand on the end-to-end latency, which should not exceed 0.5 msec. Before the user changes its location, its QoS demands on bandwidth and end-to-end latency get satisfied by the vNF-SC provisioning scheme in Fig. 6(a). However, at $t = 14$ seconds, it moves to *Node* 4. As vNFs deployed on SmartNICs do not support live migration [12], the video traffic has to be routed through the bottleneck link between *Nodes* 6 and 4, which not only brings excessive latency but also limits the video bandwidth below 5 Mbps. Hence, the user's QoS gets affected severely as shown in Fig. 7(a), *i.e.*, after $t = 14$ seconds, the luminance component's peak signal-to-noise ratio (Y-PSNR) [21] of its video playback degrades from ∼49 dB to ∼17 dB while the end-to-end delay increases from 0.27 msec to 6.65 msec.

On the other hand, if we deploy the vNF for firewall on a VM, it can be migrated to *Node* 2 lively after the user having changed its location (as shown in Fig. 6(b)). Therefore, its video traffic can be routed through 1→3→2→4 to avoid the bottleneck link between *Nodes* 6 and 4, and the live VM migration ensures that the video streaming would not be interrupted during the process. The experimental

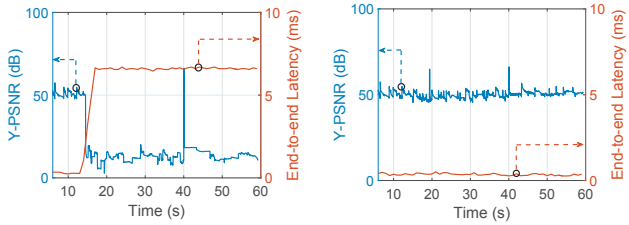


(a) vNF-SC provisioning with SmartNIC



(b) vNF-SC provisioning with VM

Fig. 6. Scenario of the second experiment.



(a) vNF-SC with SmartNIC (b) vNF-SC with VM

Fig. 7. Y-PSNR and end-to-end latency in the second experiment.

results in Fig. 7(b) confirm that with the live VM migration, both the Y-PSNR of the video playback and the end-to-end latency stay at their normal levels throughout the process. This simple experiment suggests that the heterogeneous NFV environment can also properly address the intrinsic drawbacks of hardware-based NFV platforms based on SmartNICs, and thus application-aware and on-demand vNF-SC composition can still be realized. In all, our heterogeneous NFV environment can combine the benefits of software/hardware-based NFV platforms to provide enhanced flexibility for realizing application-aware and on-demand vNF-SC composition.

VI. Conclusion

In this paper, we studied the vNF-SC composition in a heterogeneous NFV environment that combines VMs, docker containers, and SmartNICs. Based on the measurement results from a real network testbed, we first laid out the network model, and formulated an ILP model to optimize vNF-SC provisioning in the heterogenous NFV environment such that the resource cost of vNF-SC deployment is minimized while the QoS requirements are also satisfied. Then, we designed and performed two experiments to demonstrate that our heterogeneous NFV environment can combine the benefits of software/hardware-based NFV platforms to provide enhanced flexibility for application-aware vNF-SC composition.

Acknowledgments

This work was supported by the NSFC projects 61871357 and 61701472, and CAS key project (QYZDY-SSW-JSC003).

References

[1] 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.

[2] 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.

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

[4] B. Li, W. Lu, S. Liu, and Z. Zhu, "Deep-learning-assisted network orchestration for on-demand and cost-effective vNF service chaining in inter-DC elastic optical networks," *J. Opt. Commun. Netw.*, vol. 10, pp. D29–D41, Oct. 2018.

[5] I. Jang *et al.*, "Optimal network resource utilization in service function chaining," in *Proc. of NetSoft 2016*, pp. 11–14, Jun. 2016.

[6] 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.

[7] 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.

[8] T. Kuo, B. Liou, K. Lin, and M. Tsai, "Deploying chains of virtual network functions: On the relation between link and server usage," *IEEE/ACM Trans. Netw.*, vol. 26, pp. 1562–1576, Aug. 2018.

[9] C. Sun *et al.*, "NFP: Enabling network function parallelism in NFV," in *Proc. of ACM SIGCOMM 2016*, pp. 43–56, Aug. 2017.

[10] 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.

[11] J. Anderson *et al.*, "Performance considerations of network functions virtualization using containers," in *Proc. of ICNC 2016*, pp. 1–7, Feb. 2016.

[12] Y. Le *et al.*, "UNO: Uniflying host and smart NIC offload for flexible packet processing," in *Proc. of SoCC 2017*, pp. 506–519, Sept. 2017.

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

[14] 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.

[15] 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.

[16] 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.

[17] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *Proc. of CNSM 2014*, pp. 418–423, Nov. 2014.

[18] C. Kachris, G. Sirakoulis, and D. Soudris, "Network function virtualization based on FPGAs: A framework for all-programmable network devices," *arXiv preprint arXiv:1406.0309*, 2014.

[19] (2019) Amazon web services. [Online]. Available: https://aws.amazon.com/cn/ec2/pricing/reserved-instances/pricing/

[20] (2019) Colfax direct. [Online]. Available: http://www.colfaxdirect.com/store/pc/viewPrd.asp?idproduct=3017&idcategory=0

[21] X. Zhao, H. Lu, C. Chen, and J. Wu, "Adaptive hybrid digital-analog video transmission in wireless fading channel," *IEEE Trans. Circuits Syst. Vido Technol.*, vol. 26, pp. 1117–1130, Jun. 2015.