Leveraging Heterogeneous NFV Platforms for QoS-aware Reconfiguration of vNF Service Trees

(*Invited Paper*)

Tingyu Li and Zuqing Zhu, *Senior Member, IEEE*
University of Science and Technology of China, Hefei, Anhui 230027, China, Email: zqzhu@ieee.org

***Abstract*—We study the problem of how to realize quality-of-service (QoS) aware reconfiguration of virtual network function service trees (vNF-STs) over the heterogeneous network function virtualization (NFV) platforms that include virtual machines (VMs), docker containers, and programmable data plane switches (PDP-SWs), and propose an effective heuristic algorithm.**

***Index Terms*—Network function virtualization (NFV), Heterogeneous NFV platforms, Virtual network functions (vNFs).**

I. Introduction

Nowadays, the advances on physical-layer [1–4] and virtualization [5–7] technologies have greatly promoted the idea of network function virtualization (NFV) [8]. Specifically, instead of relying on special-purpose middle-boxes, NFV realizes network services by deploying virtual network functions (vNFs) on general-purpose software/hardware platforms (*e.g.*, virtual machines (VMs), docker containers (Dockers), and programmable data plane switches (PDP-SWs) [9, 10]) for high flexibility [11]. For instance, a service provider (SP) can decompose network services into atomic vNFs, instantiate them on general-purpose platforms, and steer application traffic through the required vNFs of each network service (*i.e.*, setting up network services with vNF service chains (vNF-SCs)) [12]. Meanwhile, with the fast development of multi-client network services such as webcasts, online games, and metaverse, an SP might need to organize vNFs in a tree-type forwarding graph to satisfy the demands for point-to-multiple-point communications [13]. Such a vNF forwarding graph is referred as a vNF service tree (vNF-ST), where one vNF can process the application traffic to multiple clients, and the clients on different branches of the vNF-ST can have their application traffic processed by different sets of vNFs.

Traditionally, vNFs are deployed on software platforms such as VMs and Dockers, but the emergence of in-network computing [14] has made SPs realize vNFs on forwarding devices (*e.g.*, PDP-SWs) to handle computing tasks during packet processing. PDP-SWs have superior packet processing capability and thus are suitable for carrying bandwidth-intensive vNFs [15], while VMs and Dockers are runtime programmable and provide sufficient computing and memory resources for computing-intensive vNFs [16]. Therefore, heterogeneous NFV platforms that consist of hardware (PDP-SWs) and software (VMs and Dockers) systems can unify their benefits, and provide SPs more flexibility to satisfy various quality-of-service (QoS) demands cost-effectively. This actually brings many advantages to the provisioning of vNF-STs.

One important advantage of introducing heterogeneous NFV platforms is that incremental installation of such platforms can still effectively improve the QoS of in-service vNF-STs. Specifically, after installing new NFV platforms in its network, an SP can plan the deployment of vNFs on them and reconfigure certain in-service vNF-STs to improve their QoS performance. However, to the best of our knowledge, how to design an algorithm to effectively address such a situation has not been studied in the literature. Previous studies considered how to provision vNF-SCs over heterogeneous NFV platforms [15, 17, 18] and how to upgrade an NFV network environment with heterogeneous platforms [19]. However, due to their complex topologies, the provisioning and reconfiguration of vNF-STs are much more challenging than those of vNF-SCs.

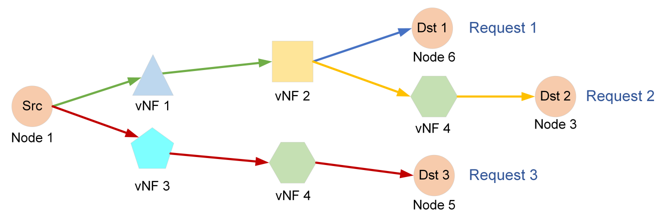
In this paper, we address the problem of how to realize QoS-aware reconfiguration of vNF-STs over heterogeneous NFV platforms, and propose a layering and aggregation-based (LAA) algorithm for it. The rest of the paper is organized as follows. Section II describes the network model and algorithm design. We present the simulation results in Section III. Finally, Section IV summarizes the paper.

II. Network Model and Algorithm Design

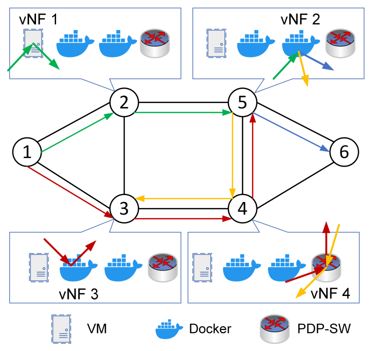
A. Network Model

We model the substrate network (SNT) as an undirected graph $G(V, E)$, where V and E are the sets of substrate nodes (SNs) and links, respectively. Each SN $v \in V$ can contain a few heterogeneous NFV platforms. In this work, three types of heterogeneous NFV platforms are considered, *i.e.*, VMs, Dockers and PDP-SWs. On them, we assume that a total of t types of vNFs can be deployed and each NFV platform can only carry one type of vNFs. Each NFV platform has its own capacities of IT and bandwidth resources. Fig. 1 shows an example on the vNF-STs considered in this work, which consists of a source and several destinations. Each branch in the vNF-ST is actually a vNF-SC between a source-destination pair, and we refer to it as a request in this work, which has its own latency requirement. As shown in Fig. 1(a), different requests in a vNF-ST can share vNFs, *i.e.*, the sequence of *Source*$\rightarrow$*vNF* 1$\rightarrow$*vNF* 2 is shared by *Requests* 1 and 2.

Based on the vNF-ST deployment in Fig. 1(b), we consider three scenarios of vNF-ST reconfiguration, namely vNF



(a) Topology of vNF-ST



(b) Deployment scheme

Fig. 1. Example on deploying a vNF-ST over heterogeneous NFV platforms.

change, latency requirement change, and platform failure, as illustrated in Fig. 2. First, *Request* 1 in the vNF-ST in Fig. 1(b) changes from *Node* 1→*vNF* 1→*vNF* 2→*Node* 6 to *Node* 1→*vNF* 1→*vNF* 3→*Node* 6. Hence, the SP changes its path from 1→2→5→6 to 1→2→3→4→6. Second, the latency requirement of *Request* 2 gets tightened, and thus the SP needs to deploy a *vNF* 2 on the PDP-SW of *Node* 5, and reconfigure *Request* 2 to use it. Finally, the *vNF* 1 that was deployed on a VM on *Node* 2 is down, and thus a new *vNF* 1 should be instantiated there (*e.g.*, on a Docker). In the following, we will explain our algorithm design to optimize the vNF-ST reconfiguration scenarios mentioned above.

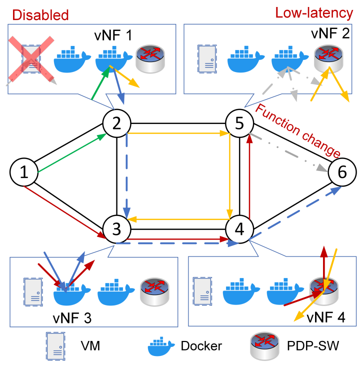


Fig. 2. Examples on vNF-ST reconfiguration scenarios.

B. Algorithm Design

To plan the vNF-ST reconfiguration cost-effectively, we design the optimization to consider two aspects jointly, *i.e.*, the reconfiguration cost and total resource usage after reconfiguration. Specifically, the objective is formulated as

$$\text{Minimize} \qquad S = \alpha \cdot (T_b + T_v) + (1-\alpha)\cdot M, \tag{1}$$

where T_b and T_v respectively denote the overall bandwidth and IT resource usages, M is the reconfiguration cost due to migrating vNFs, and $\alpha \in (0,1)$ is the weight coefficient to balance the importance between the two aspects. Specifically, when reconfiguring a vNF-ST, we might need to reroute the traffic to one vNF to a new one, and to make sure that such a reconfiguration is hitless to the service of the vNF-ST, the state information of the original vNF needs to be migrated to the new one. This actually generates additional operational cost, which can be modeled as M.

We propose an LAA algorithm to optimize the vNF-ST reconfiguration according to the objective in Eq. (1). The input to the algorithm is a set of requests that need to be reconfigured due to the three aforementioned reasons (*i.e.*, vNF change, latency requirement change, and platform failure), and it leverages the following steps to reconfigure the requests.

Step1: We conduct the preprocessing to address the platform failures and change of latency requirements. Specifically, we first remove the failed platforms from the SNT, and then find the candidate NFV platforms that might be used to satisfy the new latency requirements of requests.

Step2: We deploy new vNFs according to the layered idea as follows. In a vNF-ST, each layer of vNFs refers to the vNFs that have a equal hop-count to the source. For instance, in Fig. 1(a), the *vNFs* 1 and 3 that are directly connected to the source are in the same layer. For each layer of vNFs, we first aggregate the vNFs according to their types, and try to group the vNFs of the same type to be placed on a same NFV platform. If the IT resource demands of vNFs cannot be satisfied by one NFV platform, the remaining vNFs will be included in more groups. Then, we sort the vNF groups in descending order of their IT resource demands, and deploy the one with the largest IT resource demand first. Next, for each vNF group, we find the platform with the smallest deployment cost to place it, as long as the latency requirements of the requests that use the vNFs in the group can be satisfied.

Step3: We calculate the shortest paths to connect the deployed vNFs according to the topologies of the reconfigured vNF-STs, and obtain the new provisioning schemes of them.

III. Performance Evaluations

Our simulations use the 14-node NSFNET topology [20] for the SNT, and assign a latency to each link in it (as shown in Fig. 3). On each node in the SNT, we assume that there are one VM, two Dockers, and one PDP-SW. The weight coefficient α in Eq. (1) is set as 0.5. According to the realistic data presented in [15], the normalized unit cost of bandwidth usage is set as 2 per Gbps, and the normalized costs of instantiating a VNF on VM, Docker and PDP-SW are 1, 1.6 and 1.76, respectively. The bandwidth capacities of a VM, a Docker, and a PDP-SW are assumed to be 1.5 Gbps, 1.3 Gbps, and 10 Gbps, respectively, and the processing latencies of VMs, Dockers, and PDP-SWs are set as 200 μs, 150 μs, and 10 μs, respectively. The bandwidth requirement of each request is set as 0.1 Gbps, and its latency requirement is within $[0.2, 1]$ msec. The simulations average the results from 10 independent runs to get each data point, for sufficient statistical accuracy.

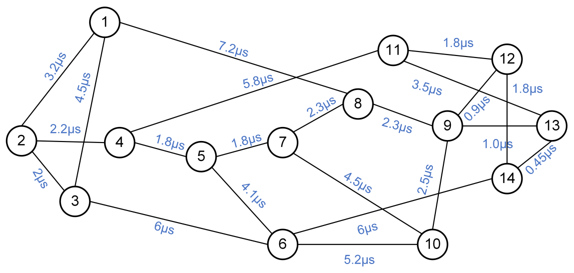
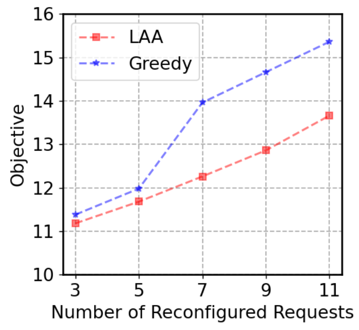
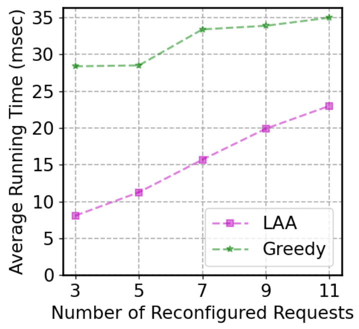


Fig. 3. NSFNET topology for SNT.



(a) Optimization objective



(b) Average running time

Fig. 4. Performance comparison of LAA and greedy algorithms.

We compare our proposed LAA algorithm (LAA) with a simple greedy algorithm (Greedy) in the simulations. Specifically, the greedy algorithm sorts the requests that need to be reconfigured according to their reasons for reconfiguration, *i.e.*, latency requirement change is the first, platform failure is the second, vNF change is the last, and then reconfigures the requests in sorted order one by one. In each simulation, we first deploy 5 vNF-STs that include 13 requests in total, where each request consists of $[1, 3]$ vNFs. Then, we consider that there are $\{3, 5, 7, 9, 11\}$ requests, which need to be reconfigured, and randomly select the reconfiguration scenario of each request.

Fig. 4(a) compares the optimization objectives obtained by the two algorithm. It can be seen that LAA outperforms Greedy to provide smaller objectives, and when the number of requests is relatively small, the performance gap is not very large, but it increases significantly with the number of requests. This confirms the effectiveness of our proposal. Fig. 4(b) compares the running time of the algorithms, and we can see that LAA actually runs faster than Greedy. This is because LAA aggregates requests before reconfiguring them. In all, the results in Fig. 4 verifies the performance of LAA.

IV. Conclusion

In this paper, we proposed an LAA algorithm for realizing QoS-aware reconfiguration of VNF-STs over heterogeneous NFV platforms. Simulation results confirmed that our proposal saves running time and obtains more cost-effective reconfiguration schemes than the greedy approach.

Acknowledgments

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

References

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

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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.

[8] M. Chiosi *et al.* (2012) Network functions virtualisation. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf.

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

[10] S. Li *et al.*, “Protocol oblivious forwarding (POF): Software-defined networking with enhanced programmability,” *IEEE Netw.*, vol. 31, pp. 12–20, Mar./Apr. 2017.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.