Dynamic VNF Placement for Mapping Service Function Chain Requests in NFV-enabled Networks

Yi Yue, Bo Cheng, Xuan Liu, Meng Wang, Biyi Li
{yueyi,chengbo,liuxuan0527,mengwang,lbyi0402}@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

ABSTRACT

Network function virtualization (NFV) brings significant performance and management benefits for the enterprises to place virtual network functions (VNFs) while reducing the operating expenses and capital expenditures. In this paper, we study the dynamic VNF placement problem for mapping users' service function requests (SFCRs) in NFV-enabled networks. The problem is formulated as a heuristic cost function aiming at minimizing the cost of mapping each SFCR. Then we propose SFCR-Mapping and Dynamic VNF-Releasing algorithms to efficiently map SFCRs and timely release the redundant VNF instances. The simulation results demonstrate that our method efficiently reduces the total activated VNF instances and improves VNF utilization compared with the benchmarks.

CCS CONCEPTS

• **Networks → Cloud computing**.

KEYWORDS

Dynamic VNF placement, resource optimization

ACM Reference Format:
Yi Yue, Bo Cheng, Xuan Liu, Meng Wang, Biyi Li. 2020. Dynamic VNF Placement for Mapping Service Function Chain Requests in NFV-enabled Networks. In *Companion Proceedings of the Web Conference 2020 (WWW '20 Companion), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3366424.3382689

1 INTRODUCTION

Traditional middleboxes or network functions (NFs) are implemented by specific hardware appliances, which bring high capital expenditures and operating expenses (CAPEX/OPEX). By exploiting network function virtualization (NFV) technology, the cloud service providers (CSPs) can realize NFs in software and flexibly instantiate them in the form of virtual network functions (VNFs) on common off-the-shelf servers.

In NFV-enabled networks, a customer's service request is implemented by a service function chain (SFC), which contains a series of VNFs concatenated in a specified order. When processing hundreds of requests from different clients, it is an intractable but important issue for CSPs to devise an optimal VNF placement scheme for mapping SFC requests (SFCRs). Moreover, since the highly dynamic features of start-up, lifetime and resource usage, SFCRs may incur the variation of network load. A good VNF placement scheme not only improve the utilization of network resources but also significantly reduce the CAPEX/OPEX. Hence, we study the VNF placement problem for mapping SFCRs and formulate the problem as a heuristic cost function, with the purpose of minimizing the mapping cost of SFCRs. Then we propose SFCR-Mapping (SFCR-M) algorithm to place and select VNF instances for SFCRs. Besides, we devise Dynamic VNF-Releasing (DVNF-R) algorithm to optimize the quantity of placed VNF instances by releasing redundant instances based on the time-varying network load.

2 DYNAMIC VNF PLACEMENT FOR SFCRS

2.1 Network Model

We denote the substrate network as an undirected graph $G = (N, E)$, where N and E represent the sets of physical nodes and links in substrate networks, respectively. And $u, v \in N$ stand for two physical nodes and the node pair (u, v) indicates the link between u and v. Moreover, we define $G_i = (N_i, E_i)$ to indicate an SFCR i, where $\bar{u}, \bar{v} \in N_i$ represent two VNF request (VNFR) nodes in SFCR i and $(\bar{u}, \bar{v})$ represents the logical link between $\bar{u}$ and $\bar{v}$.

2.2 Proposed Approach

In our work, we consider utilizing resource costs to balance network load and eliminate resource bottlenecks. Therefore, when mapping an SFCR i, we use $c_{i,u}^{cpu}$, $c_{i,u}^{mem}$ and $c_{i,uv}^{bw}$ to indicate the costs of CPU, memory and bandwidth on node u and link (u, v), respectively. And they can be calculated as:

$$c_{i,u}^{cpu} = \frac{r_{i,u}^{cpu} C_u^{cpu}}{\max\limits_{u \in N} C_u^{cpu}}, \tag{1}$$

where $r_{i,u}^{cpu}$ and C_u^{cpu} denote the CPU utilization rate and the CPU capacity of node u, respectively. The denominator $\max\limits_{u \in N} C_u^{cpu}$ indicates the maximum CPU capacity in the network. $c_{i,u}^{mem}$ and $c_{i,uv}^{bw}$ can be calculated similarly as $c_{i,u}^{cpu}$.

The resource cost functions illustrated above increase or decrease as the network load fluctuates. Therefore, these functions reflect the resource bottlenecks in the network, and we can combine them to indicate the cost of a path for mapping an SFCR i:

$$\mathbb{M} = \sum_{u \in N} \sum_{\bar{u} \in N_i} (c_{i,u}^{cpu} + c_{i,u}^{mem}) \cdot x_{\bar{u},u}^{f} + \sum_{(u,v) \in E} \sum_{(\bar{u},\bar{v}) \in E_i} c_{i,uv}^{bw} \cdot y_{uv}^{f,\bar{u}\bar{v}} \tag{2}$$

where $x^{f}_{\bar{u},u}$ and $y^{f,\bar{u}\bar{v}}_{uv}$ indicate whether $\bar{u}$ and $(\bar{u},\bar{v})$ of SFCR i are mapped on node u and link (u,v), respectively. Moreover, we formulate the VNF placement cost of mapping an SFCR i as follows:

$$\mathbb{P} = \sum_{\lambda\in\Lambda}\sum_{\bar{u}\in N_i} vpc_\lambda \cdot l^{i}_{\bar{u},\lambda} \cdot max\{k_{i,\lambda} - \hat{k}_{i,\lambda}, 0\} \tag{3}$$

where vpc_λ is the placement cost of a type-λ VNF and $l^{i}_{\bar{u},\lambda}$ denotes whether VNFR $\bar{u}$ of SFCR i requires VNF λ. Besides, $\hat{k}_{i,\lambda}$ and $k_{i,\lambda}$ respectively indicate whether VNF instance $\lambda \in \Lambda$ is deployed before ($\hat{k}_{i,\lambda} = 1$) and after ($k_{i,\lambda} = 1$) mapping SFCR i.

With sufficient network resources, the VNF placement cost $\mathbb{P}$ is much higher than the resource cost $\mathbb{M}$. So we reuse the deployed VNF instances to map SFCRs. And once $\mathbb{M}$ is much higher than $\mathbb{P}$, compared with reusing the placed VNF instances, deploying new instances can bring more benefits. We jointly exploit these two methods to map SFCRs to diminish resource bottlenecks and balance the network load. Our target is:

$$min(\mathbb{M} + \mathbb{P}) \tag{4}$$

Afterward, to describe the variation of network load, we have:

$$\Psi(t) = \frac{1}{T}\int_{t-T}^{t} \psi(t)dt. \tag{5}$$

where $\psi(t)$ is the network load at time t and $\Psi(t)$ is the mean network load during $(t - T, t]$. Then the fluctuation of network load can be calculated as:

$$\delta(t) = \frac{1}{T}\int_{t-T}^{t} |\psi(t) - \Psi(t)|\, dt. \tag{6}$$

Moreover, we use a parameter ϕ to indicate the fluctuation threshold of network load. Based on Eq. (5) and (6), we have $w(t)$ to denote the threshold of VNF utilization at time t,

$$w(t) = \begin{cases} w_l, & \Psi(t-T) > \Psi(t), \delta(t) > \phi \\ w_s & otherwise, \end{cases} \tag{7}$$

$\Psi(t - T) > \Psi(t)$ and $\delta(t) > \phi$ indicate the network load decreases gradually, so we set $w(t)$ to a large value w_l to increase the number of VNFs being checked, which facilitates the releasing of redundant VNF instances. Otherwise, we set $w(t)$ to a small value w_s to reduce the number of VNFs being checked. For each SFCR, we apply

Algorithm 1 SFCR Mapping Algorithm

```
Input: Set of SFCRs: $I$; Status of the network: $\mathcal{S}_0$;
1: for each SFCR $i \in I$ do
2:     Obtain target nodes for mapping SFCR using Eq. (4)
3: end for
4: return Set of used nodes $N_0$ and network status $S_1$.
```

Algorithm 1 to map it on nodes by minimizing the mapping cost $\mathbb{M} + \mathbb{P}$. Afterward, based on the time-varying network load, we dynamically adjust $w(t)$ to decide the number of instances to be checked. The VNF releasing process is shown in Algorithm 2. We store the instances whose utilization $\sigma_v(t) \leqslant w(t)$ to a list L, and release them when they serve no SFCRs.

Algorithm 2 Dynamic VNF-Releasing Algorithm

```
1: Calculate the current $\Psi(t)$ and $\delta(t)$ using Eq. (5) and (6).
2: Update the current $w(t)$ using Eq. (7).
3: for each used VNF instance $v \in V$ do
4:     Calculate the utilization $\sigma_v(t)$ of VNF instance $v$.
5:     Store the VNF instance $v$ whose $\sigma_v(t) \leqslant w(t)$ to a list $L$.
6: end for
7: for each $v \in L$ do
8:     Release the VNF instance $v$ when it serves no SFCRs.
9: end for
10: return Network status $S_2$
```

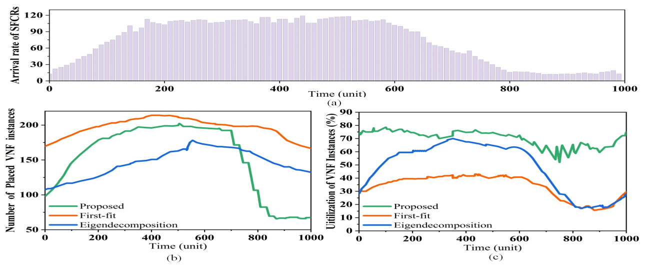

Figure 1: Performance Evaluation

3 PERFORMANCE EVALUATION

We compare our method with the First-fit (FF) and ProvisionTraffic algorithm[1]. We implement the models and algorithms based on Alevin[2], a widely used simulation environment for VNF placement. The arrival rate of SFCRs obeys the rule in Fig. 1(a). We conduct the evaluation in 1000 time units, and the execution period of DVNF-R is 50 time units. The values of ϕ, w_l and w_s are set to 50Mbps, 50% and 25 %, respectively.

From Fig. 1b, we can see that our proposed approach outperforms the benchmarks in terms of optimizing the placed VNF instances. When the network load declines during 600-1000 time units, DVNF-R perceives this phenomenon and updates the $\sigma_v(t)$ to a larger value w_l using Eq. (7). Hence, compared to the benchmarks, we can release more VNF instances whose utilization is under w_l, thereby reducing the unnecessary resource consumption. Fig. 1c shows the average VNF utilization of three methods. When network load is heavy during 200-600 time units, our approach can obtain an average utilization of more than 75%. Since our approach periodically releases low-utilization instances, when the network load declines in 600-1000 time units, it can keep the average VNF utilization at about 65%, which is higher than two benchmarks.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China (2018YFB1003804), in part by the National Natural Science Foundation of China under grant 61921003.

REFERENCES

[1] Md. Faizul Bari and et al. 2016. Orchestrating Virtualized Network Functions. *IEEE Trans. Network and Service Management* 13, 4 (2016), 725–739.
[2] J. Gil Herrera and et al. 2016. Resource Allocation in NFV: A Comprehensive Survey. *IEEE Transactions on Network and Service Management* 13, 3 (2016), 518–532.