

A Reliable Resource Scheduling for Network Function Virtualization

Daoqiang Xu¹, Yefei Li², Ming Yin³, Xin Li⁴(✉), Hao Li⁵, and Zhuzhong Qian⁵

¹ State Grid Jiangsu Electric Power Company, Nanjing 210000, China
13851759165@139.com

² Jiang Su Frontier Electric Technology Co. Ltd., Nanjing 210000, China
imliyf@163.com

³ State Grid Nanjing Power Supply Company, Nanjing 210000, China
13605146644@139.com

⁴ Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
lics@nuaa.edu.cn

⁵ State Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China
qzz@nju.edu.cn

Abstract. Network function virtualization (NFV) is designed to reduce high cost of the hardware deployment and maintenance, which will be of great importance in the future resource management. NFV makes it possible for general servers to realize fast deployment of network functions, namely virtual network function (VNF), to achieve on-demand requests. The service request is generally composed of a sequence of VNFs, which is defined as a service chain. Although one node (server) only run one VNF at a time, the VNF could be shared with several VNF chains. How to deploy VNF and share VNF to achieve high usage of both computing and bandwidth resource is a challenge. In this paper, we present the VNF request model and the formal definition of the VNF deployment problem. Then, we investigate the VNF reuse and deployment problem, and propose a greedy strategy to deploy VNFs, so that the system could effectively offer a reliable VNF service. Simulations show that this VNF deployment schema could effectively improve the resource usage compared with other classic algorithms.

Keywords: Network function virtualization · Resource management
VNF deployment · Reliable method · Virtual Network Embedding

1 Introduction

With the rapid development of Internet and the increase of network function, more and more middlebox will be deployed in the network out of technology requirement and function requirement. Middlebox which is playing a crucial role in current network can provide plentiful network functions from security aspects, such as Firewalls, collision detection and etc.; to service aspects, such as cache,

server Proxy and etc., which make hardware and deployment cost much too high. It takes hardware that requires a lot of specialization and very complicated process to deploy network middlebox. This means that hardware-based middlebox may result in significant human resources and physical resources costs, meanwhile reducing the flexibility of the network system.

Network function virtualization (NFV) aims to separate the network function from the underlying proprietary hardware and make software function running in the more general hardware instead of function running in a specialized hardware. By separating network function from specialized hardware, the deployment of physical device will be quite flexible, while the cost of redeployment of network function will be reduced by changing and repairing software instead of changing hardware device. In NFV, virtual network function (VNF) offers the basic service instead of traditional middlebox.

Since the function of single VNF is limited, the user request is often composed of several VNFs and the dataflow goes through these VNFs in a certain order, which is named as service chain. When a new service chain request is submitted, the system need to allocate new nodes (servers) to deploy VNFs and reserve related bandwidth to construct a VNF path for the coming dataflow. Generally, one node could only run one VNF at a time, while the running VNF could be shared with several VNF service chains. That is, for a new coming service chain request, we may map the VNFs to some exist VNF nodes where run the same functions. However, although this resource sharing improves computing resource utilization, it may also increase the bandwidth cost. This is because the exist running VNFs for one service chain may located in different nodes that is far away. And there is also capacity limitation for VNF sharing, we could not overload the VNF nodes. Consequently, how to allocate resources to an incoming VNF service chain request to effectively offer a reliable VNF service, is a challenge problem.

In this paper, we firstly present the VNF service chain request model and formally define the reliable resource allocation for VNF service chain. We also deeply investigate the resource sharing mechanism and propose dynamic resource sharing schema to increase the resource sharing. And then, we propose the resource scheduling mechanism, which includes the VNF reuse and VNF deployment. The simulations show that this mechanism could effectively allocate resources and have a good performance compared with other classic algorithms.

The remainder of this paper is organized as follows. We introduce the related work in Sect. 2. The problem formulation and VNF scheduling mechanism are proposed in Sects. 3 and 4. Finally we conclude our paper in Sect. 5.

2 Related Work

Derived from *Virtual Network Embedding problem* (VNE) [5] that mapping *Virtual Network* (VN) to the *Substrate Network* (SN) with guaranteed *Quality of Service* (QoS) [7], the VNF deployment problem has been proposed to be a important research challenge [9]. A strategy calculating a new VN embedding

solution efficiently when some nodes failed was proposed in Shahriar et al. [10], and Cankaya et al. [1] proposed a VNE algorithm for link protection.

There are some existing work about the VNF deployment problem. Wang et al. [11] proposed a strategy using preplanned VNF deployment solutions to minimize network communication cost. An algorithm consolidating adjacent VNFs to one server was proposed to reduce the resource cost in the network in Ye et al. [12]. Kuo et al. [8] designed a two-step VNF deployment strategy, and argued that taking both server resources and link resources into consideration leading to a better performance in VNF deployment. VNF deployment was divided into two NP-hard subproblems in Cohen et al. [2], and authors designed strategies related to the subproblems to solve the deployment problem. For reliable issue, Hmaity et al. [6] concluded 3 redundancy models of VNF deployment, while Fan et al. [4] minimized the total resources cost by consolidating VNF redundant backups. Besides, in response to dynamic traffic of NFV, [3, 13] proposed some online VNF deployment to implement reconfigurations when request traffics change.

3 Background and Problem Statement

3.1 Network Function Virtualization

In NFV, hardware and software are separated, every network function is independent from the physical environment. A physical network can be modeled as an undirected graph $G^s=(V^s, E^s)$, where V^s is a set of nodes and E^s is a set of edges, while v^s represents a node and e^s represents an edge. $Rest(e^s)$, e^s is the rest bandwidth capacity of an edge e^s , $Rest(v^s)$, v^s is the residual computing power of a node v^s . If the rest bandwidth capacity of an edge is insufficient, no more data transmission could be through this link; and if the rest computing power of a node is insufficient, no more VNF could be deployed on it. Figure 1 shows a naive physical network with only six nodes, where the number in the node represents the residual computing power of a node, and the number on the edge represents the rest bandwidth capacity of an edge. An update will be done after a deployment of a new requirement to the network.

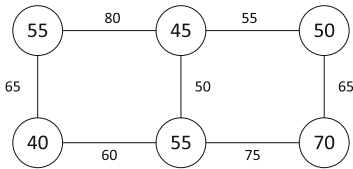


Fig. 1. Physical network

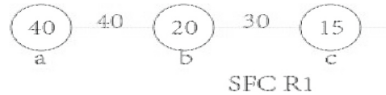


Fig. 2. Service chains

A complete network function is often composed of several VNFs in a certain order, named service chain. Similar to the definition of physical network, a service chain is defined as $R_i=(r_1^i, r_2^i, r_3^i, r_n^i)$, where r_j^i represents the j th network function in the i th service chain. $|R_i|$ is the number of the virtual network

function in the service chain. A directed subgraph $G_{R_i}^t = (V_{R_i}^t, E_{R_i}^t)$, where $V_{R_i}^t = (v_{R_1}^t, v_{R_2}^t, v_{R_3}^t, \dots, v_{R_j}^t)$ is a set of the nodes that service chain R_i passes, $E_{R_i}^t = (e_{R_1}^t, e_{R_2}^t, e_{R_3}^t, \dots, e_{R_j}^t)$. In addition, every VNF has bandwidth requirement to transfer data as well as computing capacity. $Cp(r_j^i)$ is the required computing capacity of the j th network function in the i th service chain, $\text{bandwidth}(r_j^i)$ is the requiring bandwidth between j th network function and $(j+1)$ th network function in the i th service chain. Figure 2 shows that four function nodes in service chain $R1$ require a chain of computing capacity of (40, 20, 15, 25) and bandwidth of (40, 35, 25).

Since different VNF is deployed for different purposes (e.g. network address translate, firewall, etc.), different VNF requires different bandwidth resources and computing resources. For example, an NAT will not require much computing resources, while an IDS requires a lot. In this paper, we define π_{v,v_1} as the path between node v and node v_1 , $|\pi_{v,v_1}|$ is the length of the path. $\text{Map}(R_{i,j})$ is the physical node mapped to the j th function in the R_i service chain, and obviously $\text{Map}(R_{i,j}) \in V_{R_i}^t$.

3.2 Problem Statement

The request of VNF is represented as service chains, as defined above. For a service provider, finding an optimal deployment plan for multiple service chains is a challenge problem. When a VNF request arrives, system firstly find out whether it can be accepted by the rest resources. Generally, a node (i.e. sever) can be mapped to only one VNF function, but can be shared to multiple service chains if they have the same function request in the chain. The formal goal of the deploying is defined as followed.

$$\begin{aligned} & \text{maximize} : N_R^s \\ & \text{minimize} : \text{Cost}(G^s, R, M) \end{aligned}$$

In this equation, N_R^s is the number of the requirement in R which can be accepted by network G^s , $\text{Cost}(G^s, R, M)$ is the cost of R in the network G^s .

In this paper, we consider two aspects of problem, to reduce the cost of computing resources and to reduce the cost of bandwidth resources. the process is divided into two parts:

1. Deployment of functions: optimal nodes set will be selected to load all the required VNF because virtual functions of different nodes in the physical network are different. A better deployment plan will promote the network efficiency, and make the deployment easier.
2. Selection of path: after selection of nodes, a proper path between nodes is also needed to make one node corresponding with another and achieve a splendid working efficiency.

For an incoming VNF chain requests, we need to map all the VNFs of the service chain and construct a physical path to link all the nodes. Since we can

Table 1. Denotations

G^s	a physical network
V^s	a set of nodes
E^s	a set of edges
$Rest(e^s)$	the rest bandwidth capacity of an edge e^s
$G_{R_i}^t$	a directed subgraph
r_j^i	the requiring bandwidth between jth network function and (j+1)th network function in the ith service chain
$Map(R_{i,j})$	the physical node mapped to the jth function in the R_i service chain
$p^t h$	a threshold value
$p_{v_i}^{R_i}$	the floating possibility of the service chain R_i who requires the resources of node v_i
$bandwidth(r_j^i)$	the requiring bandwidth between jth network function and (j+1)th network function in the ith service chain
$Cp(r_j^i)$	the requiring computing capacity of the jth network function in the ith service chain

reuse VNFs that already deployed in the substrate network, we may have two choices for any VNF.

1. Select a new node where no VNF is deployed.
2. Select a rest-computing-power-enough node where a same VNF has already been deployed.

It is non-trivial to construct the mapping plan for all the VNFs in a service chain to have the best performance, since choosing new nodes to deploy VNFs cost computing resources while reusing VNFs may cost more bandwidth. For example, a service chain R_i includes 3 VNF A, B and C, Fig. 3 shows the substrate network, where a, b and c are 3 nodes without any VNFs, while d, e and f are deployed VNF A, B and C respectively. Thus, we can deploy A, B and C again to node a, b and c, and the link cost is $bandwidth(a) + bandwidth(b)$. But if we reuse A, B and C from node d, e and f, the cost is $bandwidth(a) * 2 + bandwidth(b) * 2$.

Furthermore, the sharing node has the capacity constraints, we can not map all the VNF to one node which may cause performance decline. In previous works, the resource requirement of VNF is always fixed when the VNF request is submitted. In this paper, we divide the resource requirement into two parts: one is basic (fixed) resource requirement and one is dynamic part with the possibility p, which is defined as a three tuple. For a VNF r_k^i , the tuple $(r_k^i) <basic_i^k, variable_i^k, p_i^k>$ presents the resource requirement, where $basic_i^k$ is the basic resource and $variable_i^k$ is the resource with possibility p_i^k . In the Fig. 4, the basic requirement of $R_{1.a}$ is 8 and the dynamic requirement of a is 4 and the possibility is 0.3.

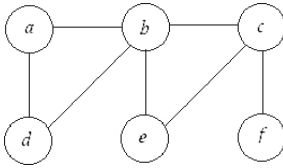


Fig. 3. A possible physical network model

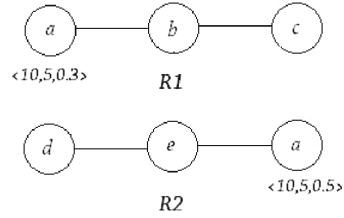


Fig. 4. Two possible service chains

If two VNFs are mapping to one node (i.e. two VNF requests share one node), if $RestCp(v_i) \geq \sum_i^j basic_i^k + variable_i^k$, the second requirement is accepted. If the condition is not satisfied, the deployment will not be refused directly yet. $p_{v_i}^{R_i}$ is the floating possibility of the service chain R_i who requires the resources of node v_i , given a threshold value p^th , if the possibility of two service chains requiring the resources at the same time below the threshold value, what is:

$$p_{v_i}^{R_i} * p_{v_j}^{R_j} < p^th$$

As the figure shows, chain R_1 and R_2 have the same VNF a . so they can share the nodes. They want to be deployed in a virtual node whose computing power is 25. For the given tuple $(r_1^1) <10, 5, 0.3>$ and tuple $(r_2^2) <10, 5, 0.5>$, and $func(r_1^1)=func(r_2^2)$, suppose $p^th = 0.2$ in this network. Since,

$$0.3 \times 0.5 = 0.15 \leq 0.2$$

The collision possibility is lower than threshold, so the VNF a in both R_1 and R_2 may map to this node and share the resource.

4 VNF Deployment Mechanism for Dynamic Requirements

Reuse VNF that has already been deployed could save computing resources, while deploying VNFs along with a shortest path will consume the least bandwidth resources. In this paper, we define VNF reuse factor x to represent the reused VNFs in a service chain and define $l(x)$ as the length of the edges when reusing x VNFs.

4.1 Finding the Reuse Factor x

For a certain service chain, it is difficult to find its optimal reuse factor. Assume that there are infinite same service chains R_i , find a x and $l(x)$ to maximum the number of the capacity of R_i . then it is considered to be a optimal plan, the aim is:

Definition 1 (Reuse Factor). *Given an infinite same service chains R_i , find a x and $l(x)$ such that the number of the capacity of R_i is maximized.*

$$\begin{aligned}
 & \max \quad n \\
 & \text{s.t.} \quad x \in \{0, 1, 2, \dots, \bar{x}\} \\
 & nTrans_i l(x) \leq \sum_{\substack{e \in E \\ e}} Res(e) \\
 & n(|S_i| - x) \leq \sum_{\substack{v \in V \\ v}} Res(v)
 \end{aligned} \tag{1}$$

The first constraint represents the range of x . Intuitively, $\bar{x}=|R_i|$, because the reuse nodes number is certainly lower than the function number of the chain. obviously, a lower \bar{x} value can be accepted because not all VNFs in the chain can be reused if some VNF is not been deployed in the network or the VNF is already overloaded. R'_i is the subchain in R_i that has the possibility to reuse, thus, $\bar{x}=|R'_i|$. The second constraint is that for every link in the R_i , the requirement must be lower than the bandwidth resources. The third constraint is for the computing power. We may try all the possible x , and choose the one that gets the maximum n .

4.2 VNF Deployment Strategy Based on Greedy

The greedy search algorithm described in this section consider both bandwidth and computing resource. The basic steps of deployment is as follows.

1. Find the reused function x for an incoming VNF chain.
2. Divide a service chain into $(x+1)$ subchains.
3. Greedily find an optimal deployment for each subchain, and finally deploy the entire VNF service chain.

In this algorithm, line 1 sorts all the incoming requests and handles the service chains according to priority. Line 3 to 5 find the accessible physical nodes and Line 6 to 11 select the optimal physical nodes for every VNF with some greedy strategy, and select an optimal path p_j , and G is updated. Here, the greedy strategy is to find the nodes with the largest flow bandwidth and computing resource. The distance between possible reused node will be controlled and costs of links will be reduced as a result of sorting the service chains by VNF reused number. How to deal with the $(x+1)$ subchains where no VNFs could be reused. These subchains may fill in one of the 4 possible situations: no reuse VNF at all, both begin and end nodes are reused VNF, one of the end node is reused VNF node.

As shown in Fig. 5, function c and e have nodes A, B and C, D respectively. Thus, the chain is divided into three subchains, $a-b-c$, $c-d-e$, and $e-f$. For $a-b-c$, this is a subchain that has one reused VNF. The algorithm will first deal with function c and then deploy a and b . For $c-d-e$, this is a subchain with two reused VNF nodes, and a shortest path can be found by breadth first search algorithm. For $e-f$, it is similar to $a-b-c$.

Algorithm 1. Greedy Deployment Algorithm

Input:

D:service chains which represent requirements set; G:physical network graph

Output:

the deployment of D in G; or D refused by G.

```

1: sorting(D)
2: for d in D do
3:   for each VNF in d do
4:     find the exist VNF nodes in G;
5:     return as  $V_i$ ;
6:   for each v in  $V_i$  except the last one do
7:     findPath  $p_j$  from  $v_j$  to  $v_{j+1}$ ;
8:   if  $p \neq \text{NULL}$  then
9:     Add  $p_j$  to P;
10:    Update(G);
11:  else
12:    return REJECT;
13: return P;

```

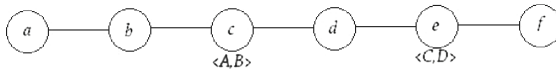


Fig. 5. A possible service chain

4.3 Simulation Results

A deployment model based on different feature networks and random requirements is designed to verify the robustness of the algorithm. Python networkX is used in the experiment. We set two substrate network topology, one has 300 vertex and 0.1 edge link possibility, the other one has 300 vertex and 0.3 edge link possibility, which is model randomly by setting the ER relationship is used to simulate real network environment. Each node has the computing power resources of 100 and every edge has the bandwidth of 100. 20 kinds of VNF and its requirement is set here, and about 20 to 120 service function chains whose length is between 2 and 6 are randomly combined. Bandwidth requirements between VNF functions in every chain are controlled between 20 and 50. We get the following results by simulations.

Figures 6 and 7 show that, two ways can not achieve the optimal result and the network simulated here is a bandwidth-free network, so the effect of the bandwidth greedy strategy is better than balanced deployment. Compare Fig. 6 with Fig. 7, when the link possibility improves, the result improves obviously. It is assumed that for more linked network, the distinct between different deployment algorithm is lower and the successful deployment is more. It is seen that the success rate is significantly improved by using a resources strategy based on dynamic requirements, which proves the effectiveness of the strategy to improve the network efficiency.

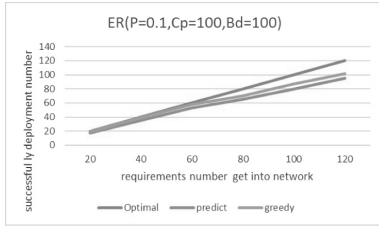


Fig. 6. ER($P=0.1$, $C_p=100$, $B_d=100$)

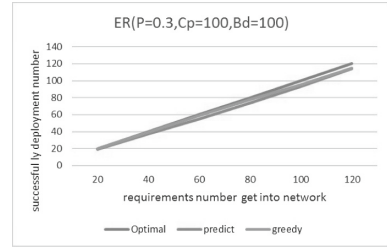


Fig. 7. ER($P=0.3$, $C_p=100$, $B_d=100$)

5 Conclusion

This paper investigates VNF service chain deployment problem. We firstly give the formal VNF request model and formalized the resource scheduling problem. And then, we present the VNF scheduling mechanism including finding VNF reuse factor and related resource scheduling algorithm based on greedy strategy. Simulations show that this mechanism works well and achieve better performance compared with other classic resource allocation algorithms.

References

1. Cankaya, H.C., Kim, I., Jue, J., Kong, J., Zhang, Q., Hong, S., Ikeuchi, T., Xie, W., Wang, X.: Availability-guaranteed virtual optical network mapping with selective path protection. In: Optical Fiber Communication Conference, p. W1B.4 (2016)
2. Cohen, R., Lewin-Eytan, L., Naor, J.S., Raz, D.: Near optimal placement of virtual network functions. In: 2015 IEEE Conference on Computer Communications (INFOCOM), pp. 1346–1354. IEEE (2015)
3. Eramo, V., Miucci, E., Ammar, M., Lavacca, F.G.: An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures. *IEEE/ACM Trans. Netw.* **25**, 2008–2025 (2017)
4. Fan, J., Ye, Z., Guan, C., Gao, X., Ren, K., Qiao, C.: Grep: guaranteeing reliability with enhanced protection in NFV. In: Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization, pp. 13–18. ACM (2015)
5. Andreas, F., Botero, J.F., Beck, M.T., De Meer, H., Hesselbach, X.: Virtual network embedding: a survey. *IEEE Commun. Surv. Tutor.* **15**(4), 1888–1906 (2013)
6. Hmaity, A., Savi, M., Musumeci, F., Tornatore, M., Pattavina, A.: Virtual network function placement for resilient service chain provisioning. In: 2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM), pp. 245–252. IEEE (2016)
7. Khan, M.M.A., Shahriar, N., Ahmed, R., Boutaba, R.: Simple: survivability in multi-path link embedding. In: International Conference on Network and Service Management, pp. 210–218 (2016)

8. Kuo, T.W., Liou, B.H., Lin, K.C.J., Tsai, M.J.: Deploying chains of virtual network functions: on the relation between link and server usage. In: The 35th Annual IEEE International Conference on Computer Communications. IEEE INFOCOM 2016, pp. 1–9. IEEE (2016)
9. Mijumbi, R., Serrat, J., Gorricho, J.L., Bouten, N., De Turck, F., Boutaba, R.: Network function virtualization: state-of-the-art and research challenges. *IEEE Commun. Surv. Tutor.* **18**(1), 236–262 (2015)
10. Shahriar, N., Ahmed, R., Chowdhury, S., Khan, A., Boutaba, R., Mitra, J.: Generalized recovery from node failure in virtual network embedding. *IEEE Trans. Netw. Serv. Manag.* **PP**(99), 1 (2017)
11. Wang, F., Ling, R., Zhu, J., Li, D.: Bandwidth guaranteed virtual network function placement and scaling in datacenter networks. In: 2015 IEEE 34th International Performance on Computing and Communications Conference (IPCCC), pp. 1–8. IEEE (2015)
12. Ye, Z., Cao, X., Wang, J., Hongfang, Y., Qiao, C.: Joint topology design and mapping of service function chains for efficient, scalable, and reliable network functions virtualization. *IEEE Netw.* **30**(3), 81–87 (2016)
13. Zhang, B., Hwang, J., Wood, T.: Toward online virtual network function placement in software defined networks. In: 2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS), pp. 1–6. IEEE (2016)