Reliable and efficient deployment for Virtual Network Functions

Jian Sun[1], Gang Sun[1,2], Dan Liao[1,3], Yao Li[1], Muthu Ramachandran[4], Victor Chang[5]

[1]Key Lab of Optical Fiber Sensing and Communications (Ministry of Education), University of Electronic Science and Technology of China, Chengdu, 611731, China
[2]Center for Cyber Security, University of Electronic Science and Technology of China, Chengdu, 611731, China
[3]Guangdong Institute of Electronic and Information Engineering, UESTC, 523808, China
[4]Leeds Beckett University, United Kingdom
[5]Xi'an Jiaotong-Liverpool University, Suzhou, 215123, China

Abstract: Network function virtualization (NFV) is a promising technique aimed at reducing capital expenditures (CAPEX) and operating expenditures (OPEX), and improving the flexibility and scalability of an entire network. However, this emerging technique has some challenges. A major problem is reliability, which involves ensuring the availability of deployed SFCs, namely, the probability of successfully chaining a series of big-data-based virtual network functions (VNFs) while considering both the feasibility and the specific requirements of clients, because the substrate network remains vulnerable to earthquakes, floods and other natural disasters. Based on the premise of users' demands for SFC requirements, we present an Ensure Reliability Cost Saving (ER_CS) algorithm to reduce the CAPEX and OPEX of telecommunication service providers (TSPs) by reducing the reliability of the SFC deployments. We employ big-data-based arbitrary topologies as the substrate network. The results of extensive experiments indicate that the proposed algorithms perform efficiently in terms of the blocking ratio, resource consumption and time consumption.

Keywords: Network Function Virtual, Service Function Chains, Reliability, Economical Big Data networking.

1 Introduction

Telecommunication service providers (TSPs) desire flexible and cost-efficient methods for dispatching network services as market demands increase. NFV provides an opportunity to efficiently and dynamically deploy service function chains (SFCs) [1–3] without modifying dedicated infrastructure, which is costly and has become complex over time. The basic idea behind NFV is to decouple these network functions (e.g., firewall, WAN optimizers, and proxies) from the underlying customized devices and accomplish equivalent network functions via software-based functions running in virtual machines deployed on devices.

Because the reliability of NFV is critical and is a prerequisite for successfully executing SFCs and satisfying service level agreements (SLAs), improving reliability while reducing the cost of network providers is a research objective in academic and industrial arenas.

In this paper, we propose an ER algorithm to solve this problem. High reliability requires TSPs to increase CAPEX and OPEX. If we can properly reduce the reliability, we can also reduce CAPEX and OPEX. We first propose the algorithm ER_CS (based on ER) that works in conjunction with the load balancing of the substrate network. However, by analyzing the deployment scheme in ER_CS, we discover that it does not appear to be the best scheme. Therefore, we further propose the ER_CS_ADJ algorithm to adjust the deployment scheme by minimizing SFC resource consumption in the physical network. We conduct massive simulations on arbitrary topologies to verify the effectiveness of these algorithms. From the simulations and results, we determine that our algorithms are profitable in terms of resource costs, block ratio and deployment time.

The remainder of this paper is organized as follows. In Section 2, we analyze related studies. In Section 3, we describe the problem in this research with some formulations. In Section 4, we propose our heuristic algorithm and provide line-by-line details. A performance evaluation of the network algorithm is presented in Section 5, and Section 6 concludes this work.

2 Related Work

To satisfy various requests from users, service providers are eager to seek a flexible, scalable, agile, effective, resource efficient and energy efficient scheme for placing VNFs. Ensuring service reliability while finding an economical and resource-efficient solution to the problem of big-data-based VNF deployment is the goal of this work.

Numerous studies are relevant to big-data-based NFV, including how to determine and place network functions. N Bouten et al. [2] presented a set of affinity and anti-affinity constraints that can be used by TSPs to define big-data-based placement constraints. They proposed a semantic conflict mechanism to evaluate SFC requests that filters invalid mechanisms to reduce the mapping time.

The performance of big-data-based NFVs with regard to resource allocation or consumption and the acceptance ratio when mapping big-data-based VNFs has been investigated for years. W Rankothge et al. [11] proposed a genetic algorithm to optimize resource allocation. They demonstrated its efficiency in optimizing resource allocation via three network function centers proposed by the authors.

Other research projects have focused on issues such as the availability of big-data-based NFV. Due to potential failures (such as node or link failures) that can be caused by earthquakes, floods, or malfunctions such as power outages, many researchers have expressed interest in the field of high availability (HA) to protect data or network functions. Unlike some schemes, which aim to solve general big-data-based VN mapping problems for unicast services (which includes two pro-

cedures: virtual node and link mapping) such as [4]. The authors of [13] proposed an efficient framework for evaluating the reliability of NFV deployments; however, they did not investigate how to adjust NFV deployments based on their framework. The proposed framework can be used only to evaluate deployment schemes but was not intended to improve the schemes based on its results.

Although numerous studies have considered the reliability of deployed SFCs, few studies have considered the needs of users while also considering the TSP revenues. In other words, few studies have focused on building an economical network environment. Therefore, we propose the ER_CS algorithm to reduce reliability under the premise of guaranteeing users' demands while also considering economical VNF deployments.

3 Problem Statement

As described in Fig. 1, a SFC request consists of several VNFs, a source node s and a destination node t. Each of these VNFs represents a network function, as described above. The thick blue dashed line represents another scheme whose reliability is 0.94 and resource consumption is 202, called service function forwarding path 1 (SFP1). The thick red dotted line represents one deployment scheme for the request whose reliability is 0.97 and resource consumption is 232, called SFP2. We assume that the demand reliability of users is 0.90. The thin blue dashed line, which represents a VNF in SFC, is deployed on a substrate network in SFP1. The red line will yield the best experience for the users, whereas the blue line will generate a better balance for the network providers because the network can hold more requests, which allows greater potential profits for TSPs. The goal of this paper is to find a deployment scheme that both satisfies users' reliability demands and minimizes resource consumption to reduce costs.

To achieve effective and reliable network services while deploying SFC requests, we need to deploy VNFs to more reliable nodes and attempt to maximize the total availability of the deployment of SFC. This goal can be notated as formula (1).

$$\begin{aligned} &\max\left\{ R^S = \prod_{v_p \in V_N^S} r_{v_p} \times \prod_{e_p \in E_L^S} r_{e_p} \right\} \\ &\forall v_p \in V_P,\ 0 < r_{v_p} < 1.0 \\ &\forall e_p \in E_P,\ 0 < \mathrm{r}_{e_p} < 1.0 \end{aligned} \tag{1}$$

Where r_{vp} and r_{ep} represent the reliability of the nodes and links deployed for SFC requests, respectively. The reliability of each node and link in the underlying network is denoted by a positive number less than 1 according to the constraint behind the optimization objective. This paper estimates the total reliability of SFC by calculating the product of the reliability of each substrate node and link involved in a SFC deployment scheme.

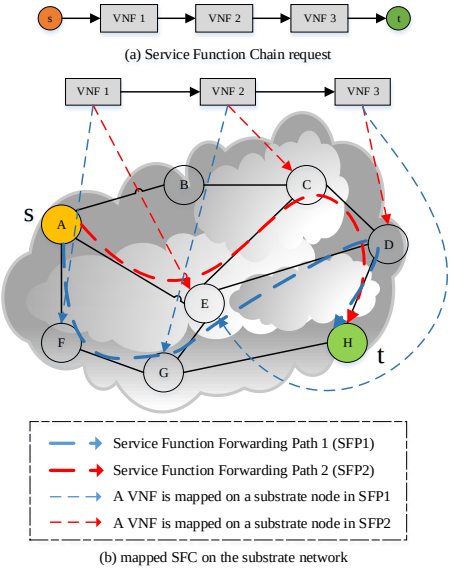

Fig. 1. Example of mapped VNFs

Due to limited resources, considering only the reliability of SFC may cause enormous resource consumption and reduce the mapping success rate. Therefore, the paper aims to solve the contradiction between the reliability and the bandwidth consumption, maintaining a balance between resource consumption and service reliability to ensure the effective use of resources.

4 Algorithm Design

4.1 Ensure-Reliability heuristic algorithm ER

In a NFV environment, many virtual networks share one substrate network; consequently, the failure of one substrate link or substrate node may cause massive failures in virtual networks, have a large-scale impact, and reduce network stability. Therefore, we propose a heuristic algorithm referred to as ER based on the reliability-aware SFC mapping problem. The pseudo-code is presented in Algorithm 1.

***Algorithm* 1:** Ensure big-data-based reliability (ER)
Input: 1. Substrate network $G_P = (V_P, E_P)$;
2. SFC request $SR = (N_S, L_S, s, t)$.
Output: SFC deployment scheme P^S.
1: Initialization: **let** $V_{remain} = V_P$;
2: **for** all VNF n_f in SR, **do**
3: **if** n_f is not the last VNF of SFC, **then**
4: initiateAllVertex() and **let** $r_{v_{so}}^{v_{so}} = r_{v_{so}}$;

```
5:          Call URSO procedure 1 to update the information;
6:          let κ_r = −∞ and v_temp = v_∞ ;
7:          for each vertex v in V_P, do
8:              if v ≠ v_si and w_v^r ≥ w_{n_f} and κ_r < r_v^{v_so} , then
9:                  κ_r = r_v^{v_so}, v_temp = v ;
10:             end if
11:         end for
12:         if κ_r = −∞ , then return null;
13:         generateScheme( κ_r , n_f), let v_so = v_temp;
14:   else
15:       repeat line 4 and 5, r_{v_si}^{v_si} = r_{v_si}
16:       call URSI procedure 2 to update the information;
17:       for each vertex v in V_P , do
18:          if w_v^r ≥ w_{n_f} and κ_r < r_v^{v_si} × r_v^{v_so} / r_v , then
19:              κ_r = r_v^{v_si} × r_v^{v_so} / r_v , v_temp = v ;
20:          end if
21:       end for
22:       repeat line 12 to line 13;
23: end for
```

When receiving a big-data-based SFC request, we firstly let $v_{so} = s$, $v_{si} = t$. When one VNF has been deployed, we will change the value of v_{so} just like line 13.

For all the VNFs other than the last one, the ER algorithm initializes the reliability of all vertexes to the source to be negative infinity and the reliability of the source vertex to be its vertex's reliability. Then, it initializes their prior vertex on the path to the source to be an inaccessible node. Next, it calls procedure 1—update all reliability to source (URSO)—to update the reliabilities of all nodes to the SFC source. In lines 6 to 11, we initialize the maximum reliability variable and the substrate node that has the maximum reliability to map the VNF, and traverse all the nodes to find the variable defined in line 6, which cannot be the sink vertex. We generate the mapping scheme and map the VNF onto the vertex v_{temp} with the reliability calculated in the previous procedure. If the reliability variable remains negative infinity, we are unable to find a mapping vertex that satisfies the demands for mapping VNF.

To map the last VNF in an SFC we must not only consider the mapping vertex's reliability to the previous VNF mapping vertex but also its accessibility and reliability at the destination node of the SFC. Similar to the previously described algorithm, we update the reliabilities of all nodes to the SFC's destination after updating the reliabilities to the SFC's source.

URSO will compute all the path's (from one underlying node in V_{remain} to the source node v_{so}) reliability, choosing and saving a path which has the max reliability. This procedure will traverse the node in V_{remain} and find a node (this node must satisfy computing resource requirement of the current VNF, and the edges in the path (from

source node to it) also need to satisfy bandwidth resource requirement of the virtual link (from prior VNF to current VNF)) that has max reliability.

Procedure 2 (i.e., update all reliability to sink (URSI)) is similar to URSO; the only difference is that rather than computing the reliability to the source, it computes the reliability to the destination.

4.2 Big-data-based Ensure-Reliability Cost Saving heuristic algorithm ER_CS based on load balancing

To maximize the reliability, SFC functions should be deployed on vertexes with high reliability, which may cause imbalanced loading in the network. Based on the algorithm ER, we introduce the idea of load balance and present the reliability-guarantee heuristic algorithm ER_CS, which is based on load balance.

In this thesis, the objective of load balance is to assign service flow transport to links with lighter loads to reduce the possibility of congestion caused by load imbalance. The following mathematical model describes load improvement:

$$\delta = \frac{1}{w_{v_i}^r} + \sum_{e_{v_i}^o \in e_i^o} \frac{1}{m_{e_{v_i}^o}^r} + m_{v_i}^{v_{so}},\ \forall v_i \in V_P \tag{2}$$

Where e_i^o denotes the set of the out-degree edge of vertex v_i, the denominator in the second fraction denotes the remaining bandwidth resource of the out-degree edge of vertex v_i, and the last symbol denotes the sum of the bandwidth cost of the path from vertex v_i to the source v_{so}. As expressed by the formula, the smaller the load factor is, the larger the vertex's remaining computing resource is, and the larger the remaining bandwidth resource of the out-degree is, the smaller the total bandwidth cost of the vertex to the source is.

Therefore, we adjust the ER algorithm to compute the δ of all the vertexes that satisfy the criteria based on satisfying R^U, the node's computing resource demands and the link's bandwidth resource demands. We add a comparison of the values of δ in URSO to find the vertexes with smaller δ values to host VNFs.

4.3 Bandwidth optimizing algorithm ER_CS_ADJ

We improve the ER_CS algorithm through bandwidth cost reduction, and we propose the bandwidth optimizing algorithm ER_CS_ADJ. We skillfully adjust the VNFs' mapping position based on the mapping scheme generated by ER_CS to lengthen the mapping paths of virtual links with low bandwidth demands and shorten it with high bandwidth demands; consequently, we reduce the bandwidth cost.

***Algorithm* 2:** Big-data-based ER_CS adjust (ER_CS_ADJ)
Input: SFC deployment scheme P^S.
Output: Adjusted SFC deployment scheme P^S.
1: **let** χ_{move} = findMinLink(SR);
2: **if** χ_{move} = 0, **then return;**
3: **while** χ_{move} > 0

```
4:    for all n_f need to be removed, do
5:       for all forwarding vertex v between two related
         function vertex, do
6:          if w_v^r ≥ w_{n_f} and B_remain^min ≥ B_request, then
7:             deploy n_f on vertex v;
8:          end if
9:       end for
10:   end for
11:   χ_move --;
12: end while
```

The function findMinLink(*SR*) finds the virtual link with the minimum bandwidth request in the SFC. The VNFs behind this link are the VNFs that must be moved; we denote the number of these as χ_{move}. When moving these VNFs, we need to traverse the VNFs in reverse order. When we adjust the mapping position of one VNF, we traverse all the forwarding vertexes on the path between this VNF and the updated VNF in reverse order. For example, when moving the last VNF, we traverse forward from the first forwarding vertex prior to the destination of the SFC. When moving the penultimate VNF, the deployment position of the last VNF is determined; thus, we traverse forward from the deployment position of the last VNF. The remaining steps can be performed in the same manner.

5 Simulation Results

5.1 Simulation Environment

To evaluate the schemes described in Section IV, we implemented an event simulation in Java. To demonstrate the applicability of the algorithm for all circumstances, we employ the Waxman 2 model from GT-ITM [15] to randomly generate small and large network instances as substrate networks. The small substrate network includes 20 nodes and the large substrate network contains 100 nodes.

During the simulation process, to compare and evaluate the performance of the three algorithms, we modified Compute followed by Network Load Balance (CNLB) [9] to the Link Mapping First (LMF) algorithm [10] without changing its core concept to be the compared algorithm in this paper.

5.2 Simulation Results and Analysis

All the left figures below was simulated in a small simulation topology, and the right one was simulated in a big simulation topology.

Fig. 2 shows the simulation results of the SFC block rate when deploying SFC requests for these four algorithms. We vary the number of functions of each SFC from 3 to 12 and randomly generate 10,000 SFC requests for each number of functions. The block rate denotes the proportion of the failed SFC deployment requests in all 10,000

SFC requests. The comparisons shown in left and right indicate that the three algorithms have a distinct advantage in block rate as the network size increases.

The results of the bandwidth overhead for SFC requests, shown in Fig. 3, reveal that the three algorithms proposed in this paper have an advantage over the LMF scheme in terms of bandwidth consumption, and that the ER_CS_ADJ algorithm performs the best.

The time consumption of each SFC mapping algorithm was evaluated by gradually increasing the number of service function chain requests, as shown in Fig. 4. The average time overhead of the SFC requests deployed by the three algorithms proposed in this paper is substantially lower than the average time overhead of the LMF algorithm.

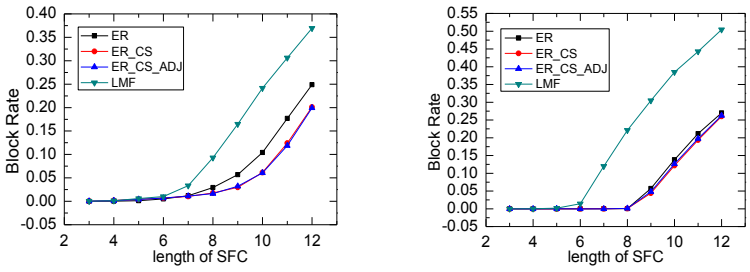

Fig. 2. Block rates of SFCs in different topology

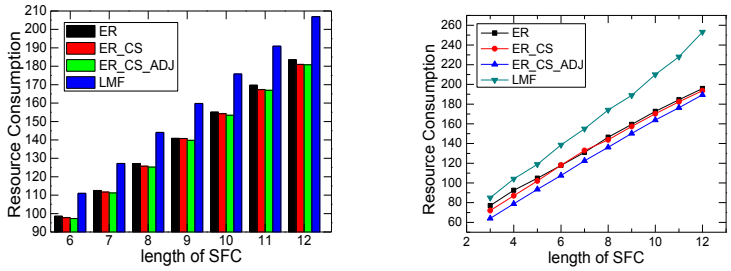

Fig. 3. Average resource consumption (i.e., computing resource and bandwidth resource) of SFCs in different topologies

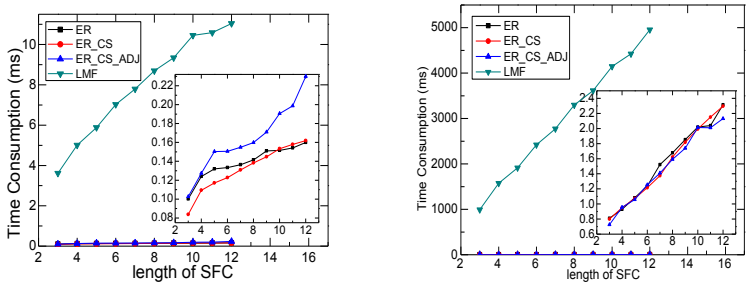

Fig. 4. Average time consumed when SFCs are deployed

6 Conclusions and future work

In this paper, we identified a problem: the high reliability requests of users reduce the CAPEX and OPEX of TSPs. Thus, we proposed ER to guarantee the basic reliability needs of users. However, considering the revenue of the TSPs, we discover that network imbalances will influence the request success rate and the resource utilization rate. Therefore, we proposed ER_CS, which is based on ER and considers the load balance factor. Although this algorithm achieved substantial progress, we discovered that the scheme used for ER_CS can be improved. Thus, we proposed ER_CS_ADJ. The simulation results indicate that ER_CS_ADJ achieves the objectives of this study. We demonstrated that our network algorithms can successfully work in a range of test environments and satisfy user demands.

Acknowledgement

This research was partially supported by the National Natural Science Foundation of China (61571098), Fundamental Research Funds for the Central Universities (ZYGX2016J217), Guangdong Science and Technology Foundation (2013A040600001, 2013B090200004, 2014B090901007, 2015A040404001, 2013B040300001).

References

1. M Mechtri, C Ghribi, and D Zeghlache. A Scalable Algorithm for the Placement of Service Function Chains. IEEE Transactions on Network and Service Management, 13(3), pp: 533-546, 2016.
2. N Bouten, R Mijumbi, J Serrat, et al. Semantically Enhanced Mapping Algorithm for Affinity-Constrained Service Function Chain Requests. IEEE Transactions on Network and Service Management, 14(2), pp: 317-331, 2017.
3. MT Beck, JF Botero. Scalable and coordinated allocation of service function chains. Computer Communications, 102, pp: 78-88, 2017.
4. N.M.M.K. Chowdhury, M.R. Rahman, and R. Boutaba, Virtual Network Embedding with Coordinated Node and Link Mapping. INFOCOM, pp: 783-791, 2009.
5. X Gao, Z Ye, et al. Virtual Network Mapping for Multicast Services With Max‐Min Fairness of Reliability. IEEE/OSA Journal of Optical Communications and Networking, 7(9), pp: 942-951, 2015.
6. R Cziva, D.P. Pezaros. Container Network Functions: Bringing NFV to the Network Edge. IEEE Communications Magazine, 55(6), pp: 24-31, 2017.
7. G Sun, D Liao, S Bu, et al. The Efficient Framework and Algorithm for Provisioning Evolving VDC in Federated Data Centers. Future Generation Computer Systems, 73, pp: 79-89, 2017.
8. G Sun, V Anand, D Liao, C Lu, et al. Power-efficient provisioning for online virtual network requests in cloud-based datacenters. IEEE Systems Journal, 9(2), pp: 427-441, 2015.
9. Z Ye, A.N. Patel, P.N. Ji, et al. Virtual Infrastructure Embedding over Software-Defined Flex-Grid Optical Networks. GLOBECOM, pp: 1204-1209, 2013.

10. Z Ye, X Cao, J Wang, et al. Joint Topology Design and Mapping of Service Function Chains for Efficient, Scalable, and Reliable Network Functions Virtualization. IEEE Network, 30(3), pp: 81-87, 2016.
11. W Rankothge, F Le, A Russo, et al. Optimizing Resource Allocation for Virtualized Network Functions in a Cloud Center Using Genetic Algorithms. IEEE Transactions on Network and Service Management, 14(2), pp: 343-356, 2017.
12. M.C. Luizelli, W.L.D.C. Cordeiro, L.S. Buriol, et al. A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining. Computer Communications, 102, pp: 67-77, 2017.
13. J Liu, Z Jiang, N Kato, et al. Reliability evaluation for NFV Deployment of future mobile broadband networks. IEEE Wireless Communications, 23(3), pp: 90-96, 2016.
14. D.E. Knuth. A generalization of Dijkstra's algorithm. Information Processing Letters, 6(1), pp: 1-5, 1977.
15. K.L. Calvert, E Zegura. Gt-itm: Georgia tech internetwork topology models (Software). Georgia Tech, [Online]. Available: http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz.