Research on Optimized Deployment of Virtual Network Functions in Network Function Virtualization Environment

Mingyue Liu¹, Feng Liu¹

¹Electronics and Communication Engineering, Beijing University of Aeronautics and Astronautics, Beijing, Beijing, 100191, China

Abstract. This paper mainly studies the optimization of dynamically arrived SFCs deployment in the SDN scenarios, and optimizes the end-to-end delay and bandwidth consumption during the deployment. First we build the model of the optimization problem, it is expressed as a 0-1 planning problem. We use ILP to get the optimal solution, cause the problem is NP hard, its runtime increases as network scales increases, so we choose the heuristic algorithm instead to reduce algorithm runtime. During our heuristic algorithm, we design a new method to sort VNFs and network nodes, then use the greedy algorithm to select nodes iteratively to place VNFs, in order to avoid local optimality, further use the simulated annealing algorithm with the results of the greedy algorithm as its initial solution, and design two methods to generate new deployments, and still repeat iterating to find a better deployment until reaching iteration limits. This paper also considers the SFCs’ lifecycle and trade-off between the two parameters. The simulation proves that the algorithm proposed in this paper can significantly reduce the end-to-end delay and bandwidth consumption than the traditional method, and 80% of its results are very close to the optimal solution with less than 5% error, and the ratio of its runtime and optimal solution’s is at most 0.003, the algorithm also has certain applicability and can be used in other scenarios.

1. Introduction

Due to the rapid development of the Internet, there are more and more types of network services [1], the demand of network users for network services is increasing rapidly. In order to ensure the availability and reliability of network equipments, the traditional communication industry uses dedicated equipments with tightly coupled functions and hardware equipment to build network services [2], but problems have occurred in actual use [3][4], using the hardware equipments to realize network functions means high cost and low flexibility. Besides, when a fault occurs, the speed of troubleshooting on the hardware device is very slow, and the data cannot be backed up in time. Moreover, when the network functions need to be upgraded, all relevant hardware equipments need to be upgraded, these factors limit the development of traditional networks seriously.

In recent years, driven by technologies such as cloud computing [5], big data [6], and software-defined networking (SDN) [7], the rise of network function virtualization (NFV) [8][9] has provided new solutions to the above problems, and it has become a research hotspot of the new generation of Internet [10][11]. NFV uses x86 and virtualization technology to softwareize network functions that rely on hardware equipments, and makes network functions stripped from specific network hardware [12], so it can help reduce the cost of expensive network equipments, and network resources can be fully flexibly shared [13]. NFV can also achieve rapidly new business on actual
needs, and greatly improve the network’s robustness, performance and user’s experience [14]. In NFV, network services are completed through a series of Service Function Chains (SFCs) [15]. SFCs are composed of Virtualized Network Functions (VNFs) in a certain order, and the deployment of SFCs is one of the most important NFV technologies [16], the deployment of the SFC is to place the VNFs that make up this SFC in a suitable position in the network, the quality of the deployment directly affects the performance of network services. The rapid development of SDN has given birth to the vigorous rise of NFV technologies. SDN decouples control functions from traditional distributed network devices and makes them directly programmable, and realizes centralized network management and control; NFV transforms network functions deployed in the form of proprietary hardware into VNFs running on common servers through virtualization, and provides the conditions for innovation in upper-level network functionality. The combination of NFV and SDN provides strong support for exploring new models of network security services.

In this paper, we mainly focus the deployment of dynamic SFCs in SDN scenarios, aim to find the optimal deployment, and we focus on two parameters during the deployment, namely the end-to-end delay and bandwidth consumption, which are considered from the perspective of network users and network service providers respectively. This paper builds the model of the deployment problem and determines its target function and constraint conditions, it is expressed as a 0-1 programming problem, we can use ILP to solve it. But the deployment problem is NP-hard, the runtime of ILP increases exponentially with the network scale increment, therefore, this paper chooses heuristic algorithms including greedy algorithm and simulated annealing algorithm to reduce the runtime. During the greedy algorithm, we design a new method to sort the VNFs and the network nodes before deployment, and consider the dynamic lifecycle of SFCs. In order to solve the local optimal problem of greedy algorithm, we further use simulated annealing algorithm for better optimization. Besides, this paper considers the trade-off of the end-to-end delay and bandwidth consumption during deployment and finds the solution to solve it.

2. System model and research problem expression

2.1. System model
This paper chooses to study the deployment of SFCs in the SDN scenarios, we suppose the underlying network consists of a set of connected OpenFlow switches that are managed and orchestrated by SDN centralized controllers. The inputs of the system include link delay, available ports of the switch, SFC policies and so on, the policies determine VNFs that the SFC consists of, the corresponding data traffic, including the source switch, the destination switch, and the characteristics of the data traffic. The outputs of the system are the VNFs deployment, and the optimized end-to-end delay and optimized bandwidth consumption.

Set the base network $W = (N, L), N$ represents a collection of nodes in the network, namely switches collection, $L$ represents a collection of links in the network. For each node $n_i \in N$, $R_i$ represents the available resources amount of $n_i$. $d_{i,j}$ represents the delay of the path between the node $n_i$ and the node $n_j$. $V$ represents the VNF collection, for any VNF $v_i \in V$, $R_i$ represents the amount of resources that $v_i$ requires. The SFC strategy set $C$ determines the VNFs’ order of the SFCs, for any SFC $c_k \in C$, $e_k$ represents the ingress node of the SFC, namely the source node, $o_k$ represents the egress node, namely the destination node, $c_k'$ represents the $ith$ VNF of the SFC, and $num_k$ represents the total number of VNFs of the SFC, $b_{i,k}$ represents the bandwidth requirement of the SFC. Define binary variable $\alpha_{i,j}$ to indicate whether the VNF $v_i$ is deployed to the node $n_j$. Define $f_{i,j}$ to represent the number of SFCs whose source node is $n_i$ and first VNF is $v_i$. $f_{i,j}$
represents the number of SFCs whose destination node is \( n_i \) and last VNF is \( v_j \). \( f_{i,j} \) represents the number of SFCs whose adjacent VNF pair is \((v_i, v_j)\). We define a variable \( h_{i,j}^p \) to represent the number of hops of the path from the node \( n_i \) to the node \( n_j \), define a variable \( b_{i,j} \) that represents the bandwidth requirements of the node \( n_i \) to \( v_j \), \( b_{i,j} \) represents the bandwidth requirements from \( v_i \) to the node \( n_j \), \( b_{i,j} \) represents the bandwidth requirements between the VNF pair \((v_i, v_j)\).

2.2. Objective function and Constraints
Considering the actual network situation, we establish several constraints. First, each VNF of the VNF set must be deployed to a suitable node in the network. Second, when the VNF is placed on the node, the resources of the node must meet the requirements of the VNF, including whether the switch has remaining available ports and whether other resources are sufficient. Last, some VNFs with special functions can only be placed in specific locations.

In order to conveniently represent the total end-to-end delay, we divide the end-to-end delay of a SFC into three parts, the delay from the source node to the first VNF, the delay from the first VNF to the last VNF, and the delay from the last VNF to the destination node. so we can get the new expression of the total end-to-end delay:

\[
\text{Delay}_{\text{total}} = \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V}} \sum_{n_k \in \mathcal{N}} d_{i,k} \cdot f_{i,j} \cdot \alpha_{i,j} + \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V}} \sum_{n_k \in \mathcal{N}} d_{i,j} \cdot f_{i,j} \cdot \alpha_{i,j} \\
+ \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V}} \sum_{n_k \in \mathcal{N}} d_{i,j} \cdot f_{i,j} \cdot \alpha_{i,j} \cdot \alpha_{i,j}
\]

(1)

Similar to the end-to-end delay, the total bandwidth consumption of the SFCs can be divided into the same three parts, we can get the total bandwidth consumption of the SFCs:

\[
\text{Bandwidth}_{\text{total}} = \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V}} \sum_{n_k \in \mathcal{N}} h_{i,j}^p \cdot b_{i,j} \cdot \alpha_{i,j} + \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V}} \sum_{n_k \in \mathcal{N}} h_{i,j}^p \cdot b_{i,j} \cdot \alpha_{i,j} \\
+ \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V}} \sum_{n_k \in \mathcal{N}} h_{i,j}^p \cdot b_{i,j} \cdot \alpha_{i,j} \cdot \alpha_{i,j}
\]

(2)

Now, we can formally define the NFV placement problem, and we can minimize the total end-to-end delay and total bandwidth consumption as expression (1) and (2).

We find that the target function (3) has a secondary term, since all decision variables are binary values, it is a 0-1 nonlinear planning problem. To solve these problems, we can use standard linear solvers to make it linearized, then use Integer Linear Planning(ILP) to solve it and get the optimal solution, this paper use Gurobi to solve it, and we call it OPTIMAL algorithm.

3. Our approach
Cause the SFC deployment is NP hard, for large-scale network, OPTIMAL’s runtime is large, so heuristic algorithms can be used instead to solve the problem. Here we mainly introduce the algorithm to optimize end-to-end delay, cause the way to optimize bandwidth consumption is similar.
3.1. Dynamic SFCs

In the actual network environment, SFCs have a certain life cycle, so we design a scheme to deal with dynamic SFCs. As shown in Figure 1, when the system is initialized, we first use the preset method to allocate resources to SFCs that have arrived, so that these SFCs begin to provide services to users, and use algorithm designed in this paper to deploy the SFCs. After the system starts running, it enters the trigger SFC resource deployment phase, called the stabilization phase. At this stage, if a new SFC arrives, the controller will use the deployment algorithm to select allocation from the remained resources for it. If a service chain ends its lifecycle, the controller recycles the resources occupied by the SFC previously and places them in a resource pool in case subsequent SFCs arriving to allocate resources. If there are no resource to use, the new arrived SFC may have to wait for other SFCs to release resources, then repeating the process until all SFCs are finished.

3.2. Greedy algorithms

This paper designs a greedy algorithm called GA to optimize the end-to-end delay, which consists of two steps. The first step, sorting the VNFs and network nodes in descending order, some VNFs are more important than others and need to handle more traffic, we count how much bandwidth the VNF assumes, and the more bandwidth it takes, the higher the priority it has. For sorting network nodes, calculating the bandwidth taked by each node when the SFC transmits according to the shortest path, called the minimum carrier bandwidth, and in our deployment, we use SCPF[17] to find routes for SFCs. The node with the largest minimum carrier bandwidth is the node with the highest priority, it is the key node in the network, and then calculating the shortest path when all SFCs pass through the node, and recording the difference between the bandwidth carried by each node and the minimum carrier bandwidth, the greater the difference, the greater the impact of the node on the SFC. Next, selecting the node with the largest difference in the last step, deploying the highest priority VNF to it, and recalculating the shortest path of all SFCs passing through the VNF, recording the difference between the node's carrying bandwidth and the minimum carrying bandwidth. Repeating the process
until all VNFs are deployed. The second step, GA iterates to place the VNFs according to the sorted order. In each iteration, and the VNF is placed in descending order of network nodes. If the node meets the resource constraint of the VNF, calculating the increment of the total end-to-end delay of the network, repeating the iterating until the end-to-end delay increment is no longer reduced.

GA actually has some problems, during the iteration process, some VNFs have not been placed, but they also cause end-to-end delay later, which directly ignored in GA. So we propose new algorithm GA-FIXED, we calculate weighted average end-to-end delay as an approximation of total end-to-end delay.

3.3. Simulated annealing
Because the greedy algorithm tends to be local optimized, so we use simulated annealing algorithm called SA to further optimize the result, cause SA can generate more deployments even with worse results, the simulated annealing algorithm can always get better results if the runtime allows. SA uses the solution obtained by the greedy algorithm as its initial solution, we also need to set the initial temperature of SA and update it during deployment. SA still iterates to place NFVs, in each iteration, the algorithm produces a random new deployment, we design two methods to generate new deployment, the first is to swap the nodes where two randomly selected VNFs are located, and the second is to randomly select a VNF to place on a randomly selected node, then calculates the end-to-end delay increments and updates the deployment until it reaches the limits of iterations.

4. Experiments and analysis
This paper chooses the MATLAB simulation platform to carry out simulation, and chooses the Abilene network and the FatTree network as the basic topology. The Abilene network has 11 nodes and 14 links, the FatTree network has 320 nodes and 2432 links, and the specific parameters used are showed in the table below, during the simulation, we choose them randomly.

In order to compare the performance of these algorithms, this paper chooses a base algorithm called SIMPLE as a comparison object, SIMPLE inherits the traditional deployment algorithm, it thinks that the SFC’s entry node is more suitable to place VNFs, it first sorts VNFs and then places VNFs iteratively to the node. In each iteration, which node has the maximum number of connections of VNFs, the node is selected to place the VNF, this algorithm does not consider the dependency between the underlying network topology and VNFs, so the optimization performance of the algorithm is not good. In the introduction below, we use DLY and BD to represent the end-to-end delay and bandwidth consumption optimization separately.

| Table 1. Simulation parameter table |
|------------------------------------|
| Bandwidth requirements.            | Average: 1-1.5, Variance: 0.2-0.5 |
| Link delay                         | Average: 1-2.5, Variance: 0.4-0.7 |
| The number of ports for the node.  | 1, 2, 3                           |
| The total number of VNFs.          | 5, 10, 15, 20                     |
| Number of VNFs on each service chain. | 4, 5, 6, 7                     |
| The total number of SFCs           | 10, 20, 30                        |

4.1. The running time of algorithms
We count the average runtime of each algorithm in 300 scenes, the results of these algorithms are compared with the ratio of the optimal solution. As shown in Figure 2, OPIMAL runs much longer than other algorithms, and the difference between the other algorithms is not large, only SA’s runtime is longer, because the simulated annealing needs to choose a higher primary temperature and slower cooling rate, which will lead to a longer optimization process, but its runtime and OPTIMAL’s are still nearly three orders of magnitude apart. The average runtime ratio between SA-DLY and OPTIMAL is about 0.00284, and ratio between SA-BD and OPTIMAL is about 0.00246.
4.2. Performance comparisons between different algorithms

We apply these algorithms to optimize the end-to-end delay and bandwidth consumption at the same scenarios and compare their performance. Figure 3 shows the bar charts while optimizing the end-to-end delay and bandwidth consumption for these different algorithms during deployment, OPTIMAL-DLY’s average end-to-end delay is $1.5433 \times 10^3$ ms, GA-DLY’s is $1.6580 \times 10^3$ ms, GA-DLY-FIXED’s is $1.6456 \times 10^3$ ms, and SA-DLY’s is $1.5701 \times 10^3$ ms. SIMPLE-DLY’s is $1.9822 \times 10^3$ ms, and from Figure 3, it can be seen that SA-DLY is the closest to the OPTIMAL solution, and SIMPLE-DLY has the biggest gap with the optimal solution, the average bandwidth consumption also has a similar result.

From the CDFs shown in Figure 4, the curves represent the ratio of different algorithm’s performance to the OPTIMAL’s, it can be seen that the performance of the SA is better than that of GA, because the principle of SA can prevent the results of the algorithm from being trapped in local optimality, and SA is very close to OPTIMAL.
4.3. The effect of parameters on algorithm performance

We change the number of ports per node and the number of VNFs on each SFC, respectively, to see how these two parameters affect the performance of algorithms. As shown in Figure 5, the average end-to-end delay and bandwidth consumption increase as the number of VNFs increases, cause SFCs need to transmit more hops and the routing is more complex.

Figure 6 shows the performance of the algorithms when the number of ports available on each node changes. The results show that as the number of available ports increases, their performance improves, because more VNFs can be connected to key nodes.

4.4. The trade-off between the two parameters.

Both the end-to-end delay and bandwidth consumption are important, so we set appropriate weights to combine the two target functions to one target function. To get the weights, we first compare the importance between the two parameters to build a comparison matrix, calculate the collection average of each row in the comparison matrix, and then the collection average is normalized to obtain a normalized matrix, and then we can get the final weight value. According to this method, the weights in the target function are $5/31$ and $26/31$, we first assume that end-to-end delay is more important, so its weight is $26/31$, then assume that bandwidth consumption is more important, and use ILP to solve the function, as shown in Figure 7, we can see that the result of the trade-off is between the results of independent optimization of the two parameters, which weight is bigger, and the trade-off is closer to which result.
Figure 7. The trade-off of the two parameters.

We also observe the change of one parameter when optimizing another parameter, and we can see from Figure 8 that the actual end-to-end delay is very close to the optimal end-to-end delay while optimizing bandwidth consumption, with an average error of 0.7851ms, and the CDF shows that 80% of the scenes have an error of less than 2% with the optimal solution. As shown in Figure 9, we can see the performance of the bandwidth consumption while optimizing the end-to-end delay. The average of the error between the actual bandwidth consumption and the optimal bandwidth consumption is 0.0337MB, and from the CDF, we can see that the error with the optimal solution in 80% of the cases is less than 1%, the error is very small, it is mainly because the factors affecting end-to-end delay and bandwidth consumption are very close.

Figure 8. The performance of the end-to-end delay while optimizing bandwidth consumption

Figure 9. The performance of the bandwidth consumption while optimizing the end-to-end delay

5. Conclusion
This paper mainly studies the deployment of dynamic SFC in SDN scenarios. The deployment problem of SFCs can be expressed as a 0-1 planning problem, we use ILP to find the optimal solution, but the computational complexity increases with the network scale, given that the deployment problem is an NP-hard problem, so we design a heuristic algorithm to find the approximate solution of the optimal solution. During our algorithm, we designed a new method to sort VNFs and network nodes,
because important VNFs should place to key nodes. And during the deployment we need to consider the SFCs’ arrival and leaving, besides, we combine the two parameters to one target function by choosing appropriate weights. From the simulation results, our algorithm can be optimized very close to the optimal solution and greatly reduce the runtime.

References

[1] Minh-Tuan Thai, Yang-Dar Lin, Yuan-Cheng Lai. A Joint Network and Server Load Balancing Algorithm for Chaining Virtualized Network Functions [C]. IEEE International Conference on Communications, 2016, 1-6.

[2] He Zhao, Yiqiang Hua, Xiaolin Guo. Advances in NFV technology and application scenarios [J]. Post and Telecommunications Design Technology, 2014 (6): 62-67.

[3] Han B , Gopalakrishnan V , Ji L , et al. Network function virtualization: Challenges and opportunities for innovations[J]. IEEE Communications Magazine, 2015, 53(2):90-97.

[4] Anderson T , Peterson L , Shenker S , et al. Overcoming the internet impasse through virtualization[J]. Computer, 2005, 38(4):36-41.

[5] Mijumbi R . Placement and Scheduling of Functions in Network Function Virtualization[J]. Computer Science, 2015.

[6] Bhagwan Singh, Sumeet Dhawan, Anish Arora, & Abhishek Patal. (2013). A view of cloud computing. International Journal of Computers & Technology, 4(2b1), 50-58.

[7] Manyika J . Big data: The next frontier for innovation, competition, and productivity[J]. http://www.mckinsey.com/Insights/MGI/Research/Technology_and_Innovation/Big_data_The_next_frontier_for_innovation, 2011.

[8] Kim H , Feamster N . Improving network management with software defined networking[J]. Communications Magazine IEEE, 2013, 51(2):114-119.

[9] Li Y , Chen M . Software-Defined Network Function Virtualization: A Survey[J]. IEEE Access, 2017, 3:2542-2553.

[10] Mijumbi R , Serrat J , Gorricho J L , et al. Network Function Virtualization: State-of-the-Art and Research Challenges[J]. IEEE Communications Surveys & Tutorials, 2017, 18(1):236-262.

[11] Mouradian C , Saha T , Sahoo J , et al. NFV Based Gateways for Virtualized Wireless Sensors Networks: A Case Study[C]// IEEE International Conference on Communication Workshop. IEEE, 2015.

[12] Jalalitabar M , Luo G , Kong C , et al. Service Function Graph Design and Mapping for NFV with Priority Dependence[C]// GLOBECOM 2016 - 2016 IEEE Global Communications Conference. IEEE, 2017.

[13] John W , Pentikousis K , Agapiou G , et al. Research Directions in Network Service Chaining[J]. 2013.

[14] Tung-Wei K , Bang-Heng L , Ching-Ju L K , et al. Deploying Chains of Virtual Network Functions: On the Relation Between Link and Server Usage[J]. IEEE/ACM Transactions on Networking, 2018:1-15.

[15] Firoozjaei M D , Jeong J P , Ko H , et al. Security challenges with network functions virtualization[J]. Future Generation Computer Systems, 2016, 67(feb.):315-324.

[16] Cohen R , Lewin-Eytan L , Naor J S , et al. Near optimal placement of virtual network functions[C]// IEEE INFOCOM 2015 - IEEE Conference on Computer Communications. IEEE, 2015.

[17] Sallam G . Shortest Path and Maximum Flow Problems Under Service Function Chaining Constraints[C]// IEEE INFOCOM 2018. IEEE, 2018.