Energy-Efficient Service Function Chain Provisioning in Multi-Domain Networks

Gang Sun[1, 2], Yayu Li[1], Guangyang Zhu[1], Dan Liao[1] and Victor Chang[3]
[1]*Key Lab of Optical Fiber Sensing and Communications (Ministry of Education),*
University of Electronic Science and Technology of China, Chengdu, China
[2]*Center for Cyber Security, University of Electronic Science and Technology of China, Chengdu, China*
[3]*Xi'an Jiaotong-Liverpool University, Suzhou, China*

Keywords: Service Function Chain, Energy Efficiency, Provisioning, Multi-Domain Networks.

Abstract: Service Function Chain (SFC) is not only helpful for saving the capital expenditure (CAPEX) and operational expenditure (OPEX) of network provider, but also can reduce energy consumption in the substrate network. However, to best of our knowledge, few researches focus on the problem of energy consumption for provisioning SFC requests in multi-domain networks. In this paper, we firstly formulate the problem of energy-efficient online SFC request provisioning across multiple domains by using integer linear programming (ILP). Then we propose a heuristic algorithm called EE-SFCO-MD for efficiently solving this problem. We conduct simulation experiments for evaluating the performance of our algorithm. The simulation results show that EE-SFCO-MD performs better than existing approaches.

1 INTRODUCTION

In traditional networks, network functions (e.g., Firewall, Network Address Translation, etc.) are implemented by middle-boxes coupled to the hardware (Kuo T W et al., 2016). However, with the development of Internet, the cost for maintaining these hardware-implemented network functions becomes higher, and the hardware devices can't easily meet the customization demand of end-users. To mitigate this problem, network function virtualization (NFV) is proposed to implement network function on commodity servers, which is called virtualization network function (VNF) (Pham C et al., 2017).

Usually, a service function chain (SFC) (Eramo V et al., 2017) (Elias J et al., 2017) is composed by several VNFs with specific order. The mapping of an SFC request is to find several physical servers to host the VNFs and physical paths to connect the servers while satisfying various constraints. Therefore, a good mapping strategy can not only save CAPEX and OPEX, but also reduce energy consumption in substrate network. In single domain network, the mapping-decision maker can obtain the global information of physical network, which makes it possible for completing the mapping policy based on a global perspective. However, in multi-domain networks, the detail information of each domain should be confident for other domains or a third part to keep the privacy of each domain, which makes it more difficult to provision the SFC request.

There are two ways to address the problem of SFC provisioning across multiple domains, i.e., the centralized and distributed approaches. The centralized approach needs each domain to share its own information with other domains or a third part, just like the research (Dietrich D et al., 2017), which violates the privacy requirements among various domains. On the other hand, as shown in the paper (Abujoda A, Papadimitriou P. 2016), the distributed method keeps the privacy of each domain during the process of provisioning SFC requests, which also results in lower performance and longer response time. Moreover, the authors don't take energy efficiency into account.

To reduce energy consumption, the provisioning result of an SFC should turn on as few servers as possible (Melo M et al., 2015) (Sun G et al., 2015). For offline scenarios, the SFC requests are given in advance and the mapping strategy usually redesigns the topology of SFC requests to reduce the number of VNFs by consolidating the same VNFs in different requests, such as the research in Yang K et al., 2016. However, for online SFC requests, the

Sun, G., Li, Y., Zhu, G., Liao, D. and Chang, V.
Energy-Efficient Service Function Chain Provisioning in Multi-Domain Networks.
DOI: 10.5220/0006770301440152
In *Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security (IoTBDS 2018)*, pages 144-152
ISBN: 978-989-758-296-7

request arrives and leaves dynamically, which makes it impossible to know all the SFC requests in advance. Therefore, it is difficult to provision the online SFC requests just like offline requests.

In this paper, we analyse the power consumption for provisioning SFC requests in substrate network and formulate the problem of energy-efficient provisioning of online SFC requests across multiple domains by using integer linear programming (ILP). The model's objective is to minimize the power consumption and we also give the constraints which must be met during mapping an SFC such as resource constraints and VNF order constraints. Since the problem of provisioning SFC requests across multiple domains is *NP-hard* (Wang Y et al., 2017), we also propose a heuristic algorithm named EE-SFCO-MD to efficiently solve the problem. The EE-SFCO-MD algorithm firstly extends node aggregation (Hong S et al., 2014) to build a domain-level function graph of the physical network. Then the algorithm generates all domain-level reachable paths between the source and destination of SFC request. For each domain-level reachable path, EE-SFCO-MD will build a local candidate graph based on the path and select a candidate provisioning solution with minimum energy consumption as the online SFC request partitioning result on the domain-level reachable path. And then the bidding mechanism is used for select the minimal energy consumption segment solution among various domain-level reachable paths as the final SFC partitioning scheme. Finally, EE-SFCO-MD maps each sub-SFC of the final partitioning result into corresponding domain.

The remainder of this paper is organizes as follows. Section 2 describes the problem of energy-efficiently provisioning for online SFC requests across multiple domains and formulates the problem as an ILP optimization problem. In Section 3, we propose a heuristic algorithm for efficiently mapping online SFC request. Section 4 gives the simulation result and analysis. And we conclude this paper in Section 5.

2 PROBLEM STATEMENT AND FORMULATION

We research the problem of how to reduce energy consumption for provisioning an online SFC request in multiple domains network, which is different from the single domain network is that the VNFs of SFC request may be deployed to several different domains and the virtual link need to be embedded on an inter-domain physical path if the two endpoints of SFC are hosted in two different domains.

Unlike offline SFC request, the online SFC requests arrive dynamically and uncertainly and thus cannot be accurately predicted, and the number of SFC requests need to be processed cannot be known in advance. Therefore, the energy-efficient online SFC request provisioning problem is not suitable for the purpose of saving energy by merging VNFs on the SFC requests as an offline problem.

To save energy, it's best for the mapping scheme to share the demand-meet active physical server as much as possible for activating fewer servers. However, in multi-domain environment, the biggest challenge of provisioning an online SFC request is the provisioning decision makers do not have all the information in each domain of substrate network. In addition, the correct order of VNFs in the SFC request should be kept in the provisioning result. Moreover, the energy cost (e.g. server computation power cost and network transmission power cost) and the response time of requests are expected to be reduced while all of the resource and function constraints are satisfied.

2.1 Primary Definitions

2.1.1 SFC Request

We model an SFC request as a directed weighted graph $G_R = (N_R, L_R, Src, Dst)$, where N_R represents the set of VNFs and L_R denotes the set of VNFs-connected virtual links on an online SFC request. Src and Dst denote the source and destination of the SFC request, respectively. For each VNF node $n_r \epsilon N_R$, $dem(n_r)$ indicates its computing resource demand, and $fun(n_r)$ denotes the function demand of n_r . For each virtual link $l_r \epsilon L_R$, $dem(l_r)$ represents the amount of link bandwidth resource demanded. We consider a scenario in which each virtual link on an SFC request has the same bandwidth resource demand but the computing resource demand and function demand of each VNF node are different.

2.1.2 Physical Network

The physical network can be modelled as an undirected weighted graph $G_P = (N_s, E_S)$, where N_s indicates the set of physical noes and E_S represents the set of substrate edges in physical network. For each physical node $n_s \epsilon N_s$, the available computing resource on n_s can be denoted by $c(n_s)$, and $f(n_s)$

represents the function category deployed on node n_s. What needs to specify is that $f(n_s)$ is equal to 0 when the server does not host a VNF. For each physical edge $e_s \epsilon E_S$, $b(e_s)$ represents its available bandwidth resource. In multiple-domain networks, the physical network consists of m domains connected by several cross-domain edges. We use $G_P^i = (N_S^i,\ E_S^i)(1 \le i \le m)$ to denote the substrate network in the *i-th* domain. N_S^i represents the set of physical nodes in the *i-th* domain. And the set of physical edge in the *i-th* domain is indicated by E_S^i. Moreover, E_L^{inter} is used for representing the set of inter-domain links in G_P. Therefore, we can also use $G_P = G_P^1 \cup G_P^2 \cup \ldots \cup G_P^m \cup E_L^{inter}$ to denote a multi-domain networks.

2.2 Extended Node Aggregation

In multi-domain networks, node aggregation technology facilitates simplifying the substrate network can clearly show the connectivity of the domains in the underlying network, which are critical to orchestrate SFC requests across multiple domains. However, if we need to take the function constraints of SFC requests into account, only the connection between the various domains is not enough. Therefore, we extend node aggregation to abstract physical network into domain-level function graph which is shown in Figure 1(b) for guiding the SFC request provisioning process in multi-domain networks.

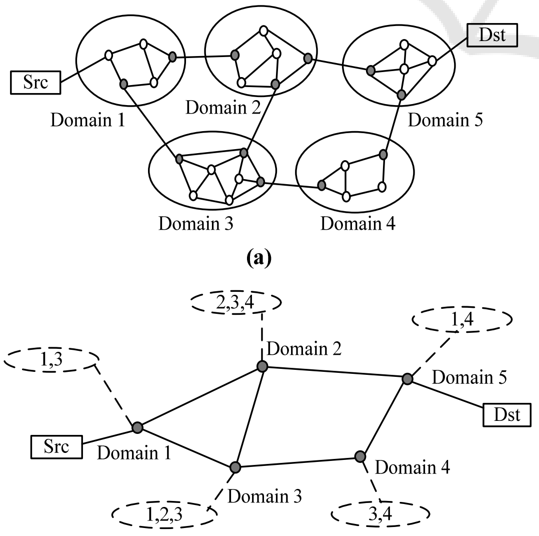

Figure 1: Extended Node Aggregation. (a)Substrate network. (b)Domain-Level function graph.

For keeping the privacy and confidentiality of each domain, only the shared public information is utilized in the extended node aggregation (ENA) approach to construct an abstracted network. In Figure 1(b), the nodes represent the domains in the substrate network and the solid lines indicate that the endpoints are connected by at least one inter-domain physical edge, which both are shared information. Moreover, the *Src* and *Dst* denote the source and destination nodes which are specified by SFC request. The numbers in each dotted circle which connects with each node represent the set of deployed function types in the corresponding domain currently. The numbers that are provided by the domain orchestrator in each domain indicate which function types have been deployed in current domain. And the numbers also represent that the domain can try to deploy the VNFs which have the same function demand to share servers for the sake of reducing the number of active servers. Although the function types in each domain are shared by the main orchestrator, the specific quantity and location of the functions in a domain are still confidential to other domains. Therefore, this ENA process is not contrary to the privacy of each domain.

In summary, all of the needed information in our algorithm includes: *i*) the domains and the connectivity between domains; *ii*) the source and destination node of SFC request; *iii*) the set of deployed function types in each domain. Obviously, our algorithm does not use the specific information in each domain.

2.3 Energy-Efficient SFC Provisioning Across Multiple Domains

2.3.1 Energy Consumption

The energy consumption for the online SFC request provisioning mainly consist of two parts: the *computing energy consumption* and the *forwarding energy consumption*. The *computing energy consumption* is generated by the servers which host the VNFs. And the *forwarding energy consumption* is generated by forwarding the traffic in the network.

If a server is active, it will consume some basic energy even if it does not host any VNF (*i.e.* the server has no workload), which is called basic power consumption. As the workload increases, the server will need to consume additional energy to process the workload. Thus the power consumption of a server also consists of two parts: the basic power consumption and the workload-dependent power consumption. And the energy cost of a physical node can be calculated according to the following formula.

$$P_n^{server} = \begin{cases} P_n^{basic} + P_n^{load} \times u, & \text{if server } n \text{ is on;} \\ 0, & \text{if server } n \text{ is off.} \end{cases} \quad (1)$$

Where P_n^{server} represents the power consumption generated by the server n to process the traffics of SFC requests; P_n^{basic} indicates the power consumption of a zero-workload server; P_n^{load} denotes the maximum additional energy cost of a full-workload server; and u is the server workload (*i.e.*, computing resource utilization). And P_n^{server} is equal to 0 when the server n is off.

In addition, when a VNF is hosted on a server, the server needs to communicate with other servers and forward traffic data to other server, which consumes some energy. The power consumption of a server in the forwarding process is related to the used ports on the server, which can be calculated by the Formula (2):

$$P_n^{link} = \begin{cases} P_n^{rack} + P_n^{port} \times c_n^{port}, & \text{if server } n \text{ is on;} \\ 0, & \text{if server } n \text{ is off.} \end{cases} \quad (2)$$

Where P_n^{link} represents the power consumption on the server n to forward data. The basic power cost on a server for data forwarding can be denoted by P_n^{rack}; and P_n^{port} indicates the power consumption for using a port on a server. c_n^{port} represents the number of used port on server n. What need to illustrate is that each server needs to use two ports for hosting a VNF, namely the ports for receiving and sending data, respectively.

In this paper, the total power consumption of all the servers to host the VNFs on a SFC request can be computed as in Formula (3).

$$P_{compute} = \sum_{v \in N_R} m_v^n \times P_n^{server} \quad (3)$$

Where $P_{compute}$ is the total computing power consumption of all the servers to deploy VNFs on a SFC request; m_v^n is a 0-1 variable that indicates whether the VNF v is deployed on the server n. $m_v^n = 1$ if v is deployed on n, and 0 otherwise.

On the other hand, the power consumption for forwarding traffic of a SFC request between VNF deployment positions can calculated according to the following formula.

$$P_{forward} = \sum_{ij \in L_R} \sum_{nk \in E_S} m_{ij}^{nk} [m_i^n \times P_n^{link} + m_j^k \times P_n^{link} + w_i^n \times (P_n^{server} + P_n^{link}) + w_i^k \times (P_k^{server} + P_k^{link})] \quad (4)$$

where $P_{forward}$ represents the forwarding power consumption in substrate network for provisioning a SFC request. $P_{forward}$ is mainly composed by two parts: one part is generated by the physical servers that have hosted VNFs due to the use of ports for exchanging data with other servers, such as the first and second parts of the above equation; and the other part is generated by the servers for forwarding purpose. Forwarding nodes not only need to use server port for forwarding data, but also need to consume basic power to active the server, such as the third and fourth parts in the above equation. ij represents a virtual link on SFC. nk is a physical edge in substrate network. m_{ij}^{nk} is a 0-1 variable that denotes whether the virtual link ij is embedded on physical edge nk. Thus $m_{ij}^{nk} = 1$ if ij is embedded on nk, and 0 otherwise. w_i^n also is a 0-1 variable which means whether physical node n is a forwarding node. $w_i^n = 1$ if n is a forwarding node, and 0 otherwise.

2.3.2 Objective Function

We formulate the online SFC request provisioning problem by using integer linear programming (ILP) to minimize the total power consumption. The objective function is presented in Formula (5).

$$Minimize \ \{P_{compute} + P_{forward}\} \quad (5)$$

Objective function tries to minimize the total power consumption for provisioning an online SFC request, *i.e.,* the sum of the computing power consumption and the forwarding power consumption.

2.3.3 Constraints

The provisioning process of an SFC request must satisfy many constraints, such as resource capacity and function constraints, especially in multi-domain network.

VNF Provisioning:

$$\sum_{v \in N_S} m_v^n = 1, \quad \forall n \in N_R \quad (6)$$

$$\sum_{v\in N_R} m_v^n \le 1, \quad \forall n \in N_S \tag{7}$$

$$f(n)\times fun(v) = f(n)\times f(n), \ \forall v \in N_R, \forall N \in N_S \tag{8}$$

$$dem(v) \le m_v^n \times c(n), \ \forall v \in N_R, \forall n \in N_S \tag{9}$$

Constraint (6) guarantees that a VNF node can only be deployed on a physical node. Constraint (7) ensures that the number of VNFs which are hosted onto a server is less than one in the provisioning process of the same SFC request. In this paper, we assume that the number of function types hosted on a server is no more than one. Thus Constraint (8) ensures that a server can only deploy the VNFs with same function. We have to note that $f(n) = 0$ if the server n has not deployed any VNF. Therefore, a VNF can be hosted on the server only if it does not deploy VNF or its function constraints are satisfied. Equation (9) is the node resource capacity constraint of each physical node, which ensures that the server must have sufficient available resource capacity for meeting the VNF's resource demand if the VNF tend to be hosted on the server.

Virtual Link Provisioning:

$$dem(ij) \le m_{ij}^{nk} \times b(nk), \ \forall ij \in L_R, \forall nk \in E_S \tag{10}$$

$$\sum_{ij\in L_R}\sum_{np\in E_S} m_{ij}^{np} + \sum_{ij\in L_R}\sum_{pn\in E_S} m_{ij}^{pn} \le 2, \ \forall n \in N_S \tag{11}$$

Constraint (10) is the bandwidth resource capacity constraint of each physical edge, which guarantees that a virtual link's bandwidth resource demand must not exceed the available bandwidth resource of physical edges which host the virtual link. Due to an SFC request is an ordered chain of VNFs, the provisioning solution of SFC should be loop-free to avoid the emergence of Ping-Pong traffic. Equation (11) restricts a physical node to be used at most once, which is helpful to obtain acyclic mapping solutions for SFC.

Order Constraints: since the VNFs are orderly connected in an SFC request, their deployment nodes should also be connected at the specific order. The *order constraints* between VNFs can be expressed as in the following formulas.

$$\sum_{ph\in E_L^{inter}} m_{ij}^{ph} - \sum_{hp\in E_L^{inter}} m_{ij}^{hp} = D_j^h - D_i^h, \forall ij \in L_R \tag{12}$$

$$\sum_{kn\in E_S^h} m_{ij}^{kn} - \sum_{nk\in E_S^h} m_{ij}^{nk} = D_j^h \times m_j^n - D_i^h \times m_i^n, \quad \forall n \in N_S^h, \forall ij \in L_R \tag{13}$$

$$m_{ij}^{nk} = m_i^n + w_i^n, \forall ij \in L_R, \forall nk \in E_S \tag{14}$$

Where D_j^h is a 0-1 variable which represents whether VNF j is deployed in domain h. And $D_j^h = 1$ if h hosts j and 0 otherwise. Therefore, Equation (12) ensures that at least one inter-domain edge can be found to connect the two domains when two VNFs are allocated to different domains. Constraint (13) guarantees that the VNF deployment results in each domain keeps the correct order. Moreover, if there are forwarding nodes on the underlying embedding path of a virtual link, the correct order of VNFs should also be guaranteed, which is constrained in Constraint (14). Furthermore, the endpoint of the provisioning solution must be the destination of the SFC request rather than a forwarding node, so we just need to constrain the first node of the embedded edges of each virtual link. If $m_{ij}^{nk} = 0$, server n can neither host any VNF nor be a forwarding node. And if $m_{ij}^{nk} = 1$, the server n is either a forwarding node or a VNF deployment node.

3 ALGORITHM DESIGN

Compared with single domain networks, the process of provisioning an SFC request can be divided into two key parts in multiple domains network, i.e., the partition of SFC request for each domain in the network and the mapping of sub-SFCs in each domain.

In this section, we propose a heuristic algorithm called EE-SFCO-MD for energy-efficient provisioning online SFC requests across multiple domains where the requests arrive and leave dynamically. Without loss of generality, we assume that the online SFC requests arrive and leave according to a Poisson process in this work. For each online SFC request, EE-SFCO-MD firstly extends node aggregation to build a domain-level function graph (DLFG) based on the substrate network to keep the privacy of each domain, as is shown in Figure 1. Then EE-SFCO-MD generates all domain-level reachable paths (DLRP) between source and destination in the domain-level function graph. As is shown in Figure 1(b), a domain-level reachable path may go through several domains. Since the VNFs of an SFC request has the order constraints, which means the deployment solution must keep the correct order of VNFs on the SFC. Moreover, for each domain-level reachable path, EE-SFCO-MD sends the SFC request to each domain on the path and collects all the candidate information from each domain to construct a local candidate graph (LCG). Based on the local candidate graph, EE-SFCO-MD partitions the SFC request into several sub-SFCs with minimum energy consumption for the domains in domain-level reachable path. And then the bidding mechanism is used for obtaining the final SFC request partitioning result among various domain-level reachable paths. Finally, EE-SFCO-MD maps the sub-SFCs of the final partitioning result into corresponding domains. The pseudo code of the EE-SFCO-MD algorithm is shown in ***Algorithm 1***.

Algorithm 1: Energy Efficient SFC request Orchestrating across Multiple Domains (EE-SFCO-MD).

```
Input:
(1) Substrate network $G_P = (N_S, E_S)$;
(2) Online SFC request queue, $ArrivedSFC$;
Output:
The set of accepted SFC requests $SFC_{acc}$ and the mapping solutions $M_{SFC}$.
1:  Initialization: $SFC_{acc} = \emptyset$, $M_{SFC} = \emptyset$;
2:  while $ArrivedSFC \neq \emptyset$, do
3:      Query SFC requests in DeployedSFC and release the resources occupied
        by the expired SFC requests, and then remove the expired SFC
        requests from DeployedSFC;
4:      Take out the first SFC request $sfc_1$ in ArrivedSFC and construct
        the domain-level function graph according to $sfc_1$, and then
        generate all domain-level reachable paths $PATH$ between source and
        destination of $sfc_1$, let the optimal mapping $M_{opt} = \emptyset$;
5:    for each $path \in PATH$, do
6:      Construct local candidate graph lcg based on $path$, and generate the
        partitioning result of $sfc_1$ according to lcg;
7:      Based on the partitioning result, premap the sub-SFC in each domain
        on $path$;
8:      if find an provisioning solution $M$ successfully, do
9:        if the energy consumption of $M < M_{opt}$, do
10:           $M_{opt} \leftarrow M$;
11:       endif
12:     endif
13: endfor
14: if $M_{opt} \neq \emptyset$, do
15:    Map $sfc_1$ into substrate network according to $M_{opt}$, and update
       the substrate network;
16:    $SFC_{acc} = SFC_{acc} \cup \{sfc_1\}$,
       $M_{SFC} = M_{SFC} \cup M_{opt}$,
       $DeployedSFC = Deployed \cup \{sfc_1\}$;
17:  endif
18:  $ArrivedSFC = ArrivedSFC \setminus \{sfc_1\}$;
19: endwhile
20: return $SFC_{acc}$, $M_{SFC}$.
```

In ***Algorithm 1***, all arrived online SFC requests are firstly buffered in a queue named *ArrivedSFC*. The notation *DeployedSFC* is used for representing the set of deployed online SFC requests. When ArrivedSFC $\neq \emptyset$, each online SFC request in the *ArrivedSFC* queue is deployed one by one (line 2). Due to the limitation of physical resource, the online SFC requests may be blocked. Therefore, we define SFC_{acc} to denote the set of accepted SFC rquests. Before deploying a new online SFC request, our algorithm firstly query the expired request in *DeployedSFC* and remove them from *DeployedSFC*, and release the physical resource occupied by these expired SFC requests (line 3). And then, EE-SFCO-MD constructs the domain-level function graph according to the first SFC request in the *ArrivedSFC* and generates all domain-level reachalbe paths (line 4). Line 5-13 in ***Algorithm 1*** are responsible for obtaining the final SFC partitioning solution with minimum energy consumption. Due to the space limitation, we omit the algorithms for constructing local candidate graph and candidate selection (line 6). Moreover, EE-SFCO-MD deploys each sub-SFC in the final segment result into substrate network (line 14-15). Finally, EE-SFCO-MD updates the substrate network and the sets SFC_{acc}, M_{SFC}, *DeployedSFC* and *ArrivedSFC* (line 16-18).

4 SIMULATION RESULT

In this section, we numerically compare the performance of EE-SFCO-MD algorithm with the algorithm proposed in (Abujoda A, Papadimitriou P. 2016) and (Wang Y et al., 2017). We first describe the simulation environment, and then present our simulation results and analysis.

4.1 Simulation Environment and Settings

We generate the multiple domains physical network by using IGEN tool. The substrate network is composed by 6 domains and each domain has 20 nodes. In each domain, the physical nodes are connected by the Delaunay model in IGEN. And the domains are connected by inter links with a probability of 0.5. The resource demands of VNFs and virtual links both follow a uniform distribution U (10, 40). The computing resource capacity of physical nodes and bandwidth resource capacity of physical intra-domain edges follow a uniform distribution U (200, 300), and the bandwidth resource capacity of physical inter-domain edges follows a uniform distribution U (4000, 6000). Moreover, a server's zero-workload power and full-workload power are set to 171W and 301W, respectively. The basic power cost for forwarding on a server is set to 5W, and the power consumption for using a port on a server is set to 1.2 W.

4.2 Simulation Result and Analysis

The average response time of EE-SFCO-MD is lower than that of Nestor and DistNSE algorithms, as shown in Figure 2. This is because our algorithm extends the node aggregation for constructing a domain-level function graph of the substrate network, which is helpful for guiding the provisioning process. Moreover, EE-SFCO-MD just needs to search on the intra-domain function graph whose scale is much smaller than the substrate network. Therefore, EE-SFCO-MD can quickly response to the online SFC requests. It needs to be mentioned that DistNSE algorithms traverses all the physical paths in each domain, which results in a significant increase in response time.

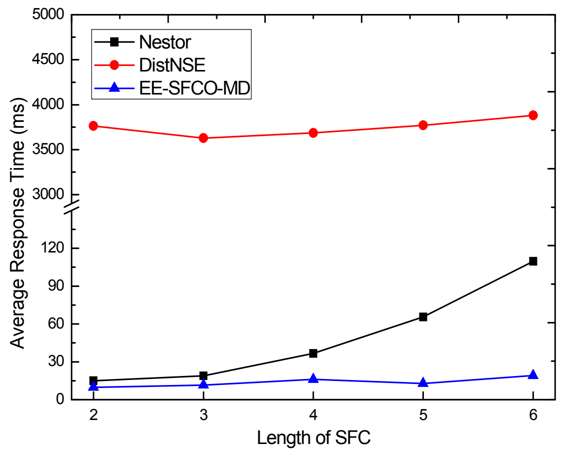

Figure 2: The average response time as a function of the length of SFC.

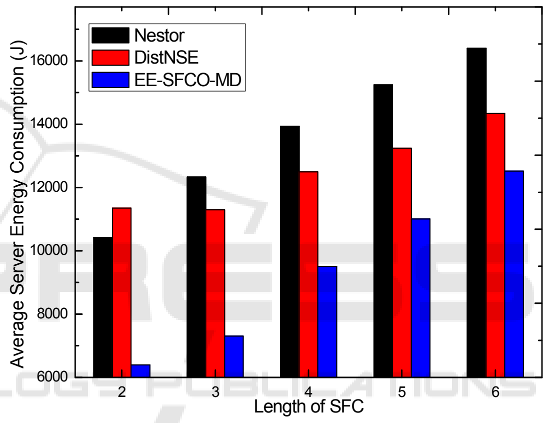

Figure 3: The average server energy consumption as a function of the length of SFC.

As shown in Figure 3, the average server energy consumption of EE-SFCO-MD algorithm is much lower than that of Nestor and DistNSE. This is because that EE-SFCO-MD takes reusing servers into account, which makes it possible for EE-SFCO-MD to find the minimal energy consumption servers for hosting VNFs while provisioning online SFC requests. Whereas Nestor and DistNSE algorithms don't consider to saving energy during mapping SFC requests, which leads to a high energy cost.

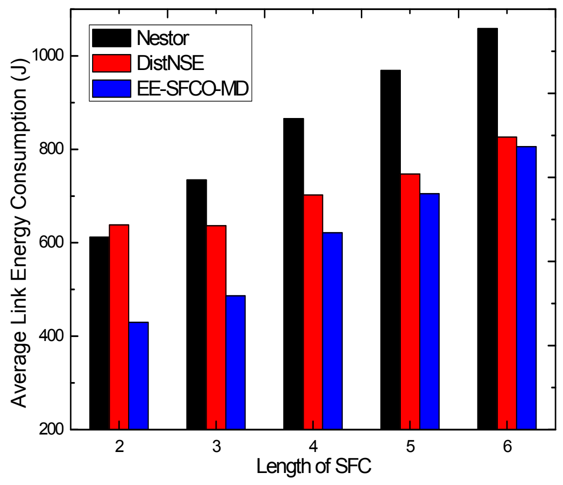

Figure 4: The average link energy consumption as a function of the length of SFC.

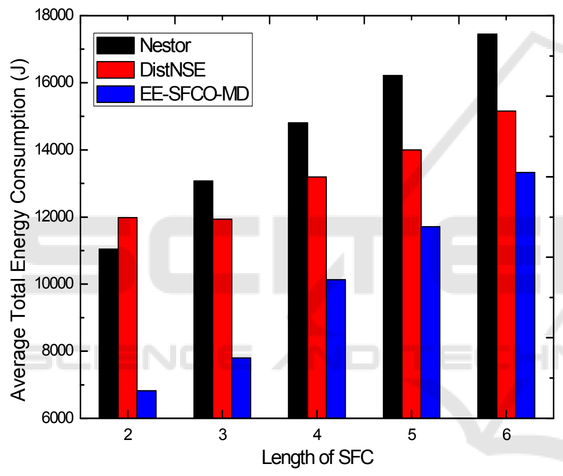

Figure 5: The average total energy consumption as a function of the length of SFC.

Similarly, by searching on the intra-domain function graph, EE-SFCO-MD can find the shorter physical paths for embedding the virtual links on the SFC request, which can reduce the energy consumption for forwarding data between servers, which is shown in Figure 4.

Figure 5 describes the average total energy consumption among three compared algorithms. From Figure 5, we can see that the EE-SFCO-MD algorithm has lower total energy consumption. This is because that EE-SFCO-MD algorithm considers saving energy during provisioning online SFC request.

5 CONCLUSIONS

In this paper we study the problem of energy-efficient provisioning for online SFC request in multi-domain networks. We firstly formulate the problem as an optimization problem by using ILP and propose a heuristic algorithm named EE-SFCO-MD for solving this problem. The simulation results show that our algorithm is promising for reducing energy consumption and perform better than existing approaches.

ACKNOWLEDGEMENT

This research was partially supported by the National Natural Science Foundation of China (61571098), Fundamental Research Funds for the Central Universities (ZYGX2016J217).

REFERENCES

Kuo T W et al., 2016. Deploying chains of virtual network functions: On the relation between link and server usage. In *IEEE INFOCOM 2016 - the IEEE International Conference on Computer Communications*. 1-9.

Pham C et al., 2017. Traffic-aware and Energy-efficient vNF Placement for Service Chaining: Joint Sampling and Matching Approach. In *IEEE Transactions on Services Computing*.

Eramo V et al., 2017. An Approach for Service Function Chain Routing and Virtual Function Network Instance Migration in Network Function Virtualization Architectures. In *IEEE/ACM Transactions on Networking*, 99:2008-2025.

Elias J et al., 2017. Efficient Orchestration Mechanisms for Congestion Mitigation in NFV: Models and Algorithms. In *IEEE Transactions on Services Computing*, 10(4):534-546.

Dietrich D et al., 2017. Multi-Provider Service Chain Embedding With Nestor. In *IEEE Transactions on Network & Service Management*, 14(1):91-105.

Abujoda A, Papadimitriou P. 2016. DistNSE: Distributed network service embedding across multiple providers. In *International Conference on Communication Systems and Networks*. IEEE, 1-8.

Wang Y et al., 2017. Cost-Efficient Virtual Network Function Graph (vNFG) Provisioning in Multi-Domain Elastic Optical Networks. In *Journal of Lightwave Technology*, 99:1-1.

Melo M et al., 2015. Optimal virtual network embedding: Energy aware formulation. In *Computer Networks*, 91(C):184-195.

Sun G et al., 2015. Power-Efficient Provisioning for Online Virtual Network Requests in Cloud-Based Data Centers. In *IEEE Systems Journal*, 9(2):427-441.

Yang K et al., 2016. Energy-Aware Service Function Placement for Service Function Chaining in Data Centers. In *Global Communications Conference (GLOBECOM)*. IEEE, 1-6.

Hong S et al., 2014. Virtual optical network embedding in multi-domain optical networks. In *Global Communications Conference (GLOBECOM)*. IEEE: 2042-2047.