A Postprint of an Optimized Deployment Mechanism for Service Function Chains Based on Software-Defined Networking

Authors: Liu Yicen, Lu Yu, Wang Shan, Chen Xingkai, Qiao Wenxin

Date: 2018-06-19T00:00:00+00:00

Abstract

To address the problem that existing service chain deployment methods in software-defined networking environments fail to fully consider network-wide resource utilization, this paper proposes an optimized service chain deployment mechanism based on an efficient heuristic algorithm. First, the overall architecture of service chain deployment is presented, and an integer linear programming model is introduced for mathematical formulation. Second, an efficient heuristic algorithm for solving the model is proposed, which employs a sort-then-greedy approach to effectively utilize network resources and balance load while satisfying resource and delay constraints. Simulation results demonstrate that, compared with other deployment algorithms, the proposed algorithm reduces both load balancing degree and time complexity while improving request acceptance rate.

Full Text

Preamble

A Service Function Chain Optimization Deployment Mechanism Based on Software-Defined Networking

Liu Yicen[1], Lu Yu[1], Wang Shan[2], Chen Xingkai[1], Qiao Wenxin[1]

(1. Equipment Training Simulation Center, Army Engineering University, Shijiazhuang 050003, China;
2. Unit 66172 of the PLA, Shijiazhuang 050000, China)

Abstract: In software-defined networking (SDN) environments, existing service function chain (SFC) deployment methods fail to fully consider network-wide resource utilization. This paper proposes an SFC optimization deployment

mechanism based on an efficient heuristic algorithm. First, we present the overall architecture for service chain deployment and introduce an integer linear programming (ILP) model for mathematical formulation. Second, we propose an efficient heuristic algorithm that operates in a "sort-first, greedy-second" manner, effectively utilizing network resources and balancing load while satisfying resource and delay constraints. Simulation results demonstrate that compared with other deployment algorithms, the proposed algorithm reduces load balancing degree and time complexity while improving request acceptance rate.

Keywords: software-defined networking (SDN); service function chaining (SFC); optimal deployment; integer linear programming (ILP); heuristic algorithm

0 Introduction

With the rapid growth of network information volume (global Internet traffic is projected to reach 1.6×10^{21} bytes by 2018 [1]), user demands for Internet services continue to expand. Traditional static service function deployment models have introduced numerous problems. First, in conventional networks, service functions (such as firewalls, intrusion detection systems, and network address translation) are tightly coupled with dedicated hardware devices, resulting in poor scalability and high network maintenance costs. Second, static and rigid service functions struggle to adapt to the dynamic demands of current Internet multi-tenant models, causing network operational instability. Consequently, research on new dynamic service models has become a recent hotspot [2-3].

Currently, Software-Defined Networking (SDN) [4] and Network Function Virtualization (NFV) [5] technologies provide technical support for dynamic service function chain development. The SDN architecture employs programmable, logically centralized control to orchestrate service functions according to fine-grained requirements. NFV technology transforms traditional embedded deployment of network functions by decoupling network control functions from hardware devices, creating opportunities for third-party companies to develop service functions. The combination of these two technologies enhances network flexibility and scalability, achieves infrastructure resource sharing, and supports dynamic service function chain technology.

Research on dynamic service function chains based on SDN is still in its infancy, with many outstanding issues in practical applications, among which SFC optimization deployment is one of the urgent problems to be solved [6-8]. Existing deployment studies are largely conceptual validations. In deployment strategy research, current methods tend to consider only a single aspect of network-wide resource utilization. For example, reference [9] proposes a service function deployment method based on tabu search, which uses a tabu list to search for optimal deployment positions for service functions in the global network, but this method only considers node resource utilization. Reference [10] proposes a

service function management mechanism for NFV environments that primarily uses the Viterbi algorithm to select service functions with minimum deployment cost among candidate nodes, effectively reducing deployment costs but lacking consideration for link deployment overhead. Reference [11] proposes a service function deployment model using an AFM heuristic algorithm for solution, but AFM only uses physical node computing resource capacity as its selection strategy without considering global service path resource utilization. Reference [12] uses a Greedy algorithm to enumerate all service paths meeting resource constraints and connectivity requirements across the network, selecting the path with minimum link resource occupation, but this approach focuses only on link optimization selection. Reference [13] proposes a service function deployment mechanism under SDN/NFV architecture that constructs a hierarchical graph model and minimizes delay as the service path selection criterion, reducing service function chain processing time but lacking consideration for node resources.

These deployment methods explore different perspectives, but their main research limitation is that they focus either solely on node resource utilization or solely on link resource utilization, lacking comprehensive study of global resource allocation schemes. Therefore, this paper takes a global perspective, simultaneously considering both node and link resource utilization, fully leveraging the flexible allocation characteristics of virtual resources to select appropriate function placement positions and service paths. This effectively utilizes resources and balances load while satisfying resource and delay constraints. We first present the overall architecture for SDN-oriented service function chain deployment and model it as an integer linear programming mathematical model. Second, addressing the difficulty that current methods fail to simultaneously consider node and link resource utilization, we design a heuristic search algorithm to solve the deployment model. Finally, we evaluate the proposed algorithm using metrics such as load balancing degree, request acceptance rate, and time complexity to verify its effectiveness.

---

1 Network Model and Problem Description

1.1 SDN-Based Service Function Chain Deployment Model

As shown in [Figure 1: see original paper], the SDN-based service chain deployment model consists of three planes: the orchestration plane, control plane, and data plane. The orchestration plane serves as the development environment for Virtual Network Functions (VNFs), primarily responsible for centralized management of VNFs in the network and constructing SFC policies according to different business requirements and application scenarios. It combines corresponding functional components to form logical function chains for network control. The control plane maps VNFs and virtual links to the physical infrastructure based on service function resource requirements and underlying resource information, following certain deployment strategies to achieve logical

service chain deployment. The data plane mainly comprises general-purpose hardware devices (such as standardized forwarding devices and x86 hardware resources) that receive rules from the control plane and carry actual service requests.

1.2 Service Function Chain Deployment Problem Description

The service function chain deployment process can be described as follows: upon receiving a user SFC service request, the resource management and VNF orchestration modules in the SDN controller find optimal deployment positions for VNFs and virtual links according to the service chain request and underlying resource status, following specified deployment strategies to generate a service path that meets specific functional and performance requirements. To abstractly represent the SDN-based service function chain deployment problem, it can be reduced to a two-level model: SFC Policy → Logical Function Chain and Logical Function Chain → Specific Service Path, as shown in [Figure 2: see original paper]. The SFC Policy → Logical Function Chain process involves applications dynamically orchestrating and combining VNFs through northbound interfaces. The Logical Function Chain → Specific Service Path process involves rational allocation of underlying resources through southbound interfaces.

Definition 1: SFC Policy. Contains a target set and a service function set. The target set represents the targets requiring processing actions, while the service function set represents the sequence of service functions that data packets must traverse. The SFC policy can be represented as the set $P_{stm} = \{v_s, v_t, (c_1, c_2, ..., c_m)\}$, where v_s represents the source node, v_t represents the destination node, and $(c_1, c_2, ..., c_m)$ represents the sequence of service functions that data packets from source node v_s to destination node v_t must pass through, where m represents the number of functions in the service request.

Definition 2: Logical Function Chain. The controller orchestrates and combines VNF functional modules to form a logical function chain, which can be represented as the set $C = \{c_1, c_2, ..., c_m\}$. A weighted directed graph $G_v = (N_v, L_v)$ represents all VNF deployable nodes and their contextual connections, where $N_v = \{n_{v_1}, n_{v_2}, ..., n_{v_i}, ..., n_{v_m}\}$ represents the set of logical nodes (i.e., $N_v = \{n_{v_i} | 1 \leq i \leq m\}$), and $L_v = \{(n_{v_i}, n_{v_j}) | 1 \leq i < j \leq m\}$ represents the set of virtual links between logical nodes.

Definition 3: Specific Service Path. According to a specified objective function, find corresponding optimal placement positions for VNFs and virtual links in the logical function chain to form a specific service path from source node to destination node. The underlying network consists of all physical nodes and connecting links, which can be represented by a weighted undirected graph $G_s = (N_s, L_s)$, where $N_s = \{n_{s_1}, n_{s_2}, ..., n_{s_j}, ..., n_{s_k}\}$ represents the set of underlying physical nodes (i.e., $N_s = \{n_{s_j} | 1 \leq j \leq k\}$), and $L_s = \{(n_{s_i}, n_{s_j}) | 1 \leq i < j \leq k\}$ represents the set of physical links between physical nodes.

Taking the SFC Policy → Logical Function Chain → Specific Service Path

process in [Figure 2: see original paper] as an example, we describe the instantiation flow from initiating a service request to successfully achieving service deployment and finally providing corresponding functional services. First, the orchestration plane constructs a logical function chain composed of 4 VNFs through the northbound interface according to tenant requests. Second, the control plane maps the logical service chain to physical underlying nodes based on underlying network resource status, controlling device forwarding through southbound protocols according to certain deployment strategies. If the logical service function chain mapping is successful, corresponding instantiation resources are allocated, forming a service path that meets specific functional and performance requirements. In the specific service path, shaded nodes represent physical nodes where VNFs are deployed, while unshaded physical nodes only serve forwarding functions.

1.3 Integer Linear Programming-Based Optimization Deployment Model

The optimization deployment strategy comprehensively considers service resource requirements and network node resource conditions to find optimal deployment positions for VNF nodes and virtual links according to a specified objective function. This section establishes an integer linear programming (ILP) model with the goal of minimizing physical resource occupation. For formal representation, the symbols used in this paper are defined in .

** Main Symbol Definitions**

The optimization deployment model is formulated as follows:

1) **Objective Function:** Equation (1) represents the objective function of the deployment model, which minimizes physical resource occupation to achieve minimal service function chain deployment cost. In the equation, α and β are scaling parameters used to adjust the influence factors of VNF mapping and link mapping costs. δ represents an infinitesimal value to ensure non-zero denominators. To improve underlying physical resource utilization, the objective function uses remaining resources as denominators to bias VNF mapping and link mapping strategies toward nodes or links with more remaining physical resources.

2) **Resource Constraints:** Equations (2) and (3) provide computing resource and bandwidth resource constraints. For computing resource constraints, the remaining resources of underlying physical nodes should satisfy the computing resource requirements for VNF instantiation. For bandwidth resource constraints, the remaining resources of underlying physical links should satisfy the bandwidth resource requirements in requests.

3) **Connectivity Constraint:** Equation (4) is the connectivity constraint. If VNF nodes n_{v_i} and n_{v_j} are mapped to underlying nodes n_{s_k} and n_{s_l} respectively during the VNF node mapping phase, then during the logical

link mapping phase, the logical link (n_{v_i}, n_{v_j}) will be mapped onto an underlying path from node n_{s_k} to node n_{s_l}. At the source node, the outgoing flow is 1 and the incoming flow is 0, so $\sum y_{\dots} = 1$; at the destination node, the outgoing flow is 0 and the incoming flow is 1, so $\sum y_{\dots} = -1$; at other nodes, both incoming and outgoing flows are 1, so $\sum y_{\dots} = 0$.

4) **Delay Constraint:** Equation (5) is the delay constraint for service paths. When service providers successfully map service requests to the underlying physical network, they construct an end-to-end service path. Delay represents the time consumed by data flows from source node to destination node, and the delay constraint ensures satisfaction of network Service Level Agreements (SLA).

5) **Variable Constraints:** Equations (6) and (7) are mapping constraints for logical function chains and binary constraints for variables, respectively. Equation (6) ensures that services in the same service request can only be mapped to a unique underlying node. Equation (7) specifies the binary constraints for variables x and y.

---

2 Algorithm Description

For the service function chain deployment model proposed above, optimization theory can be used to reduce it to a virtual network embedding problem [14-16]. The virtual network embedding problem has been proven to be NP-Hard [17], so the service function chain deployment problem is also NP-Hard. That is, under the condition $P \neq NP$, no polynomial-time algorithm exists to solve this problem, and efficient search algorithms are typically used to obtain approximate solutions. Traditional two-stage mapping algorithms in virtual network embedding [18] separate the process into node mapping and link mapping. However, logical function chain mapping in service chain deployment differs from traditional virtual network embedding in that it requires comprehensive consideration of service chain end systems, instantiation resource allocation, and orchestration order characteristics. Based on the characteristics of service function chain deployment, this paper designs a "sort-first, greedy-second" heuristic search algorithm to quickly and efficiently solve the service function chain deployment model.

2.1 Service Chain Deployment Based on Efficient Heuristic Algorithm

Current research on service function chain deployment tends to focus either solely on node resource utilization or solely on link resource utilization. To address this challenge, this paper proposes an efficient heuristic search algorithm (Greedy Node Embedding with k-Shortest Path Link Embedding Algorithm, G-kSP). This algorithm comprehensively considers tenant service resource requirements, underlying physical remaining resources, and QoS conditions, simultane-

ously focusing on both VNF placement and path selection stages during service function chain deployment to obtain a globally optimal resource allocation strategy.

To more precisely describe VNF resource requirements and underlying physical node resource remaining conditions in logical function chains, this section defines VNF comprehensive resource and physical node comprehensive remaining resource functions as follows:

- $E(n_{v_i})$ represents the set of all adjacent virtual links to VNF node n_{v_i}
- $c(n_{v_i})$ represents the computing resource requirement for VNF node n_{v_i} instantiation
- $r(e_{v_i})$ represents the bandwidth resource requirement for link instantiation
- $E(n_{s_k})$ represents the set of all adjacent physical links to underlying physical node n_{s_k}
- $r(n_{s_k})$ represents the current remaining computing resources of physical node n_{s_k}
- $r(e_{s_k})$ represents the current remaining bandwidth resources of link e_{s_k}

To simplify the search space of traditional heuristic algorithms and improve the efficiency of solving global resource allocation strategies, the G-kSP algorithm designed in this paper for the deployment model shown in Equations (1)-(7) is a sort-first, greedy-second efficient search method. The sorting approach aims to simplify the algorithm' s search space, enabling optimal service paths to be obtained without global search. During the greedy selection process, the algorithm prioritizes physical nodes or links with the most remaining resources as the greedy strategy. The greedy strategy directly impacts each sub-problem operation, producing optimal resource allocation strategies for each sub-problem and thus quickly and efficiently approximating the model' s objective function. Additionally, during iteration, the algorithm employs a backtracking mechanism: when the current iteration cannot find a solution meeting resource constraints for a VNF or virtual link, it backtracks to the suboptimal solution in the previous stage' s mappable set. The detailed operation of the G-kSP algorithm is described in .

** G-kSP Algorithm Detailed Process**

Input: SFC request P_{stm}, underlying physical network $G_s = (N_s, L_s)$
Output: VNF mapping set M_N, virtual link mapping set M_L

a) **Construct VNF Queue** Q**.** Build the SFC policy according to online arrival requests, orchestrate and combine VNFs to form a virtual network layer G_v with chain structure. Calculate the resource requirement $C(n_{v_i})$ for each VNF node in set G_v, reorder them from largest to smallest, and place them into queue Q.

b) **Construct Mappable Node Set M.** Based on underlying physical node information, calculate the comprehensive remaining resource quantity $R(n_{s_k})$ of mappable physical nodes, and use Equations (2)-(6) to determine whether the remaining resources of physical nodes satisfy resource constraints. Form the mappable set M with underlying physical nodes that meet the constraints.

c) **Select Optimal VNF Node Position.** Take the first VNF from queue Q that needs deployment, use Equation (10) to search physical nodes in mappable set M, and map the VNF to physical node n_{s_k}. If the VNF is successfully mapped to physical node n_{s_k}, then set $x_{\dots} = 1$ and record to set M_N; otherwise, backtrack to the suboptimal solution in mappable set M from the previous iteration. Update queue Q.

d) **Obtain VNF Mapping Set M_N.** Check whether there are still VNFs in queue Q that have not completed mapping. If yes, return to step 2 to execute the next VNF node in queue Q; otherwise, complete VNF mapping and obtain set M_N.

e) **Construct Path Set R.** After completing VNF mapping, based on set M_N and following the service function chain order P_{stm}, use the K-Dijkstra algorithm with remaining link bandwidth $r(e_{s_k})$ as weight to calculate k shortest paths from source node v_s to destination node v_t, denoted as path set $R = \{r_1, r_2, ..., r_k\}$.

f) **Select Optimal Virtual Link Position.** Sort set R by bandwidth resource occupation from smallest to largest, and select the first path. Then set $y_{\dots} = 1$ and record to set M_L; otherwise, backtrack to the suboptimal solution in the shortest path set R.

g) **Service Time Monitoring and Resource Allocation.** After completing virtual link mapping, use Equation (7) to determine whether the service function chain QoS metrics are satisfied. If satisfied, the underlying resource pool allocates corresponding instantiation resources to complete the SFC deployment process and updates underlying resource status; otherwise, reclaim allocated resources and mark service function chain deployment as failed.

2.2 Theoretical Analysis of Algorithm Complexity

During service function chain deployment, assume there are m VNFs requiring mapping, n physical nodes, and l physical links in the underlying network. In the VNF mapping process, G-kSP algorithm first sorts the m VNFs by required resources from largest to smallest. The computation of VNF mapping mainly involves sorting resource requirements and searching underlying remaining resources, with complexity $O(m \log m)$. In the virtual link mapping process, since each virtual link is mapped to the underlying physical network using the K-Dijkstra algorithm, its time complexity is $O(K(l \log n))$. In summary, the time

complexity of the G-kSP algorithm is $O(m \log m + K(l \log n))$, which can be equivalently expressed as $O(\log n)$.

---

3 Experimental Evaluation and Analysis

To verify the effectiveness of the proposed deployment algorithm (denoted as G-kSP), this paper selects two other typical deployment algorithms for comparison (as shown in). We evaluate the G-kSP algorithm using three important deployment metrics: load balancing degree, request acceptance rate, and time complexity. Load balancing degree reflects the algorithm' s search balance, request acceptance rate reflects the algorithm's search performance in the solution space, and request processing time reflects computational performance.

** Comparison Algorithms**

Algorithm	Description
G-kSP	The efficient heuristic search algorithm proposed in this paper based on global resource utilization
TS	Deployment algorithm based on tabu search [9] that searches for optimal deployment positions in the global network by setting tabu tables, focusing on node resource optimization
Greedy	Algorithm [12] that assigns higher priority to low-delay links, focusing on link resource optimization

3.1 Experimental Environment and Parameter Settings

The experiments run on a Linux PC with Intel Core i7-3770 3.60 GHz and 8 GB RAM. Network topologies are generated using the GT-ITM tool [19], and algorithm programs run through Matlab. Similar to reference [20], this paper uses a test case with a network topology containing 6 identical physical nodes as the underlying infrastructure, resembling a lightweight cloud data center network environment. As shown in [Figure 3: see original paper], the network topology consists of 6 nodes and 14 links (numbers on nodes represent node labels), with node 1 as the data traffic ingress and node 6 as the egress. All nodes are assumed to be deployed in a cloud data center, and all nodes can carry service functions. Physical node resources and link bandwidth remaining capacities follow a random distribution of [1000, 1500]. Each SFC request consists of different types of service functions with quantity following a random distribution of [2, 5]. Resource requirements for each SFC node and link follow a uniform

distribution of [0.5, 1]. The number of service types supported by the underlying network is set to 10, with each node randomly providing 1-5 of them.

3.2 Algorithm Performance Comparison

a) Node Load Balancing Degree: This metric represents the load accumulation of physical nodes, reflecting the algorithm' s performance in node resource utilization. Smaller values are better, indicating more balanced load across physical nodes. The node load balancing degree is calculated as:

We generate SFC request intensities from 10 to 100 and statistically measure the node load balancing degree of the three algorithms, as shown in [Figure 4: see original paper].

[Figure 4: see original paper] shows that the TS-based deployment method achieves the smallest maximum node load balancing degree while obtaining the maximum average load balancing degree, indicating better node load balancing performance. Conversely, the Greedy algorithm performs poorly, with G-kSP algorithm in between. The reason is that the TS algorithm uses underlying node computing resource capacity as its selection strategy and can search for better node selection schemes through repeated iterations, enabling effective utilization of node resources. The Greedy algorithm prioritizes nodes on low-delay links, which can lead to suboptimal load balancing on some functional nodes. The G-kSP algorithm achieves moderate performance.

b) Specific Path Load Balancing Degree: This metric represents the load accumulation of the entire service function chain, reflecting the algorithm' s performance in network-wide resource utilization. Smaller values are better. The load balancing degree of the specific execution path is calculated as:

where scaling factors ω_N and ω_L adjust the emphasis on node load and link load.

We generate SFC request intensities from 10 to 100 and statistically measure the service path load balancing degree of the three algorithms, as shown in [Figure 5: see original paper]. The figure shows that as request arrival intensity increases, available resources decrease and the curve gradually flattens. As resource bottlenecks emerge, request acceptance rate gradually decreases and service path construction failures increase. With increasing request arrival intensity, the G-kSP-based deployment method most effectively balances service path load. Since both TS and Greedy algorithms fail to allocate resources from a global optimal path perspective, they result in relatively unbalanced network service path loads. However, the Greedy algorithm assigns higher deployment priority to low-delay links, which can lead to frequent reuse of some links and premature resource bottlenecks. Therefore, TS algorithm outperforms Greedy algorithm in service path load balancing performance.

c) Request Acceptance Rate: This metric represents the proportion of requests that satisfy resource, delay, and other constraints, reflecting the algo-

rithm' s search performance in the solution space. Larger values are better. Request acceptance rate is defined as the ratio of successfully deployed SFCs to total SFC requests:

where $\sum SFC^{t}_{accepted}$ represents the number of SFCs successfully deployed to the underlying physical network from time t_0 to t, and $\sum SFC^{t}_{total}$ represents the total number of SFC requests arrived.

We generate SFC request intensities from 10 to 100 and statistically measure the request acceptance rates of the three deployment algorithms, as shown in [Figure 6: see original paper]. Under the same conditions, the average request acceptance rate of G-kSP algorithm is approximately 95.4%, TS deployment algorithm is about 92.3%, and Greedy deployment algorithm is about 81.7%. The reason is that G-kSP algorithm considers global optimal path, reduces load balancing degree of the entire service function chain, improves underlying physical resource utilization, occupies less resource space, and maximally reserves instantiation resources for other requests in the queue, thereby increasing request acceptance rate. The Greedy algorithm focuses mainly on link resource utilization, increasing node processing queue time and underlying network load, thus reducing request acceptance rate. The TS algorithm primarily considers node resource conditions and can search for optimal deployment schemes through repeated iterative calculations, resulting in improved service request acceptance rate compared to Greedy algorithm.

d) Time Complexity: This metric represents the runtime from request issuance to deployment completion, reflecting the algorithm' s computational performance. Smaller runtime values are better. We generate SFC request intensities from 10 to 100 and statistically measure the request processing delay of the three algorithms, as shown in [Figure 7: see original paper].

[Figure 7: see original paper] shows that as request arrival intensity increases, G-kSP algorithm has smaller processing delay and can complete service function chain deployment faster. Conversely, Greedy algorithm has larger processing delay, with time complexity approximately 10-12 times that of G-kSP algorithm, while TS algorithm is in between. The analysis reveals that Greedy algorithm is essentially a two-stage mapping algorithm (VNF node mapping and logical link mapping). In the VNF mapping stage, it enumerates all underlying nodes satisfying constraints, with time complexity $O(n^m)$. In the logical link mapping stage, it selects the path with minimum bandwidth occupation between nodes, with time complexity $O(m^2)$. VNF nodes that have completed mapping must wait for the next VNF node in the queue to complete mapping before establishing service paths. Therefore, Greedy algorithm not only has large search space but also suffers from virtual link mapping being affected by VNF mapping, resulting in longer waiting times for VNFs in the queue and lower algorithm efficiency. The total time complexity is $O(mn^2 + m^2)$. TS algorithm' s processing delay is affected by tabu list length λ, which increases storage space and computational delay as λ grows. TS algorithm' s time complexity can be expressed

as $O(m\lambda)$. Combined with the theoretical complexity analysis in Section 3.2, the trend of curves in [Figure 7: see original paper] is basically consistent with theoretical analysis.

4 Conclusion

The service function chain deployment method proposed in this paper simultaneously addresses node and link resource utilization during deployment. First, for the service function chain optimization deployment problem, we present the overall architecture for SDN-oriented service function chain deployment, reduce it to a two-level model of SFC Policy → Logical Function Chain → Specific Service Path, and model it as an integer linear programming mathematical model. Second, addressing the difficulty that current deployment methods fail to fully consider global resource utilization, we design a sort-first, greedy-second heuristic search algorithm. Compared with tabu search deployment algorithm and greedy heuristic algorithm, our method demonstrates good performance in metrics such as load balancing degree, request acceptance rate, and time complexity, and can obtain optimal resource allocation schemes more quickly and effectively. Currently, industry research on SDN-based service function chains continues to deepen. Future work will focus on reliability-aware service function chain deployment to meet network service chain reliability requirements and carrier-grade network service quality.

References

[1] Cisco. Cisco visual networking index: forecast and methodology [R/OL]. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html

[2] Bhamare D, Jain R, Samaka M, et al. A survey on service function chaining [J]. Journal of Network & Computer Applications, 2016, 75 (C): 138-155.

[3] Medhat A M, Taleb T, Elmangoush A, et al. Service function chaining in next generation networks: state of the art and research challenges [J]. IEEE Communications Magazine, 2017, 55 (2): 216-223.

[4] Mckeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks [J]. ACM SIGCOMM Computer Communication Review, 2008, 38 (2): 69-74.

[5] ETSI. Network functions virtualization [R]. NFV White Paper, 2012.

[6] Takacs A, Green H, Shirazipour M, et al. Network function placement for NFV chaining in packet/optical datacenters [J]. Journal of Lightwave Technology, 2015, 33 (8): 1565-1570.

[7] Huang Meitian, Liang Weifa, Xu Zichuan, et al. Throughput maximization in software-defined networks with consolidated middleboxes [C]// Proc of Local Computer Networks. 2016: 298-306.

[8] Kim S, Park S, Kim Y, et al. VNF-EQ: dynamic placement of virtual network functions for energy efficiency and QoS guarantee in NFV [J]. Cluster Computing, 2017, 20 (3): 2107-2117.

[9] Mijumbi R, Serrat J, Gorricho J L, et al. Design and evaluation of algorithms for mapping and scheduling of virtual network functions [C]// Proc of International Conference on Network Softwarization. 2015: 1-9.

[10] Bari M F, Chowdhury S R, Ahmed R, et al. On orchestrating virtual network functions [C]// Proc of International Conference on Network and Service Management. 2015: 50-56.

[11] Zeng Menglu, Fang Wenjian, Zhu Zuqing. Orchestrating tree-type vnf forwarding graphs in inter-dc elastic optical networks [J]. Journal of Lightwave Technology, 2016, 34 (14): 3330-3341.

[12] Lukovszki T, Rost M, Schmid S. It's a match!: near-optimal and incremental middlebox deployment [J]. ACM SIGCOMM Computer Communication Review, 2016, 46 (1): 30-36.

[13] Dwaraki A, Wolf T. Adaptive service-chain routing for virtual network functions in software-defined networks [C]// Proc of Workshop on Hot Topics in Middleboxes and Network Function Virtualization. New York: ACM Press, 2016: 32-37.

[14] Even G, Rost M, Schmid S. An approximation algorithm for path computation and function placement in SDNs [C]// Proc of International Colloquium on Structural Information and Communication Complexity. Cham: Springer, 2016: 374-390.

[15] Schmid S. Online Admission control and embedding of service chains [C]// Proc of International Colloquium on Structural Information and Communication Complexity. New York: Springer-Verlag, 2015: 104-118.

[16] Moens H, Turck F D. VNF-P: a model for efficient placement of virtualized network functions [C]// Proc of International Conference on Network and Service Management. 2014: 418-423.

[17] Chowdhury M, Rahman M R, Boutaba R. ViNEYard: virtual network embedding algorithms with coordinated node and link mapping [J]. IEEE/ACM Trans on Networking, 2012, 20 (1): 206-219.

[18] Yu Minlan, Yi Yung, Rexford J, et al. Rethinking virtual network embedding: substrate support for path splitting and migration [J]. ACM SIGCOMM Computer Communication Review, 2008, 38 (2): 17-29.

[19] Calvert K L, Bhattacharjee S. How to Model an Internetwork [C]// Proc of IEEE INFOCOM. 1996: 594.

[20] Sahhaf S, Tavernier W, Rost M, et al. Network service chaining with optimized network function embedding supporting service decompositions [J]. International Journal of Computer & Telecommunications Networking, 2015, 93 (P3): 492-505.

Note: Figure translations are in progress. See original paper for figures.