OpenNSP: Open Network Service Provision for Mobile Networks

Qi Xu, Deyun Gao, Huachun Zhou*, Bohao Feng, and Taixin Li
School of Electronic and Information Engineering
Beijing Jiaotong University
Beijing, China
{15111046, gaody, hchzhou, 11111021, 14111040}@bjtu.edu.cn

Abstract

In recent years, many innovative techniques are emerged for satisfying various user requirements, such as Software Definition Network (SDN) and Network Function Virtualization (NFV). However, the current solutions for network service provision still lack enough flexibility and openness that result in high expenditures and complexity. Thus, we propose an Open Network Service Provision (OpenNSP) solution in this paper, aiming to address the multi-domain problems of service function chain outsourcing for mobile networks. Particularly, OpenNSP is based on IETF hierarchical service function chaining (hSFC) architecture to implement service chains hierarchically in multi-domain networks comprising operator network and cloud datacenters. It outsources service chains, which is usually located in SGi-LAN, to the public clouds, enabling fast, flexible and cost-effective network service provision. Besides, OpenNSP achieves the programmable service function placement and chaining by the integration of SDN and NFV. Much work has been done, and the experiment results confirm its feasibility and flexibility.

Keywords: SFC, Network services, Mobile networks

1 Introduction

In recent years, the evolution of mobile communications has tremendous impacts on social developments. Recent statics show that the user number of mobile wireless broadband has exceed that of fixed wired broadband, and global mobile traffic is expected to grow from 2.6EB to 15.8EB by 2018 [1]. Generally, these mobile traffic will be handled by several middleboxes with various network functions (NFs) to get value-added network services (such as performance enhancement and security) over the normal network connectivity. Typical network functions are firewall, network address translation (NAT), deep packet inspection (DPI), and so on. Each traffic flow traverses these network functions in a certain order according to the service provider's strategies, which is called Service Function Chain (SFC).

Traditionally, in the LTE mobile network, the user traffic is classified by the packet data network gateway (P-GW) and sent to the SGi-LAN, where the operator provides value-added network services, as the red dotted line shown in Figure 1. Usually, SGi-LAN is an ethernet network with a large number of hard-wired middleboxes, and connects evolved packet core (EPC) with Internet [8]. Although this tradditional way has worked well so far, it is unable to introduce innovative services flexibly, and makes high operational expenditures (OPEX) and capital expenditures (CAPEX), especially for 5G networks. Thus, there is an increasing attention to the new solutions for network service provision.

With the rapid development of Software Defined Network (SDN) and Network Function Virtualization (NFV), many solutions have been proposed for network service provision. For example, OpenSCaaS [6] was an open service chain as a service platform by taking advantage of SDN together with

Research Briefs on Information & Communication Technology Evolution (ReBICTE), Vol. 3, Article No. 4 (October 15, 2017)
*Corresponding author: School of Electronic and Information Engineering, Beijing Jiaotong University, No.3 Shangyuancun, Haidian District, Beijing, 100044, China, Tel: +86-137-1816-8186

NFV. CATENAE [4] was a SFC system for the SGi-LAN, leveraging SFC controllers to configure SDN switches to rewrite MAC address for supporting traffic steering. The approach in [18] incorporated SDN in the current EPC architecture for supporting mobile service chaining flexibly and effectively. However, these researchs are based on an assumption that a single network domain can provide all required service functions, which may cause scalability and cost issues. Hence, there is an increasing attention in outsourcing service chains to public clouds [5], which uses service functions provided by public clouds to construct service chains, as the green line shown in Figure 1. Outsourcing SFCs improve scalability and cost efficiency. Nonetheless, there are several other problems needed to be addressed. Particularity, outsourcing service chains will lead to a multi-domain issue about how to chain several service functions deployed over multiple geographical locations in an orderly manner. Besides, it may cause additional delays as it travels through multiple public clouds, while 5G services have more stringent requirements for end-to-end delay [2].

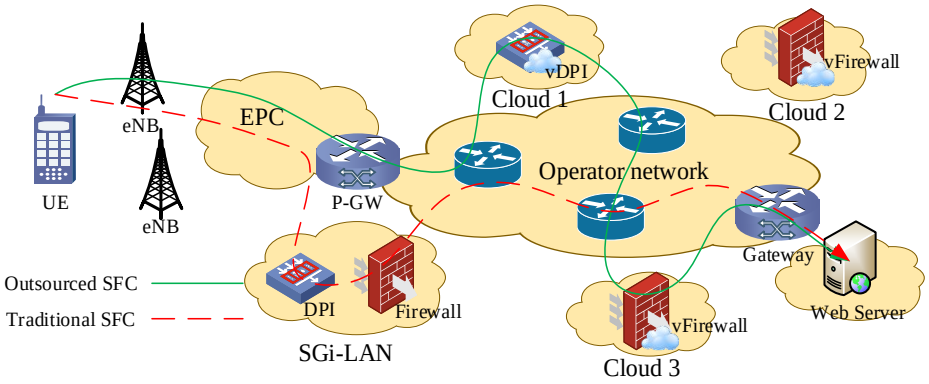


Figure 1: Tradditional SFC vs Outsourced SFC

In this paper, we propose a new solution called Open Network Service Provision (OpenNSP). It is a hierarchical framework based on IETF hierarchical SFC (hSFC) architecture [7], integrating SDN and NFV, enabling automated, flexible, and scalable network service provision for mobile networks. In OpenNSP, the service chain that represents the intergrated network service can be divided into several simple sub-chains. The sub-chain is also a sequence of service functions deployed among several cloud datacenters (DCs) at diffierent locations. In general, the main contributions of this paper are as follows: 1) We present the architecture design of OpenNSP for mobile networks. OpenNSP outsources service chains, which are originally located in the SGi-LAN, to the cloud DCs distributed in operator networks. It describes the required network services as service graphs, which will be further divided into several subgraphs processed by the selected cloud DCs through hierarchical orchestration. Then, cloud DCs achieve service functions placement and service function chaining for the received subgraphes. Finally, these sub-services in cloud DCs can be chained together to form a complete service, enabling flexible, scalable network service provisioning. 2) We implement the prototype system for OpenNSP, and conduct several experiments based on three use-cases about security services, to confirm its feasibility and flexibility. Specifically, Docker is adopted to virtualize service functions; NSH protocol patch is used to enable OpenVSwitch (OVS) to support the service function chaining; we also expand OpenDaylight (ODL) SFC controller, enabling it to support hierarchical SFC configuration and management.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes the proposed open network service provision framework. Section 4 presents our prototype implementation and three usecases of security service. Then we analyze experiment results in section 5. Section 6 concludes this paper.

2 Related Work

Recently, many researches have focused on the flexible service function chain based on SDN/NFV architecture. These researches can be divided into two categories, either within single-domain network or within multi-domain networks. With regard to the former, Ding et al. [6] presented the OpenSCaaS architecture, which achieves flexible, scalable and automated service chains without modifying SDN standards and middleboxes. Li et al. [11] proposed a method to provide services in a flexible and scalable way by using SDN/NFV/PCE technologies. It also proposed an algorithm based on grey system theory to select a proper service function path. However, none of them took the IETF SFC framework and NSH protocol into account. With regard to service chains across multi-domain networks, most of researches focuses on the scalability issues. Dietrich et al. Chen et al. [5] proposed an heuristic algorithm based on hidden markov model to resolve the service function chain outsourcing problem. The goal is to minimize the total cost with consideration of extra delay incurred by outsourcing service functions to public clouds. Bhamare et al. [3] investigated virtual network function placement in multi-cloud scenario, and presented an affinity-based heuristic algorithm to minimize delay and inter-cloud traffic. These work proved their solutions by performing simulations, while their effectiveness in actual scene was unknown. Different from the above work, Vu et al. [17] proposed to the implementation of hSFC architecture by using OpenDaylight SFC, and presented a prototype system for deploying service chains across multi-domain networks. It provides the basis for implementation of SFC across multi-domain networks, but there is still much work to be done, such as the integration with NFV and hierarchical orchestration.

Besides, the research about SFC in mobile network began to be concerned. Bifulco et al. [4] presented CATENAE system for chaining service functions readily in SGi-LAN by using SDN, without modifying any protocol or network. Taleb et al. [15] also investigated the rapid dynamic deployment of SFC in SGi-LAN, reducing cost for mobile operators. Li et al. [10] proposed a security service chaining approach based on fuzzy theory for selecting the optimal security function order in mobile-edge computing environment. However, these approaches have little consideration on the multi-domain problem.

In this article, we propose the OpenNSP framework for 5G networks. It aims to address the inevitable multi-domain problem caused by SFC outsourcing, and improve flexibility, scalability and cost-efficiency for network service provision.

3 Open Network Service Provision Framework

The basic idea of OpenNSP is: (i) using the hSFC architecture to hierarchically deploy service chains across multi-domain networks for reducing complexity; (ii) using NFV to migrate service functions from proprietary hardware middleboxes to common standard servers for improving scalability and cost-efficiency; (iii) using SDN to decouple the control plane and data plane of network forwarding devices, enabling the flexible and programmable traffic steering; (iv) using Network Service Header (NSH) protocol to encapsulate service chains, enabling topology independency, where service forwarding is independent with the topology of underlying network. Figure 2 depicts an overview of the OpenNSP framework, showing its two levels of hierarchy, namely high-level and low-level. Note that the hierarchy of OpenNSP is iterated, for example, the low-level network may be divided into several lower-level networks. Moreover, Figure 2 gives various components of different planes and the interaction between them. More details are described as following.

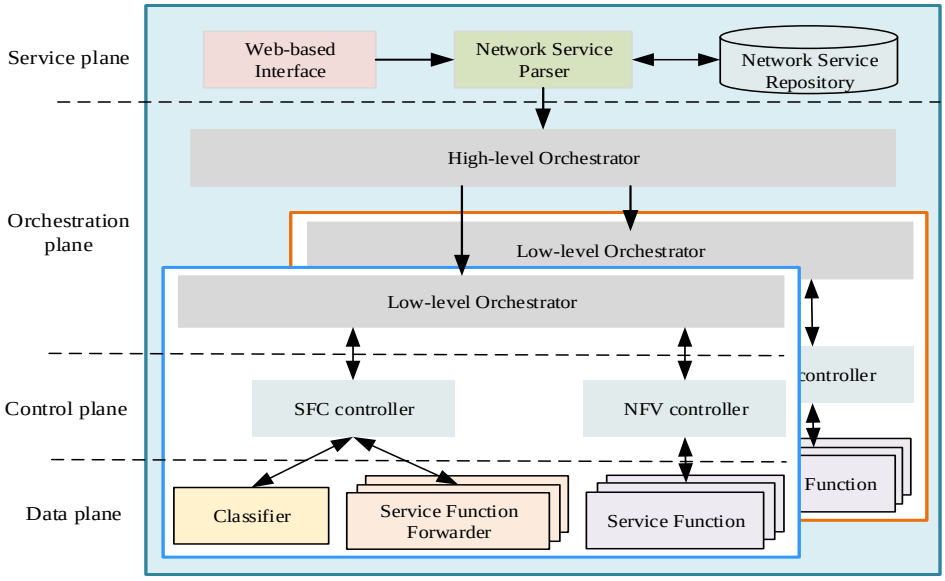


Figure 2: Overview of Open Network Service Provision Framework

3.1 Service Plane

In OpenNSP service plane, users are able to customize desired services. The first step of network service provision is to capture service requirements. Referring to the business models of current cloud service providers, a Web UI is provided as an interface for users to access the system. Users can select desired services from a list on the web, and choose different levels of quality of service such as bandwidth capacity. The network service parser can obtain the key properties of desired services by analyzing requirement information, and then match corresponding service templates stored in network service repository. Service templates describe the type and sequence of service functions. With the key properties of service requirements and service templates, a service graph is able to be constructed using service description language. An example of service graph can be described as follows:

3.2 Orchestration Plane

In OpenNSP orchestration plane, the concept of hierarchical orchestration is introduced for the consideration of multi-domain issue. Firstly, the multi-domain issue is an inherent attribute of end-to-end service deployment. Because many SFs are constrained by location scopes, it is difficult for a single domain to provide all the required SFs. As a result, several network domains are taken to provide specific SFs and composed them together to construct complete network services. Secondly, hierarchical orchestration can improve the scalability of service deployment. With hierarchy, network operators can quickly introduce new services by simply modifying network configurations of relevant low-level domains.

Hierarchical orchestration means that several different levels of orchestrators execute corresponding granularity of service orchestration in their network domains. For example, in the two levels of hierarchy shown in Figure 2, high-level orchestrator executes service graph partition, while low-level orchestrators execute service subgraph embedding, as shown in Figure 4. Specifically, high-level orchestrator maintains the resource description of the high-level domain, which is a relatively abstract functional description (such as the SF types that each sub-domain can provide). According to the current available resource, the service graph from service plane will be divided into several service subgraphs, which will be processed by low-level orchestrators. Then the high-level orchestrator controls the components of

```
<service-graph>
  <name>security service< /name>
  <property> //describe the properties of the service
    <src>10.0.6.13< /src> //the source noede of the service
    <dest>10.0.10.5< /dest> //the destination noede of the service
    <level>Enhanced< /level> //the capacity level of the service
  < /property>
  <service-function-chain> //describe the SF sequence of the service
    <symmetric>false< /symmetric> //describe whether the service is bidirectional
    <number>4< /number> //the length of the chain
    <service-functions> //describe the type of SFs in order
      <sf1>firewall< /sf1>
      <sf2>nat< /sf2>
      <sf3>nat< /sf3>
      <sf3>firewall< /sf3>
    < /service-functions>
  < /service-function-chain>
< /service-graph>
```

Figure 3: Service graph in XML format

control plane to chain these distributed subgraphs in an orderly manner and combine them into a complete service. Furthermore, similar to the NFV MANO, low-level orchestrator maintains the resource description of its autonomous domain. It is a relatively specific quantitative description, such as how much CPU or memory that each service node can provide, how much bandwidth that each physical link can have, and so on. According to the current resource description, the required resources are allocated for the received service subgraph. And then the control components is controlled to chain these required SFs in an orderly manner and combine them into a sub-service provided for high-level domain.

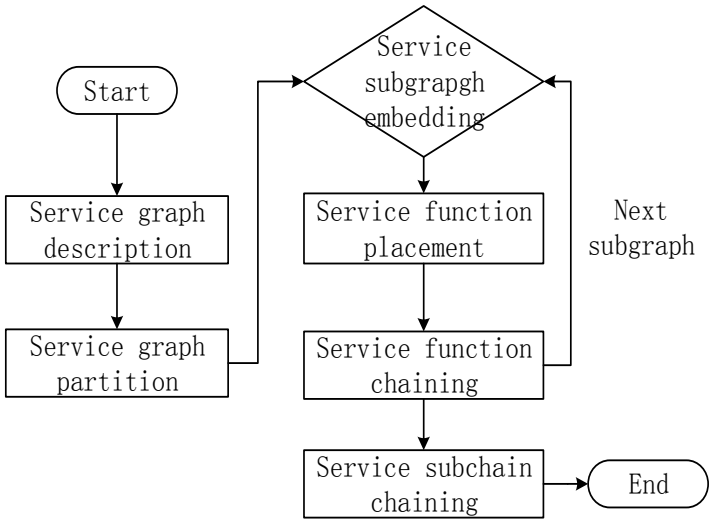


Figure 4: The workflow of the service graph

3.3 Control Plane

The design of OpenNSP control plane is based on the integration of SDN/NFV, and the SFC framework of IETF SFC working group. There are two control plane components, described as following.

NFV controller: it performs service function placement, which embeds SFs of service graph into the service nodes in the underlying network infrastructure. NFV controller manages the all service nodes in data plane, collects and maintains their resource descriptions. Following the decision from orchestration plane, NFV controller controls the selected service nodes to instantiate required SFs.

SFC controller: it performs service function chaining, which transforms the logical service chain in service graph into the actual service function path. SFC controller manages classifiers (CFs) and service function forwarders (SFFs) through southbound interface (OpenFlow or alternative protocol). Taking the advantage of SDN, the classification rules and forwarding rules can be install/update/delete programmably. These rules are derived from the information extraction of the service graph, such as IP address and SPI/SI.

3.4 Data Plane

The design of OpenNSP data plane is based on the hSFC framework, and consists of five types of components, namely classifiers (CFs), internal boundary nodes (IBNs), service function forwarders (SFFs), service functions (SFs) and service nodes (SNs). Moreover, the NSH protocol is used as the SFC encapsulation to implement service function path. Specifically, CFs maintains the classification rules of local domain, which determine the matching relation between flows and service chains. Through applying classification rules, CFs assign the required service chain for incoming data flow and add the NSH header for it. SFFs maintain a set of forwarding rules. The match between forwarding rules and the SPI/SI information in the NSH header directs the packets to the next SF or SFF along the service function path. SFs refer the network middlebox that processes data packets and implemented as software running in the virtual machine of service node. It is worth noting that the IBN is a key component of the architecture that plays a role in bridging service chains of different level domains. IBNs are logically composed of CFs and SFFs in low-level domain and controlled by the SFC controller. The classifier in IBN applies classification rules to match the packets from high-level domain to construct its NSH encapsulation of low-level service chain. Then IBN can push the high-level SPI/SI into the metadata C3 field of low-level encapsulation. When packets return to the high-level domain after its execution of low-level service chain, the high-level SPI/SI can be restored by the last SFF in low-level service function path. After the replacement and recovery of NSH encapsulation, IBNs connect service chains between high-level domain and low-level domains, which provide the feasibility for the implementation of complex service chains across multiple domains.

4 Performance Evaluation

In this section, we implement the OpenNSP framework in a prototype to verify its feasibility in the real network, and conducted experiments to evaluate its performance.

4.1 Prototype Implementation

The complete mobile network scenario should include user equipment (UE), radio access network (RAN), EPC, SGi-LAN, and Internet. But the SFC use-cases in mobile network [8] are mainly located in the SGi-LAN, and the P-GW serves as the SFC classifier. Therefore, for simplicity, in our prototype system, traffic flows are classified by the P-GW, and then encapsulated into the corresponding service chains.

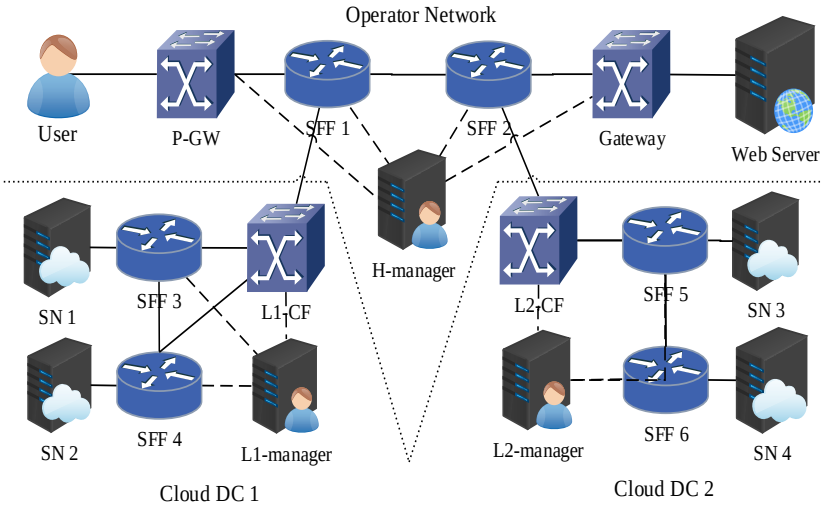


Figure 5: Illustration of prototype

Along service function path, Traffic flows will pass through the cloud DCs located in different locations providing service functions. Finally traffic flows are decapsulated by the gateway and forwarded to the target server. For this prototype, We use 18 servers with dual-core CPU and 2 GB memories, as shown in Figure 5. And the topology of the prototype consists of three network domains, namely operator network, cloud DC 1 and cloud DC 2. Each network domain has a centralized manager, which integrates SFC controller, NFV controller and orchestrator. We also implement an Apache Server in H-manager to provide a Web UI for the service plane of OpenNSP. The network service parser is developed in Python language, about 110 lines of code. It extracts the service requirement information from the Web UI and converts them into a service graph in XML format. Besides, we develop basic functions of the high and low level orchestrators using Python language for the orchestration plane. TCP Socket API is provided for the service graph interaction between high-level orchestrator and low-level ones. And a set of RESTful API are provided for the interaction between orchestration plane and control plane. The interaction information is the result of service orchestration by JavaScript Object Notation. For the control plane, we utilize ODL SFC [12] and Docker Swarm [14] as SFC controller and NFV controller, respectively. The current ODL SFC does not support hierarchical service function chaining, so we refer to the work [17] to improve it, including the development of IBN and the modification of SFF. At last, for the data plane of each cloud DC, we use 2 servers running Docker [9] as service nodes to create a lightweight container for any SF. And we also use 3 4 servers running the modified version of OVS [16] and OpenFlow1.3 protocol, as the CFs and SFFs of data plane of each network domain (the P-GW and Gateway in operator network play the role of CFs). To support NSH protocol, nsh patches [13] are installed in the OVS 2.6.1.

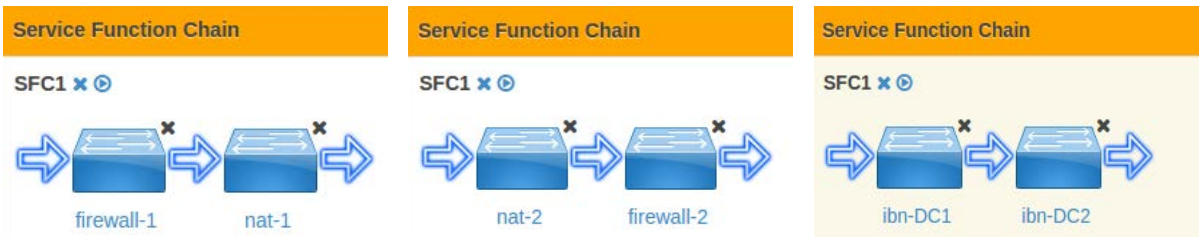
Table 1: Security services of the prototype

	Cloud DC1	Cloud DC2
Security service 1	FW →	FW
Security service 2	FW → NAT→	NAT → FW
Security service 3	FW → NAT→ VPN→	VPN → NAT → FW

Finally, we implement three kinds of end-to-end network service related to security [6]. Their composition and location are shown in Table 1. Considering current security SFs do not support the NSH protocol, we develop the simple examples of security SFs using Python, including firewall (FW), net-

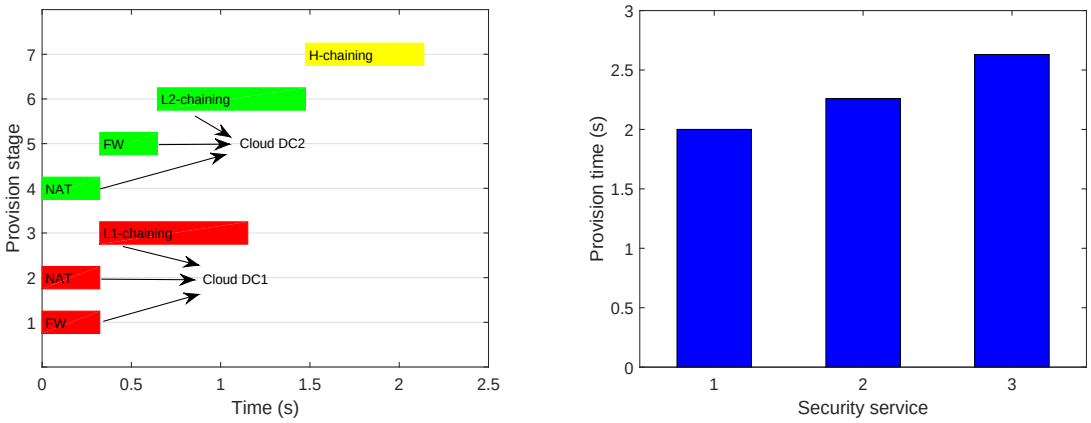
work address translation (NAT) and virtual private network (VPN). We believe that security equipment vendors will enhance their products to support NSH protocol with the promotion of NSH protocol.

4.2 Experimental Results



(a) cloud DC2 (b) cloud DC1 (c) operator network

Figure 6: Deployment of Security Service 2



(a) Provision stages (b) Provision time

Figure 7: Exeperiment Results

Before the experiment, we assume that security function containers already exist in each service node, and that each service node can only start one container at one time. And we use random orchestration algorithm to allocate network and computing resources for service chains.

Figure 6 shows the results of hierarchical deployment for security service 2. As explained in Section 3, the network service parser in the H-manager describes the security service 2 as a service graph in XML language format, as shown in Figure 3. Then, the high-level orchestrator in the H-manager divides this service graph into two subgraphs according to the location constraint shown in Table 1. Figure 6(a) and Figure 6(b) show the results of the subgraph deployment in the cloud DC 1 and cloud DC 2, respectively. Finally, the H-manager chains these two sub-chains into a complete service chain, and the deployment of sub-chains chaining in operator network is shown in Figure 6(c).

The provision stage is measured to demonstrate the feasibility of OpenNSP in the real network. As shown in Figure 7(a), we give an example of the provision for security service 2, which includes 3 main stages, marked with red, green and yellow respectively. The red/green part represents the execution procedure of service subgraphs in cloud DCs, consisting of service function placement and service function

chaining; the yellow part represents the procedure within operator network, chaining the sub-services among different domain. The results show that the complete process takes about 2.138 seconds. Among it, the execution of security function instantiation takes about 0.323 seconds, the execution of security function chaining in low-level networks takes about 0.829 seconds and the one in high-level network takes about 0.663 seconds. Moreover, it can be found that it is different for the security function instantiation process between the red and green procedures. The red one can work in parallel while the green one need to queue. This is because the two SFs are processed by two service nodes in red procedure, which are instantiated by one service node in green procedure.

The provision time is measured to evaluate the agility of our proposed architecture. We test the provision time of three security services in the prototype. Each group of experiments are tested 50 averages and average is calculated. As shown in Figure 7(b), the provision time increases with the number of service functions, distributed between 2-2.5 seconds, which shows our proposed architecture could meet the requirements of agility. Moreover, during the experiment, OpenNSP can flexibly provision service chains with different lengths.

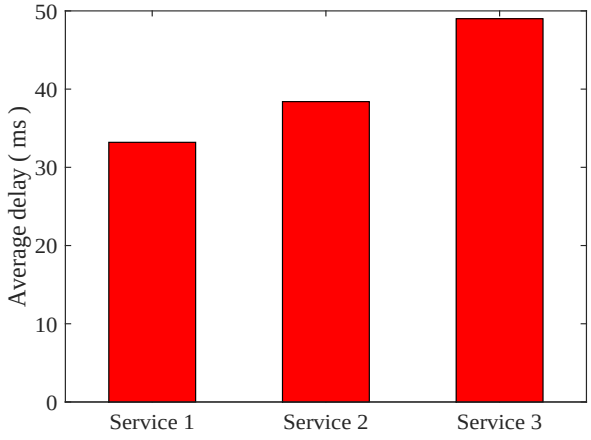


Figure 8: Average delay

We also test the average forwarding delay for different service chains by *Ping* tool. As shown in Figure 8, as the number of service functions in the service chain increases, the forwarding delay increases. And the average forwarding delay changes as the number of SFs increases. Due to the scale of the topology in Figure 5 is limited, the average delay is ranged from 33 ms to 50 ms. This experiment realizes the service chain forwarding and proves OpenNSP framework is usable.

5 Conclusion

In this paper, we propose the OpenNSP, a framework for mobile networks to solve the multi-domain problems of service function chain outsourcing. We also implement the OpenNSP prototype system and conducted several experiments through three use-cases about security services. The experiments confirm the feasibility and advantages of the OpenNSP. For example, the provision time of different services increases with the number of their service functions, distributed between 2-2.5 seconds, meeting the flexibility requirements. In the future, we will focus on standard communication protocols for interactions of different orchestrates, and integrate SFC optimization algorithm in our solution to optimize service performance such as end-to-end delay.

Acknowledgments This paper is supported by NSAF under Grant No. U1530118, National High Technology of China ("863 program") under Grant No. 2015AA015702, NSFC under Grant No. 61602030, National Basic Research Program of China ("973 program") under Grant No. 2013CB329101 and Fundamental Research Funds for the Central Universities of China (No.2017YJS013).

References

[1] S. Abdelwahab, B. Hamdaoui, M. Guizani, and T. Znati. Network function virtualization in 5g. *IEEE Communications Magazine*, 54(4):84–91, Aprill 2016.

[2] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang. What will 5g be? *IEEE Journal on selected areas in communications*, 32(6):1065–1082, June 2014.

[3] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, and H. A. Chan. Optimal virtual network function placement in multi-cloud service function chaining architecture. *Computer Communications*, 102:1–16, April 2017.

[4] R. Bifulco, A. Matsiuk, and A. Silvestro. Catenae: A scalable service function chaining system for legacy mobile networks. *International Journal of Network Management*, 27(2):1–16, April 2017.

[5] H. Chen, S. Xu, X. Wang, Y. Zhao, K. Li, Y. Wang, W. Wang, et al. Towards optimal outsourcing of service function chain across multiple clouds. In *Proc. of the 2016 IEEE International Conference on Communications (ICC'16), Kuala Lumpur, Malaysia*, pages 1–7. IEEE, July 2016.

[6] W. Ding, W. Qi, J. Wang, and B. Chen. Openscaas: an open service chain as a service platform toward the integration of sdn and nfv. *IEEE Network*, 29(3):30–35, May 2015.

[7] D. Dolson and D. Lopez. Hierarchical service function chaining. Technical report, April 2017. `https://tools.ietf.org/html/draft-ietf-sfc-hierarchical-04` [Online; Accessed on October 3, 2017].

[8] W. Haeffner and J. Napper. Service function chaining use cases in mobile networks. Technical report, April 2017. `https://tools.ietf.org/html/draft-ietf-sfc-use-case-mobility-07` [Online; Accessed on October 3, 2017].

[9] D. Inc. Docker, 2017. `https://www.docker.com` [Online; Accessed on October 3, 2017].

[10] G. Li, H. Zhou, B. Feng, G. Li, T. Li, Q. Xu, and W. Quan. Fuzzy theory based security service chaining for sustainable mobile-edge computing. *Mobile Information Systems*, 2017:13, April 2017.

[11] T. Li, H. Zhou, and H. Luo. A new method for providing network services: Service function chain. *Optical Switching and Networking*, 26:60–68, November 2017.

[12] ovs nsh patches. Opendaylight service function chaining, 2017. `https://wiki.opendaylight.org/view/Service_Function_Chaining:Main/` [Online; Accessed on October 3, 2017].

[13] ovs nsh patches. ovs nsh patches, 2017. `https://github.com/yyang13/ovs_nsh_patches/` [Online; Accessed on October 3, 2017].

[14] D. Swarm. Docker swarm, 2017. `https://github.com/docker/swarm` [Online; Accessed on October 3, 2017].

[15] T. Taleb, A. Ksentini, M. Chen, and R. Jantti. Coping with emerging mobile social media applications through dynamic service function chaining. *IEEE Transactions on Wireless Communications*, 15(2):2859–2871, April 2016.

[16] O. vSwitch. Open vswitchr, 2017. `http://openvswitch.org/` [Online; Accessed on October 3, 2017].

[17] A.-V. Vu and Y. Kim. An implementation of hierarchical service function chaining using opendaylight platform. In *Proc. of the 2016 IEEE NetSoft Conference and Workshops (NetSoft'16), Seoul, South Korea*, pages 411–416. IEEE, July 2016.

[18] A.-V. Vu and Y. Kim. Novel core network architecture for 5g based on mobile service chaining. In *Proc. of the 8th International Conference on Mobile Networks and Management (MONAMI'16), Abu Dhabi, United Arab Emirates*, pages 44–57. Springer, October 2016.

Author Biography

Qi Xu received the B.S. degree in telecommunications engineering from Beijing Jiaotong University in 2014, and then entered National Engineering Lab for Next Generation Internet Interconnection Devices at BJTU. Currently, he is pursuing the Ph.D. degree in telecommunications and information system at BJTU, China. His research interests include next generation internet, service function chain, software defined networking, delay tolerant networking and network function virtualization.

Deyun Gao received BEng and MEng degrees in electrical engineering and a PhD degree in computer science from Tianjin University, China, in 1994, 1999, and 2002, respectively. He spent one year as a research associate with the Department of Electrical and Electronic Engineering, Hong Kong University of Science and Technology, Kowloon. He then spent three years as a research fellow in the School of Computer Engineering, Nanyang Technological University, Singapore. In 2007, he joined the faculty of Beijing Jiaotong University as an associate professor of School of Electronics and Information Engineering and was promoted to a full professor in 2012. In 2014, he was a visiting scholar in University of California at Berkeley, USA. His research interests are in the area of Internet of Things, vehicular networks and next-generation Internet.

Huachun Zhou received the B.S. degree from the People's Police Officer University of China in 1986. He received the M.S. in telecommunication automation and Ph.D. degrees in telecommunications and information system from Beijing Jiaotong University in 1989 and 2008, respectively. In October 1994, he joined Institute of Automation Systems, BJTU, where he is a lecturer. From Apr. 1999 - Sep. 2009, he was a senior engineer at School of Electronic and Information Engineering, BJTU, and at Network Management Research Center, BJTU. From Oct. 2009 to now, he is a professor in National Engineering Lab for Next Generation Internet Interconnection Devices at BJTU. He has authored more than 40 peer-reviewed papers and he is the holder of 17 patents. His main research interests are in the area of mobility management, mobile and secure computing, routing protocols, network management and satellite network.

Bohao Feng received his B.S. degree in telecommunications engineering from Beijing Jiaotong University in 2011, and then entered National Engineering Lab for Next Generation Internet Interconnection Devices at BJTU. He is currently working toward his Ph.D degree in telecommunications and information system at BJTU, China. His research interests are ID/Loc Split network architecture, Software Defined Networking, Information-Centric Networking, network-based caching mechanism, Delay Tolerant Networking, mobile Internet and multicast

Taixin Li received the B.S. degree in telecommunications engineering from Beijing Jiaotong University in 2013, and then entered National Engineering Lab for Next Generation Internet Interconnection Devices at BJTU. Currently, he is pursuing the Ph.D. degree in telecommunications and information system at BJTU, China. His research interests include next generation internet, service function chain, software defined networking, delay tolerant networking and satellite networking.