Towards Traffic-Driven VNF Scaling: A Preliminary Case Study based on Container

Guanglei Li[1]*, Huachun Zhou[1], Guanwen Li[1], Bohao Feng[1], and Hyo-Beom Ahn[2]
[1]*Beijing Jiaotong University, Beijing, 100044 China*
{guangleili, hchzhou, guanwenli, bohaofeng}@bjtu.edu.cn
[2]*Kongju National University, Republic of Korea*
hbahn@kongju.ac.kr

Abstract

In Network Function Virtualization (NFV), Virtualized Network Function (VNF) scaling is one of the key lifecycle management operations to accommodate the traffic fluctuation. Compared with a reactive scaling approach based on the load threshold, proactive traffic load prediction can drive the VNF scaling ahead of time and avoid VNF states movement by only redirecting new coming flows. However, most existing online learning research is based on presumed VNF capacity or utilizes a server cluster with high cost and heavy foot-print. To provide the environment for online learning-based VNF scaling research, based on Docker container, we build a lightweight platform on a general personal computer (PC), which supports real word traffic replay and fine-grained resource allocation. Our preliminary case study evaluates the capacity of Snort-based IDS with one CPU core and a half of CPU core under different traffic replay speeds. The experiment results verify that the CPU consummation level rises with the increase of replay speed and the overhead causes packet loss and missing alerts of threads. Besides, under the same replay speed, the CPU consummation level fluctuates with traffic condition. The preliminary case study demonstrates that the container-based platform can provide the basis for online traffic-driven VNF scaling research.

Keywords: SDN, NFV, VNF scaling, container

1 Introduction

Software Defined Networking (SDN) and Network Function Virtualization (NFV) are considered as key technologies for future networks, such as 5G slicing [1–3], where resources are allocated to each tenant, slice or user elastically according to demand. NFV technologies utilize commercial off-the-shelf hardware platforms for network service function provision, where network functions are virtualized and share common infrastructure resources. SDN technologies, which config and control the forwarding plane in a logical centralized way, enables flexible traffic steering to chain network functions. Virtualized Network Function (VNF) scaling is a key lifecycle management operation in NFV [4] and allows network operators resize network services with the change of load. When the traffic volume is small, VNFs should be scaled down to save resources, and when traffic volume is large, VNFs should be scaled up and a part of traffic should be migrated to new network function instances.

There are have been efforts on the VNF scaling issue and focusing on minimizing the negative impact of flow migration of a reactive scaling operation [5,6]. In fact, in many scenarios, network traffic volume changes in a predictable way. For example, in the campus and house network, the change of network traffic is related to users' working and resting periods. The network traffic is also related to other factors

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), 9:1 (Sept. 2018), pp. 1-10
*Corresponding author: School of Electronic and Information Engineering, Beijing Jiaotong University, No.3 Shangyuan-cun, Haidian District, Beijing, 100044, China, Tel: +86-188-1038-8516

such as live events and breaking news. Thus, the research on traffic prediction and load estimation, such as machine learning-based approach [7, 8], have attracted much attention. However, most related works do not discuss and analyze the capability of specific VNF and only conduct simulation based on arbitrary parameters. Although some researchers test their algorithm based on a server cluster, they do not either give the details of each VNF, and it is usually difficult and costly to build a cluster for high-speed traffic replay.

To provide the basis and conveniences for the research of traffic-driven VNF scaling, a lightweight testbed is needed to support algorithm design and verification. For example, the Snort [9], as a popular Instruction Detection System (IDS), needs to be evaluated how much traffic it can process without packet loss if set with specific detection rules and given one CPU core, which is the most CPU resource a single thread can utilize. Besides, the network operator also needs to know how many resources should be allocated to the Snort instance for cost-efficiency if the predicted traffic volume size is small.

In this paper, utilizing a general personal computer (PC) and Docker container [10], we build a lightweight platform for VNF scaling research and deploy the Snort-based IDS function for a case study. The trace replay tool tcpreplay [11] is used to generate traffic with different speed. The OpenvSwitch (OVS) [12] is used for container networking and traffic steering. Our experiment evaluates the capability of the Snort when it is given one CPU core and a half of CPU core (50% cycles of a CPU core). We record the CPU utilization, packets loss and reported alerts under different traffic speed. Our preliminary case study demonstrates that the container-based platform can provide the basis for parameter setting of VNF scaling algorithm design and simulation. Besides, it is also suitable for online learning algorithm research, as it supports real word traffic replay.

The rest of this paper is organized as follows. Section II presents our motivation and related works. Section III shows an overview of the platform framework for VNF scaling research and describes the case study of Snort-based IDS. Section IV presents the experiment results.

2 Motivation and Related Works

With the help of SDN and NFV, network operators can provide network function services in a form of software and chain these service functions in order effortlessly. In the virtualized Service Function Chaining (SFC), the number of VNF instances changes with the network traffic condition and flows are steered dynamically during scaling operation. OpenNF [5] enables the movement of the internal state of VNF by carefully designed APIs. Co-Scaler [6] focuses on the scaling up operation and try to minimize the negative impact of flow migration. However, the scaling operation only depends on the load of the instance and the action is performed reactively when the load reaches the threshold. Online learning and proactive demand prediction have attracted much attention since the reactive approach incurs delay and packet loss [7, 8]. These proactive-based approaches can predict the traffic fluctuation and bring more agility as the scaling decision is driven by traffic condition. However, they do not give the capacity evolution of each specific VNF based on actual measurement.

The design, debugging and testing of a learning algorithm must rely on a test platform. To evaluate the capacity of a VNF and perform scaling actions, the platform must satisfy several requirements: 1) Support high-speed traffic replay based on a trace file to evaluate the ultimate capacity of a VNF and analyze real-world traffic condition. 2) Be able to allocate CPU, memory, and storage resources to VNFs and monitor resources usage with fine granularity. 3) Support lifecycle management of VNFs to enable scaling up and scaling down operation. 4) Have flexible flow steering ability for traffic migration during the scaling operation.

Current popular network emulators and NFV platforms either do not satisfy all above requirements or are too heavy and costly. For example, the Mininet, a popular SDN-enabled emulator, can create a

network of virtual hosts, switches, controllers, and links on a laptop or other PCs with little effort and provides 2Gbps total bandwidth on modest hardware [13]. In general, if the replayed traffic is limited in a host, the total bandwidth can be much higher than the situation where traffic traverses physical links among hosts and switches unless high-speed Ethernet links are used. Thus, the Mininet supports high-speed traffic replay naturally. However, it does not support fine-grained resource allocation to virtual hosts and scaling operation. Other network slicing emulators such as FlowVisor [14] and OpenVirteX [15] have similar drawbacks.

OpenStack, initially designed for creating a private or public cloud [16], now is also combined with SDN technology to build networking testbed [17]. OpenStack is also promising to become the concrete realization of the NFV architecture along with related subprojects [18]. However, it takes much effort to be skillful on all related projects and it is expensive to build a cluster to support high-speed traffic replaying.

Container-based network functions now receive great attention due to its cost efficiency and agility [19]. Thus, we prefer to carry on the VNF scaling research based on the container. Docker engine can allocate the CPU resource to VNFs by limiting a given container's access to the host machine's CPU cycles, which means a general PC can host lots of VNFs with a fine granularity of resource management. Besides, it is simple to use OVS bridge for Docker container networking and that enables flexible flow steering, which is a key capability for VNF scaling, as large traffic need to be split for load balancing during the VNF scaling up operation and small traffic flows need be merged to specific VNFs during the VNF scaling down operation for cost saving.

3 Test Paltform and Case Study

In this section, we first present the overview of the container-based platform and describes each component in detail. Then, we analyze the Snort-based IDS evaluation case.

3.1 Framework

Our preliminary case study mainly focuses on the data plane, but we present the whole framework of the test platform for future online learning research. Figure 1 shows the overview of the framework.

In the data plane, container-based VNFs are connected by the OVS bridge and a specific container is deployed with 'tcpreplay' for traffic replay. The 'tcpreplay' tool can split the trace into two sides: client and server, so two interfaces are attached to the traffic source/sink container. When replaying a traffic trace file, client and server packets are sent out via this two interfaces, respectively. Then, the OVS bridge can steer packets to VNFs in order based on SFC policy. Meanwhile, the Docker engine is responsible for instance management and states collection.

The deployment of above-mentioned components is restricted inside a single host to guarantee high bandwidth. The control plane can be deployed in another host since the control channel bandwidth should be relatively low. The SDN controller is responsible for traffic state collection and flow migration during scaling operation. The VNF manager calls the Docker client Command-Line Interface (CLI) to collect VNF load and perform instance scaling action. The learning agent should be on top of the SDN controller and VNF manager. It relays on the SDN controller and VNF manager to collect data and transfer flow migration and scaling decisions.

3.2 Snort-based IDS Case Study

Snort is one of the most popular open source IDS software, which can also work as a sniffer and packet logger. The intrusion detection model usually consumes more CPU and memory resources. The resource

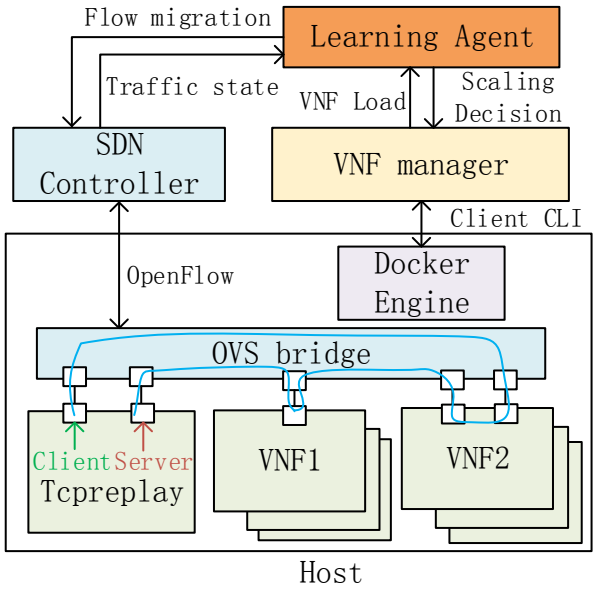


Figure 1: Overview

consummation level is also related to traffic bandwidth and the rules used. Thus, it is necessary to figure out how much traffic the Snort can deal with under specific rules and given resource. That will provide the basis for online resource allocation decision and proactive scaling operation. Based on our platform, the learning agent can learn this information based on real-time traffic replay and states collection.

In this case study, we use the bigFlow.pacp trace file provided by the 'tcpreplay' site for testing. The trace data is captured on an access point to the Internet of a private network in 5 minutes, which contains 791,615 packets with 40686 flows and 132 applications [20]. Figure 2 shows how traffic is steered into the Snort-based IDS function. As the Snort is not configured with a firewall like iptables to prevent packets, only one interface is attached to the Snort container. To direct packets into the Snort container, several flow rules are configured in the OVS flow table to modify the destination MAC address of original frames.

In this case study, we only care about CPU resources constrain and the maximal bandwidth the Snort can process. To evaluate its performance, we replay the trace with different speed and monitor the CPU utilization of Snort instances, which are configured with different CPU resource constraint. After packets are sent out, we check whether the Snort has detected all packets and miss any alerts, as that is usually most concerned for a security service.

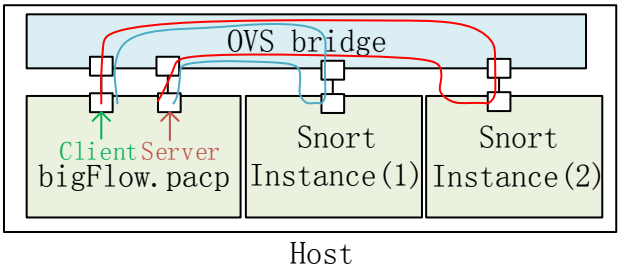


Figure 2: Snort capacity evaluation

4 Experiment Results

The experiment is on a general PC with following configuration: Intel(R) Core(TM) i5-2400S CPU @ 2.50GHz (4 cores), 1 x 8GB DDR3-1333 RAM and Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-101-generic x86_64) OS. We conduct the experiment with two Snort instances, which are configured with one CPU core and a half of CPU core, respectively. We deploy the Snort-2.9.7, which only supports a single thread, in the container running Ubuntu 16.04 system and perform the intrusion detection with default rule set.

4.1 Performance Evaluation with one CPU Core

We first evaluate the performance of the Sort instance with one CPU core. The traffic trace is replayed at speed of 150 Mbps to 210 Mbps, with the interval of 10 Mbps. Figure 3 shows the dropped packets and produced alerts under different bandwidth. The Snort instance starts to drop packets at the speed of 180 Mbps and the number of alerts also decreases. With the increase of replay speed, more packets and alerts are missed. When the bandwidth is 180Mbps, the number of dropped packets is 1256 and the number of alerts decreases from 11952 to 11923. When the bandwidth is 210Mbps, the number of dropped packets increases to 16905 and the number of alerts decreases to 11653.

Figure 4 presents the CPU utilization during the experiment. As we replay the trace with high speed, the experiment time is less than 20s, but it still presents the traffic fluctuations. Figure 4 (a) shows that when the replay bandwidth is 150 Mbps, 160 Mbps, and 170 Mbps, except the Snort initialization phase, the CPU utilization is always less than 100% and their peak value is between 80% and 90%. Figure 4 (b) shows the that when bandwidth is larger, the peak CPU utilization value reaches the limit (one core) and that leads to packet loss and missing alerts of threats.

If a learning agent has learned the capacity of Snort with one CPU core and can predict the traffic bandwidth, it can drive the scaling up operation in a proactive way once the predicted traffic volume is too large, including starting new VNF instances and steering new coming flows to them.

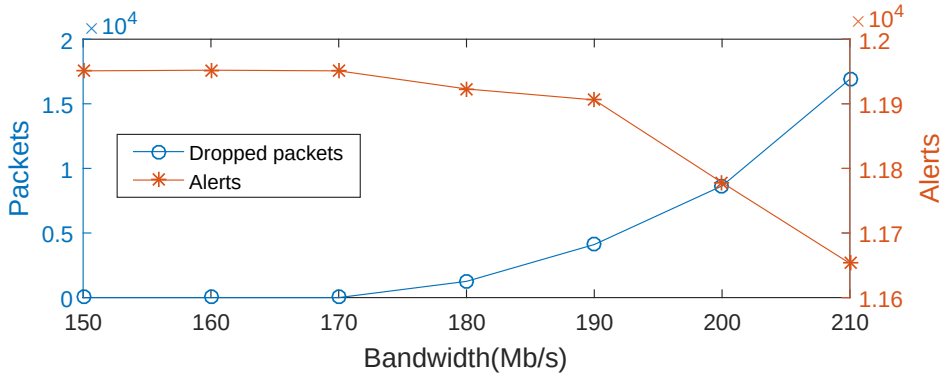
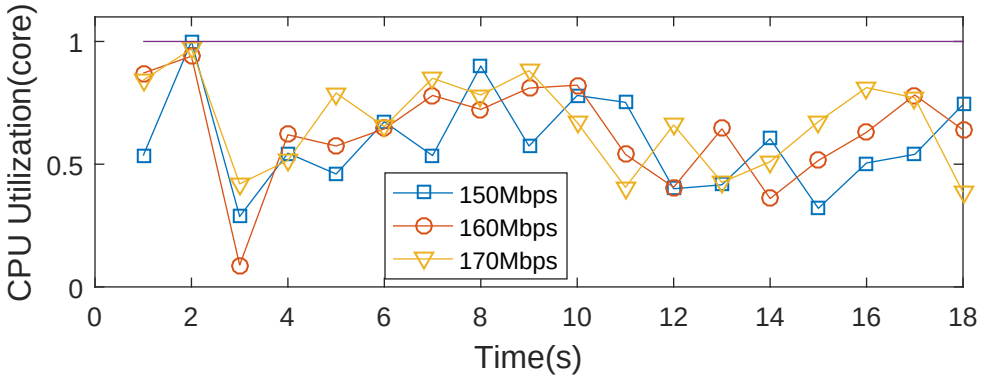


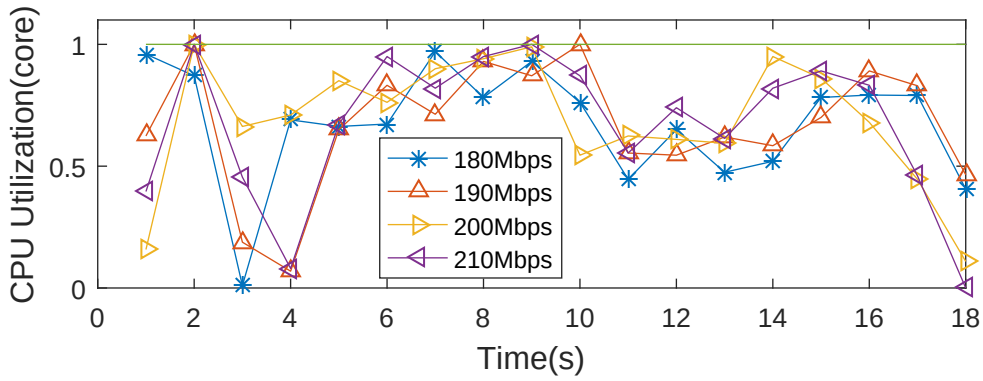
Figure 3: The number of dropped packets and produced alerts under different replay bandwidths with one CPU core

4.2 Performance Evaluation with a half of CPU Core

In the second experiment, we update the flow rules in the OVS bridge, then new coming packets are steered into the second Snort instance configured with a half of CPU core. The traffic trace is replayed at a lower speed, which is from 60 Mbps to 120 Mbps, with the interval of 10 Mbps. Figure 5 shows the



(a) Replay at the speed of 150 Mbps, 160 Mbps, and 170 Mbps



(b) Replay at the speed of 180 Mbps, 190 Mbps, and 200 Mbps and 210 Mbps

Figure 4: CPU utilization during the replay with a half of CPU core

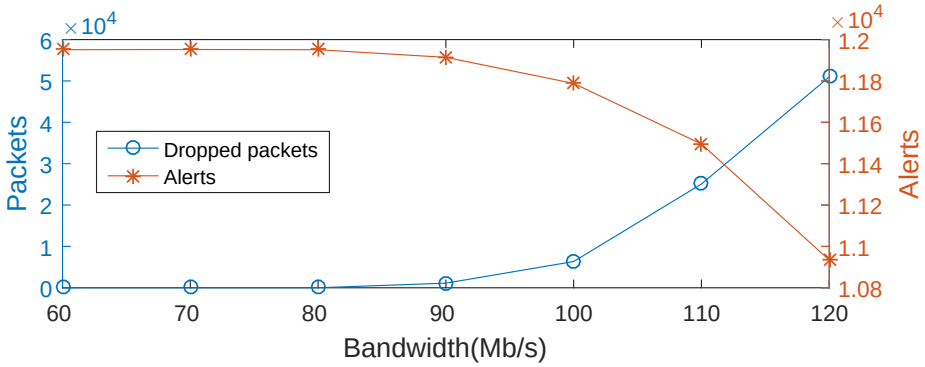
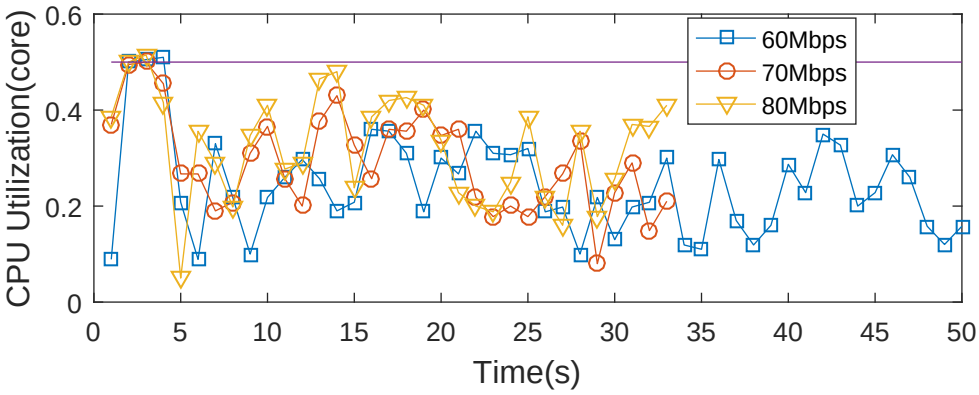
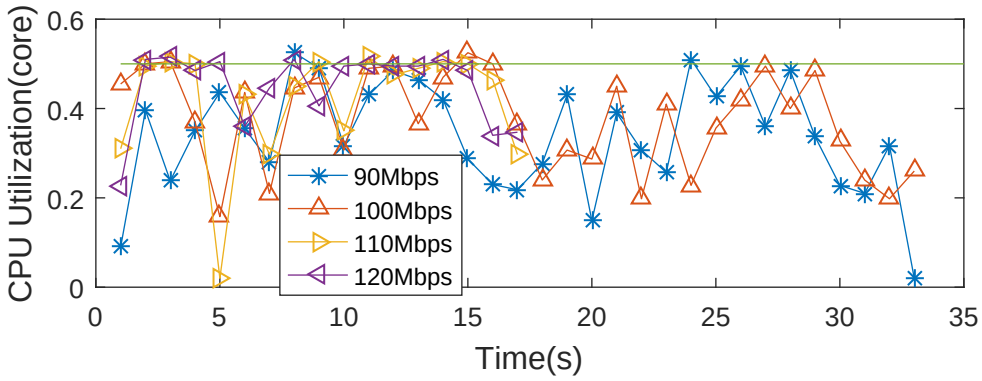


Figure 5: The number of dropped packets and produced alerts under different replay bandwidths with one CPU core



(a) Replay at the speed of 60 Mbps, 70 Mbps, and 80 Mbps



(b) Replay at the speed of 90 Mbps, 100 Mbps, and 110 Mbps and 120 Mbps

Figure 6: CPU utilization during the replay with a half of CPU core

dropped packets and produced alerts. When the bandwidth increases to 90 Mbps, the Snort starts to drop packets and miss alerts. The number of dropped packets is 1115 and the number of alerts decreases to 11914. When the bandwidth increases to 120 Mbps, the number of dropped packets increase to 50971 and the number of alerts decreases to 10934.

Figure 6 presents the CPU utilization during the experiment, which is similar to the results in the first experiment. Figure 6 (a) presents that when the replay bandwidth is less than 80 Mbps, the peak CPU utilization value is less than 50%. The peak value is between 30% and 50%. Figure 6 (b) shows that when the bandwidth increases to 90 Mbps, the peak CPU utilization value reaches the limit (50% cycles of a CPU core) and that leads to packet loss and missing alerts of threats.

In a scaling operation, the resources given to a new VNF instance can be decided based on such information. If the predicted bandwidth is less than 80 Mbps, then the learning agent can allocate no more than a half of core to the instance.

5 Conclusion and Future Work

NFV provides network service functions in a form of software and VNFs should scale based on traffic load to guarantee service quality or save resources. Compared with load threshold-based reactive approaches, proactive approaches try to predict traffic load based on online learning and perform scaling operations in advance, which brings more agility and avoids the state copying of VNFs. Based on the

Docker container, we build a lightweight test platform to support traffic-driven VNF scaling research, where a Snort-based IDS case study is presented by replaying a real word traffic trace under different speeds. Our case study experiments verify that the CPU consummation level fluctuates with traffic under a fixed replay speed, and the higher the replay the speed is, the more CPU resources the Snort needs. In our future work, we will focus on the control plane and enable online learning-based scaling operation.

Acknowledgments

This work is supported by NSAF under Grant No. U1530118, NSFC under Grant No. 61232017 and Fundamental Research Funds for the Central Universities under Grant No. 2018RC003.

References

[1] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5g: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, May 2017.

[2] A. Kaloxylos, "A survey and an analysis of network slicing in 5g networks," *IEEE Communications Standards Magazine*, vol. 2, no. 1, pp. 60–65, April 2018.

[3] N. Zhang, Y.-F. Liu, H. Farmanbar, T.-H. Chang, M. Hong, and Z.-Q. Luo, "Network slicing for service-oriented networks under resource constraints," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2512–2521, October 2017.

[4] O. Adamuz-Hinojosa, J. Ordonez-Lucena, P. Ameigeiras, J. J. Ramos-Munoz, D. Lopez, and J. Folgueira, "Automated network service scaling in nfv: Concepts, mechanisms and scaling workflow," *arXiv preprint arXiv:1804.09089*, 2018.

[5] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "Opennf: Enabling innovation in network function control," *Proc. of the 2014 ACM SIGCOMM Computer Communication Review (SIGCOMM'14)*, vol. 44, no. 4, pp. 163–174, October 2014.

[6] B. Zhang, P. Zhang, Y. Zhao, Y. Wang, X. Luo, and Y. Jin, "Co-scaler: Cooperative scaling of software-defined nfv service function chain," in *Proc. of the 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN'16), Palo Alto, California, USA*. IEEE, May 2016, pp. 33–38.

[7] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive vnf scaling and flow routing with proactive demand prediction," in *Proc. of the 2018 IEEE Internationalk Conference on Computer Communications (INFOCOM'18), Honolulu, Hawaii, USA*. IEEE, April 2018, pp. 486–494.

[8] X. Wang, C. Wu, F. Le, and F. C. Lau, "Online Learning-Assisted VNF Service Chain Scaling with Network Uncertainties," in *Proc. of the 2017 IEEE 10th International Conference on Cloud Computing (CLOUD'17), Honolulu, California, USA*. IEEE, June 2017, pp. 205–213.

[9] C. and/or its affiliates, "Snort-network intrusion detection & prevention system," https://www.snort.org/ [Online; accessed on May 3, 2018], 2018.

[10] D. Inc., "What is a container," https://www.docker.com/what-container [Online; accessed on June 2, 2018], 2018.

[11] F. Klassen and AppNeta, "Tcpreplay - pcap editing and replaying utilities," https://tcpreplay.appneta.com/ [Online; accessed on May 5, 2018], 2018.

[12] L. F. C. Project, "Production quality, multilayer open virtual switch," https://www.openvswitch.org/ [Online; accessed on May 4, 2018], 2018.

[13] M. Team, "Mininet overview," http://mininet.org/overview/ [Online; accessed on May 3, 2018], 2018.

[14] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," OpenFlow Switch Consortium, Tech. Rep. OPENFLOW-TR-2009–1, October 2009.

[15] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, W. Snow, G. M. Parulkar *et al.*, "Openvirtex: A network hypervisor." in *Proc. of of the 2014 Open Networking Summit, Santa Clara, California, USA*. USENIX, 2014.

[16] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: toward an open-source solution for cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, October 2012.

[17] T. Huang, F. R. Yu, C. Zhang, J. Liu, J. Zhang, and Y. Liu, "A survey on large-scale software defined networking (sdn) testbeds: Approaches and challenges," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 891–917, November 2016.

[18] R. C. Computing, "Tacker-openstack nfv orchestration," https://wiki.openstack.org/wiki/Tacker [Online; accessed on June 1, 2018], 2018.

[19] R. Cziva and D. P. Pezaros, "Container network functions: bringing NFV to the network edge," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 24–31, June 2017.

[20] F. Klassen and AppNeta, "Sample captures," http://tcpreplay.appneta.com/wiki/captures.html, [Online; accessed on June 1, 2018], 2018.

Author Biography

Guanglei Li received his B.S. degree from Beijing Jiaotong University in June 2015. He is currently working towards the Ph.D. degree in telecommunications and information system at the National Engineering Laboratory for Next Generation Internet Interconnection Devices, Beijing Jiaotong University, Beijing, China. His main research interests are Software Defined Networking, Network Function Virtualization, Service Function Chaining, network security, mobile netwoks, datacenter networks and space networks.

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.

Guanwen Li received his 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. He is currently working towards the Ph.D. degree in telecommunications and information system at BJTU, China. His main research is directed to the architecture of next generation internet, network service management and network security. His other research interests include satellite network and mobile internet.

Bohao Fengreceived his B.S. and Ph.D degrees from Beijing Jiaotong University in 2011 and 2017 respectively. He is currently with the School of Electronic and Information Engineering, Beijing Jiaotong University. His research interests include software-defined networking, network functions virtualization, information-centric networking, service function chaining, 5G, mobile Internet, and satellite networks. He has been served as a TPC for a number of international conferences, such as IEEE ICC, IEEE GLOBECOM, and IEEE CSCN, and a Reviewer for many international journals, such as the IEEE INTERNET OF THINGS, the Journal of Network and Computer Applications, and the IEEE COMMUNICATIONS LETTERS.

Hyo-Beom Ahn received the B.S. in Computer Science and M.S., and Ph.D. in Computer Science and Statistics from Dankook University, Korea in 1992, 1994 and 2002 respectively. Since then, he has been with the Dept. of Information and Telecommunication, Kongju National University, Rep. of KOREA. His main research interests include Computer Networks, Network Security, Smart Grid Security and Application and Industrial control system security.