A Web-based Orchestrator for Dynamic Service Function Chaining Development with Kubernetes

Ziqiang Wang
Department of Computer and System Engineering
Carleton University
Ottawa, Canada
ziqiangwang@cmail.carleton.ca

Abdullah Bittar
Department of Computer and System Engineering
Carleton University
Ottawa, Canada
abdullahbittar@cmail.carleton.ca

Changcheng Huang
Department of Computer and System Engineering
Carleton University
Ottawa, Canada
huang@sce.carleton.ca

Chung-Hong Lung
Department of Computer and System Engineering
Carleton University
Ottawa, Canada
chlung@sce.carleton.ca

Gauravdeep Shami
External Research, Office of CTO
Ciena Corporation
Ottawa, Canada
gshami@ciena.com

2022 IEEE 8th International Conference on Network Softwarization (NetSoft) | 978-1-6654-0694-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/NetSoft54395.2022.9844086

***Abstract*— The research community has been moving attention from Virtual Network Function (VNF) to Cloud-native Network Function (CNF) since cloudification has brought the Network Function Virtualization (NFV) to an advanced level. It has already been demonstrated that cloud-native technology brings high flexibility and efficiency to large-scale network service deployment compared to the traditional VNF with Virtual Machines (VMs). However, more work is needed to provide a flexible and reliable Service Function Chaining (SFC) development solution in a cloud-native environment. This paper proposes a web-based orchestrator system to deploy an SFC use case consisting of multiple CNFs in a multi-node Kubernetes cluster using Network Service Mesh (NSM). We demonstrate a cloud-native SFC framework that allows users to dynamically create container-based SFC rather than the traditional VMs with NFV/SDN controller approach. Further, additional work is presented with the support of an open-source monitoring system, Prometheus, to validate the SFC path.**

Keywords*— *Cloud-native, Service Function Chaining, Network Service Mesh, Kubernetes, Prometheus

I. Introduction

The technology evolution in the cloud-native world also brings innovation to network applications. Currently, network applications are primarily developed based on Network Function Virtualization (NFV). Traditionally, network functions runs in Virtual Machines (VMs) which are managed by a VM orchestrator and Software-defined Networking (SDN) controller such as OpenStack with ONOS [1] [8]. On the other hand, in a cloud-native environment, Virtual Network Functions (VNFs) are deployed in the form of containers running on the cloud platform [2]. This new trend has provided benefits to NFV based on the following facts: 1) containers have less overhead than VMs. VMs run in a hypervisor environment where each VM must have its operating system (OS), along with its related binary code, libraries, and application files which impose high overhead; 2) containers are adaptable and portable. The deployment of a container is independent of OS and hardware platforms.

Furthermore, containers can be configured to form a Service Function Chaining (SFC) which connects a list of networking applications in a specific order to offer network services [3]. Leveraging both SDN and NFV, SFC can automate traffic flow between services and optimize the use of network resources. The cloud-native SFC could leverage containers' agility, flexibility, and scalability. Most importantly, it does not even rely on an SDN controller to manage the application life cycle [1].

In this paper, we demonstrate the benefits of deploying an SFC over Kubernetes using the proposed web-based orchestrator system and ensuring the reliability of the SFC by adopting the state-of-art monitoring solution. The contribution of this paper is twofold:

1. Demonstrate a proof-of-concept prototype that allows users to deploy network function chains using a web interface-based orchestration dynamically.
2. Use the monitoring tool to ensure that the SFC path is reliable by collecting metrics while adding minimum overhead to the system.

The rest of the paper is organized as follows. Section II provides background on key concepts and the distinguishing aspects concerning related studies. Section III describes the system prototype and the list of individual items that will be demonstrated. The summary of the innovation is discussed and summarized in Section IV.

II. Background and Related Work

NFV describes the idea of virtualization in terms of network services at both the virtualization layer and the application layer of SDN [4]. The SDN architecture inherently promises advantages such as software and hardware decoupling, flexible network function deployment, and dynamic functioning. Cloud-native Network Function (CNF) is a software implementation of NFV deployed in a cloud-native platform such as Kubernetes. Kubernetes, also known as K8s, is a cloud-based platform that offers a Container-as-a-Service (CaaS) layer for managing containerized workloads and services [5]. A Pod is the smallest deployable unit in the K8s cluster. Combined with other features of K8s, cloud-native SFC offers real-time and dynamic provisioning along with flexible traffic steering, and could benefit from network upgrades.

The authors in [6] proposed a novel traffic steering algorithm to route traffic in cloud-native SFC using a dynamic weighted round-robin algorithm. Also, in [1], the authors offered a Ketama algorithm-based [7] traffic steering to maximize the Quality of Service (QoS) satisfaction rate by load-balancing the traffic over the SFC path using the Contiv-VPP network plug-in. On the other hand, the author in [8] focused on integrating Kubernetes with OpenStack to deploy a complex system that leverages both the advantages of VM and container to build an SFC. The authors in [9] proposed a novel fault management system that can monitor the current SFC deployment and dynamically recover the fault container-based VNF by creating a backup VNF with the same functionality to replace it. In [10], the author noticed that the security and integrity of the container-based SFC are usually ignored compared to the VM-based SFC. Hence, they proposed a framework that can be integrated into NFV controller to detect anomalies using an extreme learning machine (ELM).

Unlike the existing SFC solution provided in [6]-[10] using Contiv-VPP or OVN4NFV plug-in, our approach focuses on dynamic creation of customized SFC across multiple nodes using Network Service Mesh (NSM) and integration with a web-based orchestrator. NSM provides the SFC data plane for steering the traffic based on network policies [11]. The web-based orchestrator automates the SFC developing process while allowing users to choose different container-based microservices and define the routing rules. The system also integrates a resource and network state monitoring solution supported by the Prometheus system [12]. The monitoring framework provides metrics that can optimize network performance and validate the SFC path. Furthermore, previous research works mainly focus on the development of cloud-native SFC but not on network reliability and SFC path validation. To the authors' best knowledge, this work is the first to automate the SFC development process using a web-based orchestrator system in a cloud-native environment practically and concretely.

III. Prototype Model

The graphical interface is a Python-based front-end web server of the orchestrator system connected with both the Kubernetes API to conduct user commands and the Prometheus API for traffic validation. The network of K8s control plane is provided by WeaveNet while the adopted NSM plug-in creates SFC tunnels called Virtual-wires (V-wires) between Pods. The overall system model is depicted in Figure 1. With reference to Figure 1, the system contains several necessary elements of K8s, NSM, and Prometheus.

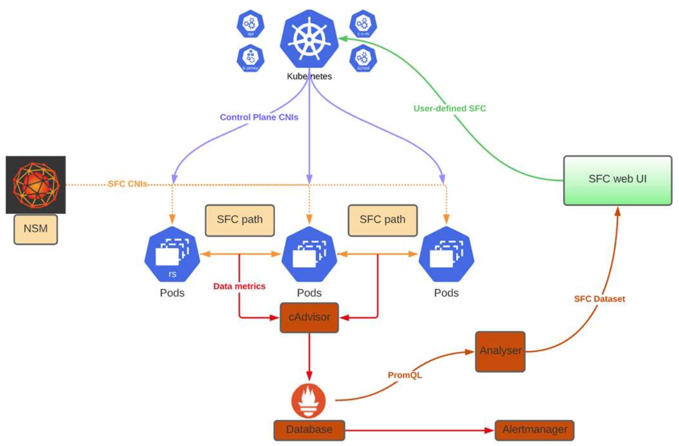

Figure 1 The overall design of the NSM-based SFC with Kubernetes

A new SFC is created by a client dragging and dropping the pre-configured microservices' icons from the network application library in the web UI or using an existing pre-designed SFC. This process automatically creates the corresponding manifest files and deploys SFC components in the K8s cluster. When the NSM manager receives an SFC request from a client, it examines the existence of registered Network Service Endpoints (NSEs). It establishes the chained V-wire connections between the client and NSEs if the required NSEs and the interface mechanism are available in the cluster. New NSEs may get registered during this process to satisfy the SFC creation according to the requested application label. Figure 2 illustrates this entire process with the step numbers. The web UI displays the Pod creation process and network metrics so that the user can clearly understand the states of the cluster.

Meanwhile, the Prometheus monitors the node's hardware resources, QoS and validates the SFC traffic flows. There are a few metrics that imply the performance of SFC, such as 1) the latency for Pods initialization, 2) the network throughput for each network device during the sample period, 3) the CPU and memory utilization, 4) the HTTP request delay and error rate.

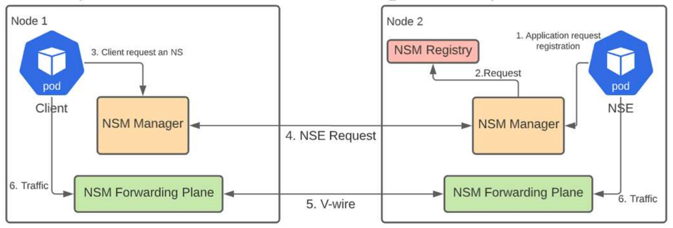

Figure 2 NSM communication flow chart

The SFC demonstrated in this paper consists of 3 NSEs distributed in two cluster nodes. All the NSEs are created from the customized docker image built for this experiment. The NSE_1 located at the first node contains the function of video streaming supported by the Nginx-RTMP and FFmpeg module. It has one Kernel interface that connects with V-wire for sending and receiving videos. The NSE_2 is an ACL (Access

Control List)-based firewall located at the second node. It has two Shared Memory Packet Interfaces (MEMIF), which connect with the Kernel interface in NSE_1 and NSE_3, respectively. The NSE_3 located at the end of the chain has the function of video size reduction which reduces the size of a video using the FFmpeg module to ensure that the user can stream video faster when facing the network bottleneck. In this case, the Network Service Client (NSC) implemented as an Alpine Linux container is integrated with the NSE_1 in the same Pod, making the SFC tightly organized.

Figure 3 demonstrates the system architecture of the video streaming SFC. The NSM-admission-webhook injects the corresponding interfaces into NSC and NSEs. After the SFC is declared and containers are created, V-wires will be made in the forwarder plane using the Vector Packet Processing (VPP) traffic forwarder or the Kernel traffic forwarder. It is worth noting that some of the V-wires have two different types of interfaces at the two ends. An initial request is sent from the NSC located at the first node along with the video that will be compressed at NSE_3 residing at the second node. Theoretically, the traffic passes through NSE_2 and reaches the destination NSE_3. The traffic is steered between two physical nodes and among 3 Pods. Then, after NSE_3 processed the video, the traffic containing the compressed video is steered back from NSE_3 to NSE_1 and passes NSE_2. Hence, a content distribution network that uses cloud-native SFC to distribute videos is achieved via NSE_2. Meanwhile, the entire workload will be registered with the Spire agent/server to ensure that the SFC path is not compromised. Simultaneously, the Prometheus system will monitor the traffic, such as the latency of requests, the network throughput, node CPU and memory utilization, etc. The collected dataset is analyzed to validate the SFC traffic path.

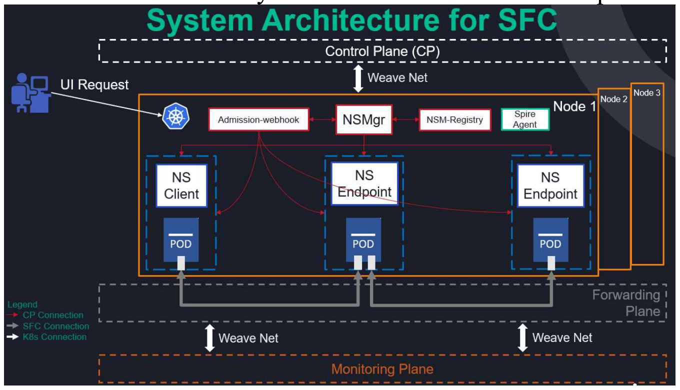

Figure 3 The Overall System Architecture of NSM-based SFC

This section demonstrates an emulated content distribution network using the proposed web-based SFC orchestrator system. Inclusion, the following list contains the individual items that will be demonstrated.

- A user web interface where the user can configure network applications and SFC. It also displays the collected metrics to validate the SFC path.
- An SFC consists of 3 individual CNFs deployed across two different physical servers.
- Multi-node K8s cluster with various applications such as WeaveNet, Spire, NSM, and Prometheus.

IV. SUMMARY OF INNOVATION

SFC plays a vital role in the next-generation networks by benefiting different technologies such as 5G, IoT, and edge computing. However, cloud-native SFC is still a novel approach in industry and academia. Traditionally, developing SFC is usually a time-consuming task that involves linking various components together and configuring them individually to ensure the functionality. With the help of a graphical interface that facilitates the user to perform otherwise complex operations, developers can focus on fostering new algorithms for container-based service chaining.

The SFC, by itself, cannot provide any insight into the system, especially in terms of resource usage trends and network traffic flows. Developers are not able to know how the SFC deployment occupies the hardware resources and where the traffic is dedicated. The reliability of the SFC may also be at risk considering the security level of the container. More importantly, any further improvement requires feedback to provide the capacity and ability of the system from the central and historical view. The proposed platform takes advantage of state-of-the-art monitoring systems that help control the performance and reliability of the deployed SFCs, which allows the operator to adopt suitable tasks.

REFERENCES

[1] A. Bouridah, I. Fajjari, N. Aitsaadi and H. Belhadef, "Optimized Scalable SFC Traffic Steering Scheme for Cloud Native based Applications," Proc. of IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), 2021, pp. 1-6.

[2] Pantheon.tech, "Cloud-Native Network Functions," CDNF, https://cdnf.io. (accessed August. 9, 2021)

[3] J. M. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," RFC 7665, Oct. 2015.

[4] Y. Li and M. Chen, "Software-Defined Network Function Virtualization: A Survey," IEEE Access, vol. 3, pp. 2542-2553, 2015.

[5] T. K. Authors, "What is Kubernetes," The Linux Foundation, https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/. (accessed May. 24, 2021).

[6] B. Dab, I. Fajjari, M. Rohon, C. Auboin and A. Diquélou, "An Efficient Traffic Steering for Cloud-Native Service Function Chaining," Proc. of 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), 2020, pp. 71-78.

[7] libketama: Consistent Hashing library for memcached clients. [Online]. Available: https://www.metabrew.com/article/libketamaconsistent-hashing-algo-memcached-clients

[8] H. R. Kouchaksaraei and H. Karl, "Service Function Chaining Across Openstack and Kubernetes Domains," Pro. of the 13th ACM International Conference on Distributed and Event-based Systems, pp. 240–243.

[9] S. -Y. Song and F. J. Lin, "Dynamic Fault Management in Service Function Chaining," 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), 2020, pp. 1477-1482.

[10] S. -T. Cheng, C. -Y. Zhu, C. -W. Hsu and J. -S. Shih, "The Anomaly Detection Mechanism Using Extreme Learning Machine for Service Function Chaining," Prof. of International Computer Symposium (ICS), 2020, pp. 310-315.

[11] The Network Service Mesh authors, "Architecture", networkservicemesh.io. (accessed 2022)

[12] YunlZhang, "Best Practices: 4 Golden Indicators and the USE Method" in *Prometheus-book* (accessed Jan. 14, 2021)