Poster: A Lightweight Network Slicing Orchestration Architecture

Biyi Li, Bo Cheng, Meng Wang, Meng Niu, Junliang Chen
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications, Beijing, China

CCS CONCEPTS

• **Networks** → *Programming interfaces*; *Network management*; *Mobile networks.*

ACM Reference Format:
Biyi Li, Bo Cheng, Meng Wang, Meng Niu, Junliang Chen. 2019. Poster: A Lightweight Network Slicing Orchestration Architecture. In *The 17th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '19), June 17–21, 2019, Seoul, Republic of Korea.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3307334.3328643

1 INTRODUCTION

With the explosive growth of large scale services, traditional mobile networks have become increasingly unable to guarantee the efficient operation of services. However in the fifth generation (5G), supported by Software Defined Network (SDN) and Network Function Virtualization (NFV), network slicing technology [1] makes mobile networks more intelligent and flexible. 5G network slicing allows a set of logically independent virtual networks to be created on a common physical infrastructure and provides appropriate monitoring, management and resource allocation for a variety of different types of communication services [2].

However, there is still a lack of an efficient network slicing orchestration system to reduce the complexity of slices deployment for users [3]. Most of the existing slicing orchestration systems build Virtual Network Functions (VNFs) in the form of virtual machines, which have the disadvantages of slow startup and large memory occupation. And, a relatively discrete distribution of virtualized resources requires a lightweight management interface. Secondly, the exiting slices are mostly specific and difficult to expand and reuse. In addition, the deployment and management of slices often require users to have deep professional knowledge, which is unrealistic. To this end, we propose a lightweight network slicing orchestration architecture. In this poster, our main contributions are as follows:

- We design a network slicing model template based on the extended TOSCA [4]. This template provides a design method for slices, including resource requests, Service Level Agreement (SLA), and priority protocols.

MobiSys '19, June 17–21, 2019, Seoul, Republic of Korea

ACM ISBN 978-1-4503-6661-8/19/06.
https://doi.org/10.1145/3307334.3328643

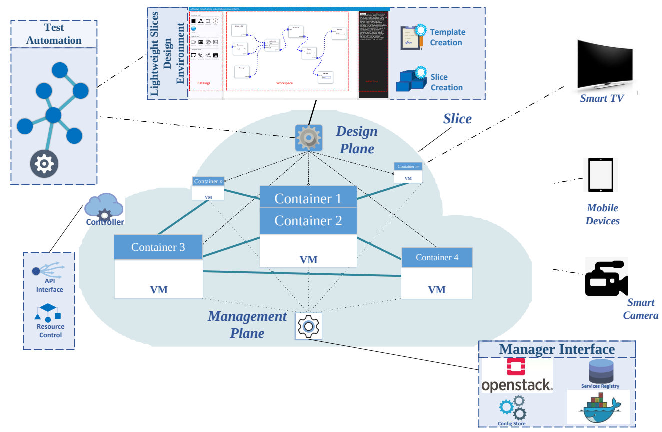

Figure 1: Network Slicing Architecture

- Over more, we implement an easy-to-use design plane for network slicing, where users can encapsulate the common VNFs and build services in the form of drag and drop.
- Based on micro-service management architecture, we make use of VM-based containers for lightweight inter-cluster access control and lifecycle management while preserving the excellent isolation between the different slices.

2 NETWORK SLICING ARCHITECTURE

As shown in Figure 1, the architecture is mainly divided into three planes: design plane, management plane, and control plane. Next, we introduce the architecture in detail.

2.1 Design plane

To reduce the complexity of service construction, we design an easy-to-use interface. And we have encapsulated some common VNFs in advance which users can connect them into the workspace and build a slice with specific demands. Once a service is confirmed to be deployed, the design plane automatically tests the correctness of connection and converts the relevant information into an extended TOSCA template.

As Figure 2 shows, this template provides detailed parameters for service construction, including resource requirements, QoS policies, priority protocols and SLA. We can also add policies at the design plane and perform secondary development for slices in the running state.

2.2 Management plane

The management plane is responsible for the creation, management, and monitoring of slices. Based on the micro-service architecture, we make use of VM-based containers to create the VNFs on the

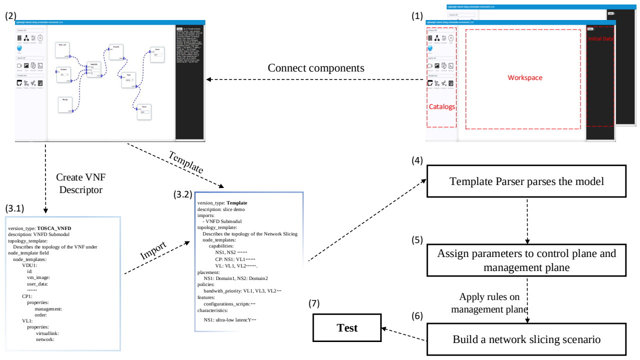

Figure 2: The Process of Network Slicing

slice. Compared with traditional VMs, this method greatly reduces the time of service creation, while container-based services are more lightweight and can efficiently utilize resources. The specific management process is as follows.

- **Template Parser**. At this stage, the parser analyzes templates to extract the necessary information. Then this information will be sent to the orchestrator through design layer unified interface and the corresponding resources are allocated to the generated slices by leveraging virtualization technologies.
- **Detection**. After analyzing the template, the orchestrator applies for a logical resource of a corresponding scale according to the description of the logical resource therein. If the current system can satisfy the minimum guarantee of slicing requirements, the slice will be established according to the mapping rules issued under the control plane.
- **Deployment**. When the above steps are all successfully executed, the pulled VNFs will be deployed in the appropriate virtual machines in the form of containers. This makes services on slices lightweight while preserving the excellent isolation between the different slices.

2.3 Control plane

Learning from the network solution of OpenStack, we introduced OpenDaylight, an open source SDN controller, into the architecture to steer traffic from VNF container clusters and virtual machines. The control layer mainly analyzes two parameters of the template: VNF mapping and QoS policy.

- **VNF mapping**. We integrate a mapping algorithm [5] to the controller. Of course, users can also design the mapping algorithm according to their own model. After calculating the optimal mapping, management plane orchestrator builds the slice with reference to the results uploaded by the control layer.
- **QoS policy**. As mentioned above, each user has inconsistent priority requirements for the business. Therefore, we generate a QoS parser to divide network functions into multiple priorities by analyzing QoS parameters in templates. When users access to multiple services, the control layer sends a flow table to guide traffic, adjust bandwidth and ensure the normal operation of high-priority service.

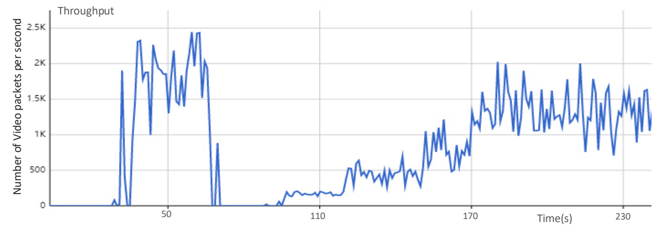

Figure 3: Number of packets per second of video service

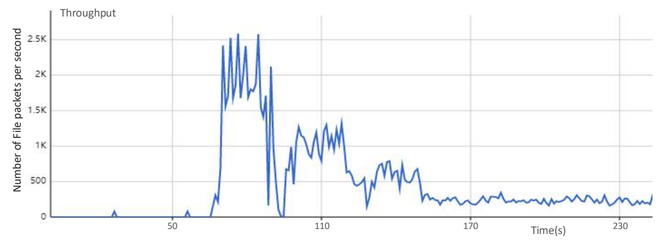

Figure 4: Number of packets per second of database service

3 EVALUTION

Figure 2 shows an illustration of the design plane and some pivotal steps that facilitate users to learn how to generate slicing instances. Our template can also import VNFD standard models [6], which greatly increases the reusability of slices. And building services in a drag-and-drop form at the design plane is a great way to reduce the complexity of template design for users. Next, we evaluate the performance of our architecture by building a sample business scenario.

Figures 3 and 4 show the changes in throughput for the interaction of two services on the same slice. And we judge the change of bandwidth by measuring the number of real-time packets per second. In order to prevent bandwidth preemption of database, we added a detailed QoS hierarchy in the design plane. As shown in these figures, the number of packets per second of video service is increasing while the number of packets per second of database service is decreasing. Finally, both of them are stabilized at an appropriate location.

ACKNOWLEDGMENT

This work was supported in part by the National Key Research and Development Program of China No.2018YFB1003804, in part by the National Natural Science Foundation of China No.61772479.

REFERENCES

[1] NGMN Alliance. Description of Network Slicing Concept. (20), November 2016.
[2] Meng Wang, Bo Cheng, Xuan Liu, Yi Yue, Biyi Li, and Junliang Chen. A SDN/NFV-Based IoT Network Slicing Creation System. *Conference on ACM MOBICOM 2018 Conference*, pages 666–668, 2018.
[3] 3GPP. Study on architecture for next generation system. 2016.
[4] ETSI GS NFV-SOL 001. Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV Descriptores Based on TOSCA Specification. March 2018.
[5] T.W.Kuo, B.H.Liou, K.C.J.Lin, and M.J.Tsai. Deploying chains of virtual network functions: On the relation between link and server usage. *Conference on IEEE INFOCOM 2016 Conference*, 2016.
[6] Tacker - OpenStack NFV Orchestration. https://wiki.openstack.org/wiki/tacker.