Scientific Research Publishing

Communications and Network, 2021, 13, 79-92
https://www.scirp.org/journal/cn
ISSN Online: 1947-3826
ISSN Print: 1949-2421

Evaluation of 5G Core Slicing on User Plane Function

Cheng-Chin Tsai[1], Fuchun Joseph Lin[1*], Hiroshige Tanaka[2]

[1]Department of Computer Science, College of Computer Science, National Yang Ming Chiao Tung University, Taiwan
[2]Okinawa Open Laboratory, NTT Communications Corporation, Okinawa, Japan
Email: tcc.cs07g@nctu.edu.tw, *fjlin@nycu.edu.tw, hiroshige.tanaka@ntt.com

How to cite this paper: Tsai, C.-C., Lin, F.J. and Tanaka, H. (2021) Evaluation of 5G Core Slicing on User Plane Function. *Communications and Network*, **13**, 79-92.
https://doi.org/10.4236/cn.2021.133007

Received: May 11, 2021
Accepted: August 1, 2021
Published: August 4, 2021

Abstract

Network slicing is one of the most important features in 5G which enables a large variety of services with diverse performance requirements by network virtualization. Traditionally, the network can be viewed as a one-size-fits-all slice and its services are bundled with proprietary hardware supported by telecom equipment providers. Now with the network virtualization technology in 5G, open networking software can be deployed flexibly on commodity hardware to offer a multi-slice network where each slice can offer a different set of network services. In this research, we propose a multi-slice 5G core architecture by provisioning its User Plane Functions (UPFs) with different QoS requirements. We compare the performance of such a multi-slice system with that of one-size-fits-all single slice architecture under the same resource assignment. Our research objective is to compare the performance of a network slicing architecture with that of a "one-size-fits-all" architecture and validate that the former can achieve better performance with the same underlying infrastructure. The results validate that our proposed system can achieve better performance by slicing one UPF into three with proper resource allocation.

Keywords

Network Slicing, NFV, 5G, User Plane Function, QoS

1. Introduction

The 5th-Generation (5G) network [1] is going to provide plethora of network services with diverse performance requirements. Unlike the 4G designed for achieving mobile broadband services with the "one-size-fits-all" architecture, 5G is being designed to provide a large variety of vertical services with heterogene-

ous requirements such as Enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-Latency Communications (URLLC) and Massive Machine Type Communications (mMTC) [1]. To fulfill diverse service types over the same network infrastructure, network slicing in 5G is the key enabling concept.

According to 5G.co.uk network slicing is "providing dedicated virtual networks with functionality specific to the service or customer over a common network infrastructure". To achieve network slicing, Network Function Virtualization (NFV) is applied to transform services bundled with proprietary hardware to software-based Virtual Network Functions (VNFs) on the commodity infrastructure [2]. NFV provides a programmable, flexible, and modular network environment to run these VNFs [3] [4] [5]. We would leverage NFV technologies to build our slicing environment.

The 5G core (5GC) is designed to be "cloud native" where NFV is leveraged to create network slices. A 5G core slice is composed of a collection of 5G core VNFs that are chained together to support a specific use case [1]. Though an end-to-end 5G network slice may consist of different network slice subnets including 5G core, access, or transport networks [6], in this research we will focus on 5G core slicing. Also, we will follow one of the major characteristics of 5G, Control and User Plane Separation (CUPS), to decouple a 5G system into two parts and deploy Control Plane (CP) as a commons slice and configure User Plane (UP) into multiple customized slices, each with different bandwidth requirements.

Several related works on the design of 5G core slicing are surveyed below. In [7], potential challenges in 5G core slicing such as slice creation, slice management, and security in network slicing were elaborated. Our system tackles these challenges by leveraging APIs provided by OpenStack and Tacker for slices creation and management. Also, to protect the VNFs in slices our system relies on the security group provided by OpenStack and Tacker.

In [8], the modularization of 5G Core is identified as an important feature since this would allow independent evolution of its modules in the future. Our system thus adopts NCTU free5GC which follows a modularized functional design.

Another important issue related to network slicing is how to guarantee hard isolation between slices [9]. Our system resolves this issue by utilizing the VM-based system architecture where the policy of strict no-resource-sharing between VMs is enforced.

The KPIs for network slicing based on ETSI NFV MANO architectural framework were defined in [10]. On the other hand, the main concepts and principles of network slicing such as the NFV, SDN and cloud technologies are well elaborated in [11]. The above two papers provide a comprehensive overview of network slicing.

Our research objective is to compare the performance of a network slicing architecture with that of a "one-size-fits-all" architecture and validate that the

former can achieve better performance with the same underlying infrastructure. On the proposed system, 5G User Plane Functions (UPFs) would be sliced according to different QoS requirements to optimally support different services. On the compared system, only one UPF will be used to serve different services. Also, both systems will be given the same amount of computing/communication resources.

The rest of the paper is organized as follows: Section II introduces the background of the technologies used in our system and discusses the related work on network slicing. Section III introduces our proposed multi-slices 5G system based on the NFV MANO (MANagement and Orchestration) architecture. Section IV describes the implementation and evaluation of our system and compares our system with the one-size-fits-all single slice system under the same resource allocation. Finally, Section V presents our conclusion and future work.

2. Background

We construct our proposed 5GC slicing system by leveraging several technologies such as 5GC platform, ETSI NFV-MANO architectural framework and MANO-based 5GC slicing. In this section, we explain the background of these techniques and related work.

2.1. 5GC Platform

The 5G system is proposed to support a diverse set of vertical services with different QoS requirements. It is designed as a set of network functions (NFs) and follows CUPS to separate its CP from UP. These NFs are modularized to enable flexible Network Service (NS) development. The 5G CP follows the Service-Based Architecture (SBA) that provides a common interface for inter-communications among NFs. The 5G core architecture consists of eleven NFs with ten in CP while one in UP. Figure 1 shows 5G non-roaming reference architecture as

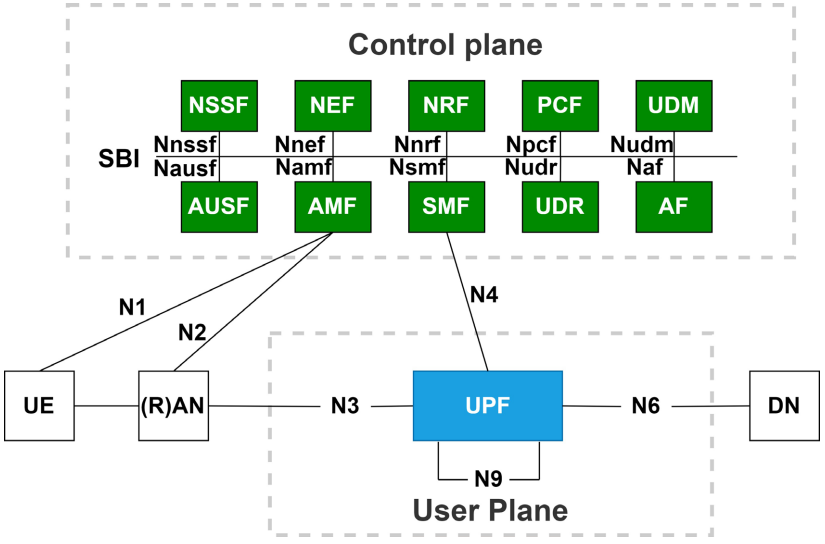

Figure 1. 3GPP 5G non-roaming system reference architecture.

defined by 3GPP [1].

- The ten 5GC CP network functions (NFs) in CP include Access and Mobility Management Function (AMF), Network Repository Function (NRF), Session Management Function (SMF), Unified Data Repository (UDR), Policy Control Function (PCF), Unified Data Management (UDM), Network Slice Selection Function (NSSF), and Authenticate Server Function (AUSF). In addition, 5GC also allows Application Function (AF) which enables service providers to create their own services cooperating with 5GC through the interaction with Network Exposure Function (NEF). The N1 and N2 interfaces support signaling exchange between UE (User Equipment) and AMF and between RAN and AMF, respectively.
- The only 5GC UP network function is User Plane Function (UPF) which is responsible for creating and maintaining PDU (Packet Data Unit) sessions to route and forward data packets to various external Data Networks (DNs). The N3 interface connects RAN and UPF. The N4 interface supports signaling exchange between UP and CP. The N6 interface is the connection to the external DN. The N9 interface supports signaling exchange between UPFs.

5G CUPS makes the deployment of modularized network functions of 5GC more flexible and agile. In our experiment, we aim to leverage the CUPS characteristic in 5G to create event slices [12], defined as network slices created specifically to handle the sudden rise of traffic in special events such as New Year's Eve party. As a result, we would structure our 5GC slicing experimental system with UP as customized event slices and CP as a common slice shared by those event slices.

There are not many 5GC open sources available currently. Most of the mobile network open sources such as nextEPC and OpenAirInterface only provide 4G EPC. Since free5GC [13], an open-source project for 3GPP 5GC Release 15 and beyond, developed by NCTU in Taiwan, provides a complete 5GC architecture, we adopt it to construct our 5GC slicing experimental system.

2.2. ETSI NFV-MANO Architectural Framework

ETSI NFV-MANO [14] is an architectural framework for enabling Network Services (NSs) (and thus network slices) under virtualized infrastructure. Each NS is composed of one or multiple VNFs. These VNFs are connected by Virtual Links (VLs) and their interactions are regulated by VNF Forwarding Graphs (VNFFGs). To deal with different QoS requirements of various vertical services, operators need to provide NSs of different characteristics.

As depicted in **Figure 2**, the ETSI NFV-MANO architecture consists of the following three major function components [14]:

- VNF Manager (VNFM): This is responsible for the lifecycle management of VNFs on the VNF platform, such as creation, modification and termination. Note that each VNF is a network function thus should be coupled with an Element Management (EM) system which is responsible for FCAPS[1] management

[1]FCAPS—Fault, Configuration, Accounting, Performance and Security Management.

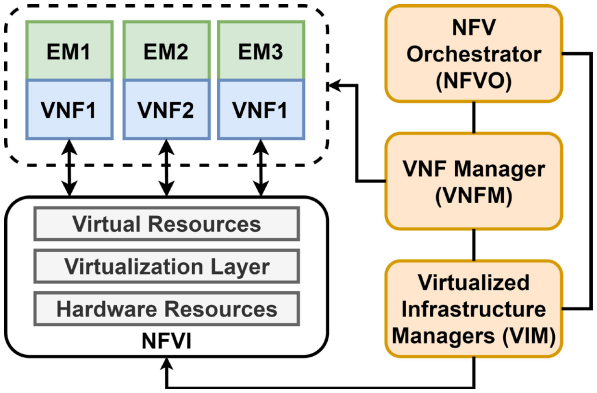

Figure 2. ETSI NFV MANO architecture framework.

functionalities for VNF.

- Virtualized Infrastructure Manager (VIM): This is responsible for managing the NFVI resources including compute, storage and network. These resources are transformed from hardware resources to virtual resources via a virtualization layer. There are two types of VIMs: Virtual Machines (VM) enabled by OpenStack and containers enabled by Kubernetes.
- NFV Orchestrator (NFVO): This is responsible for the orchestration of NFV Infrastructure (NFVI) resources, the lifecycle management of NSs and the repository and management of deployment templates for NSs such as Network Service Descriptor (NSD) and Virtual Network Function Descriptor (VNFD).

MANO provides a framework to manage network slices. There are several MANO open sources available such as Tacker, OSM, OpenBaton, ONAP. We would leverage Tacker [15] as our NFVO/VNFM and OpenStack [16] as our VIM in our experiment.

- OpenStack is an open-source cloud operating system for virtualizing and managing resources including compute, network and storage. It provides multiple managing services such as compute (NOVA), network (Neutron), storage (Cinder), orchestration (Heat) and user interface (Horizon). We leverage Open vSwitch to create our experimental networks where switch virtualization techniques are used to manage the bandwidth limitation for slicing.
- Tacker is an OpenStack project that provides the functionalities of NFVO and VNFM to deploy and manage NSs and VNFs on either OpenStack or Kubernetes-based NFVI. Tacker is compliant to ETSI MANO Architectural Framework but requires some enhancements to smoothly orchestrate end-to-end NSs based on VNFs.

2.3. MANO-Based 5GC Slicing

According to 3GPP, a network slice is an end-to-end network architecture consisting of multiple network slice subnets. Each network slice subnet represents a different segment in an end-to-end network such as access network, core network, transport network. Each network function can be deployed as VNF or

Physical Network Function (PNF). NFV MANO, as the key enabling technology for network services, maps a 3GPP-defined network slice/network slice subnet to an NS. Figure 3 shows the mapping between a 3GPP network slice and an ETSI NFV NS [17]. The left side of Figure 3 shows 3GPP-defined communication service, network slice, network slice subnet, network function and their relations. On the other hand, the right side of Figure 3 shows NFV-defined network service, VNF, PNF and their relations.

The MANO-based 5GC slicing system can be divided into three parts: user plane, control plane and management plane. The ETSI NFV-MANO architecture is the management plane while the 5GC platform serves as the control and user planes. The management plane is responsible for the life cycle management of NSs, VNFs and other virtual resources. On the other hand, the 5GC supports all network functions required by the control plane and the user plane to provide 5G services.

In this research, we assume a 5G core with multiple slices pre-provisioned and instantiated. As 5G service traffic arrives at the core network, its control signaling will be directed to a common network slice that consists of all CP VNFs. On the other hand, its user data traffic will be routed to multiple customized UP slices preconfigured with different QoS characteristics, then forwarded to different data networks

3. Design of Multi-Slice Architecture

As depicted in Figure 4, we propose to deploy four slices: a common network slice consisting of all CP VNFs and three customized network slices, each with a single UP UPF configured with different bandwidth limits. These slices will be pre-provisioned before the system starts its operations. Our purpose is to verify that under the slicing system, there will be more flexibility to allocate limited resources.

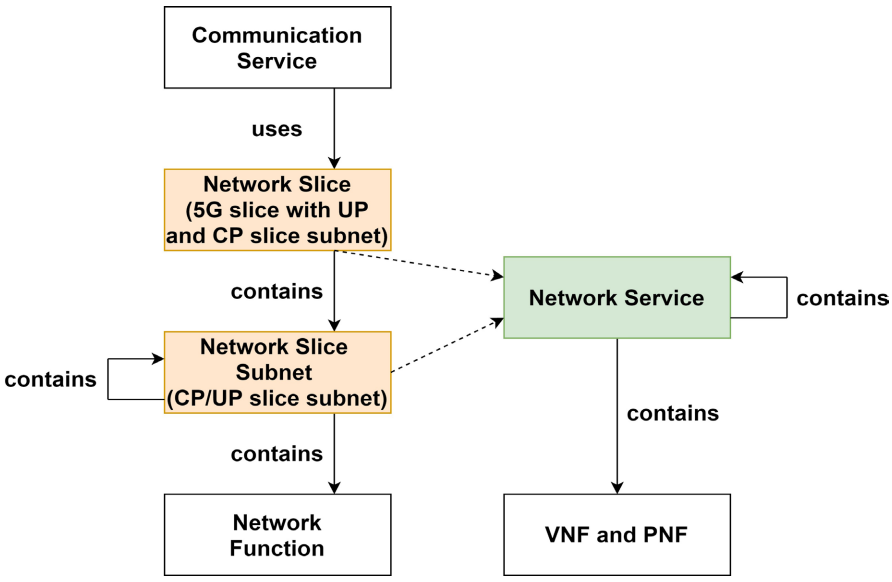

Figure 3. Mapping from network slice (subnet) to network service [12].

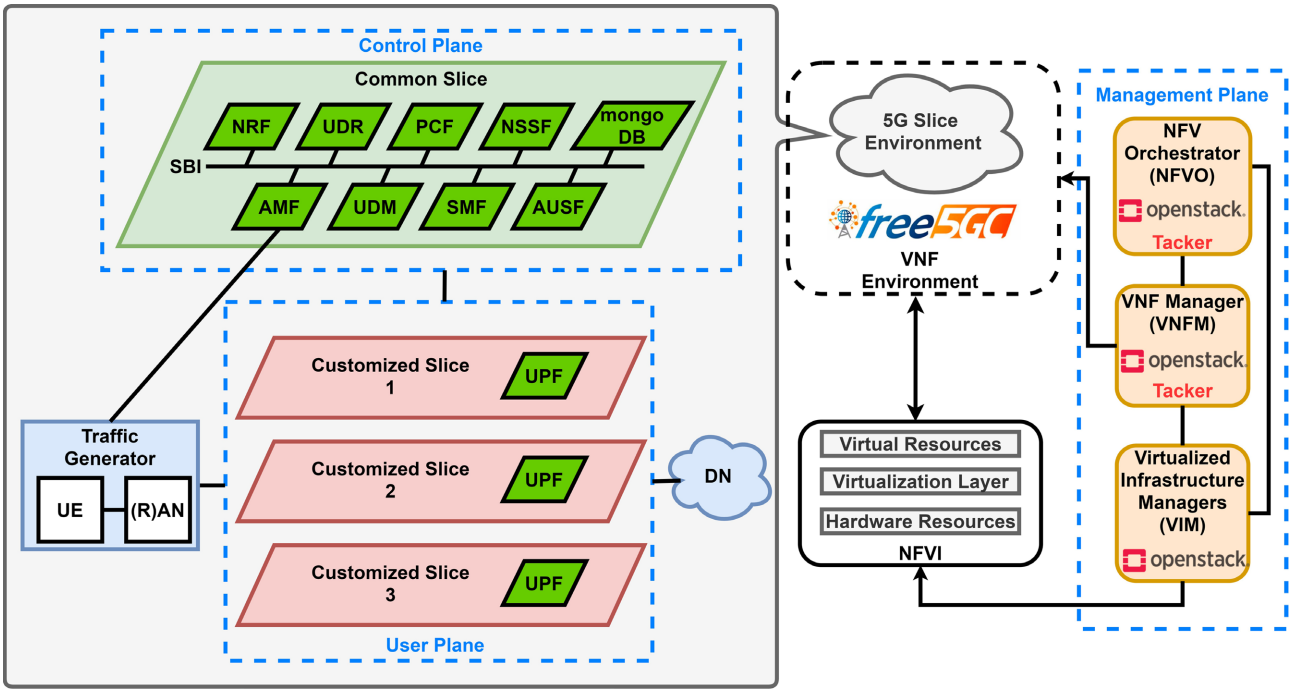

Figure 4. Our proposed 5G slice architecture.

Table 1. Expected traffic and max bandwidth of UPF slices.

VNF	UPF1	UPF2	UPF3
Expected Traffic	50 Mbps	100 Mbps	210 Mbps
Max Bandwidth	100 Mbps	200 Mbps	420 Mbps

3.1. Design of 5G Network Slices

Our slicing design is focused on customized slices. In three customized slices, only one UPF is deployed but each with a different bandwidth setting. These customized slices are designed to optimize resource allocation. Table 1 shows the bandwidth allocation and expected traffic of each customized slice. These three slices are to handle three different kinds of 5G eMBB traffic with low, medium and high bandwidth requirements.

In the compared one-size-fits-all single slice system, we set its user plane bandwidth as the summation of those in three customized slices of our proposed system. In the proposed system, we specify 3 vCPU, 16G RAM and 40 GB disk for each customized slice. In the compared system, we specify 9 vCPU, 48G RAM and 120 GB disk. All the UPFs in UP are equipped with the Debian 7 operating system. Consequently, the total virtualized resources of the compared and the proposed systems are exactly the same except that the proposed system would set the bandwidth limitation on each customized slice.

On the other hand, the common slice consists of 9 CP VNFs including AMF, SMF, PCF, NRF, UDM, AUSF, UDR, NSSF and mongoDB. Note that mongoDB is not the formal NF provided by 5G. Nevertheless, free5GC separates it as a storage NF for all other NFs. It is used to store the subscription and policy data

in UDR, the network function profiles in NRF, etc. We configure all the VNFs in the common slice with 1 vCPU, 2 GB RAM, and 40 GB disk, equipped with the Ubuntu 18 operating system.

3.2. Design of 5G Deployment Templates

Before instantiating our proposed 5GC slicing system, both NSD and VNFD need to be designed [18]. The VNFD describes the behavior and deployment information of a VNF [19]. It may be constituted by the following deployment components:

- Virtual Deployment Unit (VDU): This represents the specification of a VNF Component (VNFC) that can be run on a VM or container. It includes deployment properties such as name, image, flavor and key. The image is the executable that can be instantiated on a VM or a container. The flavor includes disk, RAM, vCPU, etc. and the key specifies the key pair for authentication.
- Internal Virtual Link (VL): This is an internal VL providing connectivity between VDUs. In OpenStack, it specifies as a network.
- Connection Point (CPt): CPt is used to connect to the internal VL. It specifies the connectivity between Internal VL and VDU. We could assign the IP address for VDU or specify the security group for traffic management in CPt.

Figure 5 shows the relation between VDU, VL and CPt in Tacker. In the current implementation of Tacker, a valid VNFD must include VDU, VL and CPt. Each VDU needs to connect to a VL through a CPt. The communications between VDUs would go through the same VL. In our system, each 5G VNF consists of only one VDU and each VDU connects to a private network we create in advance to simulate the VL bound by CPt.

To enable network services, the NSD describes the behavior and deployment information of an NS [20]. It consists of:

- VNF Descriptor (VNFD): NSD would reference one or multiple VNFDs it needs.
- Virtual Link Descriptor (VLD): VLD would specify the connections between VNFs. We can call these external VLs to distinguish from the internal VLs between VDUs.
- VNF Forwarding Graph (VNFFGD): This defines the connection topology of VNFs in network slices. VNFFGD is usually used to create a service function chain.

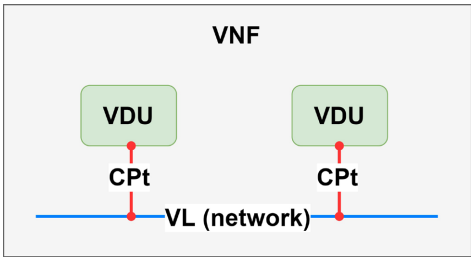

Figure 5. Relation between VDU, VL, CPt in VNF.

In NSD, we need to import one or multiple required VNFDs which have been on-boarded and specify their topology. In our system, we deploy CP and UP as a network slice subnet respectively. The combination of CP and UP network slice subnets builds up the network slice of 5GC. In our system, we don't have any service function chain. Therefore, there is no need to specify any VNFFG. In our implementation, we only instantiate the VNFs we need in NSD but no virtual links between VNFs (Tacker does not support VLD). Instead, all the 5GC VNFs would be deployed to a private network (Internal Virtual Link) we create in advance to simulate the inter-VNF communications.

Note that Tacker only supports VNFD and VNFFGD in NSD in its current implementation [21]. It doesn't support PNFD (PNF Descriptor) and VLD in the NSD level for inter-VNF communications. Instead, the connection between VNFs is specified in VL and CP at the VDU level in VNFD.

3.3. 5GC System Operation Workflow

We deploy the proposed 5GC network slicing system using MANO. Below we show the workflow of our system operations including the setup of the 5GC network slice environment, the onboarding of required NSD and VNFDs and the testing of the proposed system.

1) The MANO system is brought up to construct the 5GC network slice environment. Both VNFM and NFVO (Tacker) would be set up first then OpenStack needs to be registered as VIM to Tacker in order to manage the NFVI resources.

2) The VNFDs of all required network functions need be developed, then on-boarded to VNF Catalogue one-by-one through Tacker after verification.

3) The NSD of the network slice/network slice subnet based on the VNFs just on-boarded needs to be designed and on-boarded to NS Catalogue through Tacker after verification.

4) Once both NSDs and VNFDs have been on-boarded, we can proceed to instantiate the network slice/network slice subnet instance. The Tacker would create VNFs with proper requirements based on the NSD and its constituent VNFDs, then activate them.

5) After all the VNFs are up and running, we can configure the network bandwidth by CLI through VIM manually. This is to set the bandwidth requirements of the UP slices. The automation of this step is currently being developed.

6) We would register Traffic Generator simulating UEs to the 5GC to set up the testing sessions for data transmission to the specific user planes. Each UE is pre-assigned to a specific user plane.

7) After the PDU sessions are set up, UEs would start to send data simultaneously to simulate real traffic.

8) Measurement was done by testing the throughput and response time with UDP and ICMP packets under three different type of eMBB traffic: low, medium and high.

4. System Evaluation

To verify our design, we compare our proposed system with a one-size-fits-all single slice system. We define three evaluation metrics: 1) CPU utilization; 2) Throughput; 3) Response time.

Below we introduce the experiment environment we use and the experiment results.

4.1. 5G Slice Environment

We followed the ETSI MANO NFV framework discussed in Section III to build our slicing environment. We employed two rack servers configured according to Table 2 to deploy Tacker and OpenStack. An NFV-MANO testbed is hence created to manage and orchestrate our virtualized slice environment.

4.2. Traffic Simulation

The testing scenarios were sending two types of data: 1) ICMP; 2) UDP. UDP is used to create traffic load while ICMP is used to discover the response time. The whole experiment is done in the virtual environment. We create three traffic generators to simulate three sets of UEs generating three types of data traffic with different bandwidth requirements: High at 210 Mbps, medium at 100 Mbps and low at 50 Mbps.

Each traffic generator would register to the 5GC to set up the PDU session on UP. After the session's setup, it would generate UDP packets toward a specific UPF. At the same time, ICMP data would be sent periodically to discover the response time. In the data network, we would set up a server to process the packet from the specific traffic generator. Figure 6 shows the data paths for the proposed system and the compared system. The UPF of the proposed system is sliced into three with different bandwidth requirements while the UPF of the compared system remains as one with the aggregated bandwidth of all UPFs in the proposed system.

4.3. Performance Evaluation

We evaluate the performance by collecting throughput, response time and CPU utilization under the UDP traffic. Only UDP packets with upLink traffic were

Table 2. Specification of physical machines.

Entity	Configuration	Version
Rack Server # 1 hosting Tacker as NFVO & VNFM	-OS: Ubuntu 18 -Processor: Intel E5-2678V3 2.5 Ghz 10 Cores -RAM: 128 GB -DISK: 960 GB	Stable Rocky
Rack Server # 2 hosting OpenStack as VIM	-OS: Ubuntu 18 -Processor: Intel E5-2678V3 2.5 Ghz 10 Cores -RAM: 128 GB -DISK: 960 GB	Stable Train

sent. In our testing, we collected the measurements for the throughput by sending the UDP packets to the server in DN and the response time by sending the ICMP packets to the server in DN. The latter is calculated by the traffic generator as the round-trip time of the ICMP packet under three different types of UDP traffic.

Figure 7 shows the results of average throughput of the proposed system and the compared system under three type of traffic. The bars show the comparison of average throughputs under three types of data traffic: high at 210 Mbps, medium at 100 Mbps and low at 50Mbps. Overall, the average throughput of our proposed system is 1.27 times higher than that of the compared system because a UPF dedicated to handle each type of traffic instead of one UPF handling all three types of traffic.

Figure 8 shows the results of average response time of the proposed system

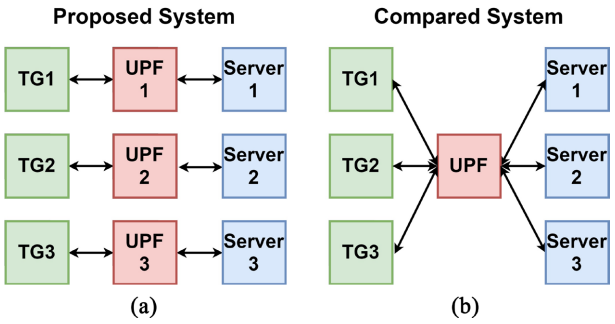

Figure 6. Experiment Traffic Path. (a) Proposed System; (b) Compared System.

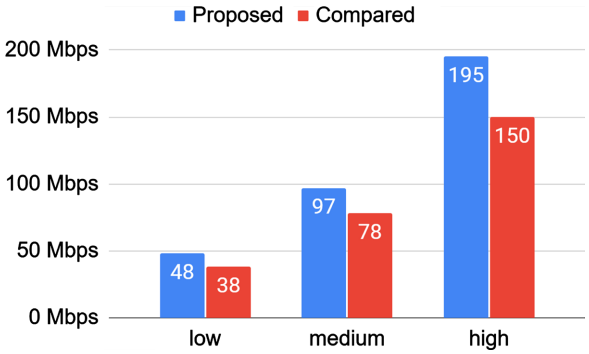

Figure 7. Average throughput.

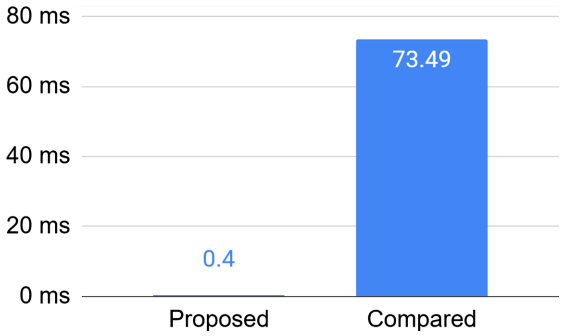

Figure 8. Average response time.

and the compared system. Here the average is calculated from all three types of data traffic. Since the response time of the compared system will increase dramatically when the traffic becomes very large, the proposed system gets a much lower average response time than the compared system. We suspect this was caused by the computational limit of the UPF design in free5GC.

In CPU Utilization, we only measure the user plane slices. The result of CPU Utilization test for the proposed system and the compared system are shown in Figure 9. We allocate the same resource and bandwidth limitation to both systems. Since the overhead of operating system and the heavier usage of computing resources with three UPFs instead of one UPF in the proposed system, the average CPU utilization of the proposed system is approximately three times higher than that of the compared system.

5. Conclusions and Future Work

In this paper, we propose a 5G multi-slices system that consists of one common CP slice and multiple UP customized slices. To support different types of services, we specify each customized slice subnet in 5G networks with different bandwidths. Our system was constructed by leveraging open source projects including Tacker, OpenStack, free5GC and compliant with the NFV MANO framework. We provided a framework for deploying network slices on a VM-based virtual environment using the aforementioned open sources. With such a framework, our 5G multi-slices architecture could achieve parallelization on UP to get better performance than the single slice architecture under the same resource allocation.

In our experiment, we design traffic generators to simulate three eMBB services of different bandwidth requirements: low, medium and high. We measure the average throughput, response time and CPU utilization under three types of traffic and validate that our proposed system could get better response time and throughput under proper resource allocation. We suspect the gigantic difference of response time between the proposed and the compared systems is because the UPF in free5GC is currently still a single process design. The larger vCPU allocation would not increase the processing speed of the one-size-fits-all slice. Instead, the response time is determined by the strength of a single core. On the other hand, network slicing enabled by virtualization techniques greatly improves the response time with multiple numbers of UPF under the same virtualization

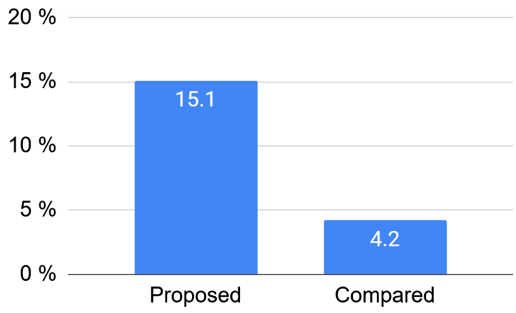

Figure 9. Average CPU utilization.

resources due to its better allocation of resources.

In the future, we plan to expand the system to dynamically add user plane functions according to the increasing traffic load. Moreover, we also want to apply docker which leverages lightweight virtualization technologies such as Container to lower the overhead and reduce the cost of resource usage.

Acknowledgements

This work was financially supported by the Center for Open Intelligent Connectivity from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the MOE in Taiwan.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] 3GPP (2018) Technical Specification Group Services and System Aspects. System Architecture for the 5G System, TS 23.501 Version 15.3.0 Release 15.

[2] Afolabi, I., Taleb, T., Samdanis, K., Ksentini, A. and Flinck, H. (2018) Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions. *IEEE Communications Surveys & Tutorials*, **20**, 2429-2453. https://doi.org/10.1109/COMST.2018.2815638

[3] Ordonez-Lucena, J., Ameigeiras, P., Lopez, D., Ramos-Munoz, J.J., Lorca, J. and Folgueira, J. (2017) Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges. *IEEE Communications Magazine*, **55**, 80-87. https://doi.org/10.1109/MCOM.2017.1600935

[4] Foukas, X., Patounas, G., Elmokashfi, A. and Marina, M.K. (2017) Network Slicing in 5G: Survey and Challenges. *IEEE Communications Magazine*, **55**, 94-100. https://doi.org/10.1109/MCOM.2017.1600951

[5] Kim, D. and Kim, S. (2019) Network Slicing as Enablers for 5G Services: State of the Art and Challenges for Mobile Industry. *Telecommunication Systems*, **71**, 517-527. https://doi.org/10.1007/s11235-018-0525-2

[6] Kaloxylos, A. (2018) A Survey and an Analysis of Network Slicing in 5G Networks. *IEEE Communications Standards Magazine*, **2**, 60-65. https://doi.org/10.1109/MCOMSTD.2018.1700072

[7] Li, X., *et al.* (2017) Network Slicing for 5G: Challenges and Opportunities. *IEEE Internet Computing*, **21**, 20-27. https://doi.org/10.1109/MIC.2017.3481355

[8] Kaloxylos, A. (2018) A Survey and an Analysis of Network Slicing in 5G Networks. *IEEE Communications Standards Magazine*, **2**, 60-65. https://doi.org/10.1109/MCOMSTD.2018.1700072

[9] Kozłowski, K., Kukliński, S. and Tomaszewski, L. (2018) Open issues in Network Slicing. 2018 9*th International Conference on the Network of the Future* (*NOF*), Poznan, 19-21 November 2018. https://doi.org/10.1109/NOF.2018.8598130

[10] Kukliński, S. and Tomaszewski, L. (2019) Key Performance Indicators for 5G Network Slicing. 2019 *IEEE Conference on Network Softwarization* (*NetSoft*), Paris, France, 24-28 June 2019, 464-471. https://doi.org/10.1109/NETSOFT.2019.8806692

[11] Afolabi, I., Taleb, T., Samdanis, K., Ksentini, A. and Flinck, H. (2018) Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions. *IEEE Communications Surveys & Tutorials*, **20**, 2429-2453. https://doi.org/10.1109/COMST.2018.2815638

[12] Zhou, X., Li, R., Chen, T. and Zhang, H. (2016) Network Slicing as a Service: Enabling Enterprises' Own Software-Defined Cellular Networks. *IEEE Communications Magazine*, **54**, 146-153. https://doi.org/10.1109/MCOM.2016.7509393

[13] free5GC.org (2020) https://www.free5gc.org/

[14] ETSI (2014) Network Functions Virtualisation (NFV). Management and Orchestration, NFV-MAN 001 V1.1.1.

[15] (2020) OpenStack Tacker Project. https://wiki.openstack.org/wiki/Tacker

[16] (2020) OpenStack VIM. https://docs.openstack.org/tacker/latest/install/openstack_vim_installation.html

[17] ETSI (2017) Network Functions Virtualisation (NFV) Release 3; Evolution and Ecosystem. Report on Network Slicing Support with ETSI NFV Architecture Framework, GR NFV-EVE 012 V3.1.1.

[18] Ordonez-Lucena, J., Adamuz-Hinojosa, O., Ameigeiras, P., Muñoz, P., Ramos-Muñoz, J.J., Chavarria, J.F. and López, D. (2018) The Creation Phase in Network Slicing: From a Service Order to an Operative Network Slice. 2018 *European Conference on Networks and Communications* (*EuCNC*), Ljubljana, Slovenia, 18-21 June 2018, 1-36. https://doi.org/10.1109/EuCNC.2018.8443255

[19] (2020) OpenStack VNFD. https://docs.openstack.org/tacker/latest/contributor/vnfd_template_description.html

[20] ETSI (2019) Network Functions Virtualisation (NFV) Release 2. Management and Orchestration, Network Service Templates Specification, GS NFV-IFA 014 V3.3.1.

[21] (2020) OpenStack NSD. https://docs.openstack.org/tacker/latest/user/nsd_usage_guide.html