Scientific Research Publishing

Advances in Internet of Things, 2020, 10, 17-35
https://www.scirp.org/journal/ait
ISSN Online: 2161-6825
ISSN Print: 2161-6817

Enabling IoT Network Slicing with Network Function Virtualization

Ting-An Tsai, Fuchun Joseph Lin

Department of Computer Science, College of Computer Science, National Chiao Tung University, Taiwan
Email: a9103100.cs07g@nctu.edu.tw, fjlin@nctu.edu.tw

How to cite this paper: Tsai, T.-A. and Lin, F.J. (2020) Enabling IoT Network Slicing with Network Function Virtualization. *Advances in Internet of Things*, **10**, 17-35.
https://doi.org/10.4236/ait.2020.103003

Received: June 26, 2020
Accepted: July 27, 2020
Published: July 30, 2020

Abstract

Numerous Internet of Things (IoT) devices are being connected to the networks to offer services. To cope with a large diversity and number of IoT services, operators must meet those needs with a more flexible and efficient network architecture. Network slicing in 5G promises a feasible solution for this issue with network virtualization and programmability enabled by NFV (Network Functions Virtualization). In this research, we use virtualized IoT platforms as the Virtual Network Functions (VNFs) and customize network slices enabled by NFV with different QoS to support various kinds of IoT services for their best performance. We construct three different slicing systems including: 1) a single slice system, 2) a multiple customized slices system and 3) a single but scalable network slice system to support IoT services. Our objective is to compare and evaluate these three systems in terms of their throughput, average response time and CPU utilization in order to identify the best system design. Validated with our experiments, the performance of the multiple slicing system is better than those of the single slice systems whether it is equipped with scalability or not.

Keywords

NFV, NFV MANO, Network Slicing, Scalability, IoT, OpenStack, Kubernetes

1. Introduction and Related Work

5G networks meet the different needs of various vertical services flexibly through network slicing. The Third Generation Partnership Project (3GPP) provides four standardized slice/service types including enhanced Mobile BroadBand (eMBB), Ultra-Reliable Low-Latency Communications (URLLC), massive Internet of Things (mIoT) and Vehicle-to-X (V2X) [1] [2]. Among them, mIoT, URLLC and V2X exemplify the large varieties of IoT services. Such a diversity of IoT services is demanding a network with the support of large QoS heterogeneity. However, such support doesn't exist in the currently deployed 4G networks [3]. On the other hand, with the emergence of 5G network and its network slicing, we will be able to allow operators to support a diverse set of services by slicing a physical network into multiple virtual networks [4] [5].

Both Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) technologies can be used to enable network slicing for IoT [6]. NFV decouples network functions from the hardware and runs them on the software platform. NFV provides the flexibility and agility to deploy network functions. With NFV, the resources for a service can be allocated efficiently. In this research, we use NFV technologies to enable IoT network slicing. However, it is also possible to enable IoT network slicing by SDN. In [7], a network slicing environment for IoT is established via an OpenDaylight SDN controller to address the diverse QoS requirements of different IoT/M2M application. In our research, IoT network slicing is enabled by NFV based on the MANO framework. We run virtualized IoT platforms as our VNFs and customize network slices through our NSD (Network Service Descriptor) to support IoT services of various QoS.

We propose to customize each network slice first with different bandwidth to handle different types of IoT services. In addition, because of the virtualization of IoT platforms, we can scale out/in their instances rapidly and dynamically to support the variation in service load [8]. Hence three different slicing systems are constructed for our research. The first system consists of only a single network slice for all IoT services, while the second one consists of three customized slices to handle each type of IoT traffic separately. The last system is similar to the first one but can scale out/in VNFs on the slice. To evaluate the performance of each system, we design a Traffic Generator to simulate three types of IoT services with different bandwidth requirements. We show the advantages and disadvantages of each system and articulate performance tradeoffs by analyzing their pros and cons.

The rest of the paper is organized as follows: Section 2 introduces the background information of oneM2M, ETSI NFV architectural framework and network slice. Section 3 presents our system design and system workflow in OpenStack. Section 4 describes three different systems and compares their performance evaluation. Section 5 presents our system design, implementation and evaluation in Kubernetes. Finally, Section 6 shows the conclusion and future work of this paper.

2. Background

In this section, we explain the oneM2M IoT platform we used in our system, the NFV architectural framework and the concept of network slicing.

2.1. oneM2M IoT Platform

oneM2M [9] is a global standard for Machine-to-Machine (M2M) communications and IoT platforms. It provides common service functions and uniform APIs via a common framework to support the development of IoT applications and services. The oneM2M architecture consists of two domains: Field Domain and Infrastructure Domain as depicted in **Figure 1**.

- Field Domain is the domain where sensors, actors, aggregators and gateways are deployed. It is the M2M area network and resides at the edge of the network.
- Infrastructure Domain is the M2M core network and is normally located in a cloud environment where IoT servers and applications reside.

There are 4 different kinds of nodes in the oneM2M network: Application Dedicated Node (ADN), Application Service Node (ASN), Middle Node (MN) and Infrastructure Node (IN). The Field Domain consists of ADN, ASN and MN while IN is located in the Infrastructure Domain. We only apply ASN and IN in our system. A Traffic Generator is designed as the ASN to simulate IoT devices sending traffic. Three different types of ASN devices including video camera, light pole and parking detector are simulated for three IoT services, respectively. On the other hand, virtualized IoT servers are used as INs in the cloud to receive application traffic from Traffic Generator.

We utilize OM2M, which is an open source implementation of oneM2M developed by LAAS-CNRS [10], as our IoT platform. OM2M provides RESTful APIs for creating and managing M2M resources. It offers Common Services Entities (CSEs) to implement horizontal M2M servers, gateways, and devices. We deploy OM2M IN CSEs as VNFs and use these VNFs to construct network slices in NFV to conduct our experiments with three different slicing systems.

2.2. ETSI NFV Architectural Framework

NFV MANO (Management and Orchestration) [11] is a framework developed by ETSI (Europe Telecommunication Standards Institute) for the management and orchestration of all virtualized resources including compute, network, storage,

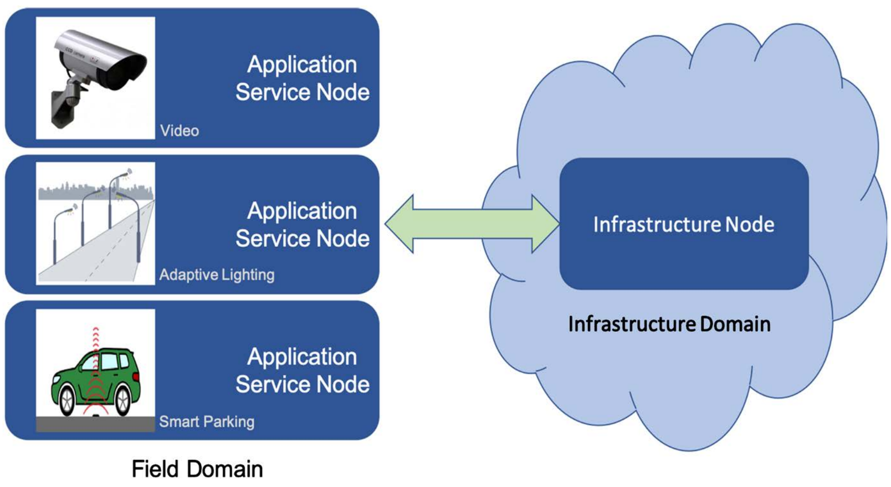

Figure 1. oneM2M functional architecture in our system.

and VNFs. Whenever there is a demand for virtual resources, NFV MANO will coordinate, verify and authorize requests for these resources. It is also responsible for managing the life cycle of VNFs, such as instantiation, scaling, update and termination. In addition, it manages the policy of network services, the collection and transfer of performance measurement, and the allocation of resources related to infrastructures. The NFV MANO framework is adopted in our research to construct the network slicing environment.

As illustrated in Figure 2, NFV MANO consists of three major components.

- NFV Orchestrator (NFVO), which is in charge of the lifecycle of Network Services (NS) and responsible for onboarding Network Service Descriptor (NSD).
- VNF Manager (VNFM), which is responsible for the lifecycle management of the VNFs including VNF scaling out/in and their performance and fault management.
- Virtualized Infrastructure Manager (VIM), which is in charge of allocating and releasing NFV infrastructure (NFVI) including compute, storage, and network resources upon requests of the VNFM and NFVO.

Virtualized Network Functions (VNFs) are the software implementations of network functions such as firewall, load balancer. VNF can improve network scalability and agility and also make better use of network resources. Network Service (NS) consists of one or more network functions, which can include VNFs and Physical Network Functions (PNFs).

The NFV Infrastructure (NFVI) is composed of hardware and software resources that allow VNFs to be deployed, managed and executed. The physical infrastructure includes compute, storage and network. The virtualization layer decouples the virtual resources from the underlying hardware resources. Virtual resources are abstractions of the physical resources through the virtualization layer.

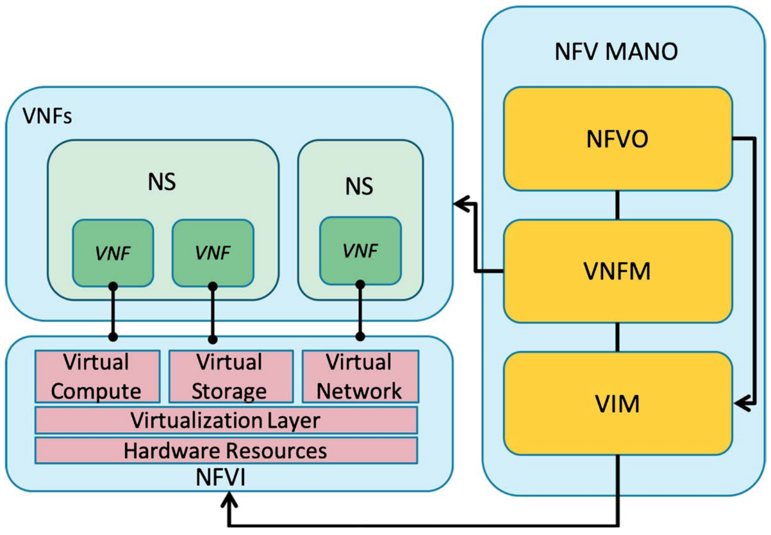

Figure 2. NFV MANO architectural framework.

2.3. Network Slice

According to 3GPP [1], network slice is an end-to-end network architecture. It consists of multiple network slice subnets. Each network slice subnet represents different components in an end-to-end network such as access network, core network, transport network. NFV MANO, as the key enabling technology of network slicing, maps each network slice subnet to an NS in MANO. Each NS is defined by an NSD that consists of a set of attributes and several constituent descriptors including VNF descriptors (VNFDs), VL descriptors (VLDs), and VNFFG descriptors (VNFFGDs) [12]. The attributes of NSD are used to specify how NS instances should be deployed.

Network slicing enables the operator to divide a physical network into multiple virtual and logically independent end-to-end networks. Each network slice is tailored to fulfill different service requirements, such as delay, bandwidth, security, and reliability to cope with diverse network application scenarios [13]. Mobile operators can use network slicing to provide customized 5G networks to various vertical services based on specific needs of each [14].

3. System Design in OpenStack

In our design, we utilize Tacker [15] as NFVO and VNFM, and OpenStack [16] as VIM to set up the NFV MANO framework [17] [18] [19].

- 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 Nova for compute, Neutron for network [20], Cinder for storage, Keystone for key management, Horizon for dashboard and Heat for orchestration.
- Tacker is an official OpenStack project. It is an open source implementation of the ETSI MANO architecture. It provides a generic VNFM and NFVO to deploy and operate NSs and VNFs based on VIM. It supports both OpenStack and Kubernetes as its VIM.

There are other orchestrators that can be used as NFVO and VNFM, such as Open Network Automation Platform (ONAP) [21], Open Source MANO (OSM) [22] and Open Baton [23]. Since Tacker is an official OpenStack project, compared to other open sources, it is highly compatible with OpenStack VIM. Moreover, its design has the advantage of simplicity that allows users to easily deploy and operate. Therefore, we adopt Tacker as NFVO and VNFM in the NFV MANO framework.

In our system, OM2M IN instances are deployed as the VNFs. On the other hand, NSs are the composition of OM2M IN instances and a Load Balancer to be introduced next.

We construct the three slicing systems for our research experiments to support IoT services including: 1) a single slice system, 2) a multiple customized slices system and 3) a single but scalable network slice system. Our objective is to compare and evaluate these three systems in terms of their throughput, average response time and CPU utilization in order to identify the best system design.

3.1. System Architecture

We first explain the functional blocks of our three systems, then show the Network Service lifecycle management flows and the system workflows. The general architecture of our systems is illustrated in Figure 3 and our three systems are depicted in Figure 4.

Note that we design three new system components Master Node, Load Balancer and Traffic Generator on top of OpenStack and Tacker open sources in order to complete our systems.

- Master Node is incorporated in VNFM to monitor the CPU status of VNFs on each network slice in order to trigger scale-out or scale-in actions [24]. When the average CPU usage of VNFs exceeds an overload threshold, it will trigger the scale-out of the VNFs on the network slice. On the other hand, if the CPU usage is lower than an underload threshold, it will trigger the scale-in of the VNFs.

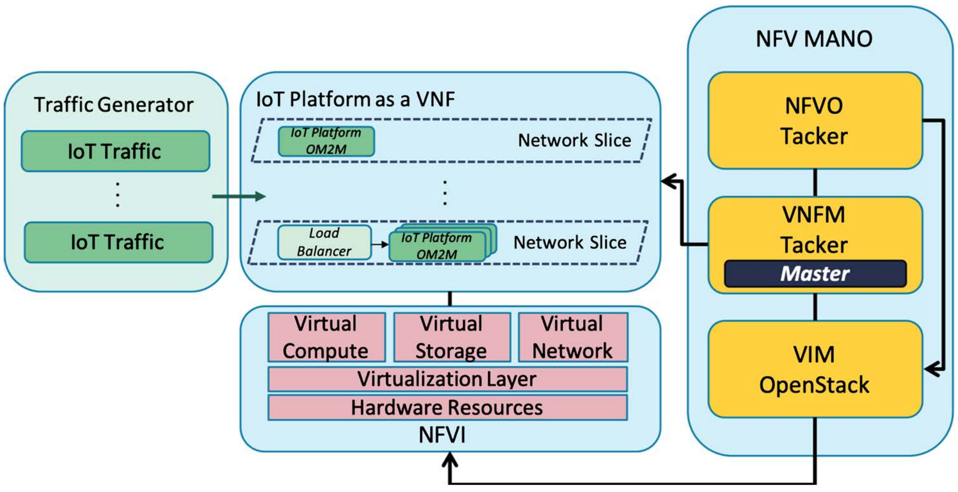

Figure 3. System architectural for OpenStack as VIM.

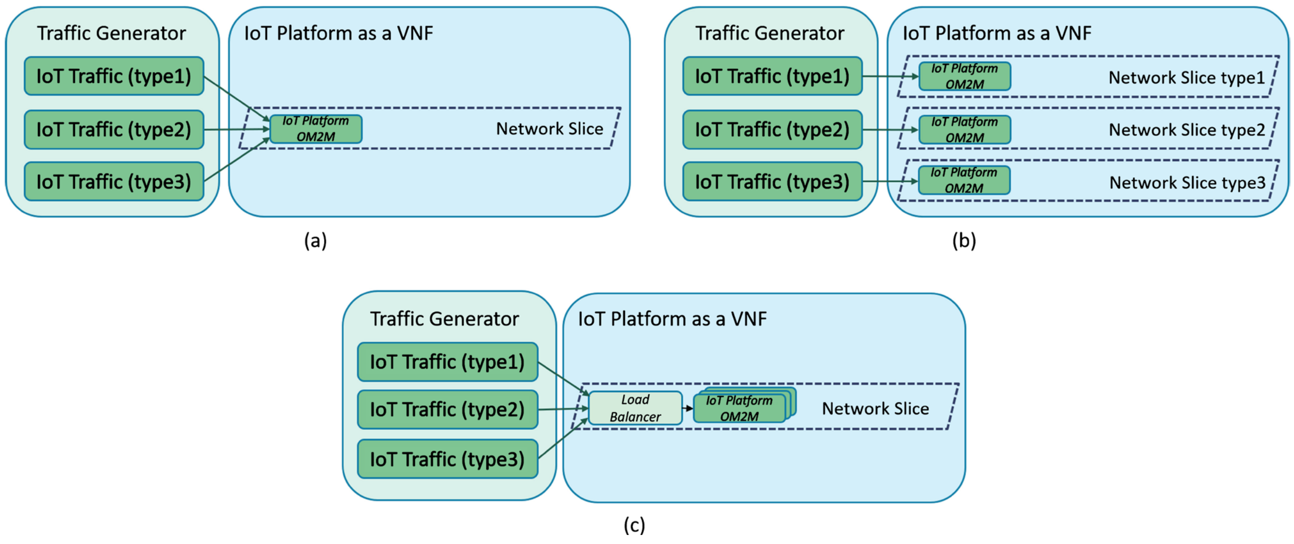

Figure 4. Architecture of (a) Single Slice System, (b) Multiple Slicing System, and (c) Single Slice Scalable System.

- Load Balancer is designed to fairly dispatch the incoming traffic to each VNF [25]. It is used only in the single slice scalable system (see **Figure 4(c)**) for distributing traffic. We use RabbitMQ [26], an open-source message-broker software implementing Advanced Message Queuing Protocol (AMQP) [27], based on Remote Procedure Call (RPC) to design our load balancer. In our system, Traffic Generator sends HTTP requests to Load Balancer and Load Balancer sends those requests to a load balancing queue. On the other side, each OM2M VNF as a server consumes those requests from the queue and replies a response back to Load Balancer.
- Traffic Generator is a multi-thread program that we design to simulate three types of IoT traffic. It can set the number for each kind of ASN devices and the frequency of sending data. Three types of IoT traffic generated include video, adaptive lighting and smart parking. Each traffic is a stream of HTTP requests.

1) Video: This is to simulate a security surveillance service enabled by the video camera. It provides monitoring services for road traffic and crowd movement. This service has the highest bandwidth demand among all three types of traffic.

2) Adaptive lighting: This is to simulate an adaptive lighting service by the smart street light pole that monitors weather conditions and adapts the brightness of street lighting accordingly based on the inputs from temperature, humidity, air pollution and light sensors.

3) Smart parking: This is to simulate a smart parking service that monitors the availability of parking spaces based on geomagnetic sensors embedded in parking areas. This service has the lowest bandwidth requirement among these three types of IoT services.

Figure 4 shows the differences of network slices in three systems under study. In our experiment, Traffic Generator will simulate the same three types of traffic. But the three systems would handle traffic in different ways.

As depicted in **Figure 4(a)**, the single slice system architecture has only one network slice with one IoT platform as a VNF. The only one VNF needs to handle all types of IoT traffic. Then **Figure 4(b)** shows the multiple slicing system architecture that has three types of network slices. Each network slice is provisioned with a different customized bandwidth for dealing with a particular type of IoT service. At last, **Figure 4(c)** shows the design of the single slice scalable system architecture where only one network slice is provisioned but this slice supports scalability. Similar to the single slice system, its network slice needs to handle all three types of IoT traffic. But unlike the single slice system, it is capable of scaling its VNF to multiple instances and thus is equipped with a Load Balancer to evenly distribute IoT traffic to multiple VNF instances. However, because of Load Balancer, IoT traffic must go through an additional VNF, which may result in a longer response time than other systems.

3.2. Network Service Lifecycle Management Flows

Figure 5 shows the Network Service (NS) lifecycle management flows [28].

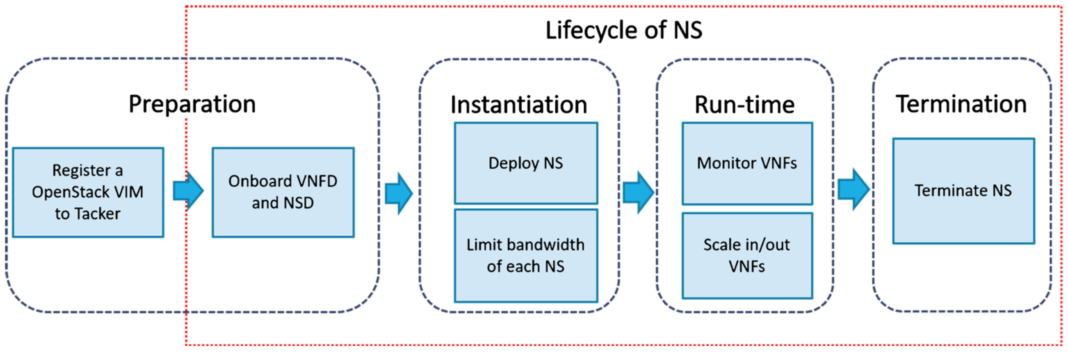

Figure 5. Network Service lifecycle management flows.

There are four phases to run a network slicing system.

In the preparation phase, we set up the environment by first registering an OpenStack VIM to the Tacker MANO system. This is done by setting up an account with authentication URL, username, password, project name and certificate. After registration is complete, we first onboard VNFD and then NSD to NFVO. NFVO will verify the integrity and authenticity of the NSD and check whether the VNFD required by the NSD exists. If acknowledged, it means that the NSD is successfully onboarded.

In the instantiation phase, NFVO receives a request to instantiate an NS. After receiving the request, NFVO validates the request and checks with VNFM whether the VNF instances required exist. If any of them does not exist, NFVO will request VNFM to instantiate it. NFVO will also check with VIM about the availability of required network resources and request the instantiation of virtual resources needed by the NS. Then NFVO will proceed to instantiate the NS by instantiating all its needed VNFs one by one. NFVO will also set up the connectivity among VNFs according to VLDs and VNFFGDs in the NSD. After deploying NS, we limit the bandwidth of each NS according to the attribute in NSD.

In the run-time phase, Master Node will keep monitoring the load status of each VNF and scale out/in VNFs according to the load. When there is only one VNF left, no scale-in action will be triggered.

In the termination phase, NFVO receives a request to terminate an NS instance and requests VNFM to terminate every required VNF in the NS if it is not used by another NS. VIM then deletes the compute, storage and network resources required by the VNFs. After NFVO acknowledges the completion of the Network Service termination, the lifecycle of NS ends in this phase.

3.3. System Workflow for Scaling

The workflow of our system for scaling is shown in Figure 6. This scaling mechanism is only used by the single slice scalable system in our experiment. Note that Master Node will keep monitoring the CPU utilization of each VNF of the NS as Traffic Generator sends requests to the NS. When the average CPU utilization of all VNFs on the network slice exceeds the scale-out threshold, the scale-out action will be triggered. On the other hand, when it is lower than the

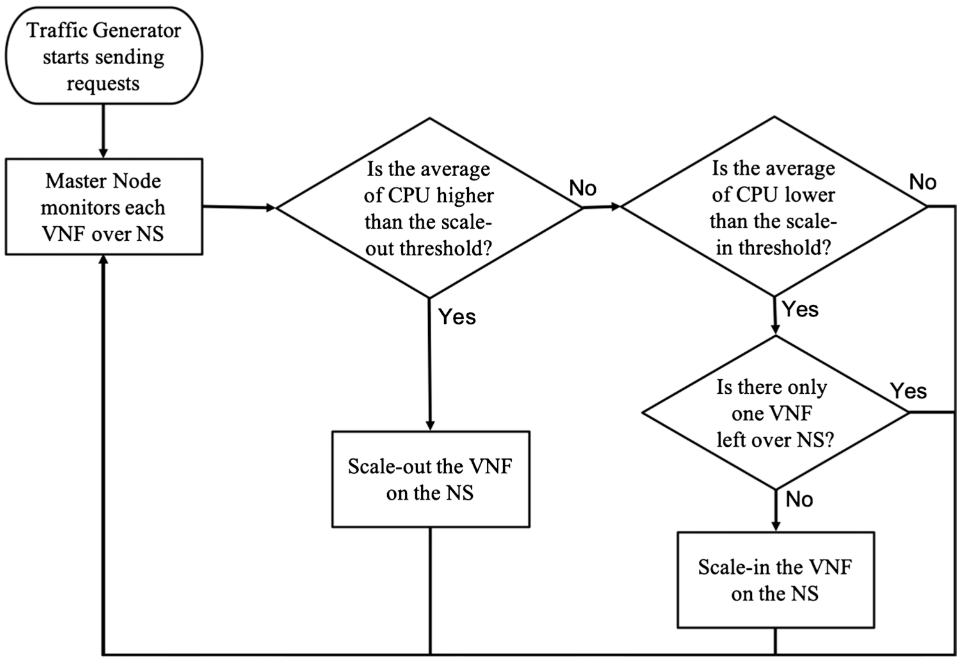
Traffic Generator starts sending requests
Master Node monitors each VNF over NS
Is the average of CPU higher than the scale-out threshold?
No
Yes
Scale-out the VNF on the NS
Is the average of CPU lower than the scale-in threshold?
No
Yes
Is there only one VNF left over NS?
Yes
No
Scale-in the VNF on the NS

Figure 6. Workflow for scaling mechanism.

scale-in threshold, it will trigger the scale-in action. However, if there is only one VNF left for the NS, the scale-in action will not be triggered. In addition, if there is already an action being executed, the next scale-out or scale-in action will not be triggered until the previous one ends.

4. Implementation and Evaluation in OpenStack

In this section, we show our test environment setup and experimental results. Three types of traffic are simulated through Traffic Generator designed to evaluate the performance of each system. The evaluation metrics include throughput, average response time and CPU utilization.

4.1. Test Environment Setup

Our test environment consists of two servers. Both Tacker and OpenStack are running on these two servers configured as shown in Table 1. Table 2 shows the virtual resource allocation of each VNF in our environment.

4.2. Experimental Results

In our experiment, we use Traffic Generator to simulate three types of traffic and send the HTTP requests to each VNF on the network slice. For the single slice system, we send the traffic to the OM2M IoT platform directly. For the multiple slicing system, each type of IoT traffic will be sent to the VNF on the corresponding network slice. For the single slice scalable system, we send all HTTP requests to the load balancer on the network slice first; the load balancer then distributes the requests to each VNF.

In order to meet the different requirements of each type of IoT requests we simulate, we set the required bandwidth of each network slice according to the setting in Table 3.

The required bandwidth is set to twice the expected traffic throughput to avoid temporary excessive traffic. For the single slice system and the single slice scalable system, we set the bandwidth to 1400 Kbps. For the multiple slicing system, the bandwidth limits are 1000 Kbps, 300 Kbps and 100 Kbps respectively with a total at 1400 Kbps that is the same as the other two systems for the fairness of comparison. The configuration of Traffic Generator in Table 4 would meet the expected traffic throughput in Table 3.

To test each system, there are three stages in our experiments. The whole process takes a total of 240 seconds. The payload size of each request sent by Traffic Generator will be based on the settings defined in Table 4. For the single slice scalable system, we follow the workflow for scaling as illustrated in Figure 6, and set the scale-out threshold to 50% and the scale-in threshold to 10% as in [29] [30].

Table 1. Specification of implement environment.

Entity	**Operating System**	**CPU**	**RAM**	**Version**
Tacker	Ubuntu 18.04	Intel E5-2678V3 2.5Ghz 10 Cores	128 GB	Stable Rocky
OpenStack			128 GB	Stable Train

Table 2. Resource information of VNF in our system.

Entity	Image	vCPU	RAM	Disk Size
IoT Platform (OM2M)	xenial-server-cloudimg-amd64-disk1	1	1 GB	10 GB
Load Balancer (RabbitMQ)		2	4 GB	40 GB

Table 3. Bandwidth of each network slice.

System	**Bandwidth (Expected Traffic Throughput/Max Bandwidth)**	
Single Slice System	700 Kbps/1400Kbps	
Multiple Slicing System	Video	500 Kbps/1000Kbps
	Adaptive Lighting	150 Kbps/300Kbps
	Smart Parking	50 Kbps/100Kbps
Single Slice Scalable System	700 Kbps/1400Kbps	

Table 4. Configuration of traffic generator.

Application	**Data Frequency**	**Number of User Threads (In the First and Third Stages/In the Second Stage)**	**Payload Size (Bytes)**
Video	1 request/s	1/3	20,000
Adaptive Lighting	1 request/s	1/3	6500
Smart Parking	3 requests/s	1/3	700

- In the first stage, we will follow the configuration in Table 4 to send data to each system for 30 seconds. The requests of each application will be sent with different frequency and payload size.
- In the second stage, we triple the number of user threads as shown in Table 4 and send data for 120 seconds. We simulate higher traffic in this stage. For the scalable system, the scale-out action will be triggered.
- In the final stage, we return to the same configuration as the first stage for 90 seconds. During this stage, the scalable system will trigger the scale-in action back to its original status.

Table 5 shows the throughput of each application type in each stage. These three systems had a similar result of the throughput and reach the values of expected traffic throughput shown in Table 3.

Figure 7 shows the average response times of all applications in each system and Figure 8 shows the total CPU utilizations of three systems. Integrating the information from these two charts, we conclude that the multiple slicing system achieves the best response time at all stages. Also, its total CPU utilization is only slightly higher than the single slice system so it is an acceptable tradeoff. Overall, the performance of the multiple slicing system is better than those of the single slice systems whether it is equipped with scalability or not.

Comparing the single slice system with the single slice scalable system, it is clear that the average response time of the single slicing system with scalability is better than the one without scalability. In the first stage, since the single slice scalable system must go through Load Balancer which is an additional VNF, the response time is longer than the single slice system. However, when the traffic load increases in the second stage, the response time of the single slice scalable system is similar to that of the single slice system. Moreover, the result of the single slice scalable system is even better in the final stage. This is because the single slice scalable system can deal with increasing traffic loads better than the single slice system. However, the total CPU utilization of the single slice scalable

Table 5. Throughput of each network slice.

Throughput (Kbits/second)				
System	**Type**	**Stage 1**	**Stage 2**	**Stage 3**
Single Slice System	Video	168.6	506.9	168.4
	Adaptive Lighting	51.3	154.1	51.4
	Smart Parking	16.4	49.1	16.4
Multiple Slicing System	Video	168.8	505.9	168.9
	Adaptive Lighting	51.5	154.2	51.6
	Smart Parking	16.4	48.9	16.4
Single Slice Scalable System	Video	169.5	506.3	169.1
	Adaptive Lighting	51.1	153.8	50.6
	Smart Parking	16.0	47.9	15.9

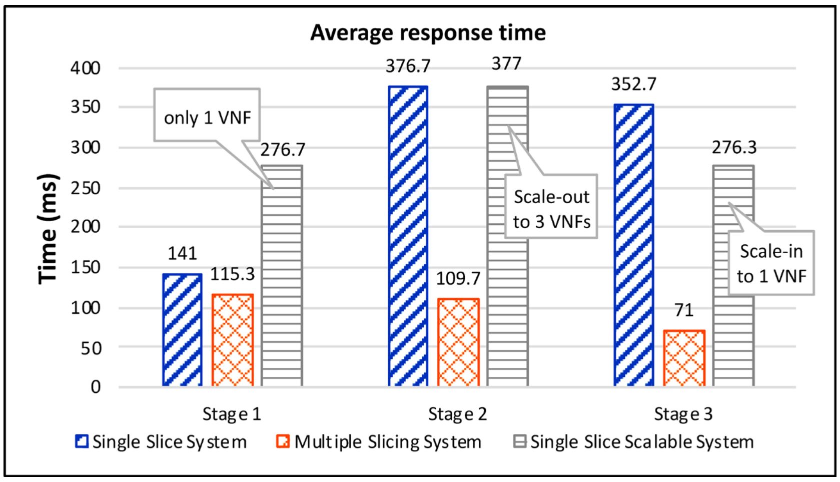

Figure 7. Average response time in each stage over all applications.

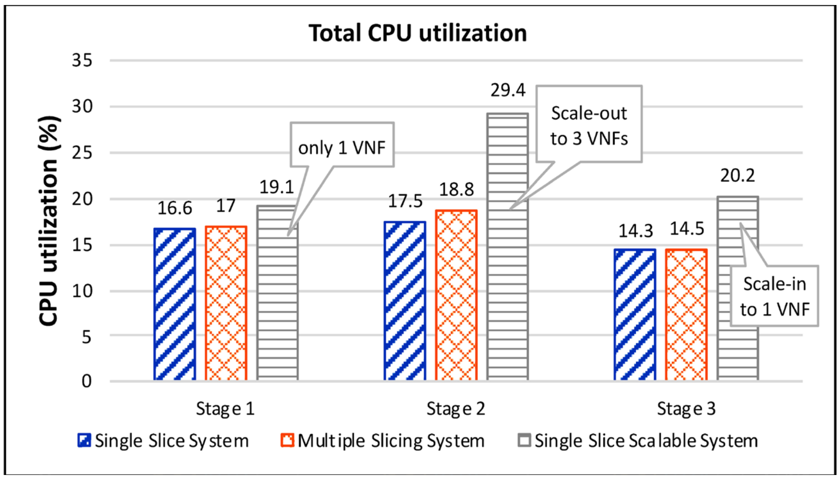

Figure 8. Total CPU utilization of three systems.

system is always higher than the other two systems due to the overhead of Load Balancer and scalability.

The average response times of each application type in the single slice system over all testing stages are shown in Figure 9. The results show that the response time of each application type under the same system architecture only differs a little bit. Similar results are also exhibited in the other two systems. Figure 10 shows that the multiple slicing system gets the lowest response time regardless of the type of applications. We can thus conclude that the system architectures instead of the application types have deeper impact on the system performance.

According to the above results, we speculate that implementing the horizontal scalability across the multiple slicing system may improve its performance and stability, which will be our future work. The research in [7] proved that network slicing can improve IoT/M2M scalability and fulfill different QoS requirements, which is also proved by our experimental results. However, the network slicing in [8] is based on SDN, while ours is based on NFV that has been adopted by the upcoming 3GPP 5G architecture.

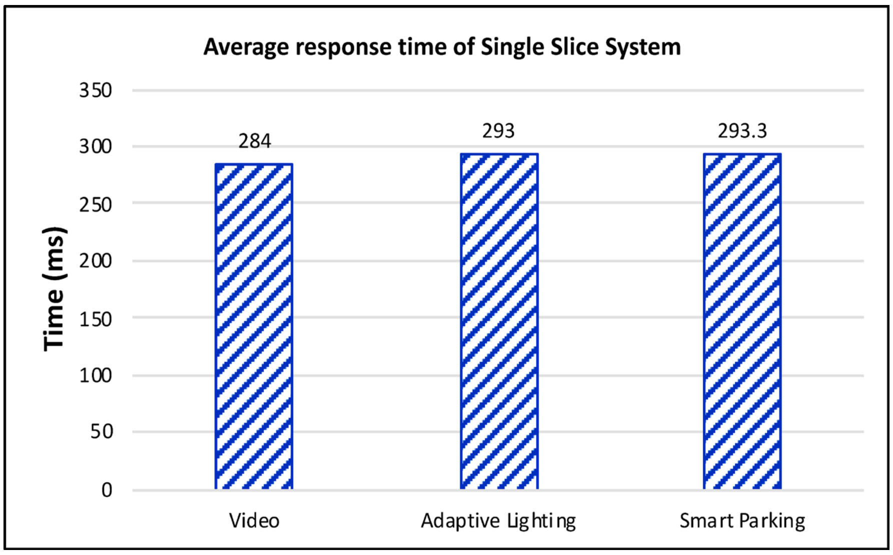

Figure 9. Average response time of each application in single slice system.

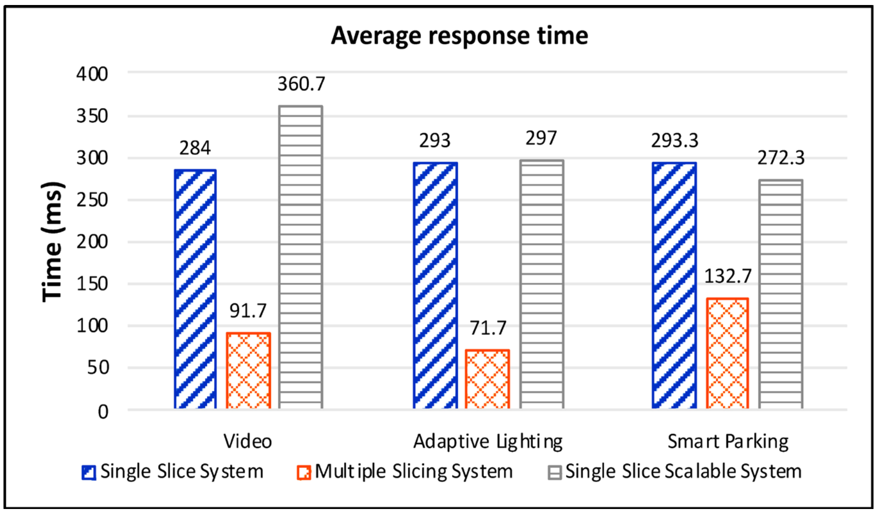

Figure 10. Average response time of each application over all stages.

5. System Design, Implementation and Evaluation in Kubernetes

In this section, we report our research results of building the NFV MANO framework with Tacker as NFVO/VNFM and Kubernetes [31] as VIM. In this design, OM2M IN instances are deployed as the containerized VNFs.

Kubernetes is an open-source system for automating application deployment, scaling, and management. It provides a platform for deploying, managing, and scaling containerized applications across clusters of hosts. It works with a variety of container tools, including Docker.

We construct only two slicing systems in this experiment including: 1) a single slice system and 2) a multiple slicing system. At the end of this section, we will compare and evaluate these two systems in terms of their average response time and CPU utilization.

5.1. System Architecture

Figure 11 shows the general architecture of the two slicing systems. Each functional block of our systems has been presented before. Also, Tacker which is utilized as NFVO and VNFM has been introduced in Section 3. The only difference is that we now use Kubernetes instead of OpenStack as VIM.

Traffic Generator which is our design will simulate the same three types of traffic as our experiments in OpenStack. As depicted in Figure 12(a), the Single Slice System has only one network slice with one IoT platform as a containerized VNF. The only one network slice will deal with all types of IoT traffic. Figure 12(b) shows the architecture of the multiple slicing system where each network slice would handle a specific type of IoT services.

5.2. Test Environment Setup

Our test environment consists of two servers. Tacker and Kubernetes are each running on a server configured as shown in Table 6. Table 7 shows the virtual resource allocation of each containerized VNF in our environment.

5.3. Experimental Results

In this experiment, we use Traffic Generator to simulate three types of traffic and send the HTTP requests to each containerized VNF on the network slice. For the single slice system, we send all three types of traffic to the OM2M IoT

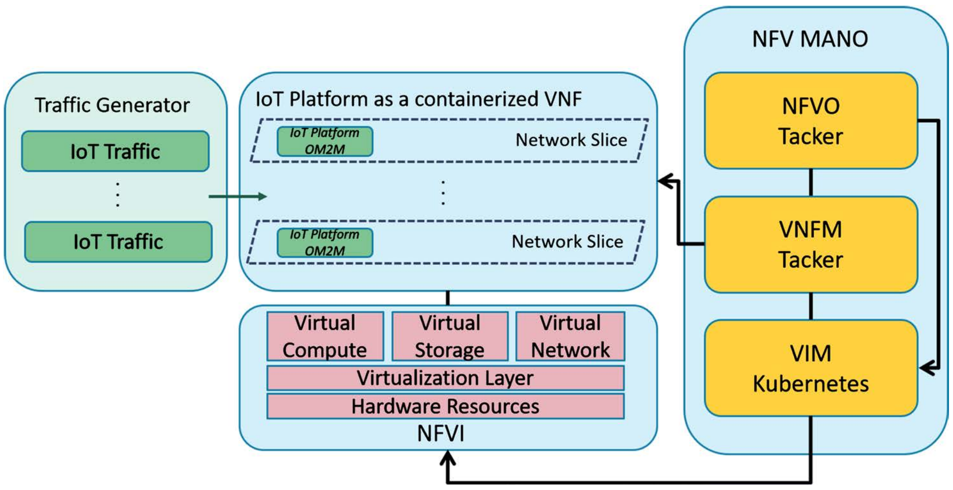

Figure 11. System architecture for Kubernetes as VIM.

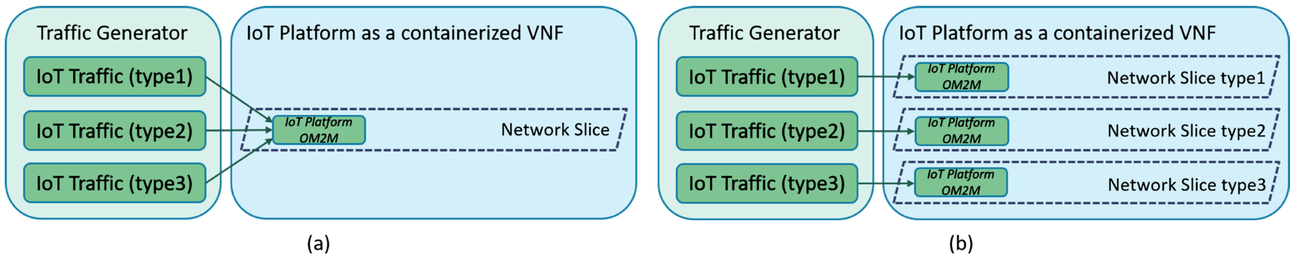

Figure 12. Architecture of (a) Single Slice System and (b) Multiple Slicing System for Kubernetes as VIM.

platform directly. For the multiple slicing system, each type of IoT traffic will be sent to the containerized VNF on the corresponding network slice.

The configuration of Traffic Generator is the same as the one used for OpenStack as shown in Table 4 of Section 4. It will generate the expected traffic throughput required for each application type in this experiment as shown in Table 8.

To test each system, there are three stages in our experiment. The whole process takes a total of 90 seconds. The payload size of each request sent by Traffic Generator will be based on the settings defined in Table 4 of Section 4.

- In the first stage, we will follow the configuration in Table 4 to send data to each system for 30 seconds. The requests of each application will be sent with different frequency and payload size.
- In the second stage, we triple the number of user threads as shown in Table 4 and send data for 30 seconds. We simulate higher traffic in this stage.
- In the final stage, we return to the same configuration as the one in the first stage for 30 seconds. During this stage, the systems will approach stability.

Because Kubernetes has its own scaling functions and policy for scalability, we only construct a single slice system and a multiple slicing system. Also, the time spent in the experiment for Kubernetes as VIM is different from the previous one for OpenStack as VIM. Since there was no need to do scalability, we shortened the total time of the experiment.

Figure 13 shows the average response times of all applications in each system. It shows that the multiple slicing system achieves better response time at all stages although the response times of the two systems are similar in the first stage and the third stage. When the traffic load increases in the second stage, the response time of the multiple slicing system is half of that of the single slice system.

Table 6. Specification of implement environment.

Entity	Operating System	CPU	RAM	Version
Tacker	Ubuntu 18.04	Intel E5-2678V3 2.5 Ghz 10 Cores	128 GB	Stable Rocky
Kubernetes	Ubuntu 18.04	Intel® Core™ i7-8700 CPU 3.2 Ghz 6 Cores	64 GB	v1.15.9

Table 7. Resource information of containerized VNF in our system.

Entity	Image	vCPU	RAM	Disk Size
IoT Platform (OM2M)	tingan0531/om2m	1	1 GB	10 GB

Table 8. Throughput of each application type.

Throughput generated through Traffic Generator (Kbits/second)			
Type	Stage 1	Stage 2	Stage 3
Video	168.2	507.4	170.3
Adaptive Lighting	51.4	154.1	51.4
Smart Parking	16.4	49.1	16.1

This result shows that the multiple slicing system has better performance when it encounters high traffic.

On the other hand, as depicted in Figure 14 the CPU utilization of the multiple slicing system is always higher than that of the single slice system in all three stages because it has three network slices to handle different services. Note that the CPU utilizations of both systems are at their peak in the second stage due to the highest traffic load. On the other hand, because the first stage is the warm-up stage, the CPU utilizations of both systems are higher than those in the final stage when the systems become stable.

Integrating the information from these two charts, we conclude that the performance of the multiple slicing system is better in general as its total CPU utilization is only slightly higher than that of the single slice system but it can achieve faster response time than the single slice system.

6. Conclusions and Future Work

In this paper, we propose three different slicing systems enabled by NFV, based on the MANO framework including: 1) a single slice system, 2) a multiple customized slices system and 3) a single but scalable network slice system to support IoT services. We utilize several open sources such as OpenStack, Tacker, Kubernetes, OM2M and RabbitMQ for constructing our system. In our system,

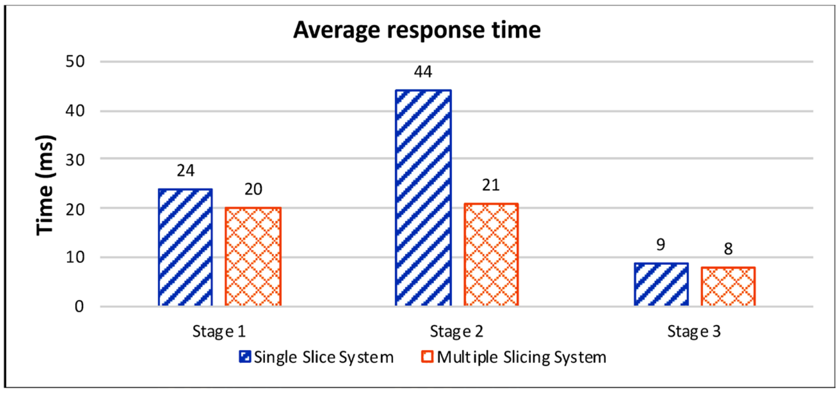

Figure 13. Average response time of all applications for Kubernetes as VIM.

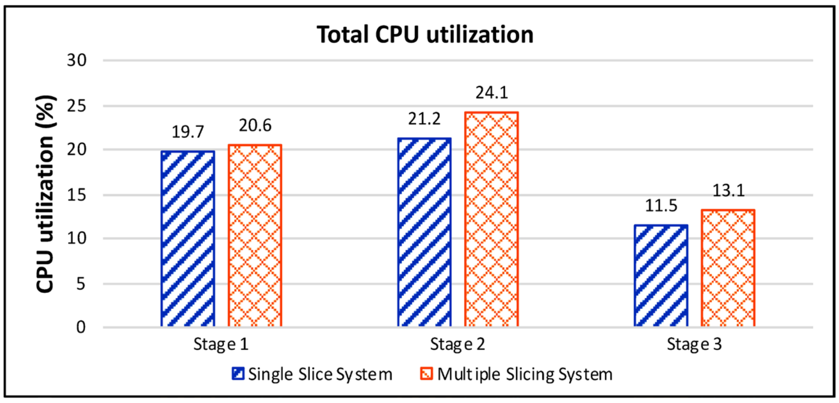

Figure 14. Total CPU utiliztion of two systems for Kubernetes as VIM.

we leverage Tacker as NFVO and VNFM, and OpenStack/Kubernetes as VIM and OM2M as VNFs to set up our NFV-enabled network slicing system for IoT. To support different kinds of IoT services, we customize each network slice with a specific QoS. Moreover, we design a Master Node to monitor the CPU usage of each VNF and scale out or scale in VNFs on the slice according to this information. Also, Load Balancer is designed for the single slice scalable system to dispatch traffic fairly.

In our experiment, we design Traffic Generator to simulate three types of IoT traffic including video, adaptive lighting and smart parking. The test traffic consists of three stages with different traffic loads. We measure the average response time and the CPU utilization of these three systems to identify the best system design. Comparing the results of these three systems, the multiple slicing system has the best performance among them. In addition, the single slicing system with scalability is more stable than the system without scalability with the tradeoff of higher CPU utilization.

Combining the results of the two experiments, the multiple slicing system is the best system design. Although in our experiment with Kubernetes as the VIM, we only constructed the first two systems. The results also show that the performance of the multiple slicing system is better than that of the single slice system.

In the future, we plan to construct a network slicing system with vertical scalability by adapting to changing QoS requirements dynamically. We also plan to experiment the horizontal scalability across multiple slices than just on a single slice. Moreover, constructing a hybrid system of horizontal and vertical scalability to meet more diverse requirements of IoT services is also a potential future research direction [32].

Acknowledgements

This work was financially supported by Ministry of Science and Technology (MOST) of Taiwan Government under Project Number MOST 109-2221-E-009-083 and the Center for Open Intelligent Connectivity from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.

Conflicts of Interest

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

References

[1] 3GPP (2020) System architecture for the 5G System (5GS) (Release 16). 3GPP TS 23.501 V16.4.0.

[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] Zhang, S.L. (2019) An Overview of Network Slicing for 5G. *IEEE Wireless Communications*, **26**, 111-117. https://doi.org/10.1109/MWC.2019.1800234

[4] 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

[5] Lin, F.J. and De La Bastida, D. (2019) Achieving Scalability in the 5G-Enabled Internet of Things. In: Wu, Y.L., Huang, H.J., Wang, C.X., Pan, Y., Eds., *5G-Enabled Internet of Things*, CRC Press, Boca Raton, 396 p. https://doi.org/10.1201/9780429199820-5

[6] Chatras, B., Steve Tsang Kwong, U. and Bihannic, N. (2017) NFV Enabling Network Slicing for 5G. *IEEE 20th Conference on Innovations in Clouds*, *Internet and Networks* (*ICIN*), Paris, 7-9 March 2017, 219-225. https://doi.org/10.1109/ICIN.2017.7899415

[7] De La Bastida, D. and Lin F.J. (2018) Extending IoT/M2M System Scalability by Network Slicing. *IEEE/IFIP Network Operations and Management Symposium*, Taipei, 23-27 April 2018, 1-8. https://doi.org/10.1109/NOMS.2018.8406254

[8] Miladinovic, I. and Schefer-Wenzl, S. (2017) A Highly Scalable IoT Architecture through Network Function Virtualization. *Open Journal of Internet of Things* (*OJIOT*), **3**, 127-135.

[9] oneM2M (2020) Functional Architecture. oneM2M Technical Specification TS-0001 V3.20.0.

[10] OM2M. https://www.eclipse.org/om2m/

[11] ETSI (2014) Network Functions Virtualisation (NFV): Architectural Framework. ETSI GS NFV 002, V.1.2.1.

[12] ETSI (2019) Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Network Service Templates Specification. ETSI GS NFV-IFA 014, V3.2.1.

[13] 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

[14] Zhou, X., Li, R.P., Chen, T. and Zhang, H.G. (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

[15] OpenStack Tacker. https://docs.openstack.org/tacker/latest/

[16] OpenStack. https://www.openstack.org/

[17] Mijumbi, R., Serrat, J., Gorricho, J.L., Bouten, N., De Turck, F. and Boutaba, R. (2016) Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys & Tutorials*, **18**, 236-262. https://doi.org/10.1109/COMST.2015.2477041

[18] Bolivar L.T., Tselios C., Mellado Area, D. and Tsolis, G. (2018) On the Deployment of an Open-Source, 5G-Aware Evaluation Testbed. 2018 *6th IEEE International Conference on Mobile Cloud Computing*, *Services*, *and Engineering* (*MobileCloud*), Bamberg, 26-29 March 2018, 51-58. https://doi.org/10.1109/MobileCloud.2018.00016

[19] Kim, M., Do, T. and Kim, Y. (2016) TOSCA-Based Clustering Service for Network Function Virtualization. 2016 *International Conference on Information and Communication Technology Convergence* (*ICTC*), Jeju, 19-21 October 2016, 1176-1178. https://doi.org/10.1109/ICTC.2016.7763398

[20] Callegati, F., Cerroni, W., Contoli, C. and Santandrea, G. (2014) Performance of Network Virtualization in Cloud Computing Infrastructures: The OpenStack Case. 2014 *IEEE 3rd International Conference on Cloud Networking* (*CloudNet*), Luxembourg, 8-10 October 2016132-137. https://doi.org/10.1109/CloudNet.2014.6968981

[21] Open Network Automation Platform (ONAP). https://www.onap.org/

[22] Open Source MANO (OSM). https://osm.etsi.org/

[23] Open Baton. https://openbaton.github.io/index.html

[24] Hirashima, Y., Yamasaki, K. and Nagura, M. (2016) Proactive-Reactive Auto-Scaling Mechanism for Unpredictable Load Change. 5*th IIAI International Congress on Advanced Applied Informatics* (*IIAI-AAI*), Kumamoto, 10-14 July 2016, 861-866. https://doi.org/10.1109/IIAI-AAI.2016.180

[25] Jain, A. and Kumar, R. (2016) A Multi Stage Load Balancing Technique for Cloud Environment. 2016 *International Conference on Information Communication and Embedded Systems* (*ICICES*), Chennai, 25-26 February 2016, 1-7. https://doi.org/10.1109/ICICES.2016.7518921

[26] RabbitMQ. https://www.rabbitmq.com/

[27] Advanced Message Queuing Protocol (AMQP). https://www.amqp.org/

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

[29] De La Bastida, D. and Lin, F.J. (2017) OpenStack-Based Highly Scalable IoT/M2M Platforms. 2017 *IEEE International Conference on Internet of Things* (*iThings*) *and IEEE Green Computing and Communications* (*GreenCom*) *and IEEE Cyber*, *Physical and Social Computing* (*CPSCom*) *and IEEE Smart Data* (*SmartData*), Exeter, 21-23 June 2017, 711-718. https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2017.110

[30] Chen, H.L. and Lin F.J. (2019) Scalable IoT/M2M Platforms Based on Kubernetes-enabled NFV MANO Architecture. 2019 *International Conference on Internet of Things* (*iThings*) *and IEEE Green Computing and Communications* (*GreenCom*) *and IEEE Cyber*, *Physical and Social Computing* (*CPSCom*) *and IEEE Smart Data (SmartData)*, Atlanta, 14-17 July 2019, 1106-1111. https://doi.org/10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00188

[31] Kubernetes. https://kubernetes.io/

[32] Adamuz-Hinojosa, O., Ordonez-Lucena, J., Ameigeiras, P., Ramos-Munoz, J.J., Lopez, D. and Folgueira, J. (2018) Automated Network Service Scaling in NFV: Concepts, Mechanisms and Scaling Workflow. *IEEE Communications Magazine*, **56**, 162-169. https://doi.org/10.1109/MCOM.2018.1701336