Scientific Research Publishing

Advances in Internet of Things, 2021, 11, 123-139
https://www.scirp.org/journal/ait
ISSN Online: 2161-6825
ISSN Print: 2161-6817

Tackling IoT Scalability with 5G NFV-Enabled Network Slicing

Pei-Hsuan Lee, Fuchun Joseph Lin

Department of Computer Science, College of Computer Science, National Yang Ming Chiao Tung University, Taiwan
Email: j0856524.cs08g@nctu.edu.tw, fjlin@nycu.edu.tw

How to cite this paper: Lee, P.-H. and Lin, F.J. (2021) Tackling IoT Scalability with 5G NFV-Enabled Network Slicing. *Advances in Internet of Things*, **11**, 123-139.
https://doi.org/10.4236/ait.2021.113009

Received: May 7, 2021
Accepted: July 6, 2021
Published: July 9, 2021

Abstract

With emerging large volume and diverse heterogeneity of Internet of Things (IoT) applications, the one-size-fits-all design of the current 4G networks is no longer adequate to serve various types of IoT applications. Consequently, the concepts of network slicing enabled by Network Function Virtualization (NFV) have been proposed in the upcoming 5G networks. 5G network slicing allows IoT applications of different QoS requirements to be served by different virtual networks. Moreover, these network slices are equipped with scalability that allows them to grow or shrink their instances of Virtual Network Functions (VNFs) when needed. However, all current research only focuses on scalability on a single network slice, which is the scalability at the VNF level only. Such a design will eventually reach the capacity limit of a single slice under stressful incoming traffic, and cause the breakdown of an IoT system. Therefore, we propose a new IoT scalability architecture in this research to provide scalability at the NS level and design a testbed to implement the proposed architecture in order to verify its effectiveness. For evaluation, three systems are compared for their throughput, response time, and CPU utilization under three different types of IoT traffic, including the single slice scaling system, the multiple slices scaling system and the hybrid scaling system where both single slicing and multiple slicing can be simultaneously applied. Due to the balanced tradeoff between slice scalability and resource availability, the hybrid scaling system turns out to perform the best in terms of throughput and response time with medium CPU utilization.

Keywords

NFV, NFV MANO, Network Slicing, Fifth Generation Networking, Scalability, IoT

1. Introduction

IoT (Internet of Things) connects a huge number of devices to collect data,

conduct data analytics, and make near real-time decisions. IoT has created a multi-faceted technological innovation and a large number of value-added services [1] [2]. With the rapid increasing of IoT services, the scalability of IoT systems has become an important problem. However, designing scalability over the one-size-fits-all 4G networks is no longer adequate to serve various types of IoT applications.

The upcoming 5G networks propose the adoption of network slicing enabled by Network Function Virtualization (NFV) to support various 5G services such as Enhanced Mobile Broadband (eMBB), Ultra-reliable and Low Latency Communications (URLLC), and Massive Machine Type Communications (mMTC) [3]. 5G network slicing thus allows IoT applications of different Quality of Service (QoS) requirements to be served by different virtual networks [4] [5]. Moreover, these network slices are equipped with scalability that allows them to grow or shrink their instances of Virtual Network Functions (VNFs) when needed. However, all current researches only focus on scalability on a single network slice rather than multiple network slices. The design of a single network slice will not only be restricted by its capacity limitation but also not be able to adapt to the unpredictable user traffic. Hence, we propose a new IoT scalability architecture by reallocating resources when needed to the network service with multiple network slices in this research.

In our proposed system architecture, each IoT network slice will have multiple Instantiation Levels (ILs). First, the scalability will migrate between different ILs on a single network slice to achieve the best resource allocation. But, when the network slice reaches its upper limit of capacity and becomes congested, the service will be scaled out by adding new instances of network slices rather than increasing the number of VNF instances on the same slice. Our unique contribution lies in the design and implementation of scalability at the NS level instead of at the VNF level. Meanwhile, migration between different ILs will still be applied to optimize the resource allocation of each IoT slice. Such an architectural design is very important for service providers because when a burst of traffic flows in, the design of a single network slice will eventually reach the upper limit of affordable traffic and cause the unavailability of IoT services. The proposed architectural design can provide the high service availability critically needed by the service provider. By implementing the proposed system architecture, the IoT services can be free from the capacity limitation of a single network slice because dynamically increasing the number of slices according to the amount of incoming traffic becomes feasible.

Three systems will be compared in terms of throughput, response time, and CPU utilization under three different types of IoT traffic. These three systems include the multiple slices scaling system, the single slicing scaling system, and the hybrid scaling system where both single slice and multiple slices scalability can be simultaneously applied. Due to the tradeoff between slice scalability and resource availability, the result shows that the hybrid scalability system can perform the best in terms of throughput and response time with medium CPU uti-

lization.

The rest of the paper will be organized as follows: Section 2 introduces the background knowledge of the ETSI NFV framework and the used IoT platform, and then gives a survey of the related work. Section 3 explains the proposed system architecture in detail, from the related open sources in our testbed to the relationship between the components, including the scalability strategy used by the proposed system. Section 4 discusses the implementation details of our experiments and their evaluation results. Finally, Section 5 summarizes the entire research and its potential future work.

2. Background

This section introduces ETSI NFV framework, network service/network slice and IoT platform used in the research and discuss related work in this area.

2.1. ETSI NFV Framework

The European Telecommunications Standards Institute (ETSI) proposed the NFV architecture framework [6] to decouple software from hardware so that vendors can do flexible network function deployment and dynamic operation. As shown in Figure 1, the NFV architecture framework is mainly composed of four functional blocks, including Operations Support Systems and Business Support Systems (OSS/BSS), NFV Management and Orchestration (NFV MANO), IoT Network Slices, and NFV Infrastructure (NFVI).

1) OSS/BSS: OSS/BSS is a system used by telecommunication service providers to manage their networks and services. It is where the need for creating network slices is determined. When this happens, OSS/BSS will send a creation request to NFV MANO.

2) IoT Network Slices: IoT Network Slice comprises one or more Network Service (NS) to provide the IoT service, where an NS consists of at least one VNFs as its network functions. VNFs are softwarization and virtualization of

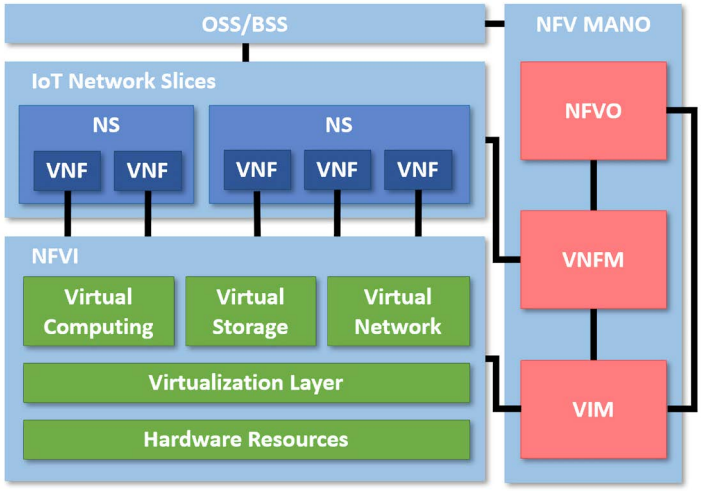

Figure 1. NFV architecture framework.

hardware-software closely coupled network functions in the legacy network. Consequently, VNFs allow new network services to be created more flexibly.

3) NFV Infrastructure (NFVI): NFVI provides an infrastructure with both physical and virtual resources in order to deploy, manage, and execute VNFs. Hardware resources including computing, storage, and network are abstracted by Virtualization Layer to provide processing, storage, and connectivity to VNFs with independent lifecycles.

4) NFV MANO: NFV MANO is in charge of management and orchestration of NSs and VNFs. It handles the demands for NS from OSS/BSS. NFV MANO can be subdivided into three components: NFV Orchestrator (NFVO), VNF Manager (VNFM), and Virtualized Infrastructure Manager (VIM).

- NFV Orchestrator (NFVO): NFVO is in charge of orchestration and management of NS's lifecycle.
- VNF Manager (VNFM): The lifecycle of VNFs is managed by VNFM. VNFM receives the demands from NFVO to create and delete VNFs. It also coordinates with NFVO to request VIM for the resources required by VNFs.
- Virtualized Infrastructure Manager (VIM): VIM is responsible for resource management, which includes controlling and managing the computing, storage, and network resources of VNFs. Moreover, it is allowed to deploy multiple VIMs under NFVO and VNFM.

2.2. Network Service/Network Slice

According to 3GPP, Network Slice is an end-to-end logical network architecture enabled by the NFV technology. It comprises multiple network slice subnets such as access network, transport network, and core network [7]. Each network slice subnet is mapped to a Network Service as defined by ETSI NFV. In this paper, Network Service and Network Slice will be used interchangeably.

1) Network Service Descriptions (NSD)

NSD is a deployment template that describes the composition of NS. It includes VNF Descriptors (VNFDs), Virtual Link Descriptors (VLDs), VNF Forwarding Graph Descriptors (VNFFGDs), and nested NSD if necessary. Each of these descriptors in NSD specifies how the elements in the NS will be instantiated [8]. After onboarding NSD, NFVO will start to manage the lifecycle of the NS until it is terminated [9]. Because of the flexible design of the NSD, NS enables the support of heterogeneous services of different requirements on the same physical network.

2) VNF Descriptor (VNFD) and Instantiation Level (IL)

Similar to NSD, VNFD includes Virtual Compute Descriptors (VCDs), Virtual Storage Descriptors (VSDs), and internal VLDs, which are lower level descriptors describing the constituents of a VNF. Both NSD and VNFD contain their own deployment flavors, which assign which combination of the descriptors should be applied at the time of deployment. Each deployment flavor defines one or more ILs for implementing horizontal and/or vertical scalability.

Horizontal scalability scales the system by adding or deleting the number of the instantiated instances. On the other hand, vertical scalability scales the system by increasing or decreasing the capacity of the existing instances. When applying horizontal scalability or vertical scalability, the system will follow the definition of ILs and migrate between these ILs.

2.3. IoT Platform

With the advent of diverse IoT applications, the need for a global IoT platform standard is emerging in 2011. Via the joint effort of Standard Development Organizations (SDOs) in seven regions, oneM2M was created as the standard functional architecture and specifications for Machine-to-Machine (M2M) communications and IoT technologies.

1) oneM2M Functional Architecture

oneM2M functional architecture contains three kinds of entities:

- Application Entity (AE), which is a logical entity that implements the M2M application service.
- Common Services Entity (CSE), which is a set of oneM2M-specified common service functions that can be used by AE, such as data storage, access control, event detection, etc.
- Underlying Network Services Entity (NSE), which mapping CSE to the underlying network.

oneM2M also defines four kinds of nodes, which comprise CSEs or/and AEs, as logical entities in the oneM2M system, namely Infrastructure Node (IN), Middle Node (MN), Application Service Node (ASN) and Application Dedicated Node (ADN).

As shown in Figure 2, all these nodes reside in either the Field Domain or the Infrastructure Domain. The former is where sensors/actors/aggregators/gateways are deployed while the latter normally resides in a Cloud hosting applications and servers [10]. This research only focuses on multiple ASNs in the Field Domain and an IN in the Infrastructure Domain.

2) OM2M

OM2M is an open source initiated by LAAS-CNRS, which implements oneM2M

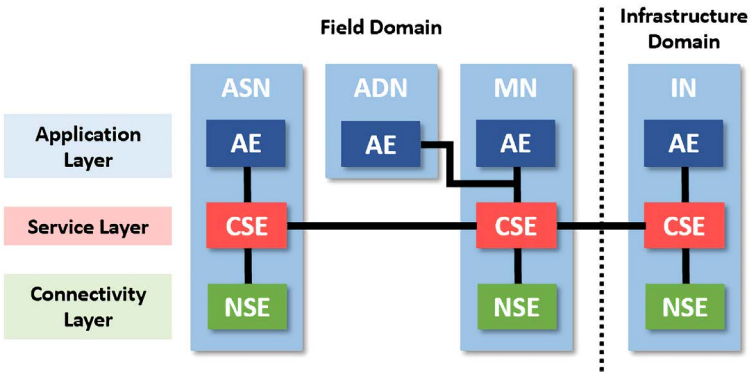

Figure 2. oneM2M functional architecture.

standards. It provides a horizontal M2M service platform with CSEs and leverages RESTful APIs to access those common service functions [11]. OM2M is employed as the IoT platform while OM2M IN CSE is deployed as a VNF in our research. On the other hand, the Traffic Generator is used to simulate the traffic produced by various ASNs.

2.4. Related Work

Network slicing is one of key technologies for 5G networks and its derived issues such as scalability [12] [13], security and resource allocation [14] [15] have become hot research topics in academia due to their importance. Several works related to the research are discussed below.

1) Related Research in Network Slicing

Several SDOs such as ETSI and 3GPP have dedicated to develop standards for network slicing [16]. In addition to standardization efforts, researchers in academia are also very active in developing solutions to solve open issues such as interoperability, security and scalability.

In [17], Samdanis *et al.* proposed a capacity broker, called 5G Network Slice Broker, to act as an intermediary and map the Service Level Agreement (SLA) requests of multiple tenants to physical network resources through the interfaces provided by Master Operator-Network Manager (MO-NM). To support end-to-end network slicing, Santos *et al.* [18] proposed the architecture of hierarchical orchestration where a higher-level orchestrator, called Hyperstrator, is introduced to manage orchestrators in different network segments. Hyperstrator also can translate high-level end-to-end requirements into specific requirements for each network segment. There are also studies like [19] that focus on reducing the complexity of orchestration caused by a large number of software services across multiple clouds and edge data centers. Taleb *et al.* proposed adding a VMN (Virtual Mobile Network) slice orchestrator in each slice to manage the functionality within the slice and providing flexible service function chaining as a service. All these researches provide ways to enhance network slicing standards with more flexibility and effectiveness.

2) Related Research in Scalability

Researches in scalability can be divided into several categories: VM-based (Virtual Machine) scalability and NFV-based (Network Function Virtualization) scalability. E. Cerritos *et al.* [20] focused on VM-based (Virtual Machine) horizontal scalability based on Linux KVM (Kernel-based Virtual Machine). D. de la Bastida *et al.* [21] utilized the OpenStack Cloud for VM horizontal scalability in order to offer faster response time with the tradeoff of higher energy consumption. Both of the above researches focused only on horizontal scalability and operated in the VM-based environment. On the other side, there are also researches in NFV-based horizontal scalability where the scalability is based on NFs such as [22] and [23]. The former realized horizontal scalability by adding Container-based Network Functions (CNFs) dynamically instead of VM-based Network Functions (VNFs) to reduce overall scaling time and computational

cost, while the latter explored both possibilities of horizontal scalability for CNFs/VNFs within a single slice or across multiple slices.

In our research, we propose a new scalability architecture where horizontal scalability is applied in a single slice and across multiple slices based on VNFs. Our architecture demonstrates the benefits of adopting both VM-based and NFV-based approaches.

3. Design and Architecture of Scalability Systems

In this section, the design of three different system architectures and each of their functional blocks will be introduced. Also, both the scalability strategies in a single slice and across multiple slices will be explained.

3.1. System Architecture

This research is based on the NFV architectural framework and plans to investigate three system architectures with 1) single slice scalability 2) multiple slices scalability and 3) mixed single and multiple slices scalability. Multiple open sources are leveraged including OpenStack, Tacker, JMeter, RabbitMQ, Nginx, OM2M to construct the designed NFV testbed.

- OpenStack: OpenStack [24] is utilized as a VIM. It is a cloud operating system that manages infrastructure resources including compute, networking, and storage. It provides various services such as Nova for computing, Neutron for networking, Swift and Cinder for storage, Keystone for authentication as well as authorization, and Glance for image service.
- Tacker: Tacker [25] is a project inside OpenStack that provides the functionalities of NFVO and VNFM in NFV MANO. It is responsible for onboarding the descriptors of NSs and VNFs as well as managing their lifecycles. Because Tacker lacks the support of ILs, we design a Master Node inside its NFVO to provide three additional features: 1) parsing NSD, 2) monitoring the loading of each VNF instance and 3) performing scalability actions. **Figure 3** illustrates the workflow of Master Node: First, Master Node would parse the NSD designed to include ILs. Then, according to the information parsed from the NSD, Master Node would set the initial environment through APIs [26] [27]. Finally, it would start to monitor OM2M VNF instances as well as manage the scalability.
- JMeter: JMeter [28] [29] is open source software based on Java in order to perform load testing and measure system performance. JMeter is used to simulate multiple ASNs in oneM2M. Serving as the Traffic Generator of our testbed, it is used to measure various performance metrics of three system architectures.
- RabbitMQ: RabbitMQ [30] is an open source message broker using Remote Procedure Call (RPC) through Advanced Message Queuing Protocol (AMQP) [31]. It is used as a Load Balancer in our system to fairly distribute all the traffic coming to a subslice. Due to the dynamic creation and deletion of instances, a Load Balancer is required in our testbed to support scalability.

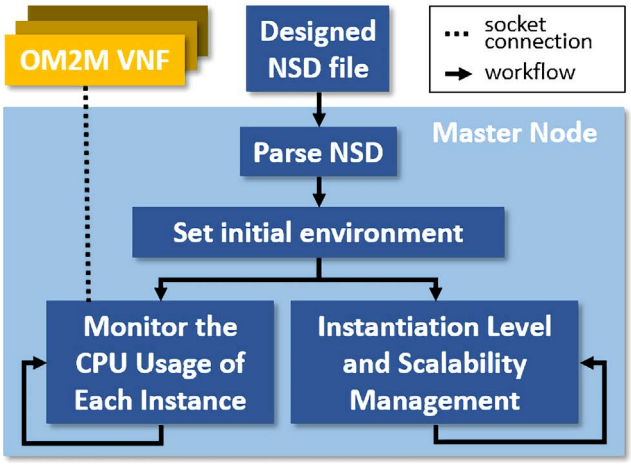

Figure 3. Workflow of master node.

- Nginx: Nginx [32] is an open-sourced HTTP and reverse proxy server with light-weight and high performance. As a master Load Balancer, it is able to evenly distribute incoming traffic to multiple slices at the NS level. It is used to avoid the conflict with the RabbitMQ Load Balancer for a subslice at the VNF level.

1) Single Slice Scaling System Design

Figure 4 shows the entire system architecture of the single slice scaling system. The network service is only served by one single network slice. It contains a Load Balancer and at least one OM2M IN CSE as the VNF to provide the services of IoT/M2M servers. While the system receives the traffic sent by Traffic Generator, Load Balancer will act as an RPC publisher and fairly distribute the traffic to all the VNF instances which act as RPC consumers. Simultaneously, Master Node keeps polling the status of each VNF instance to manage scalability according to the stored definition of ILs which are parsed from NSD.

2) Multiple Slices Scaling System Design

The system architecture of the multiple slices scaling system is shown in Figure 5. It allows service providers to provision a Network Slice consisting of a master Load Balancer and multiple subslices where each of them contains a Load Balancer and multiple OM2M IN CSE instances as VNFs. Moreover, master Load Balancer resides in the NS instead of in any of the subslices. The traffic is first received by master Load Balancer, which provides preliminary distribution based on the capacity of the subslices over the NS. Then, the traffic received by each subslice will be further distributed to its VNFs via another Load Balancer.

3) Hybrid Scaling System Design

Hybrid scalability means the system can apply both multiple slices scalability and single slice scalability. Therefore, it can only be deployed when there is more than one NSs. Service providers are able to make the decision between these two kinds of scalability for different NSs. The decision made by service providers will affect the overall performance of the whole system.

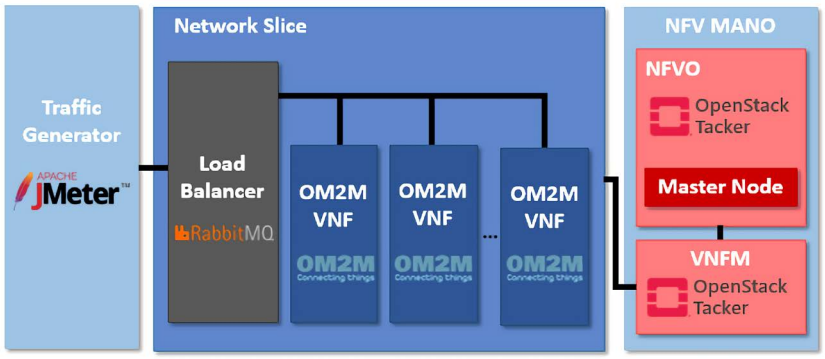

Figure 4. Architecture of single slice scaling system.

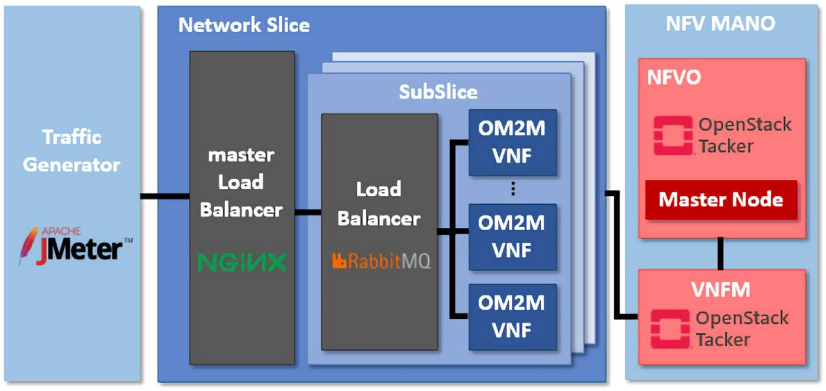

Figure 5. Architecture of multiple slices scaling system.

3.2. Scalability Flow

There are two types of scalability in our research, including 1) scaling in a single slice and 2) scaling across multiple slices. Moreover, we design the scaling strategy based on the concept of ILs that complies with the ETSI specifications [33]. Below the workflow for each type of scalability is described.

1) Scalability in a Single Slice

Figure 6 illustrates the workflow of the single slice scalability. This scalability is applied by all three systems. There are three levels of NS-ILs defined in our testbed including IL#1 with 1 VNF, IL#2 for 2 VNFs, and IL#3 for 3 VNFs. To simplify the scalability in a single slice and focus on the effect of scalability across slices, only horizontal scalability without vertical scalability is applied in the IL design. When Traffic Generator starts sending traffic to the system, Master Node will circularly poll the status of each VNF instance in the network slice.

- Upward migration along ILs: If the current IL is not the highest IL and the average CPU utilization of VNFs is higher than the scale-out threshold, the system will scale out by upward migrating along ILs.
- Downward migration along ILs: On the other hand, if the current IL is not the lowest IL and the average CPU utilization is lower than the scale-in threshold, the system will scale in by downward migrating along ILs.

2) Scalability Across Slices

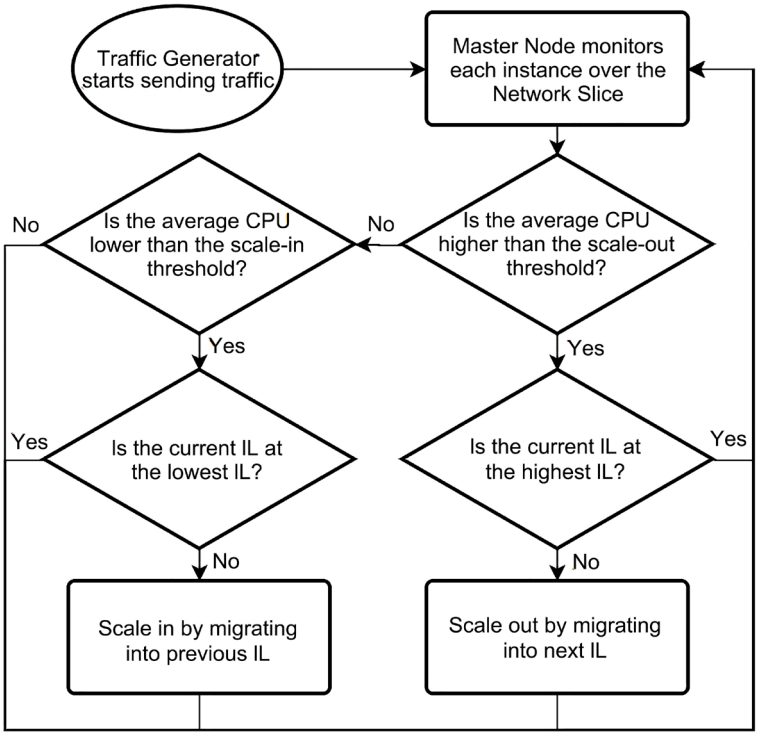

Figure 6. Workflow of the single slice scalability.

The scalability across slices is extended from the scalability in a single slice and applied only by the multiple slices scaling system and the hybrid scalability system. Before the system starts scaling across slices, the scaling strategy is the same as that for a single slice. However, when the slice achieves its upper limit of capacity, the system acts differently from the single slice scaling system. Figure 7 shows the complete workflow of scalability across multiple slices.

- Dynamic creation of slices: If the current IL is already the highest IL and the average CPU utilization is higher than the scale-out threshold, the system will scale out by creating a new subslice automatically.
- Dynamic deletion of slices: Similarly, if the current IL is already the lowest IL and the average CPU utilization is lower than the scale-in threshold, the system will either scale in by deleting a subslice if there are more than one subslices, or do nothing if there is only one subslice left.

4. Implementation and Evaluation

This section shows the environment setup of the design implementation and the traffic design of three different traffic types to simulate the actual IoT environments. Based on the environment set, the results of three system architectures are analyzed and evaluated.

4.1. Design Implementation

Two rack servers are utilized to construct our experimental testbed. OpenStack and Tacker are deployed on the servers as shown in Table 1. Moreover, Table 2

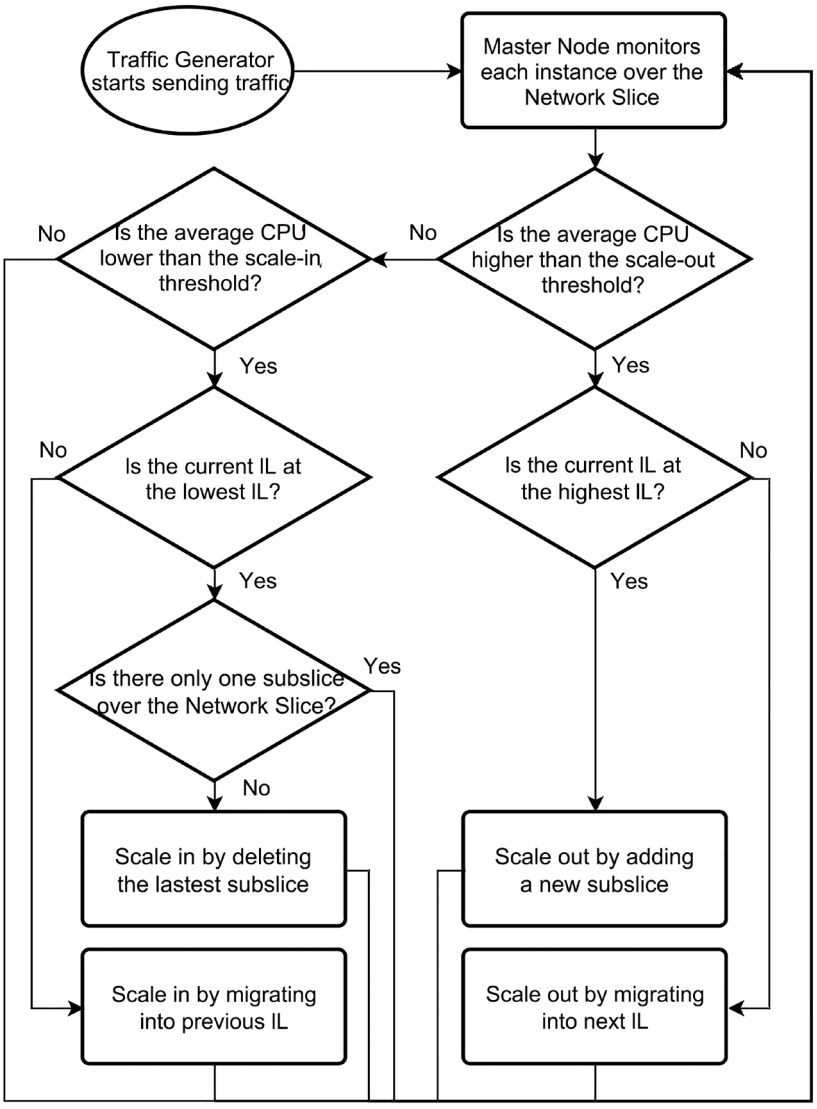

Figure 7. Workflow of the multiple slices scalability.

Table 1. Specifications of servers.

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

Table 2. Resource information of VNFs in our research.

Entity	Image	vCPU	RAM	Disk Size
IoT Platform (OM2M) Load Balancer/master Load Balancer	xenial-server-cloudimg-amd64-disk1	1	1 GB	10 GB

shows the setup of all VNFs including IoT Platform (OM2M) and Load Balancers/master Load Balancer.

4.2. Traffic Design

To simulate the actual situation in the IoT environment, three different traffic types are provided to achieve the effect. Table 3 shows the three traffic types with different payload sizes in the experiment; each type is generated in three phases with different data frequencies.

1) Experiment Phases

Three phases of traffic generation are explained below:

- In the first phase, the Traffic Generator will send data at a frequency of 5 requests per second. This phase lasts for 60 seconds. The main purpose is to let the system reach its stable operation with a small amount of user data.
- In the second phase, the Traffic Generator will send data at a frequency of 40 requests per second. This phase lasts for 360 seconds. As the system continuously receives a large amount of user data from the Traffic Generator, the network slice eventually reaches its capacity limit, then triggers scale-out actions which would migrate between ILs and adding new subslices to increase system capacity.
- Finally, in the third phase, the Traffic Generator will return to data sending at the frequency of 5 requests per second. This phase lasts for 120 seconds to trigger scale-in actions.

2) Three Different Traffic Types

The Traffic Generator generates three traffic types with different payload sizes to simulate heterogeneous data in an IoT environment. Also, in order to analyze the advantages and characteristics of the multiple slices scaling system, a large payload size is used to trigger the scale-out actions of the multiple slices scaling system at the NS level.

4.3. Result Analysis and Evaluation

We tested the implementations of three systems with multiple traffic types. Note that the hybrid scalability system in our experiment is based on the strategy of balancing the tradeoff between slice scalability and resource availability; thus, it would apply the multiple slice scaling for the high traffic, but the single slice scaling for low traffic and medium traffic.

Table 4 shows the results of the throughput comparison between the multiple slice scaling system, the single slice scaling system, and the hybrid scalability system under each traffic type and phase. The total throughput is calculated then produces the visual chart as shown in Figure 8. If only focusing on the systems

Table 3. Traffic design in entire implementation.

Traffic Type	Data Frequency	Payload Size (bytes)
Low Traffic	In the 1st phase: 5 (request/sec)	16,000
Medium Traffic	In the 2nd phase: 40 (request/sec)	22,000
High Traffic	In the 3rd phase: 5 (request/sec)	38,000

other than the hybrid scalability system, it shows that the multiple slice scaling system has better throughput than the single slice system. However, if all three systems are compared, the hybrid scalability system provides the best throughput among these three systems.

On the other hand, Table 5 shows the results of the response times for all three systems under three traffic types in each phase and visualizes the calculated average response time in Figure 9. It shows that the hybrid scalability system has the shortest response time among the three systems in the second phase of the experiment. Also, the response time of the multiple slices scaling system is longer than that of the single slice scaling system due to the reason that the former has to pass two levels of Load Balancers while the latter only needs to pass one.

Finally, Figure 10 shows that the multiple slices scaling system has the highest

Table 4. Throughput of three systems under different traffic types in each phase.

Traffic Type	Multiple Slices Scaling System (KB/sec)			Hybrid Scalability System (KB/sec)			Single Slice Scaling System (KB/sec)		
Phase	1	2	3	1	2	3	1	2	3
Low Traffic	58.07	607.42	35.81	59.89	616.54	53.65	60.16	581.34	52.47
Medium Traffic	84.15	816.22	67.85	87.73	836.15	67.31	84.14	790.50	62.13
High Traffic	148.44	1354.08	116.89	149.67	1336.14	123.69	144.73	1333.57	112.26

Table 5. Response time of three systems under different traffic types in each phase.

Traffic Type	Multiple Slices Scaling System (ms)			Hybrid Scalability System (ms)			Single Slice Scaling System (ms)		
Phase	1	2	3	1	2	3	1	2	3
Low Traffic	122.38	471.62	93.16	114.25	268.10	99.59	112.00	393.57	91.54
Medium Traffic	132.67	551.03	90.54	130.11	421.85	96.77	129.62	382.79	91.36
High Traffic	138.78	634.77	92.38	138.95	597.60	94.15	135.79	555.19	104.94

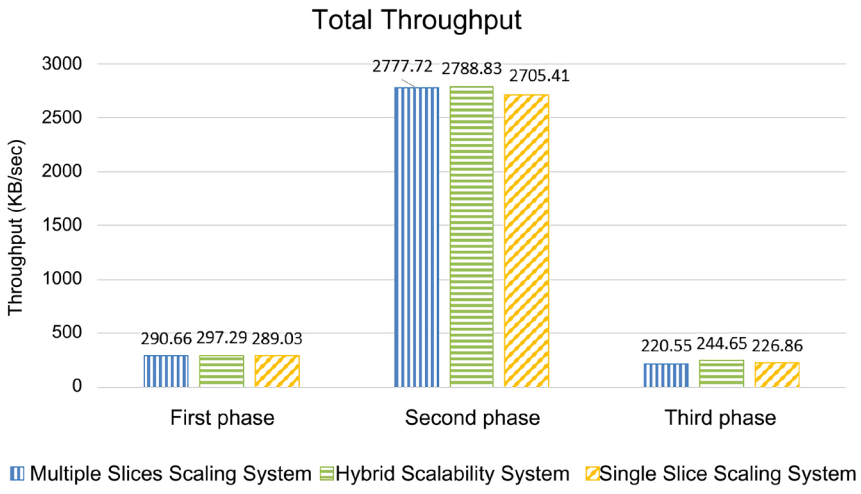

Figure 8. Total throughput of three systems.

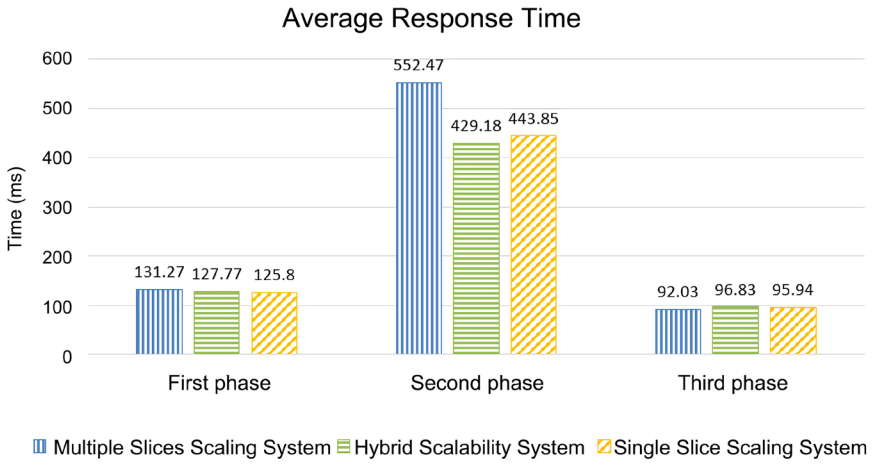

Figure 9. Average response time of three systems.

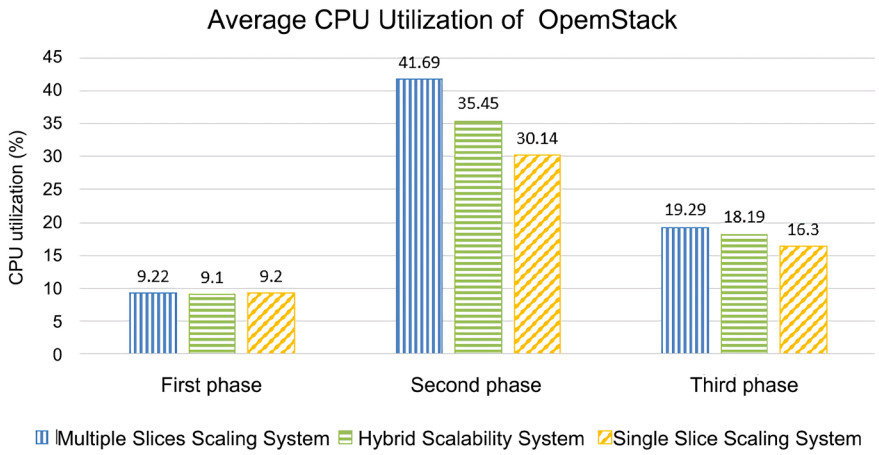

Figure 10. Average CPU utilization of three systems.

CPU utilization while the single slice scaling system has the lowest one. On the other hand, the hybrid scalability system utilizes medium CPU utilization as compared to the other two systems due to its balanced tradeoff.

5. Conclusions and Future Work

In this research, we proposed three different slicing scalability designs, including multiple slices scalability, single slice scalability, and hybrid scalability.

The testbed leverages Tacker as NFVO and VNFM and OpenStack as VIM. Moreover, RabbitMQ and Nginx are utilized as Load Balancers to fairly distribute traffic sent from JMeter. On top of open sources, a Master Node is designed to parse NSDs, monitor VNF instances, and manage the scalability of the system. The concept of Instantiation Levels (ILs) defined by 3GPP is also adopted for scalability migration. The system would first scale out in the same slice by migrating between different ILs, then scale out at the NS level by adding a new slice when the existing slice reaches its capacity limit.

In our experiment, three types of traffic are designed to simulate the IoT en-

vironment. Comparing the results of three systems under three traffic types, the hybrid scalability system turns out to perform the best in terms of throughput and response time with medium CPU utilization due to its good tradeoff between slice scalability and resource availability. This leads us to conclude that the throughput and the response time of the system do not simply depend on its capacity but are largely impacted by how to arrange the best suited architecture according to different applications. Blindly increasing the system capacity may not ensure the improvement of system efficiency. Even so, the proposed multiple slices scaling system in this research still confirms that the scalability at the NS level is useful and effective for the IoT systems.

In the future, the integration of these systems with 5G core network architecture is planned in order to expand to the areas beyond IoT. Also, the combination of the proposed scalability at the network slice level with the hybrid scalability at the instantiation level is planned to further expand the scope of scalability.

Acknowledgements

This work was supported by the Ministry of Science and Technology (MOST) of Taiwan under MOST 109-2221-E-009-083.

Conflicts of Interest

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

References

[1] Čolaković, A. and Hadžialić, M. (2018) Internet of Things (IoT): A Review of Enabling Technologies, Challenges, and Open Research Issues. *Computer Networks*, **144**, 17-39. https://doi.org/10.1016/j.comnet.2018.07.017

[2] Shen, Y., Zhang, T., Wang, Y., Wang, H. and Jiang, X. (2017) MicroThings: A Generic IoT Architecture for Flexible Data Aggregation and Scalable Service Cooperation. *IEEE Communications Magazine*, **55**, 86-93. https://doi.org/10.1109/MCOM.2017.1700104

[3] Gupta, A., Christie, R. and Manjula, R. (2017) Scalability in Internet of Things: Features, Techniques and Research Challenges. *International Journal of Computational Intelligence Research*, **13**, 1617-1627.

[4] Wang, D., Chen, D., Song, B., Guizani, N., Yu, X. and Du, X. (2018) From IoT to 5G I-IoT: The Next Generation IoT-Based Intelligent Algorithms and 5G Technologies. *IEEE Communications Magazine*, **56**, 114-120. https://doi.org/10.1109/MCOM.2018.1701310

[5] Li, S., Xu, L.D. and Zhao, S. (2018) 5G Internet of Things: A Survey. *Journal of Industrial Information Integration*, **10**, 1-9. https://doi.org/10.1016/j.jii.2018.01.005

[6] ETSI (2014) Network Functions Virtualisation (NFV); Architectural Framework. ETSI GS NFV 002, V1.2.1.

[7] ETSI (2020) 5G; Management and Orchestration; Concepts, Use Cases and Requirements. ETSI TS 128 530, V16.2.0.

[8] ETSI (2016) Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Network Service Templates Specification. ETSI GS NFV-IFA 014, V3.2.1.

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

[10] oneM2M. https://www.onem2m.org/

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

[12] Alawe, I., Hadjadj-Aoul, Y., Ksentini, A., Bertin, P., Viho, C. and Darche, D. (2018) Smart Scaling of the 5G Core Network: An RNN-Based Approach. 2008 *IEEE Global Communications Conference* (*GLOBECOM*), Abu Dhabi, 9-13 December 2018, 1-6. https://doi.org/10.1109/GLOCOM.2018.8647590

[13] Ren, Y., Phung-Duc, T., Chen, J. and Yu, Z. (2016) Dynamic Auto Scaling Algorithm (DASA) for 5G Mobile Networks. 2016 *IEEE Global Communications Conference* (*GLOBECOM*), Washington DC, 4-8 December 2016, 1-6. https://doi.org/10.1109/GLOCOM.2016.7841759

[14] Fossati, F., Moretti, S., Rovedakis, S. and Secci, S. (2020) Decentralization of 5G Slice Resource Allocation. 2020 *IEEE/IFIP Network Operations and Management Symposium*, Budapest, 20-24 April 2020, 1-9. https://doi.org/10.1109/NOMS47738.2020.9110391

[15] Su, R., Zhang, D., Venkatesan, R., Gong, Z., Li, C., Ding, F, *et al*. (2019) Resource Allocation for Network Slicing in 5G Telecommunication Networks: A Survey of Principles and Models. *IEEE Network*, **33**, 172-179. https://doi.org/10.1109/MNET.2019.1900024

[16] 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

[17] Samdanis, K., Costa-Perez, X. and Sciancalepore, V. (2016) From Network Sharing to Multi-Tenancy: The 5G Network Slice Broker. *IEEE Communications Magazine*, **54**, 32-39. https://doi.org/10.1109/MCOM.2016.7514161

[18] Santos, J.F., Liu, W., Jiao, X., Neto, N.V., Pollin, S., Marquez-Barja, J.M., *et al*. (2020) Breaking down Network Slicing: Hierarchical Orchestration of End-to-End Networks. *IEEE Communications Magazine*, **58**, 16-22. https://doi.org/10.1109/MCOM.001.2000406

[19] Taleb, T., Mada, B., Corici, M., Nakao, A. and Flinck, H. (2017) PERMIT: Network Slicing for Personalized 5G Mobile Telecommunications. *IEEE Communications Magazine*, **55**, 88-93. https://doi.org/10.1109/MCOM.2017.1600947

[20] Cerritos, E., Lin, F.J. and De La Bastida, D. (2016) High Scalability for Cloud-Based IoT/M2M Systems. *IEEE International Conference on Communications*, Kuala Lumpur, 22-27 May 2016, 1-6. https://doi.org/10.1109/ICC.2016.7511050

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

[22] Chen, H. and Lin, F.J. (2019) Scalable IoT/M2M Platforms Based on Kubernetes-Enabled NFV MANO Architecture. *IEEE International Conference on Internet of Things*, Atlanta, 14-17 July 2019, 1106-1111. https://doi.org/10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00188

[23] Tsai, T. and Lin, F. (2020) Enabling IoT Network Slicing with Network Function

Virtualization. *Scientific Research Journal of Advances in Internet of Things*, **10**, 17-35. https://doi.org/10.4236/ait.2020.103003

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

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

[26] OpenStack NFV Orchestration API v1.0. https://docs.openstack.org/api-ref/nfv-orchestration/v1/

[27] OpenStack Networking API v2.0. https://docs.openstack.org/api-ref/network/v2/

[28] Apache JMeter™. https://jmeter.apache.org/

[29] JMeter Plugins. https://jmeter-plugins.org/

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

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

[32] Nginx. https://nginx.org/

[33] 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