Received: 30 January 2019 | Revised: 19 August 2019 | Accepted: 20 August 2019

DOI: 10.1002/spe.2755

RESEARCH ARTICLE

WILEY

CloudSimSDN-NFV: Modeling and simulation of network function virtualization and service function chaining in edge computing environments

Jungmin Son | **TianZhang He** | **Rajkumar Buyya**

Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Melbourne, VIC, Australia

Correspondence
Rajkumar Buyya, Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Melbourne, VIC 3010, Australia.
Email: rbuyya@unimelb.edu.au

Funding information
ARC Discovery Project

Summary

Software-defined networking (SDN) has evolved and brought an innovative paradigm shift in computer networks by utilizing a programmable software controller with open protocols. Network functions, previously served on dedicated hardware, have shifted to network function virtualization (NFV) that enabled functions to be virtualized and provisioned dynamically upon generic hardware. In addition to NFV, edge computing utilizes the edge resources close to end-users, which can reduce the end-to-end service delay and the network traffic volume. Although these innovative technologies gained significant attention from both academia and industry, there are limited tools and simulation frameworks for the effectiveness evaluation in a repeatable and controllable manner. Furthermore, large-scale experimental infrastructures are expensive to setup and difficult to maintain. Even if they are created, they are not available or accessible for the majority of researchers throughout the world. In this paper, we propose a framework for simulating NFV functionalities in both edge and cloud computing environments. In addition to the basic network functionalities supported by SDN in CloudSimSDN, we added new NFV features, such as virtualized network functions allocation, migration, and autoscaling with the support of corresponding network functionalities, such as flow load balancing, rerouting, and service function chaining (SFC) maintenance. We evaluated our simulation framework with autoscaling and placement policies for SFC in the integrated edge and cloud computing environments. The results demonstrate its effectiveness in measuring and evaluating the end-to-end delay, response time, resource utilization, network traffic, and power consumption with different algorithms in each scenario.

KEYWORDS

cloud computing, edge computing, network function virtualization (NFV), service function chaining (SFC), simulation software, software-defined clouds, software-defined networking (SDN)

1 | INTRODUCTION

In the past decade, cloud computing has been evolved to elastically provision computing resources to their tenants in pay-as-you-go basis. Organizations and start-up companies can build their application services upon cloud data centers

without investing huge up-front costs to purchase computing servers and network infrastructure. Instead, they can utilize as many resources as needed within minutes by several mouse clicks from cloud service providers and pay only for the duration and the amount of provisioned resources.[1] In cloud computing, physical machines are virtualized through hypervisors where the virtualized resources (virtual machines [VMs]) can be allocated directly to the tenants or through platforms and services.

Similarly, virtualization technology has brought a new concept of network virtualization in the telecommunication field along with software-defined networking (SDN).[2] Network functions, such as firewall, proxy, and intrusion detection system (IDS), used to be served by an expensive hardware purpose-built only for certain network functions. As network functions are CPU intensive tasks, the network providers have to purchase the dedicated device to provide the required functions to their customers. Recently, network functions virtualization (NFV)[3] has been evolved drastically thanks to the advancement of virtualization technology, which we have seen in cloud computing. As VMs can be dynamically scaled, provisioned, and migrated in clouds, virtualized network functions (VNFs) can be also provisioned throughout generic physical machines to provide a certain network function. Telecommunication providers build their own cloud data centers within their networks for NFV and place VNFs in their data center to elastically manage its computing and networking resources.

In addition to cloud computing, the increasing popularity of Internet of Things (IoT) leads to introducing edge and fog computing, which utilizes more edge resources closer to the end-users and IoT devices.[4] In IoT, the massive amount of network traffic generated by IoT sensors becomes challenging to service providers and network operators. In order to reduce the network traffic between the IoT sensor devices and the central computation servers usually residing in a cloud, the edge resources (eg, network routers, wireless access points, and radio towers) can be utilized for filtering and preprocessing the collected data. In addition, placing service functions nearby the end-users can reduce the end-to-end latency by reducing the network transmission delay between the end-user and the service function. In NFV, utilizing edge resources for VNFs has been studied by many researchers in recent years.[5-8]

Despite the increased attention of NFV and edge computing, there are limited tools that can verify and evaluate new techniques in the literature. Several proof-of-concept platforms have been proposed by researchers[8-10] to show the potential performance and orchestration abilities of NFV over edge computing. However, evaluating a new method in a large-scale is troublesome with the proposed platforms because the empirical system has to be deployed to a large-scale infrastructure in order to evaluate at such scale. Traditional network simulation tools such as *ns3* do not support the new paradigm of NFV and SDN technologies. Thus, an accessible, adaptable, and scalable simulation toolkit have to be introduced and developed in order to foster emerging research and realization of NFV and edge computing in SDN-enabled cloud computing environments.

In order to fill the gap, in this paper, we propose a new simulation framework, **CloudSimSDN-NFV**, for NFV and edge computing simulation in SDN-enabled clouds extended from CloudSimSDN.[11] We developed and presented CloudSimSDN in 2015 to simulate SDN functionalities in cloud computing. CloudSimSDN has helped to simulate different allocation and provisioning policies for both computing and networking resources,[12] as well as the workload and application-aware scheduling methods in clouds.[13] In addition to the basic networking functionality supported by SDN in CloudSimSDN, this paper presents the design and implementation of new features to support NFV functionalities, such as VNF allocation, migration, and autoscaling. To facilitate these functionalities, the simulation framework is based on the mapping of architecture and components of ETSI NFV management and orchestration (MANO),[14] which includes NFV orchestrator (NFVO), VNF manager (VNFM), and virtual infrastructure manager (VIM). Moreover, the concept of edge computing is integrated and supported by the new simulation framework, including multiple data centers, intercloud networks, and differentiated data center capacities.

The key contributions of this paper are as follows:

- modeling of resource provisioning for NFV in the edge computing environment;
- architecture and design of the simulation framework for NFV in edge and cloud computing;
- detailed development and implementation of the simulation framework and the challenges;
- performance evaluation of the framework with use case scenarios;
- potential extensions and directions of the proposed framework.

The rest of this paper is organized as follows. Section 2 discusses existing platforms and simulation tools in the literature. Section 3 presents the modeling and simulation of NFV in edge and cloud computing environments, followed by detailed design and implementation of the new simulation framework in Section 4. Use case scenarios and evaluation results

using the simulation framework are presented in Section 5. In Section 6, we discuss the potential extensions of proposed framework, which can be implemented for supporting in different scenarios. Finally, Section 7 summarizes and concludes the paper.

2 | RELATED WORK

Several works have been proposed and presented in the literature to simulate cloud, edge, and fog computing, and networking. Many proof-of-concept systems are also developed for NFV evaluation. In this section, we review some of the related works and compare with our proposed framework.

Calheiros et al[15] developed CloudSim toolkit to simulate events and interactions of cloud data centers, such as VM placement, provisioning resources, and scheduling workloads. It is a discrete event simulation tool, where every interaction is modeled as an event between cloud entities (eg, cloud data center, broker, etc). Simulation results are calculated based on the sending and receiving time of an event. Its simplicity and ease of use have attracted significant attention from both academia and industry, which led to initiating various descendant projects based on CloudSim, such as NetworkCloudSim,[16] ContainerCloudSim,[17] iFogSim,[18] CloudSim Plus,[19] and CloudSimSDN.[11]

iFogSim[18] was developed for simulation of fog computing environment. The concept of fog is similar to edge computing that utilizes not only central clouds but also edge resources closer to the end-users, whereas fog computing includes central cloud and other intermediate nodes in the architecture. With iFogSim, it is possible to create an integrated edge-cloud environment to evaluate resource management policies for both edge and clouds. However, it focuses on managing computing resources (CPU, memory, and storage), which result in limited functionality in networks such as NFV and dynamic network configuration. Our proposed framework, on the other hand, can simulate sophisticated network functionalities available in the SDN and NFV paradigm.

CloudSimSDN[11] was introduced in 2015 to provide simulation framework for SDN functionalities on the cloud computing environment. After the first presentation, it is utilized for evaluations in multiple projects, such as oversubscription-based VM and network allocation policy,[12] and priority-aware VM allocation policy considering network topology.[13] The tool can simulate dynamic network flow scheduling, joint VM and network optimization, and monitoring CPU and network utilization in both physical host level and VM level. Although it provides extensive evaluation of network functionality with potential expandability, it is lack of supporting NFV and edge computing models.

Mininet[20] was developed by Stanford University to emulate SDN controllers in a single Linux machine. In Mininet, network switches and hosts are virtually created within the Linux operating system with the network virtualization functions provided by the Linux kernel. With Mininet, any OpenFlow-compatible SDN controllers can be used for emulating SDN functions with real-world traffic and scenarios. As it uses network drivers in the Linux kernel, the emulation reflects more accurate empirical results from the implemented SDN control logic. However, its scalability is limited due to the limitation of the operating system and its resource usage, which prevents specifically cloud-scale (eg, thousands of machines) simulation.

Many researchers developed proof-of-concept systems to evaluate their approaches in NFV. LightMANO is proposed by Riggio et al[21] to test NFV deployment in distributed cloud-edge environment and implemented as a small-scale proof-of-concept system. As the name suggests, the system is aligned with NFV management and orchestration (MANO) architecture[14] and utilizes the edge resources in addition to the central data centers. Cziva and Pezaros[7] proposed container-based network function framework supporting edge computing that can create a network function as a container and migrate to other sites in the edge of the network as like container migration. The proof-of-concept system is developed in Glasgow using OpenDayLight, OpenVSwitch, and Linux systems.

Although these practical systems provide the working example of the proposed approach, building such a system needs expensive equipment and time-consuming effort to setup the environment and implement the proposed policies. In addition, it is difficult to evaluate the new approach in a large-scale with the proof-of-concept systems, as building the large-scale system is in need of even more efforts and resources. Therefore, an efficient and quick toolkit is still necessary for simulation and evaluation in a large-scale and cost-effective manner. The work proposed in this paper meets this requirement by developing software simulation toolkit for modeling and simulation of NFV and service function chaining (SFC) for their use in evaluating policies for edge and cloud computing environments.

3 | SIMULATION AND MODELING NFV IN EDGE AND CLOUD COMPUTING

In order to evaluate and test new algorithms regarding the NFV and SFC management as well as corresponding network functionalities, a scalable methodology has to be employed to see the effectiveness of the algorithm in a large-scale. It is impractical to test such experimental algorithms in a production environment due to unpredictable results from the new method. An experiment in a small-scale testbed is an alternative practical solution to validate the effectiveness, but the impact at large-scale can be hardly captured in the small-scale testbed system. Moreover, empirical implementation of the new algorithm to deploy onto the testbed can be time consuming and challenging especially if it is in the early stage. Therefore, simulation has been widely adopted in science to simplify the evaluation with reasonable accuracy, which can be compared with baselines under the same condition. With simulation, multiple experiments with various configuration and settings can be performed in a short time with automated scripts.

In the same way, simulation of NFV in edge-cloud environment is critical in order to reduce the evaluation time and to simplify the process. Ultimately, the accessible, adaptable, and scalable simulation toolkit will foster the innovation in NFV by reducing the effort of evaluating new methods and algorithms in the field. The innovative ideas can be implemented and evaluated with the simulation toolkit without having to spend enormous time preparing testbed infrastructure and deploying the new algorithm onto the system.

The overall architecture of the NFV implementation in edge-cloud environment is presented in Figure 1. Our simulation framework is based on the architectural components noted in the figure. Edge data centers are microscale data centers (eg, a set of network routers, access points, or a base station with a few computing servers), located close to the end-users. End-users can access application services from edge data centers or cloud data centers through intercloud networks (ie, backbone network). The VNFs can be placed either in the edge or cloud data centers. Cloud data centers can provide VMs for application services and VNFs for network services. More detailed models and abstractions are described in the following subsections.

3.1 | Modeling NFV and edge computing

The NFV MANO architecture[14] has been proposed by European Telecommunications Standards Institute (ETSI) and widely accepted in the communication field. The MANO defines three main components for orchestration and management. The NFVO receives the business and operational requirements and coordinates VNFs in the system to provide the required services. The NFVO can manage various VNFs through VNFMs to actually create and provision VNFs interoperating with SDN elements. Underneath VNFM resides VIM, which controls infrastructure resources hosting VNFs.

Our simulation framework aligns with MANO architecture to provide standardized NFV simulation and evaluation method as shown in Figure 2. We modeled the network infrastructure as physical resources, such as physical hosts, switches, and links. The modeled infrastructure is managed by the infrastructure manager (ie, VIM) for network configuration, address assignment, and physical path. As VNFM can manage VNFs in the system, the simulation framework can create, delete, or migrate VNFs upon the infrastructure as VM instances. The VNF management is decided by the orchestrator (ie, NFVO), which is modeled as various policies in our framework. With the input requirements, scenarios,

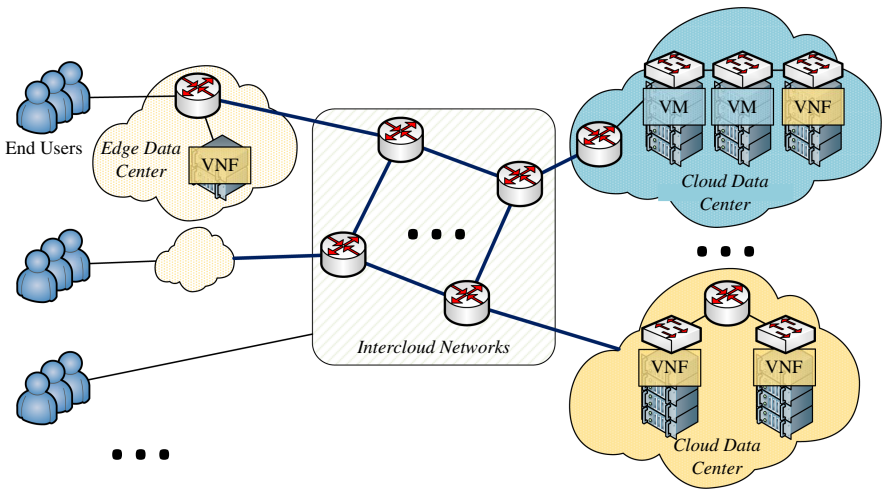

FIGURE 1 Architecture of network functions virtualization (NFV) in edge-cloud environments. VM, virtual machine; VNF, virtualized network function [Colour figure can be viewed at wileyonlinelibrary.com]

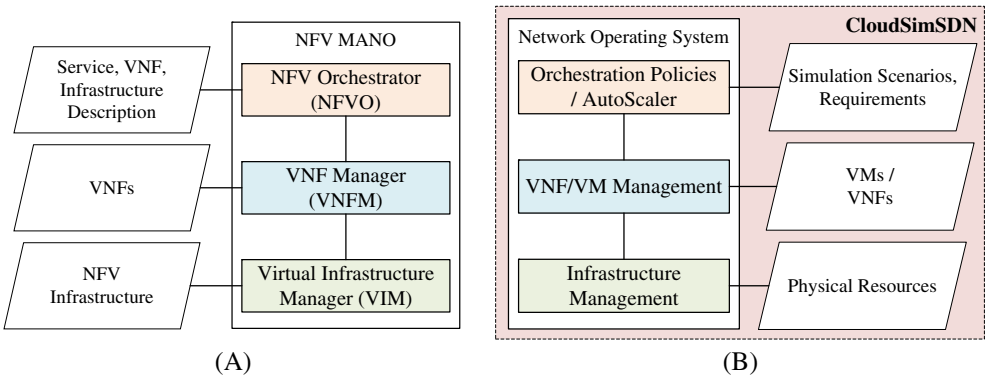

FIGURE 2 CloudSimSDN-NFV architecture aligned with NFV MANO. A, NFV MANO architecture; B, CloudSimSDN-NFV architecture. MANO, management and orchestration; NFV, network functions virtualization; SDN, software-defined networking; VM, virtual machine; VNF, virtualized network function [Colour figure can be viewed at wileyonlinelibrary.com]

and virtual network configurations, user-specified policy in the simulation framework decides the orchestration policies among VNFs, eg, increasing resources for a VNF, decreasing resources in case of underutilization, or migrating to a different infrastructure.

In addition to NFV functionalities, we modeled and integrated edge computing simulation in the framework. Similar to the central cloud data center, an edge data center is modeled with a set of physical infrastructure (hosts, switches, and links) interconnected to each other. Edge data centers are differentiated with the limited resource capacity and the type of data center. For intercloud network between the edge and central clouds, intercloud switches and links are modeled to connect those data centers. Each data center can run its own network and computing resource management policies with separate intercloud network policies to connect among data centers. Location information on VMs, VNFs, and physical hosts is visible globally in order to simplify lookup and management for simulation purposes.

4 | DESIGN AND IMPLEMENTATION

The new simulation framework is designed and developed upon CloudSim,[15] a discrete event-driven simulation framework implemented in Java language. It follows the object-oriented programming model same as its fundamental CloudSim. Components in NFV and edge computing are designed as Java Classes, which can be extended or substituted based on the requirements and simulation scenarios according to the object-oriented model. In this section, we present the design and implementation details of each component in CloudSimSDN-NFV for simulating NFV and edge computing.

4.1 | Event-driven simulation

In CloudSim, every simulation occurs by sending and receiving events between entities. For example, when a VM is created in a data center, a VM request event is sent to the data center entity. Then, the event is received by the data center that allocates resources for the requested VM using a VM allocation policy assigned to the data center. Similarly, a CPU workload can be sent to the VM through the data center entity. The processing scheduler of the VM will receive the workload submission event, calculate the workload end time, then send back the workload completion event with the calculated end time.

CloudSimSDN-NFV follows the same principle to simulate network transmissions, VNF creation and deletion, and intercloud events. It sends and receives events between entities, and the event delay is calculated by policies and schedulers, which can be customized with various scheduling methods and algorithms.

4.2 | Virtualized network function (VNF)

The VNF is designed and implemented by extending the VM Class from the original CloudSimSDN.[11] Being virtualized network appliance with high CPU requirements, VNF is designed to have the same characteristic, such as processing capacity modeled as the number of cores (processing elements) and MIPS (million instructions per second), memory size, and storage size. In addition to those general VM specifications, VNFs have a specific field named MIPO (million instructions per operation), which models the throughput of the VNF. The MIPO specifies the CPU workload length for

TABLE 1 Parameters for service function chaining descriptors template

Type	Parameters						
Node	name	type	size	pes	mips	mipoper	ram
	datacenter	subdatacenters	host				
Links	name	source	destination	bandwidth			
Policies	name	source	destination	flowname	sfc	expected_time	

Abbreviation: SFC, service function chaining.

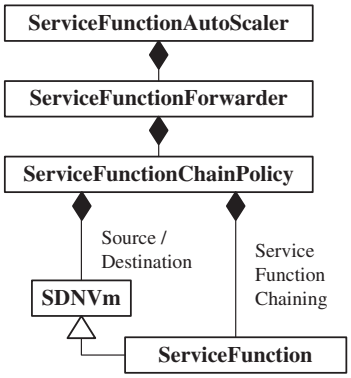

FIGURE 3 Class diagram for simulating service function chaining

a single network operation provided by the VNF, which can provide the throughput of the VNF along with MIPS. For example, a VNF with 1000 MIPS capacity and 10 MIPO can handle 100 requests (operations) per second throughput. The MIPS is also used by VM/VNF allocation policies, which decides a physical host to place the VM. If a host is lack of available MIPS for the requested VM/VNF, the allocation policy will try to allocate it in the other host.

4.3 | Service function chaining (SFC)

We offer the simulation capability of SFC in the framework (see Figure 3). The SFC is defined as a chain of multiple service functions (ie, VNFs) as an ordered and directed list. Enforcing SFC is determined in ServiceFunctionForwarder, which is in charge of forwarding the matched packet to the chain of SFs. The forwarder checks every ServiceFunctionChainPolicy defined in the current simulation configuration, which is composed of the source and destination VMs. For example, if a network flow from VM1 to VM2 has to go through a chain of two VNFs, VNF1, and VNF2, then we can simulate such behavior by creating two VMs and two VNFs in the data center, defining a SFC with VNF1 and VNF2, and enforcing a policy to redirect the network flow from VM1 to VM2 to pass through the defined SFC. These SFCs and enforcing policies are defined in the simulation configuration, loaded and deployed when the simulation started and enforced for all network traffics matching the policy. The simulation configuration of virtual topology (shown in Table 1) described in JSON language contains three categories: (1) Node template defines the name of the instance, type (VNF, VM, Container), resources requirement (disk size, CPU [pe], CPU speed [mips], Memory size [ram]), and packet processing speed MIPO for VNF node type (mipoper), as well as optional allocation requirement for the instance (datacenter, subdatacenters, and host); (2) The template for virtual links defines the name of virtual links between source VM/container to destination VM/container with specific bandwidth as the option. If the name of the virtual link is "default," the packet scheduler will allocate residual bandwidth to the link after satisfying nondefault links. The policy for packet processing can be changed accordingly based on the scenarios of the user simulation; (3) The third part of the template is the same as the VNF forwarding graph descriptors (VNFFGD) used in the Tacker[22] of OpenStack for creating forwarder and SFC. It defines the name of the SFC police, the source and destination of the SFC and which link should be enforced in the SFC (flowname), the directed VNF chaining sequence for the link (eg, "sfc":["vnf1", "vnf2", "vnf4"]), as well as the expected Quality of Service (QoS) in terms of end-to-end delay for the link (expected_time). However, the users can also change and create new VMs/VNFs anytime during the simulation and change the connectivity as well as allocated resources according to the user-defined algorithms dynamically.

On top of the SFC policies and the forwarder, we implement the autoscaling policy for SFCs. In every time interval, ServiceFunctionAutoScaler retrieves the average end-to-end latency for packets in the SFC and the utilization of SFs in the SFC. Based on the predefined autoscaling policy, the autoscaler can increase the capacity of the SF (vertical scale) and/or the number of SF machines for the specific function (horizontal scale). It can also increase the allocated bandwidth for

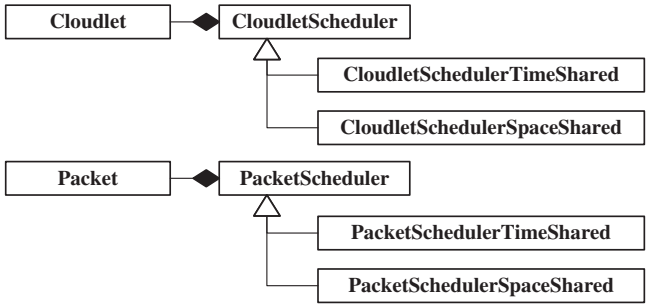

FIGURE 4 PacketScheduler to simulate network packets along with CloudletScheduler of the original CloudSim

the specific SFC if the bottleneck is at the network. We present a use case scenario of SFC autoscaling in the evaluation section.

4.4 | Packet scheduler

We model the packet scheduler similar to the Cloudlet scheduling in the CloudSim, as shown in Figure 4. In the original CloudSim, a computing workload for CPU processing is modeled as a Cloudlet, which has the length of the processing workload. CloudletScheduler is in charge of scheduling the processing workload in each VM based on the simulation scenario. In CloudletSchedulerTimeShared, the processor capacity is evenly shared by all Cloudlets submitted and currently processed at the VM. For example, if five Cloudlets are submitted to a VM, the CPU capacity of the VM is shared among them so that each Cloudlet can be assigned 20% of the total CPU capacity. On the other hand, CloudletSchedulerSpaceShared processes only one Cloudlet at each time so that 100% of the CPU capacity will be assigned to the first Cloudlet submitted to the VM. The other Cloudlets are in the waiting list and processed in the queue once the earlier Cloudlet completes the processing.

In CloudSimSDN-NFV, we design the network packet scheduler with Packet and PacketScheduler Classes similar to Cloudlet and CloudletScheduler. Packet Class represents the network transmission workload, which has the size of the network packet. PacketScheduler distributes the available network bandwidth among currently transferring Packets with the same source and destination VMs. If multiple flows share the same physical link, the bandwidth of the physical link is distributed among these flows,[11] and then PacketScheduler can allocate the distributed bandwidth onto Packets in the scheduler. Similar to CloudletScheduler, we implement PacketScheduler with two models, time-sharing and space-sharing. In time-sharing, the available bandwidth is equally shared among Packets from the same source VM to the same destination VM. In SpaceShared, the entire bandwidth of the virtual network is allocated to the first Packet submitted to the network, and the rest are waiting in the queue until the transmission of the first Packet is completed.

4.5 | Edge computing

For simulating edge computing, we regard the edge resource as part of an edge data center, which has the similar characteristic of a cloud data center. For example, the edge data center will have physical hosts with limited capacity and switches to connect the hosts and end-users. The difference between edge and cloud data centers will be the size and scale of the resources provided in the data center. In terms of simulation, an edge data center and a cloud data center can be regarded as the same entity, whereas the edge data center would provision less amount of computing resources compared to the cloud data center. Other than the resource capacity, both edge and cloud data centers can have its own computing resource management policies and network rules. Thus, we use the data center Class implemented in the original CloudSimSDN to represent both edge and cloud data centers.

However, simulating edge and cloud data centers in the same scenario has to consider running multiple data centers simultaneously. The original CloudSimSDN was potentially capable of supporting multiple data centers in its design, but it was not implemented properly. In this work, we consider the multicloud data center scenario case and implement the correlated entities to work without interfering objects of other data centers. The new framework can create multiple data centers with different configurations, eg, an edge data center with a few low-capacity hosts, and a large-scale data center with thousands of high-performance hosts. The network configurations can be independently set up to reflect the real-world scenario.

Upon properly implementing multiple data centers, intercloud networks have been designed and implemented in this work to support network traffics between data centers. We extended the original Switch Class to InterCloudSwitch Class in order to represent backbone switches connecting distributed data centers. Additionally, GatewaySwitch Class is

introduced to indicate gateway switches in a data center. Network traffics from one data center to another have to pass through the gateway of the origin data center, the distributed intercloud switches, the gateway of the destination data center, and then to the destination host. Various network topologies can be defined individually for different data centers and intercloud networks. For example, an edge data center can be defined with a canonical tree topology, a cloud data center with fat-tree, and the intercloud network with mesh topology. Furthermore, as the SDN routing elements are separated from the physical elements in the principle of SDN, every hosts and switches are considered as a Node Class extension for virtual routing. Therefore, data centers can also be defined with the hybrid or server-centric topology, such as BCube, Dcell, and hypercube, without switches.

4.6 | Customizable policies

In CloudSimSDN-NFV, we implemented the functionalities to evaluate and simulate the SDN-NFV–enabled edge and cloud computing in terms of computing, networking, monitoring, and corresponding virtualization (shown in Table 2). Policies, virtualization technology, and algorithms can be implemented and evaluated based on the optimization of these parameters, such as dynamic flow rerouting based on the processing speed of VNF and end-to-end delay along the SFC, and optimizing the allocation for the new replicated VNFs during autoscaling based on the connectivity, network topology, and available bandwidth. Based on the current simulation framework, we have evaluated the proposed latency-aware VNF dynamic provisioning algorithm to utilize both edge and cloud resources[23] and the unified framework of different autoscaling algorithms to minimize the cost with the guarantees of service-level agreement (SLA) and end-to-end delay along the SFCs.[24]

Beside the autoscaling policies discussed in Section 4.3, we modularize the VM/VNF/Container allocation and network mapping policies for simple customization as shown in Figure 5. Note that we do not show all algorithms that we have implemented but the basic template policies. VmAllocationPolicy defined in the original CloudSim is extended with the separated HostSelectionPolicy, which can select the best host to allocate a VM among hosts in the candidate list which sorting by the optimization policy. We implement two policies as the basic template for selecting a host: first fit and most full methods. First fit is to return the first host appeared in the list, which has enough resources available for the requested VM. The most full method is to find the most occupied host in the list, which can still provide the requested resources. As shown in Table 2, according to the objectives of optimization, the sequence of host list can be based on other parameters, such as computing, networking, and monitoring. Based on the connectivity among VNFs, VMs, and containers, we also implement policy VmAllocationPolicyGroupConnectedFirst to showcase the potential extensions. Furthermore, we implement more complex allocation policies OverbookingVmAllocationPolicy for the beginning and consequent dynamic resource management based on the CPU, memory, disk, and bandwidth overbooking by utilizing migration. Therefore, other policies can be easily implemented by following the existing source code with minimal extension effort.

For a virtual network, VirtualNetworkMapper Class is in charge of mapping the requested virtual network topology onto the physical network elements. Similar to HostSelectionPolicy, a separate LinkSelectionPolicy Class implements which

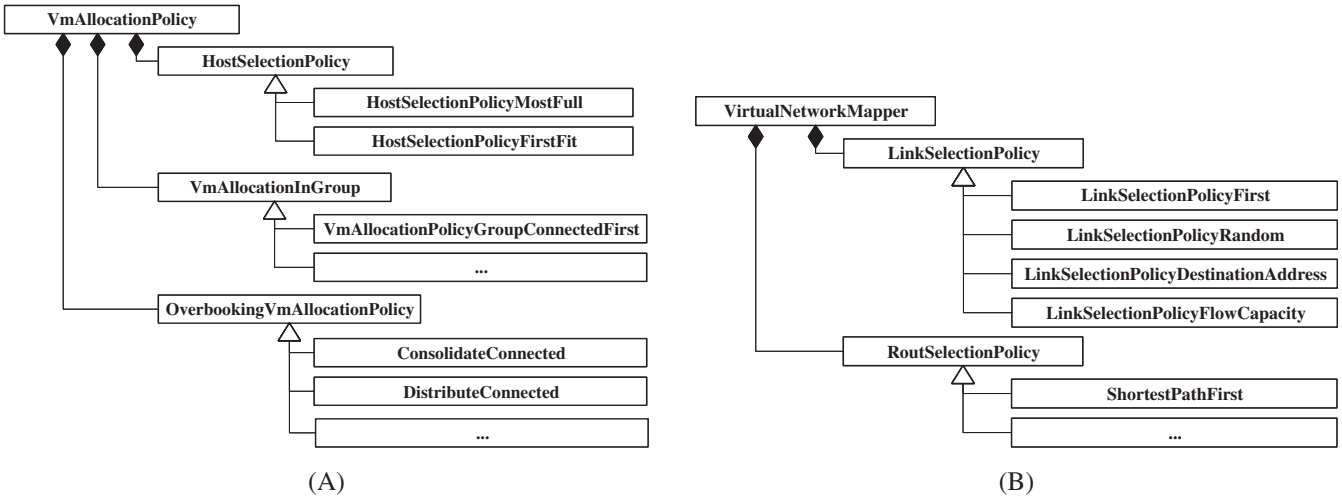

FIGURE 5 Allocation and network traffic engineering policies implemented in the framework. A, Host selection policy; B, Link and route selection policy

TABLE 2 Basic parameters supported for simulation

Type	Parameters						
Computing	CPU	Memory	Disk	Workloads	Task scheduling	Task priority	Overbooking ratio
Networking	Bandwidth	Topology	Switch buffer	Ports	Channel priority	Control plane	Data plane
Monitoring	Statistic	Energy consumption	Utilization	Response time	Network delay	Fault handling	
Virtualization	Connectivity	Allocation	Lifecycle	Flow scheduling	QoS	SLA	

Abbreviations: QoS, Quality of Service; SLA, service-level agreement.

link to choose among multiple links in the physical topology. We implement several methods in the framework as an example. LinkSelectionPolicyFirst selects the first link in the candidate list regardless of the network capacity. Similarly, a random link can be selected with the Random policy. More practically, LinkSelectionPolicyDestinationAddress looks up the destination address of the network and assigns a certain link calculated by a modulo operation. For example, if there are two available links, the destination address with the odd number selects the first link, and the even number selects the second link. LinkSelectionPolicyFlowCapacity selects the link with the most remaining capacity, such as available bandwidth or remaining buffer in switches. It checks currently occupied network capacity of the links in the candidate list and returns the most empty one among them. Thus, the network transmission can be evenly distributed among multiple paths. LinkSelectionPolicy is also readily customizable to implement a new method. Moreover, as shown in Figure 5B, we also implement the policies for proactive flow routing in SDN virtual network. Based on both global physical and virtual network topology provided in the CloudSimSDN-NFV, we can proactively push every specific virtual or physical routing into the forwarding tables of every Node, such as based on the shortest path first algorithm using the global available bandwidth. Furthermore, researchers can also extend the proactive traffic engineering algorithm, eg, the simulation of the policies for network security and jurisdiction: some flows must go through to specific groups of servers within the jurisdiction and cross-region data flows. In general, for researchers who want to evaluate new approaches in VM allocation and/or network mapping, they can modify these policies with the proposed method and compare with the baselines to evaluate the effectiveness of the new approach.

4.7 | Energy consumption and utilization monitor

The new framework also includes monitoring components for energy consumption and resource utilization. A Java interface is defined to record the utilization history of each element (eg, a switch, a host, and a VM). After the preconfigured monitoring interval time, the monitoring event is triggered to measure the amount of processed workload and calculate the utilization of the element for the previous time period. At the end of the every monitoring event, the same event with the monitoring time interval is sent again so that the utilization history is recorded periodically. In addition to recording utilization, the estimated energy consumption for the time interval is also calculated with a linear model. In the current version, we used the linear model to estimate power consumption for hosts[25] and for switches,[26] which can be replaced with other models upon the simulation scenario and requirement.

5 | EVALUATION USING TWO USE CASES

We evaluated the proposed simulation framework with two use case scenarios for NFV and edge computing. This section discusses and compares the effectiveness of different policies in NFV with two use case scenarios. In the first scenario, we evaluate the end-to-end delay and the estimated cost with different autoscaling policies of SFC. We implement the autoscaling policy for processing capacity of VNFs and bandwidth capacity for the chaining upon the simulation framework to compare the measured delay and resource usage. In the second scenario, the edge data center with limited resources is created to host some VNFs. We implement two VNF placement policies to choose between edge and cloud data centers: utilizing edge nodes for placing VNFs, or all VNFs placing in the cloud. These two use case scenarios present the potential usage of the simulation framework in various NFV and edge computing research. For both scenarios, we generated synthetic workloads based on Wikipedia trace and following the three-tier application model.[27]

5.1 | Experiment 1: SFC autoscaling policies in NFV

The first use case is to evaluate different SFC autoscaling policies within a single cloud data center. In this scenario, we created a large-scale cloud data center with 128 physical machines connected through 32 edge switches, 32 aggregation

TABLE 3 Configurations of different instances in experiments

Policy	Type (VNF/VM)	Mem (GB)	CPU (cores)	Disk (GB)	Bandwidth (Bytes/s)	Number of VNFs/VMs
NoScale-Max	lb/ids/fw	8	10/12/16	8	2 000 000	1/3/3
NoScale-Min	lb/ids/fw	8	2/6/8	8	500 000	1/1/1
Flavor	web/app/db	256	8/4/12	1000	1 500 000	8/24/2

Abbreviations: VM, virtual machine; VNF, virtualized network function.

TABLE 4 Service function chaining (SFC) policies enforcing network transmission to divert to virtualized network functions (VNFs)

Source	Destination	Subset of VNFs
Web server	Application server	{VNF1, VNF2}
Application server	Database	{VNF3, VNF4}
Database	Application server	{VNF4, VNF3}
Application server	Web server	{VNF2}

switches, and 16 core switches, forming 8-pod fat-tree[28] network topology. Thus, each pod consists of 4 edge switches, 4 aggregation switches, and 16 physical machines, where 4 machines are connected to each edge switch. Each physical machine is configured with 16 cores, and each core has 10 000 MIPS capacity. The network bandwidth of all links between switches and physical machines is equally set to 200 MBytes/sec.

5.1.1 | Virtual topology and SFC configuration

For virtual topology, we simulate three-tier web applications consisting of web, application, and database servers to form an entire application. There are 8 web servers, 24 application servers, and 2 database servers. The configuration of different types of servers is shown in Table 3. All servers are created as VMs in the simulation, which receives workloads consisting of CPU processing and network transmission. In order to evaluate NFV functionalities, we prepare 4 VNFs from three different types of VNF Firewall (fw): VNF1, Load Balancer (lb): VNF2, VNF3, and IDS: VNF4 as shown in Table 3, which are enforced by 4 different types of SFC policies. Furthermore, we assign MIPO to Load Balancer, IDS, Firewall as 20, 200, and 800, respectively. Network traffics between the VM servers are forwarded to different SFCs consisting of a subset of the 4 VNFs as shown in Table 4. As all network traffic between these VMs is diverted to transmit through the SFCs, these 4 VNFs can be a bottleneck of the application performance if the initial capacity of the VNFs is not enough to serve the entire application.

5.1.2 | Autoscaling policies

In this scenario, we evaluate three algorithm (*NoScale-Min*, *NoScale-Max*, and *AutoScale*) for different SFC autoscaling policies in combination with two VM/VNF allocation policies (*LFF*, *MFF*). The configuration of *NoScale-Min* and *NoScale-Max* of different types of VNFs is shown in Table 3. In *NoScale-Min*, the autoscaling feature has turned off and the minimal resources are allocated to the VNFs for the entire experiment duration. As VNFs have allocated the least amount of resources without any scaling, they are always in lack of resources, which makes the network packets delayed to wait in the queue at VNFs. On the other hand, by using more CPU, bandwidth, and the same type of VNF in one SFC police, *NoScale-Max* policy allocates more than enough resources to VNFs from the beginning of the experiment without autoscaling. As a large amount of resources are allocated for VNFs from the beginning, they can be underutilized if the submitted workload is less than estimated. In *AutoScale* policy, the initial resource allocation is the same as *NoScale-Min*, which allocates the minimal resources at the beginning of the experiment. However, it continuously monitors the utilization of VNFs and increases the size of the VNF when the utilization level is over the defined threshold. In this experiment, we set the threshold at 70% for autoscaling. If it is possible to increase the capacity to twice of the current size for scaling-up in the current host (double the capacity of CPU and allocated bandwidth in this case), then scaling-out policy by duplicating the VNF of the same type is not an option. If not, we create another VNF of the current one and allocate it according to the LFF or MFF policy.

In addition to different autoscaling policies, we evaluate the effectiveness of different VM and VNF allocation policies. The *LFF* (least full first) algorithm finds the least full physical machines with less amount of allocated resources for a VM. In this algorithm, VMs and VNFs are dispersed across the data center, as a physical machine hosting a VM has less priority than a machine without any hosted VM. In contrast, *MFF* (most full first) algorithm selects the most allocated machine that has enough capacity to host the VM. The selected physical machine still has enough resources for the VM, but more resources are already allocated to the other VMs compared to the other candidate machines. The VMs are consolidated

into a smaller number of physical machines with this algorithm, which can reduce the number of machines and the energy consumption of the data center.

Furthermore, in scenarios where IDS is optional, we evaluate the influence of SFC length and the VNF processing delay on the end-to-end response time. By removing VNF4 from the SFC policies shown in the Table 4, the total length of SFC policies is changed from 7 to 5. We combine three autoscaling policies with two VM/VNF allocation policies to create six cases and present in the following.

5.1.3 | Evaluation results

At first, we measure the response time of all workloads submitted to the application as shown in Figure 6. The average end-to-end response time is depicted in Figure 6A. As expected, *NoScale-Min* policies in both LFF and MFF result in exceptionally longer response time due to the lack of resources in VNFs. On the other hand, the delay is short in *NoScale-Max* policy as the sufficient resources are allocated to VNFs. In *AutoScale* policy, the response time is slightly longer than *NoScale-Max* but not as long as *NoScale-Min*, since the highly utilized VNFs can acquire more resources by the autoscaling policy after the short period of time. The observed difference is not significant between LFF and MFF in every autoscaling policy. To further explain the average response time (end-to-end delay), we separate it into processing time of application servers and network delays along SFCs.

Figure 6B and Figure 6C, respectively, show the average VM processing time at the service application servers (web, application, and database VMs) and the average network transmission time including delays at VNFs enforced in SFC. We can observe that the application processing time at VM servers remains the same across all six combinations, which explains that the processing power for web, application, and database servers is enough to process all workloads within short time. However, the network transmission time has a significant difference between different policy combinations

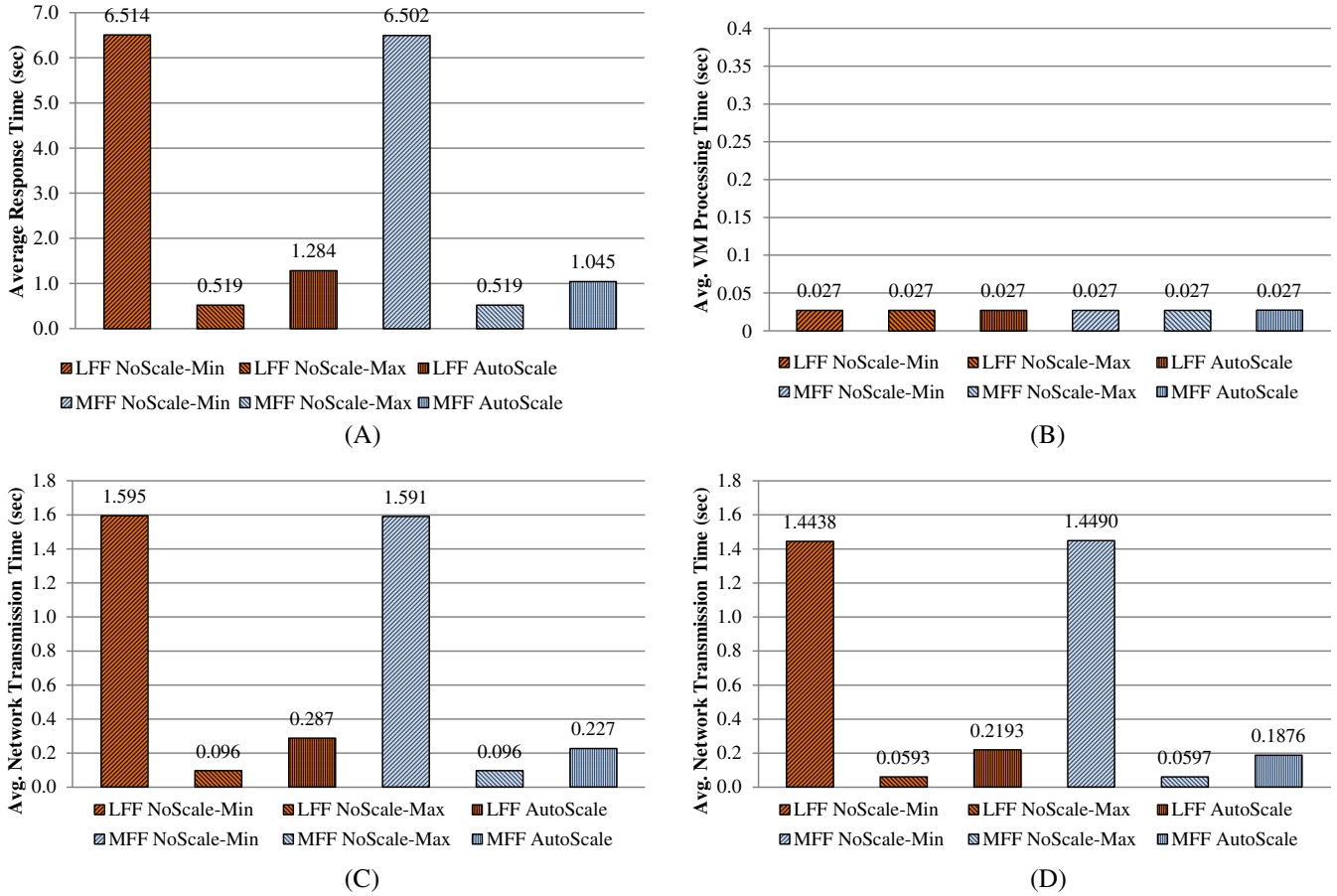

FIGURE 6 Average response and processing time with various autoscaling policies. A, Average response time; B, Average processing time at three-tier application servers; C, Average network delays enforcing through service function chaining (SFC) policies; D, Average network delays through SFC policies without intrusion detection system (IDS). LFF, least full first; MFF, most full first [Colour figure can be viewed at wileyonlinelibrary.com]

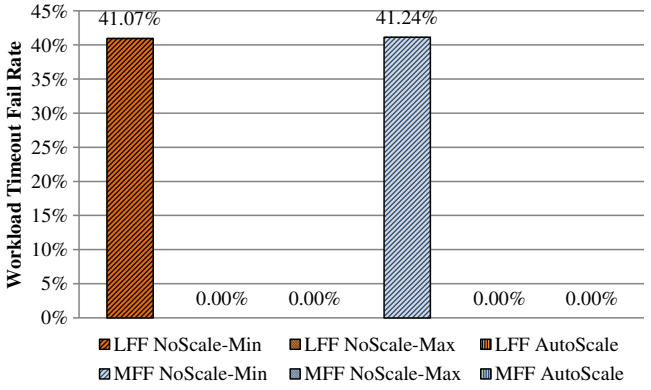

FIGURE 7 Workload failure timeout rate. LFF, least full first; MFF, most full first [Colour figure can be viewed at wileyonlinelibrary.com]

due to the delay in VNFs occurred by SFC enforcement. As the bandwidth is much sufficient for the services, it is the VNF delays along the SFCs contribute the overall end-to-end delay, which includes packet processing time and buffer waiting time. In *NoScale-Min*, the initial resources allocated for VNFs are insufficient to process the network traffic, which results in the packets delayed in SFC. *NoScale-Max* and *AutoScale* policies reduce the network transmission time significantly as the resources for VNFs are sufficient from the beginning (*NoScale-Max*) or increased to be sufficient after a while (*AutoScale*). In the case of *AutoScale*, the average network delay of LFF allocation policy is larger than the MFF. The LFF policy allocates new replicated VNF to new hosts. As a result, a more distributed allocation leads to a larger network transmission time due to more network hops. Furthermore, in the scenario of no IDS in SFCs, Figure 6D indicates that the network delays along SFCs reduced accordingly as the result of shorter SFC length.

We also measured the network timeout rate in each policy, which leads to the workload failure. The timeout threshold is set to 5 seconds so that a workload taking longer than 5 seconds is dropped and counted as timeout packet. Figure 7 depicts the timeout rate among all workloads with different policies. *NoScale-Min* policy results in dropping over 40% workload by timeout, whereas none of the workloads is dropped in the other two policies.

Then, we measured the maximum number of simultaneously used hosts. As shown in Figure 8, overall LFF algorithm uses more hosts. For *AutoScale* SFC algorithm, *MFF* is better policy compared to the LFF, which fully utilizes the computing resources. Although the simultaneously used hosts' number of *MFF AutoScale* is bigger than the *MFF NoScale-Min* as necessary to alleviate the workload failure, it is smaller compared to the *MFF NoScale-Max*. By using fewer hosts, the *MFF AutoScale* maximizes revenue generation for service provider and also offers cost saving for cloud users.

Finally, we measure the estimated energy consumption in the data center for hosts and switches, as shown in Figure 9. A clear difference can be seen between LFF and MFF, as LFF consumes more energy in any autoscaling policies compared to MFF. For example, LFF with AutoScale consumed 984.49 Wh in hosts and 474.56 Wh in switches totaling 1459.05 Wh, whereas MFF with AutoScale consumed 476.49 in total (379.66 in hosts plus 96.83 in switches). In MFF, more numbers of VMs are allocated in a single host, which leads to consolidating VMs and workloads into less number of hosts. The unused hosts can be turned to the idle mode, which can save power consumption significantly. Interestingly, more power was consumed in *AutoScale* compared to *NoScale-Max* in LFF allocation policy because more numbers of smaller VNFs were created in AutoScale due to LFF policy. In LFF, VNFs are distributed across the data center leading in utilizing more numbers of physical hosts. The duplicated VNFs created from the autoscaling policy after the initial placement are also

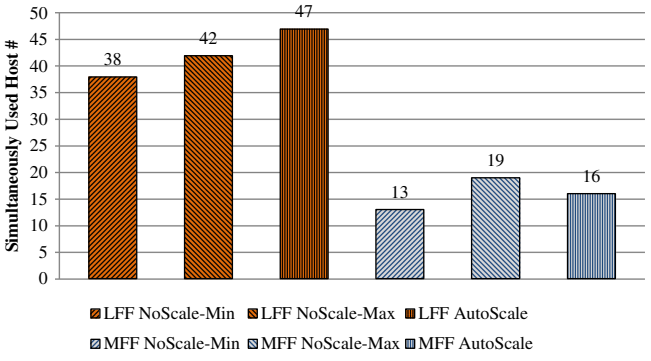

FIGURE 8 Simultaneously used hosts number. LFF, least full first; MFF, most full first [Colour figure can be viewed at wileyonlinelibrary.com]

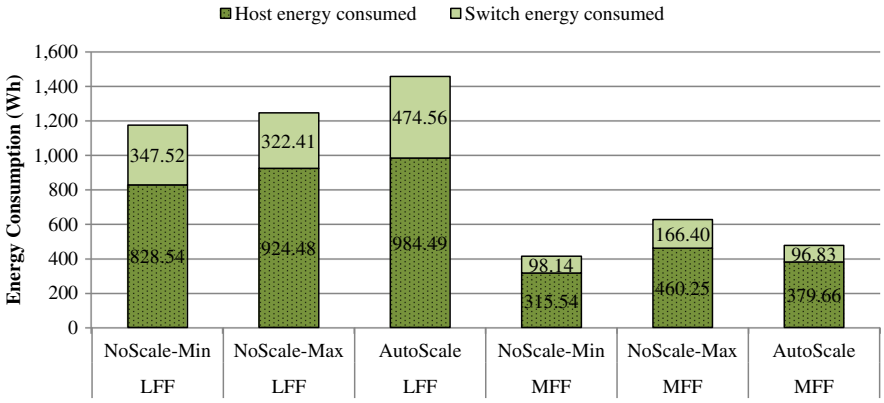

FIGURE 9 Energy consumption in hosts and switches in the data center. LFF, least full first; MFF, most full first [Colour figure can be viewed at wileyonlinelibrary.com]

dispersed leading in utilizing more physical hosts to be used. In addition, these scattered VNFs increase the volume of network traffic in the system, which results in more energy consumption in switches.

5.2 | Experiment 2: NFV placement policy in edge-cloud environment

In the second experiment, we evaluate different VNF placement policies in edge-cloud environments. The composition of various service components is a crucial technique to optimize the resource allocation and utilization of edge and cloud resources, as well as to reduce end-to-end latency for end-users. A cloud data center is built with an excessive amount of resources to provide infrastructure to serve vast applications and customers, whereas edge data centers are equipped with limited resources such as a base station or a wireless access point. Therefore, the limited number of VNFs can be placed in edge data centers, although the edge resources can help to reduce end-to-end delays for end-users as the network traffics can be processed closer to the users.

In this experiment, we created an edge and a cloud data center (ie, network data center) to place VNFs, in addition to the extra data center (ie, application data center) to host application servers. The edge data center has limited capacity with 4 hosts (each with 16 cores and 10 000 MIPS/core), whereas two cloud data centers (one for application servers and another for network functions) have the very large amount of resources to host an almost infinite number of VMs and VNFs. End-users are placed within the edge data center. The network latency of a link within the same data center is set to 1 ms. Edge and cloud data centers are connected through four intercloud switches linearly connected to each other with the network latency set to 100 ms. Thus, a network latency for intercloud traffic can be up to 400 ms, while less than 10 ms within the same data center. In this experiment, we use a canonical tree topology to simplify the creation of various data centers.

The application is composed of 12 servers running as VMs and placed in the application data center, which is accessed by 4 end-users created in the edge data center. All network traffics from the end-users to the application servers are enforced to go through VNF1, and the traffics of opposite direction are through VNF2. In this experiment, the autoscaling policy is always turned on, so that the number of VNFs to serve the same type of function can be increased if the utilization of the VNF is over the threshold, same as the autoscaling policy experimented in the previous scenario. When a VNF is duplicated to serve more network traffics, the new VNF can be placed either in edge or clouds. We use different policies to select where to place the duplicated VNFs.

5.2.1 | VNF placement policies

We implemented two different placement policies in the simulation to select a place for VNFs. *Cloud-only* policy is to utilize only network cloud data center for placing VNFs and no edge resources. As cloud data centers can provide large capacity, this policy does not consider the resource requirement of the VNF and places all VNFs in the cloud. On the other hand, *Cloud+Edge* policy considers the edge data center as the option, so that if there are available resources in the edge, the new VNF is placed in the edge first. If the edge data center has insufficient resources, the cloud data center will host the VNF. Note that the edge data center itself has limited resources insufficient to place the required VNFs for the whole workload, therefore the edge-only policy was not considered in the experiment.

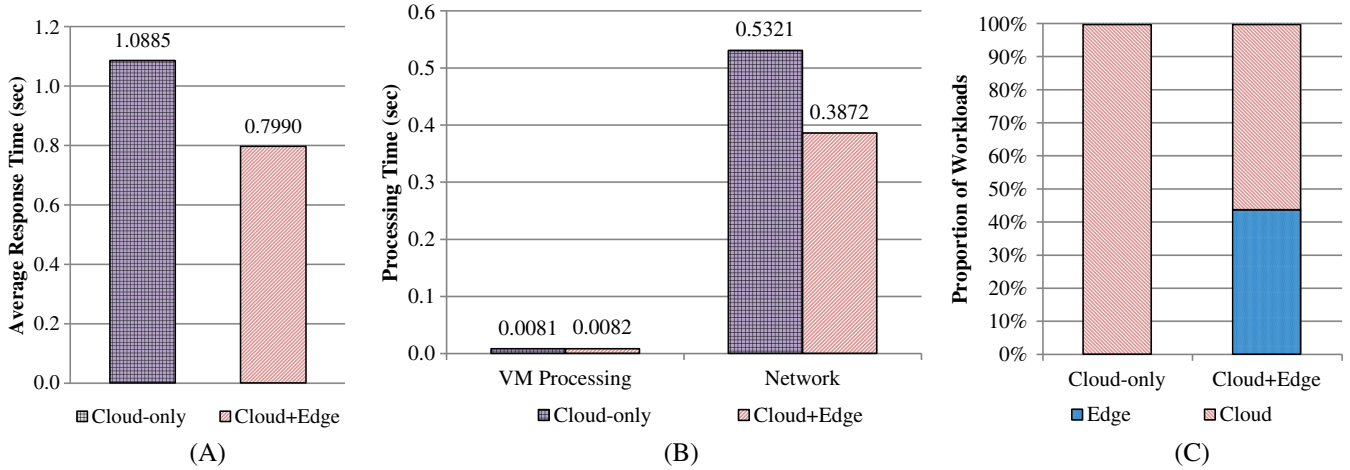

FIGURE 10 Performance evaluation results of virtualized network function (VNF) placement policies in edge-cloud environment. A, Average response time; B, Virtual machine (VM) processing time at application servers and network transmission time including service function chaining (SFC) delays; C, Workload proportion [Colour figure can be viewed at wileyonlinelibrary.com]

5.2.2 | Evaluation results

Figure 10 shows the average response time, CPU and network processing time, and the proportion of workloads processed in each data center, measured in the simulation. Average application response time is measured from the time that end-users send a request to the application server via VNFs until the response arrives at the end-users. Both request and response are enforced to forward through VNFs. As depicted in Figure 10A, the average response time is reduced by 26.6% in Cloud+Edge policy from 1.0885 seconds with Cloud-only policy to 0.7990 seconds, thanks to the reduced network latency by placing VNFs in edge resources closer to end-users. The VM processing time in application servers is measured similarly at approximate 0.008 seconds in both policies, but the network transmission time has a significant difference between two policies (see Figure 10B). This result shows that the reduction of network transmission time in Cloud+Edge policy is the reason for the improved response time. The last figure (10C) shows the proportion of VNF workloads processed in edge and cloud data centers. In Cloud-only policy, all packets are obviously processed only in the network cloud data center, whereas about 45% packets are processed in the edge data center with Cloud+Edge policy. The rests (55% of workloads) are still processed in the network cloud, due to the lack of resources in edge data centers for the newly duplicated VNFs by autoscaling policy.

6 | FUTURE DIRECTIONS AND EXTENSIONS

As an event-driven simulation framework, CloudSimSDN-NFV supports evaluation of both networking and computing resource management in the context of edge and cloud computing. For the networking processing, we have already evaluated the accuracy compared with Mininet.[11] However, when it comes to the scenarios that require high precision evaluation and prediction, such as low level infrastructure-related parameters, it depends on the profiling and modeling. By profiling the real data and identifying the acquired parameters, we can translate the mathematical models to the simulation framework with higher precision. In this section, we discuss our thoughts on how CloudSimSDN-NFV can be extended comprehensively to support (1) OpenFlow-like Capabilities, (2) live migration evaluation, and (3) energy modeling.

6.1 | OpenFlow-like capabilities

Current CloudSimSDN-NFV enables several SDN functionalities such as dynamic proactive packet routing, forwarding a packet to set of ports, and modifying the destination of a packet or dropping the packet. As OpenFlow[29] is the de facto SDN protocol, it can facilitate the functionalities through proactive and reactive flows based on packet-in and packet-out between controller and switch. For OpenFlow packet-in and packet-out simulation, we can extend the current class of packet and specify the packet label for the OpenFlow protocol. Furthermore, based on the event-driven simulation principles, several events according to the user-defined algorithms and policies can be created in the class of *SDNHost*,

SDNDataCenter, and *NetworkingOperationSystem*. These features can be used in evaluating new algorithms for placement of distributed SDN controllers. Such algorithms aim at minimizing the communication delays of packet-in/packet-out messages between SDN-enabled switches and the SDN controllers.

6.2 | Comprehensive modeling of live migration

Generally, existing works use live migration as an approach for dynamically allocating VMs[8,12] to realize the objectives such as optimal end-to-end delay and energy consumption. However, live migration can significantly influence the QoS and SLA by using both computing and networking resources. The current framework supports the mechanism of live migration that allows instance (VNF/VM/container) stops on the source host and resumes at the destination host with available resource checking. Nevertheless, it does not consider the emulation model of iterative dirty memory copy through network transmission during the migration. It only pauses the processing of workload in the *CloudletScheduler* and reroutes the packet to the new destination when the migration is finished. For a comprehensive live migration model, we can extend it based on the different phases and the prediction modeling.[30] That is, we can define the migration behaviors in the event processing functions of the *SDNDataCenter* and *NetworkingOperationSystem* according to the live migration phases: premigration, dirty page synchronization, stop-and-copy, and postmigration phases.[30] For example, with the support of the current Channel manager and Cloudlet scheduler, it pauses the packet transmission and workload processing within the migrating instance during the downtime of live migration. Therefore, the overheads and performance of live migration such as live migration time, downtime, and the amount of transferred data can be evaluated. On top of the live migration modeling, we can schedule the migration flows and multiple live migration planning with the global network topology.

6.3 | Energy modeling

Researchers who focus on the performance comparison between the current energy models and the state-of-the-art model of NFV can use CloudSimSDN-NFV directly to produce relatively accurate results because NFV as the software is still using the same networking resources (switches)[26] and computing resources (physical hosts in cloud data centers).[25] Power modeling of the virtualized data center can be categorized into analytic power estimation modeling[25] and empirical measurement-based characterization.[31] In addition, we need to consider other parameters for comprehensive and integrated energy modeling, such as (1) power consumption model for other devices, ie, storage and memory, and (2) temperature flow model between different components and servers. Based on the currently available monitoring ability and extension flexibility, many researchers have proposed new solutions for energy modeling. Louis et al[32] extended the CloudSim to simulate the storage power consumption in the data centers. Ilager et al[33] also extended the components of the server energy model in CloudSim by taking runtime temperature into consideration. They modeled the hot spots generated by the running physical hosts, which have a significant impact on the energy cost for the whole cooling and power solution of the cloud and edge computing. Furthermore, researchers can also implement other power consumption models for VM/VNFs/container as mathematical models[34] in virtualized environments and OpenFlow switches[35] for SDN network. In summary, future directions for energy modeling of NFV and other virtualized features in the cloud and edge computing should integrate the energy models of memory, disk, and cooling by characterizing the low-level software behavior.

7 | SUMMARY AND CONCLUSIONS

The advancement of virtualization technology and exploding computing capacity lead to several innovative paradigm shifts in computing and networking industry, including the emergence of cloud computing, NFV, SDN, and edge computing. Cloud computing and SDN have been studied in the research community for a decade and widely accepted in the industry, whereas NFV and edge computing are still in the early stage of research and implementation in real world. Although many state-of-the-art works are presented in the literature, the simple and quick evaluation framework can foster the advancement exponentially.

In this paper, we propose CloudSimSDN-NFV, a new simulation framework to evaluate NFV functionalities in edge and cloud computing along with other SDN functionalities and cloud computing environments. The framework is designed and developed upon CloudSimSDN, which is built on the well-studied CloudSim toolkit. We described the modeling and simulation of NFV and edge computing and the detailed design and implementation of our framework. Two use

case scenarios are presented to help understand how to use the new tool, and several algorithms are implemented and evaluated upon the framework. The results show that our framework can be efficiently exploited for quick evaluation of various approaches in the simulation.

To improve the proposed framework, more in-built policies can be added and implemented for VNF placement and SFC composition. For example, a network topology-aware VNF policy can be implemented and tested within the simulation, as well as an SFC composition policy working with multiple network data centers. Under the proposed simulation framework, we elaborated the potential extensions and future directions in respect of OpenFlow-like capabilities, comprehensive live migration modeling, and energy modeling. Therefore, we expect that our simulation framework can empower researchers in conducting advanced investigations in NFV, edge, and cloud computing to foster new innovations.

ACKNOWLEDGEMENTS

This work is partially supported by an ARC Discovery Project. We thank Huaming Wu and Adel Nadjaran Toosi for the valuable discussion and feedback to improve the quality of the simulation framework.

SOFTWARE AVAILABILITY

This software is released in open source as CloudSimSDN v2.0. The code can be downloaded from https://github.com/Cloudslab/CloudSimSDN.

ORCID

Jungmin Son https://orcid.org/0000-0001-6278-2368
TianZhang He https://orcid.org/0000-0002-5472-7681
Rajkumar Buyya https://orcid.org/0000-0001-9754-6496

REFERENCES

1. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. *Future Gener Comput Syst*. 2009;25(6):599-616.
2. Son J, Buyya R. A taxonomy of software-defined networking (SDN)-enabled cloud computing. *ACM Comput Surv*. 2018;51(3). Article No. 59.
3. European Telecommunications Standards Institute (ETSI). Network functions virtualisation. 2018.
4. Chang C, Srirama SN, Buyya R. Internet of Things (IoT) and new computing paradigms. In: *Fog and Edge Computing: Principles and Paradigms*. New York, NY: John Wiley & Sons; 2019.
5. Boubendir A, Bertin E, Simoni N. On-demand, dynamic and at-the-edge VNF deployment model application to Web Real-Time Communications. In: Proceedings of the 12th International Conference on Network and Service Management (CNSM); 2016; Montreal, Canada.
6. Dominicini CK, Vassoler GL, Meneses LF, Villaca RS, Ribeiro MRN, Martinello M. VirtPhy: fully programmable NFV orchestration architecture for edge data centers. *IEEE Trans Netw Serv Manag*. 2017;14(4):817-830.
7. Cziva R, Pezaros DP. Container network functions: bringing NFV to the network edge. *IEEE Commun Mag*. 2017;55(6):24-31.
8. Cziva R, Anagnostopoulos C, Pezaros DP. Dynamic, latency-optimal vNF placement at the network edge. In: Proceedings of the IEEE INFOCOM 2018 - IEEE Conference on Computer Communications; 2018; Honolulu, HI.
9. Mijumbi R, Serrat J, Gorricho J, Latre S, Charalambides M, Lopez D. Management and orchestration challenges in network functions virtualization. *IEEE Commun Mag*. 2016;54(1):98-105.
10. van Lingen F, Yannuzzi M, Jain A, et al. The unavoidable convergence of NFV, 5G, and fog: a model-driven approach to bridge cloud and edge. *IEEE Commun Mag*. 2017;55(8):28-35.
11. Son J, Dastjerdi AV, Calheiros RN, Ji X, Yoon Y, Buyya R. CloudSimSDN: modeling and simulation of software-defined cloud data centers. In: Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing; 2015; Shenzhen, China.
12. Son J, Dastjerdi AV, Calheiros RN, Buyya R. SLA-aware and energy-efficient dynamic overbooking in SDN-based cloud data centers. *IEEE Trans Sustain Comput*. 2017;2(2):76-89.
13. Son J, Buyya R. Priority-aware VM allocation and network bandwidth provisioning in software-defined networking (SDN)-enabled clouds. *IEEE Trans Sustain Comput*. 2019;4(1):17-28.
14. European Telecommunications Standards Institute (ETSI). Open Source NFV Management and Orchestration (MANO). 2018.
15. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw Pract Exper*. 2011;41(1):23-50.
16. Garg SK, Buyya R. NetworkCloudSim: modelling parallel applications in cloud simulations. In: Proceedings of the Fourth IEEE International Conference on Utility and Cloud Computing; 2011; Washington, DC.

17. Piraghaj SF, Dastjerdi AV, Calheiros RN, Buyya R. ContainerCloudSim: an environment for modeling and simulation of containers in cloud data centers. *Softw Pract Exper*. 2017;47(4):505-521.
18. Gupta H, Vahid Dastjerdi A, Ghosh SK, Buyya R. iFogSim: a toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments. *Softw Pract Exper*. 2017;47(9):1275-1296.
19. Filho MCS, Oliveira RL, Monteiro CC, Inácio PRM, Freire MM. CloudSim plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. In: Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM); 2017; Lisbon, Portugal.
20. Lantz B, Heller B, McKeown N. A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks; 2010; Monterey, CA.
21. Riggio R, Khan SN, Subramanya T, Yahia IGB, Lopez D. LightMANO: converging NFV and SDN at the edges of the network. In: Proceedings of the 2018 IEEE/IFIP Network Operations and Management Symposium; 2018; Taipei, Taiwan.
22. OpenStack Tacker. OpenStack service for NFV orchestration. 2019. https://docs.openstack.org/tacker/latest/. Accessed July 5, 2019.
23. Son J, Buyya R. Latency-aware virtualized network function provisioning for distributed edge clouds. *J Syst Softw*. 2019;152:24-31.
24. Toosi AN, Son J, Chi Q, Buyya R. ElasticSFC: auto-scaling techniques for elastic service function chaining in network functions virtualization-based clouds. *J Syst Softw*. 2019;152:108-119.
25. Pelley S, Meisner D, Wenisch TF, VanGilder JW. Understanding and abstracting total data center power. In: Proceedings of the Workshop on Energy-Efficient Design (WEED 2009); 2009; Austin, TX.
26. Wang X, Yao Y, Wang X, Lu K, Cao Q. CARPO: correlation-aware power optimization in data center networks. In: Proceedings of the 2012 IEEE INFOCOM; 2012; Orlando, FL.
27. Ersoz D, Yousif MS, Das CR. Characterizing network traffic in a cluster-based, multi-tier data center. In: Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS '07); 2007; Toronto, Canada.
28. Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture. In: Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication; 2008; Seattle, WA.
29. McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Comput Commun Rev*. 2008;38(2):69-74.
30. He T, Toosi AN, Buyya R. Performance evaluation of live virtual machine migration in SDN-enabled cloud data centers. *J Parallel Distributed Comput*. 2019;131:55-68.
31. Aroca JA, Chatzipapas A, Anta AF, Mancuso V. A measurement-based characterization of the energy consumption in data center servers. *IEEE J Sel Areas Commun*. 2015;33(12):2863-2877.
32. Louis B, Mitra K, Saguna S, Åhlund C. CloudSimDisk: energy-aware storage simulation in CloudSim. In: Proceedings of the 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC); 2015; Limassol, Cyprus.
33. Ilager S, Ramamohanarao K, Buyya R. ETAS: energy and thermal-aware dynamic virtual machine consolidation in cloud data center with proactive hotspot mitigation. *Concurrency Computat Pract Exper*. 2019;31(17):1-15.
34. Pedram M, Hwang I. Power and performance modeling in a virtualized server system. In: Proceedings of the 39th International Conference on Parallel Processing Workshops; 2010; San Diego, CA.
35. Kaup F, Melnikowitsch S, Hausheer D. Measuring and modeling the power consumption of OpenFlow switches. In: Proceedings of the IEEE 10th International Conference on Network and Service Management (CNSM) and Workshop; 2014; Rio de Janeiro, Brazil.

How to cite this article: Son J, He TZ, Buyya R. CloudSimSDN-NFV: Modeling and simulation of network function virtualization and service function chaining in edge computing environments. *Softw: Pract Exper*. 2019;1–17. https://doi.org/10.1002/spe.2755