VFCSIM: A simulation framework for real-time multi-service Virtual Function Chains deployment

Vassilios Tsakanikas
Smart Internet Technologies (SuITE)
Division of Computer Science and Informatics
School of Engineering
London, UK
tsakaniv@lsbu.ac.uk

Tasos Dagiuklas
Smart Internet Technologies (SuITE)
Division of Computer Science and Informatics
School of Engineering
London, UK
tdagiuklas@lsbu.ac.uk

***Abstract*—Modelling service deployment on distributed, heterogenous and dynamic environments usually involves the integration of multiple software components under a common deployment. Such deployments have gained focus lately, not only because of the rise of the Edge Computing paradigm but also due to Clouds and micro-Clouds Computing infrastructures. A novel approach for this deployment is Virtual Function Chains, according to which a service is decomposed to a set of Virtual Functions and each Virtual Function is undertaken by a different device. We propose a Virtual Function Chain Simulation ($VFCSIM$) Framework, an opensource library for building simulation models of both edge and cloud networked systems, based on Virtual Function Chains. VFCSIM's central structure is a heterogeneous network, which can describe, via scripting, a wide set of devices, network links, cloud resources, cost models and services. A case study is presented, according to which final users request on demand surveillance services from an edge network. After describing the simulation setup steps, both the summarization results and the real-time probing metrics are presented. VFCSIM is expected to enhance and facilitate the deployment of the Virtual Function Chaining paradigm both to cloud and edge infrastructures.**

***Index Terms*—simulator, edge computing, cloud computing, streaming services, multiprocessing**

I. Introduction

The technological developments of the last decade in the area of networking and embedded systems have resulted to an exponential increase of micro-processing units, which can be deployed near the production of primal data. These micro-processing units facilitated in the formulation of the Edge Computing paradigm. Edge Computing is an umbrella term which describes an extensive range of computing concepts, such as mobile computing, multi-access edge computing (MEC) [1], open edge computing [2] and fog computing [3]. The underline denominator of the aforementioned approaches is the effort to minimize the network distance and thus increase proximity between data (usually generated by sensing devices and communicated through streaming protocols) and processing devices. Time sensitive services benefit from the edge computing model, as the round-trip network delay, implied by the cloud computing based services, is omitted. Additionally, services which can not access cloud infrastructures can explore Edge computing for deployment.

Furthermore, the network infrastructure of edge computing provides a lower transmission delay than the cloud computing because the clients do not encounter the wide area network (WAN) delay in the edge computing. Thus, services like video analytics or biosignals processing can be implemented more efficiently on the edge of the network [4].

Both academia and industry are showing an increasing interest towards edge computing and many researchers and engineers are designing new models and novel approaches. Such approaches though pose a challenge from the design point of view, as a wide set of parameters and performance criteria need to be considered. During the design phase, there are three options to be explored: (i) cloud environments, (ii) experimental test beds and (iii) simulators, in order to evaluate the proposed schemes. For each option, certain advantages and disadvantages can be discussed. An actual cloud environment is usually costly and requires special virtualization frameworks for deployment. Likewise, designing on experimental test beds brings in difficulties regarding the expandability of the experiments and scalability of the proposed architectures. On top of that, as the main category of edge processing units are mobile systems, emulating their software environment is a challenging task.

As a result, engineers and researchers usually use simulation frameworks to optimize and evaluate their edge computing architectures and designs. While simulators offer many advantages, various modelling challenges need to be tackled by developers. Compared with mainstream cloud computing simulators, edge computing environment simulators need to incorporate heterogeneous devices, limited computing resources and networking protocols, which require a more complex framework to be developed. Thus, while cloud environment simulators can inspire and direct the device of edge based simulators, they can not be used 'off the shelf' for simulating edge environments.

A second challenge to be addressed when designing an edge environment simulator is the mobility of the edge devices. Relative positioning of the edge devices plays an important role when point-to-point communication protocols are applied (e.g., BLE – Low Energy Bluetooth). Finally, edge based setups are highly fluctuating environments where edge devices

become available or unavailable constantly. This environment fluctuations is a characteristic that is not explored in cloud based simulators, as the infrastructure is expected to be stable at a high degree.

A service deployed on an Edge environment can be conceived as a set of independent functions which communicate with each other to realize the service's objective, and typically interact in a sequential order, especially in IoT scenarios demanding for sense-process-actuate workflows [5]. These functions are composed of atomic commands and are expected to be deployed on virtualized infrastructures (e.g., container-based virtualization) or run natively on an edge device. For instance, Docker containers are explored in Microsoft Azure IoT Edge for deploying computational processes on edge devices, and Amazon AWS Greengrass can deploy and execute Lambda functions on the Greengrass core software in edge devices.

An umbrella term for these functions is called Virtual Functions (VFs) and are inspired by the Virtual Network Functions, as a set of interconnecting and communicating VFs as Virtual Function Chains ($VFCs$). Concerning VFC models, a wide set of problems have been addressed and discussed by the research community [18]. Among the most challenging ones are:

1) VF **placement problem**, which refers to the optimal assignment of each VF instance to an edge device, so that the services constrains are met and the total environment cost is minimized.
2) VF **dimensionality problem**, which refers to the estimation of the minimum number of VF replicas required to implement the service. It is important to mention that VFC models may consider deploying multiple replicas of a VF in order to distribute the data to the edge and thus reducing the processing time of the specific VF. This approach, inspired by the micro-services model [17], can enable the deployment to heavy processing VFs to low-end edge devices.
3) VF **migration problem**, which refers to the transparent transfer of the running status of a VF when the undertaking device needs to be removed from the VFC. While the aforementioned problems have been discussed in the literature, new approaches appear constantly which need to be evaluated.

For this, we consider a simulation framework suitable for modeling VFC architectures and schemes that would be of great usefulness to engineers and researchers. This work proposes $VFCSIM$, a simulator for VFC architectures, both in the Edge and in Cloud. While other simulation environments can be used for simulating $VFCs$, they require substantial effort for doing so.

The novelty of $VFCSIM$ is that is can natively support VF and $VFCs$. Most important, $VFCSIM$ can support real-time interaction with the simulations, enabling the ad hoc alteration of the edge environment, edge network and deployed services. $VFCSIM$ can be used for simulations on Edge environments, as well as in Cloud environments. Finally, $VFCSIM$ can support multi user access support, which is crucial for edge environments. $VFCSIM$ is an open source project and is publicly available at github.com/vtsakan/VFCSIM.

The organization of this paper is as follows. In Section II, the background of the edge computing and the related works are detailed while in Section III, the design and the architecture of the $VFCSIM$ is detailed. In Section IV, a case study which utilize the proposed simulation framework is presented while Section V concludes the wok and describes the possible directions for the future work.

II. State of the Art

Edge computing, fog computing, open edge computing, and MEC, while presenting some dissimilarities, can be considered as similar conceptions. In the present study we use the term edge computing, for consistency. A useful review on

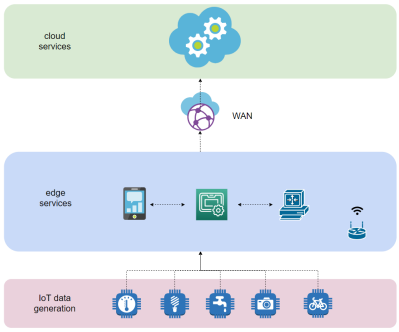

Fig. 1. Cloud - Edge - IoT conceptual architecture.

the evolution of the edge computing paradigm is provided by Taleb *et al.* [6]. Satyanarayanan *et al.* introduced the Cloudlet paradigm with the work reported in [7]. Cloudlets can be thought as micro-clouds close to mobile users, which can deploy and manage their own virtualized environments. About the same period, Cisco © proposed fog computing [3], a concept comparable to the main ideas as cloudlets. Fog computing architectures most usually comprise devices placed in the edge of the network, with the capacity of wireless communication.

The aforementioned approaches draw the attention of the telecommunication service providers and public institutions. Thus, European Telecommunications Standards Institute (ETSI) introduced the mobile edge computing (MEC) concept, aiming to adjust the edge computing paradigm in mobile cellular networks [8]. The objective of this concept is to enable the support of real time access to high-end services by adjusting the edge and cloud computing capabilities into the edge of the radio access network (RAN). To expand the MEC paradigm from cellular networks to alternative wireless access technologies (e.g., WiFi), ETSI modified the name of the concept to multiaccess edge computing (keeping though the abbreviation the same). A typical edge computing architecture and end user devices are depicted in Fig. 1. The proliferation of cloud based services resulted to numerous cloud computing

TABLE I
COMPARISON OF THE WELL-ESTABLISHED SIMULATORS.

		Resource modelling	Dynamic simulation	Virtualization modelling
Cloud	**GreenCloud**	-	-	✓
	iCanCloud	✓	-	-
	CloudSim	✓	✓	-
Edge	**IoTSim**	✓	-	✓
	SimIoT	-	✓	-
	iFogSim	✓	-	✓

simulators [9]. Basically, most of these simulators provide the computational models for the virtualized cloud environments, which can simulate the basic aspects of the virtual nodes, such as CPU, RAM memory, storage and energy consumption. The models incorporated in the simulators usually consider at least three of the aforementioned characteristics.

A. Cloud simulators

Three of the most well established cloud simulators are GreenCloud [10], iCanCloud [11] and CloudSim [12]. GreenCloud (an add-on of the well-known NS2 simulator) is a framework which has advanced models for energy usage and consumption for both communication and computational tasks. GreenCloud can utilize the full TCP/IP stack from the NS2 library, enabling the detailed modeling of the energy consumption on the network elements. Nonetheless, NS2 backend requires substantial CPU and memory capacity, affecting the simulation time of the produced scenarios. Built on OMNeT++, iCanCloud is another well-established cloud simulator. The comparative advantage of iCanCloud is the capacity to simulate large-scale environments with thousands of nodes, by supporting great extensibility, scalability and performance indicators. iCanCloud provides a graphical user interface to describe the simulation scenario.

When it comes to simulate Infrastructure-as-a-Service cloud computing environments, CloudSim is probably the optimal option, as it is designed for modeling both cloud components such as datacenters, virtual machines and hosts, and source provisioning policies such as CPU, RAM memory, storage, and network communication models.

B. Edge simulators

Edge environments, compared with the cloud setups, share diverse characteristics, as far as devices, virtualization environments, networking, user access and service deployment. For this, additional aspects need to be integrated in the simulation frameworks which plan to support edge computing environments. These aspects can be summarized in three categories: (1) Edge node mobility profiles, as mobility is usually not considered in cloud environments, (2) Data generation devices (e.g. sensors) integration and (3) Reliability profiles for the edge devices. For supporting these capabilities, simulators specialized for edge environments have been introduced. Among the most popular ones are SimIoT [13], IOTSim [14], and iFogSim [14]. IOTSim is simulator which is built upon the engine of CloudSim. It is proposed for simulations which require large volumes of data to be sent out to cloud infrastructures. SimIoT is developed by extending the SimIC([16]) simulator and, in principle, it integrates an IoT layer to the SimIC, allowing edge devices to request and access cloud resources. Finally, iFogSim runs on top of CloudSim and it has been developed for simulating IoT and fog environments by modeling components like the sensors, actuators, fog devices and the cloud infrastructures. iFogSim supports the establishment service access from edge devices from fog servers. Yet, this information is static and can not be updated during a running scenario. Table I summarizes the reviewed simulators, in terms of their capacity to model resource allocation, virtualization technologies and dynamic changes during the execution of a simulation.

$VFCSIM$ introduces a simulation framework which can be used to simulate scenarios based on virtual functions, both on edge and cloud environments. The main novel aspects of the proposed framework are: (i) native support of VF and $VFCs$, (ii) real-time interaction with the simulation environment, (iii) common simulation environment for Edge and Cloud based simulations and (iv) multi user access support.

III. $VFCSIM$ SIMULATION FRAMEWORK

With the progress of modeling frameworks, and the diverse scenarios where VFC could be integrated, the scripting approach has been applied, according to which the modeler can utilize already established entities and constitute a simulation scenario, without requiring advance programming skills, especially with Python involved in the pathway. When a flexible structured framework is provided, scripting can create complex simulation scenarios by modelers, after a short training period.

$VFCSIM$ is a modelling and simulation framework created to build VFC scenarios on edge and cloud infrastructures. $VFCSIM$ is based on an object-oriented approach to define the network environment, the computational nodes (edge-cloud nodes), the services, described as a set of sequential VFs their properties, as well as the demand from the users side. This particular design approach is general enough for enabling modular, multi-services scenarios and specific enough, so that the modeler can easily deploy different scenarios.

$VFCSIM$'s novelty lies on the fact that it is a generic VFC simulation framework, written in Python, capable of supporting multi-service modelling. The use of Python offers $VFCSIM$ particular advantages, such as:

- Python integrates numerical, statistical, scientific and visualization libraries, like numpy, pandas, seaborn, matlibplot and scipy-kit.
- Python is a recognized framework for resource modelling.
- Python integrates pyomo, a native optimization framework.

In $VFCSIM$ documentation, the following terminology is used to describe its elements: *Module*: A file containing a Python class definition, *Object*: An instance of a class (as defined within a Module), *Type*: A file containing a superclass definition, along with the relative Python decorators and

Simulation: An instance of a model with a particular set of model parameters.

A. Design

The basic classes of the simulation framework are presented in Fig. 2. More specifically, the basic entities of the framework are:

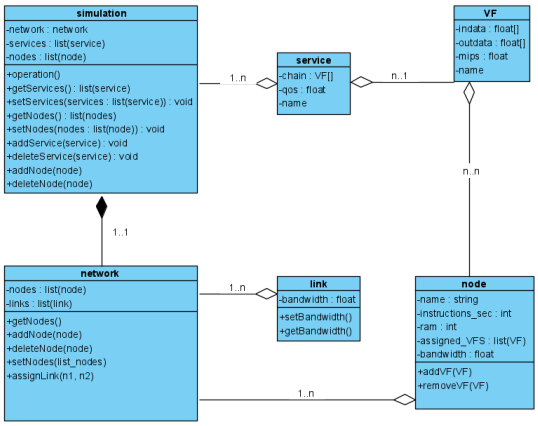

Fig. 2. UML design of the proposed simulation framework.

- ***Node***, which is used to model a computational and communication entity, as an edge device or a virtual machine. The basic attributes of the Node entity are (i) CPU mips (million instructions/sec), RAM memory volume, storage capacity and a cost function, which refers to the total operational cost of the node
- ***VF***, which is used to model a virtual function with attributes: (i) input data size, (ii) output size data, (iii) required CPU instructions and (iv) required RAM volume.
- ***NetworkLink***, which is used to model a point-to-point or a point-to-multipoint network communication link.
- ***Service***, which models a deployed service. Service's attributes include a set (list) of VFs, which actually constitute a VFC and QoS requirements which refers to the specific quality of service contract of a service.
- ***Simulation***, which is used to create a script with a specific simulation scenario.

In order to set up a simulation scenario, one should create a script which describes (i) the device environment, (ii) the deployed services and the networking infrastructure. Each simulated entity is materialized as a Python process, utilizing the multiprocess Python library. Thus, $VFCSIM$ considers a no-shared memory environment, promoting a realistic framework for the simulations. $VFCSIM$ handles the evolution of time as discrete time points. It is important to mention that the time discretization is involved on probing the relative pre-defined metrics and logging them. The actual simulation is performed on a realistic basis, according to the specific simulation scenario. Fig. 3 illustrates the basic steps for the execution of a simulation.

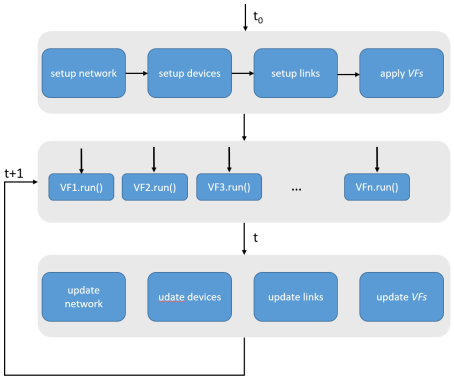

Fig. 3. Steps for simulation execution.

B. Visualization

$VFCSIM$ offers two visualization modules, which are expected to facilitate the work of the modelers. The first module parses the designed environment and produces a graph network by utilizing the NetworkX library. Thus, the modeler can inspect the designed network. It is important to mention that $VFCSIM$ offers live interaction with the simulation, which means that the modeler can intervene with the environment during the execution of a simulation. The second visualization module refers to the real-time probing of specific simulation metrics. This user interface explores the Python-Qt library and communicates with the up-running processes via UDP multicast sockets. Thus, when an edge node for example needs to report a metric, it creates a multicast packet and send to a pre-defined port. The visualization module dynamically creates a set of views for each node and for each VF, which can be later selected by the modeler (Fig. 10).

C. Validation

Aiming to verify that the entities built in $VFCSIM$ are valid, and that the simulation environment is functioning properly, we have implemented a virtual function chaining scenario in $VFCSIM$ and in a well-established simulation environment (iFogSim). The scenario involved the modeling of a set of virtual function chains, each one of them modeling a mock service. Each VFC required 3 - 8 nodes, with a different VF to be deployed to a different edge node. Each VF produced a throughput of 200kbps. By implementing the same setup in both iFogSim and $VFCSIM$, the overall network utilization has been compared (Fig. 4). One can observe that the difference between the two simulation frameworks, is approximately 1.8% (*maximum value*), and remains relatively small, even when the number of the edge devices increases.

IV. Case Study: Surveillance Services

Aiming to test the proposed simulation framework, a specific scenario has been built. More specifically, an edge environment with dynamic number of devices has been considered, upon which, users can request on demand different surveillance services with different demand probabilities. The

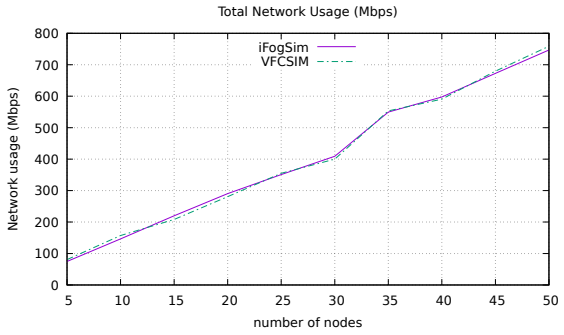

Fig. 4. Network utilization on $VFCSIM$ and iFogSim.

TABLE II
PARAMETERS ON SIMULATION SETUP.

Parameter	Value	Description
cpu_{load}	$10^4 N(10, 0.8)$	cpu mips of edge devices
m_k	$10^3 N(20, 0.9)$	VF_k required mips per data instance
Wtt'	$10^5 N(10, 0.7)$	bps between VF_t and VF'_t
$c_g(l)$	$\frac{l^2+0.8}{1000}$	function which describes the cost of g edge device when performing l instructions
$r(l)$	$\frac{20l+9}{m_k}$	function which describes required time of an edge device to perform l instructions

simulated services have been modeled as a set of $nVFs$, where n is a random integer $\in [3, 7]$, aiming to simulate different sizes of $VFCs$. Each VF could be either a light VF, a moderate VF (4 times higher computational requirements than a light VF) or a heavy VF (4 times higher computational requirements than a moderate VF), with relative computational characteristics each. Within the simulation platform, two different Setups of a surveillance service have been implemented.

- **Setup I**: The surveillance service has been implemented under a monolithic approach. This means that all VFs of the same service have been hosted in one edge device.
- **Setup II**: A VFC model, where the different VFs were deployed on different edge devices.

The two aforementioned setups have been tested under different service demand probability distributions. By service demand probability distribution, we refer to the probability a user requests a service at a specific time-point t_d. More specifically, various Poisson distributions have been used. For the Poisson distributions, three different λ parameters have been used, aiming to simulate low ($\lambda = 20$), normal ($\lambda = 40$) and high ($\lambda = 100$) user demand rates. For the same Setup, the cumulative number of the requested services was the same. The parameters of the simulation environment are given in Table II.

For this specific scenario, the modeler aims at exploring the scalability of VFC model. For this, a script has been developed which implements the aforementioned Setups under two dependent variables: (i) the number of edge devices n and (ii) the service demand distribution probability. The

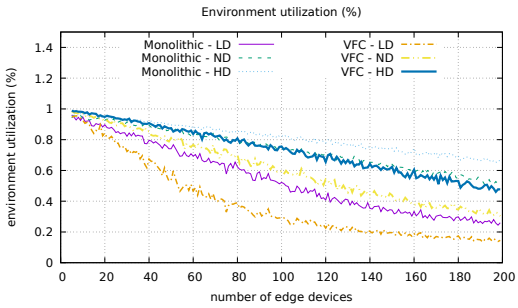

Fig. 5. Edge environment utilization on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices.

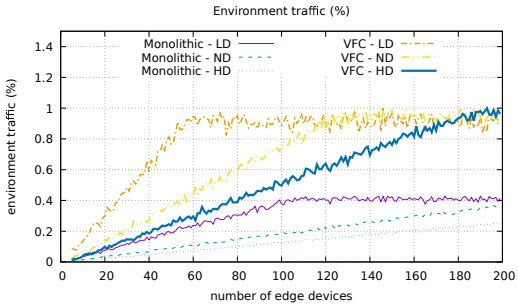

Fig. 6. Edge environment network traffic on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices

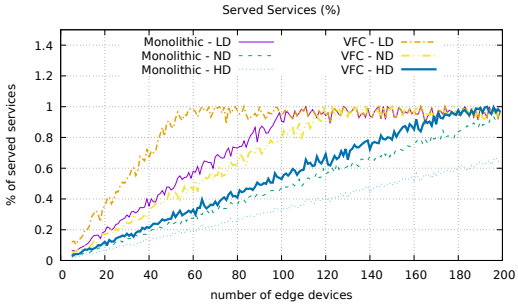

Fig. 7. Percentage of served services on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices.

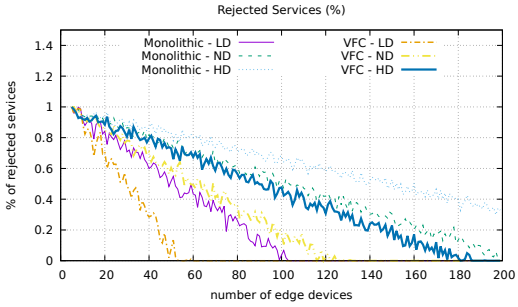

Fig. 8. Percentage of rejected services on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices.

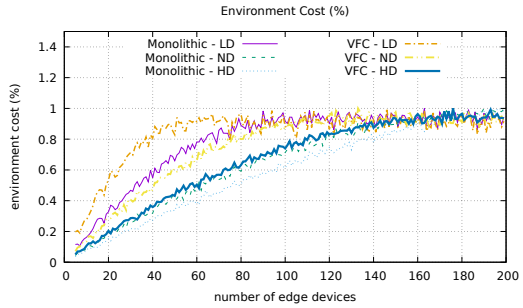

Fig. 9. Edge environment cost on different user demands (LD - Low Demand, ND - Normal Demand, HD - High Demand) and number of edge devices.

measurements of the simulation study include the following metrics: (i) the total edge environment utilization (Fig. 5), (ii) the total network traffic (Fig. 6), (iii) the percentage of successfully served services (Fig. 7), (iv)the percentage of rejected services (Fig. 8) (inadequate resources) and (v) the total edge environment cost (normalized) (Fig. 9). Finally, the $VFCSIM$ visualization module can probe specific metrics during the execution of the simulation. For example, in Fig. 10, the available RAM of a randomly selected edge device is presented, after 60 secs of simulation time.

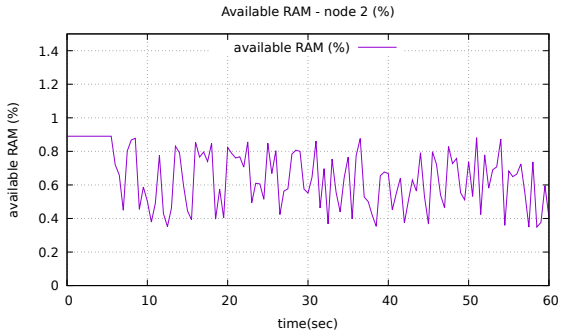

Fig. 10. Real time graph - Available RAM of an edge device.

V. Discussion

$VFCSIM$ has been applied successfully to a case study which aims to design an edge network with the capacity to undertake a set of surveillance services. The feature under study was the number of required edge devices against the service deployment mode (Monolithic vs. VFC). Via scripting, a set of simulation scenarios have been considered, with different end user demand probability distributions. From the simulation results, certain conclusions can be drawn. For instance, in order to avoid down-time of the designed system (down-time refers to the time the system can not undertake new services), the edge environment would require 58 devices for low demand, 125 devices for normal demand and about 190 devices for high demand, if the VFC model is applied. In case of the Monolithic model, the relative numbers are 110, 195 and $>$ 200 edge devices. As virtualization and dockerization technologies are recently considered not only to the cloud but also to the edge computing paradigm, an abstract approach for developing services is required. VFC model can support this design approach. For this, we propose a simulation framework which can support modellers, engineers and researchers to test architectures and solutions based on VF. The limitations of the framework summarize our future work. An integrated user interface for designing networks, devices and VF is expected to facilitate the work of the modellers. On the same direction, the integration of libraries with well-established devices and communication protocols will result to a more concrete simulation framework.

References

[1] Multi-access edge computing. 2021. [Online] http://www.etsi.org/technologies-clusters/technologies/ multi-access-edge-computing. Accessed April 02, 2021.

[2] Dinh HT, Lee C, Niyato D,Wang P. A survey of mobile cloud computing: architecture, applications, and approaches. Wirel CommunMob Comput. 2013;13(18):1587-1611.

[3] Fog computing and the internet of things extend the cloud to where the things are. [Online]. www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf Accessed Apr. 2021.

[4] Orsini G, Bade D, Lamersdorf W. CloudAware: empowering context-aware self-adaptation for mobile applications. Trans Emerg Telecommun Technol. 2017;29(4):e3210.

[5] Brogi A., Forti S., "QoS-aware deployment of IoT applications through the fog," IEEE Internet Things J., vol. 4, no. 5, pp. 1185–1192, 2017.

[6] Taleb T. et al. On multi-access edge computing: a survey of the emerging 5G network edge cloud architecture and orchestration. IEEE Commun Surv Tutor. 2017; 19(3):1657-1681.

[7] Satyanarayanan M,Bahl P,Caceres R,Davies N.The case for VM-based cloudlets in mobile computing.IEEE Perv. Comput. 2009;8(4):14-23.

[8] Farris I, Taleb T, Flinck H, Iera A. Providing ultrashort latency to user-centric 5G applications at the mobile network edge. Trans Emerg Telecommun Technol. 2017;29(4):e3169.

[9] Byrne J et al. A review of cloud computing simulation platforms and related environments, 7th International Conference on Cloud Computing and Services Science; 2017; Porto, Portugal.

[10] Kliazovich D,Bouvry P,Audzevich Y,Khan SU. GreenCloud: a packet-level simulator of energy-aware cloud computing data centers,IEEE Global Telecommunications Conference GLOBECOM 2010; Miami,FL.

[11] Núñez A, Vázquez-Poletti JL, Caminero AC, Castañé GG, Carretero J, Llorente IM. iCanCloud: a flexible and scalable cloud infrastructure simulator. J Grid Comput. 2012;10(1):185-209.

[12] 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 Exp. 2011;41(1):23-50.

[13] Sotiriadis S, Bessis N, Asimakopoulou E, MustafeeN. Towards simulating the internet of things, 28th International Conference on Advanced Information Networking and Applications Workshops; 2014; Canada.

[14] Zeng X, Garg SK, Strazdins P, Jayaraman P, Georgakopoulos D, Ranjan R. IOTSim: a cloud based simulator for analysing IoT applications. arXiv preprint arXiv:1602.06488; 2016.

[15] GuptaG, DastjerdiAV,Ghosh SK, Buyya R. iFogSim: a toolkit for modeling and simulation of resourcemanagement techniques in internet of things, edge and fog computing environments. arXiv preprint arXiv:1606.02007; 2016.

[16] Sotiriadis S, Bessis N, Antonopoulos N, Anjum A. SimIC: designing a new inter-cloud simulation platform for integrating large-scale resource management. Paper presented at: 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA); 2013; Barcelona, Spain.

[17] Sun, L., Li, Y., & Memon, R. A. (2017). An open IoT framework based on microservices architecture. China Communications, 14(2), 154-162.

[18] Gouareb R.,Friderikos V., Aghvami A. (2018). Virtual Network Functions Routing and Placement for Edge Cloud Latency Minimization. IEEE Journal on Selected Areas in Communications, 36(10)