Received: 3 December 2017 | Revised: 11 May 2018 | Accepted: 1 July 2018
DOI: 10.1002/ett.3493

SPECIAL ISSUE ARTICLE

WILEY

EdgeCloudSim: An environment for performance evaluation of edge computing systems

Cagatay Sonmez[1] | Atay Ozgovde[2] | Cem Ersoy[1]

[1]Department of Computer Engineering, Boğaziçi University, Istanbul, Turkey
[2]Department of Computer Engineering, Galatasaray University, Istanbul, Turkey

Correspondence
Cagatay Sonmez, Computer Networks Research Laboratory (NETLAB), Department of Computer Engineering, Boğaziçi University, 34342 Istanbul, Turkey.
Email: cagatay.sonmez@boun.edu.tr

Present Address
Department of Computer Engineering, Boğaziçi University, 34342 Istanbul, Turkey

Funding information
Galatasaray University Research Foundation, Grant/Award Number: 16.401.002; Boğaziçi University Research Fund, Grant/Award Number: 12663; Republic of Turkey Ministry of Development, Grant/Award Number: 2007K120610

Abstract
Edge computing is a fast growing field of research that covers a spectrum of technologies bringing the cloud computing services closer to the end user. Growing interest in this area yields many edge computing approaches that need to be evaluated and optimized. Experimenting on the real cloud environments is not always feasible due to the operational cost and the scalability. Despite increasing research activity, this field lacks a simulation tool that supports the modeling of both computational and networking resources to handle the edge computing scenarios. Existing network simulators can model the network behavior at different levels of granularity. The cloud computing simulators support the modeling and simulation of the computational infrastructures and services efficiently. Starting from the available simulators, a significant programming effort is required to obtain a simulation tool meeting the actual needs. On the other hand, designing a new edge computing tool has many challenges such as the scalability, extensibility, and modeling the mobility, network, and virtualized resources. To decrease the barriers, a new simulator tool called EdgeCloudSim streamlined for the edge computing scenarios is proposed in this work. EdgeCloudSim builds upon CloudSim to address the specific demands of edge computing research and support the necessary functionalities. To demonstrate the capabilities of EdgeCloudSim, an experiment setup based on different edge architectures is simulated. In addition, the effect of the edge server capacity and the mobility on the overall system performance are investigated.

1 | INTRODUCTION

Edge computing covers a wide range of computing concepts such as mobile cloud computing,[1] cloudlet-based computing,[2] fog computing,[3] and multiaccess edge computing (MEC).[4] The common objective of these computing paradigms is getting the service from a nearby server located in the geographical proximity of the client. Generally, the edge computing paradigm provides a clear advantage for the time sensitive applications and the mobile environments with no cloud access availability. Executing the whole application or a part of it on the edge of the network brings a significant advantage for the resource-constrained mobile devices as well. For example, the mobile devices with low computation capability can handle the complex operations while consuming less battery by utilizing the edge server. In addition, the 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. As a result, the delay intolerant applications such as real time face or speech recognition can be executed more efficiently on the edge of the network.[5]

There is an increasing trend toward the edge computing and many researchers are devising new schemes for the implementation of new approaches. A crucial challenge in proposing a new edge computing solution is to realistically assess the performance of the envisioned system while considering the relevant engineering parameters. The researchers can use the real cloud environments,[6] experimental test beds,[7] or simulators[8] to evaluate their approaches. There are pros and cons in selection of these options. Using a real cloud environment is not a cost-effective option due to the deployment requirements. On the other hand, working on the experimental test beds brings in difficulties regarding the repeatability of the experiments and scalability of the proposed architectures.[9] In addition, the clients of the edge computing systems are mostly mobile. Operating mobile clients on the real cloud environments or test beds is a challenging issue. Therefore, the researchers generally prefer to use simulators to evaluate and optimize their edge computing approaches and designs.

The simulators provide many advantages to the researchers. However, from the simulator developers' perspective, there are some modeling challenges.[10] When compared to the traditional cloud computing simulators, we envision that simulating edge computing environment will bring more complex questions and challenges. *Modeling the virtualized resources* (eg, CPU, memory, and storage) and their provisioning is the first challenge to be addressed. Due to the cloud computing simulators, we have some current solutions to exploit or get inspired from. However, the applications running on these computing paradigms have different characteristics when considering the clients (brokers). For example, the wireless sensor device applications generally have a Sense-Process-Actuate model that is beyond allocating a virtual machine (VM) on the datacenter. As a result, integrating a computational model of the virtualized resources and the fog devices requires more complex application models. *Modeling the mobility* is another challenge in this domain. In the cloud computing simulators, the mobility is generally ignored since the main focus is how to provision the virtualized resources with respect to the client requests. However, in the edge computing architectures, mobility of the fog devices and the edge servers is an important factor that cannot be ignored. As the devices become more mobile, the system design will be more challenging since the mobility affects many other modules. The third challenge is *the network modeling*. In edge computing, the interactions among the users, edge nodes, and the cloud may require different access technologies and topologies. For instance, a mobile user can transmit data to the edge node over a wireless local area network (WLAN), and the edge node forwards the incoming data to the cloud over a WAN. Using a fixed end-to-end transmission delay in such systems is not reasonable since the number of users is usually considered very large. Therefore, the modeling tool should be capable of employing different delay models to simulate various access technologies such as Wi-Fi, Bluetooth, 4G, and 5G. *The scalability* becomes another significant challenge to handle high number of the entities used in the edge computing architectures. There is a trade-off between the scalability and high level of details. If the components are abstracted at a very high level, the tool becomes unrealistic. On the other hand, if too many details are considered, it provides poor scalability. *The extensibility* is also an important factor for the research studies while choosing a simulator. The researchers usually use different architectures and application models in their studies. As a result, they need to customize and extend the modeling tool according to their requirements. Providing extensible modules is another challenge that should be taken into consideration. Finally, *the ease of analysis* is a crucial feature required by the simulator users. The results of the simulations can be analyzed regarding various aspects. Determining how many logs will be collected and the output format of them are the last challenges to be considered while developing an edge computing simulator.

A unique requirement specific to edge computing systems is the need to consider the mobility, network, and computational resources when evaluating their performances. To the best of our knowledge, there is no simulation environment within the current solutions, which handles these concepts in an efficient and easy way. Figure 1 depicts the main aspects of the edge computing simulation modeling in independent axes. Conventional network simulators such as NS2,[11] Omnet++,[12] and Riverbed Modeler[13] can model various types of network protocols and topologies effectively. Conventional network simulators such as NS2,[11] Omnet++,[12] and Riverbed Modeler[13] can model various types of network protocols and topologies effectively. However, the cloud computing elements and scenarios are not handled in these tools. Therefore, a significant implementation effort is required for modeling the computational aspects relevant to edge computing. When it comes to the cloud computing simulators, they allow the modeling of cloud computing infrastructures and the application services. One of the most popular simulation tools streamlined for the cloud computing setup is CloudSim.[14] The main drawbacks of CloudSim from the point of view of edge computing are (i) lack of the dynamic WLAN and WAN communication models, (ii) lack of the mobile nodes and mobility support in general, and (iii) lack of the realistic edge type load generator model. Generally, the researchers extend CloudSim by implementing the lacking parts with respect to the needs of their subject matter. For example, IOTSim[15] and iFogSim[16] are implemented by extending CloudSim. Although they are useful for generating relevant workload of IoT and fog computing scenarios, the mobility and dynamic network model are not considered by these tools. When all the aforementioned shortcomings are taken into

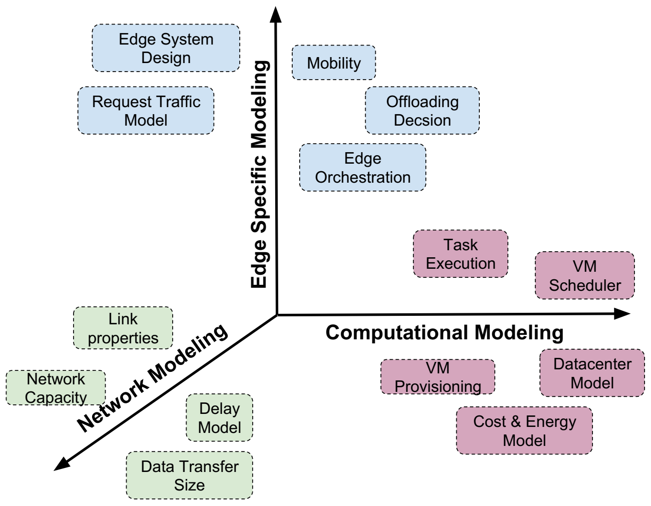

FIGURE 1 Aspects of edge computing simulation modeling. VM, virtual machine

consideration, the necessity of a new tool that can satisfy the need for realistic simulation of the edge computing scenarios can be seen.

In this work, we introduce a new simulator called EdgeCloudSim that covers the whole capabilities space presented in Figure 1. EdgeCloudSim provides a modular architecture to provide support for a variety of crucial functionalities such as network modeling specific to WLAN and WAN, device mobility model, and realistic and tunable load generator. EdgeCloudSim by design supports simulation of multitier scenarios where the multiple edge servers are running in coordination with upper layer cloud solutions. Specific to this need, EdgeCloudSim provides an edge orchestrator module to enable its users to model the orchestration actions that typically arise in edge computing scenarios. For modeling the computational tasks such as VM creation with a given capacity, EdgeCloudSim relies on the capabilities of CloudSim that has a long known and reliable code base for the simulation of computational actions. EdgeCloudSim is publicly available as an open-source project.[17] Modular design and open-source code base of EdgeCloudSim allow its users to incorporate the specific needs of their scenarios in their simulation experiments.

EdgeCloudSim can be used to analyze different engineering problems in the edge computing domain. The solution of the problems can be applied on different time scales. For example, to improve the service time, various solution methods such as increasing the capacity, changing the offloading decision, and using more efficient provisioning algorithm can be applied. Here, increasing the capacity can be handled in a broader time scale, since it requires a new deployment. On the other hand, changing the offloading decision will show its effect in a narrower time scale, because it can be done on the edge server dynamically. EdgeCloudSim can be used for evaluating different design decisions in a wide range of time scales. To demonstrate these capabilities of the EdgeCloudSim, we designed three sample simulation setups. In the first setup, different edge computing architectures with different internal organizations are examined. In the second setup, the edge server capacity planning problem is analyzed. Finally, in the third setup, the effect of the mobility is explored. These simulation setups are evaluated via an extensive number of experiments carried out with EdgeCloudSim. The results of these experiments clearly depict the effect of the networking, mobility, and architectural parameters on the overall performance of the edge computing systems.

The organization of this paper is as follows. In Section 2, the background of the edge computing and the related works are detailed. In Section 3, the design and the architecture of the EdgeCloudSim is explained. In Sections 4 and 5, example simulation setups are summarized and the results of the experiments are evaluated. Section 6 concludes our study and provides the possible directions for the future research studies.

2 | EDGE COMPUTING BACKGROUND AND RELATED WORKS

The main motivation of edge computing is basically performing computation on the nearby resources that are located at the edge of the network. The resource in the proximity may be a network resource or a computational resource in between the client and the cloud datacenters. Edge computing, cloudlet-based computing, fog computing, and MEC are in essence similar concepts. In our study, we focus on the term edge computing to maintain the consistency. The history of edge computing paradigms is explained in the work of Taleb et al.[18] The Cloudlet concept formed in light of the definition of Satyanarayanan et al was the first study in this domain.[2] The cloudlets can be considered as the micro-clouds in the proximity and the mobile users deploy and manage their own VMs over wireless networks. Among the various edge computing approaches, the cloudlets deserve further attention. Then, Cisco proposed fog computing[3] that has the similar idea with cloudlets. Fog computing architectures generally consist of many devices that are located somewhere in the edge of the network and capable of wireless communication. Finally, these approaches attracted the attention of the telecommunication service providers. European Telecommunications Standards Institute (ETSI) proposed the mobile edge computing (MEC) concept to speed up the expansion of edge computing in the cellular networks.[19] Their main motivation is providing a real time access to the services by bringing the edge and cloud computing capabilities into the edge of the radio access network (RAN). In order to cover not only the cellular networks but also different access technologies such as Wi-Fi, ETSI changed the *mobile* term in the MEC to *multiaccess* by keeping the abbreviation the same. The access points could be Wi-Fi only, cellular only, or a mix of both access technologies in MEC architectures. A typical edge computing architecture and end user devices are depicted in Figure 2.

Today, there have been proposed many cloud computing simulators as a result of the growing popularity of cloud computing studies.[20] In essence, these simulators provide the computational models for the virtualized cloud environments and they are very useful for CPU, memory and the storage models, and the energy consumption models. Some popular cloud simulators are CloudSim,[14] GreenCloud,[21] and iCanCloud.[22]

CloudSim is mainly designed for modeling Infrastructure-as-a-Service cloud computing environments.[14] It supports both modeling of the cloud components such as datacenters, hosts, and VMs and the resource provisioning policies such as CPU, storage, memory, and bandwidth utilization models. CloudSim is the most commonly used cloud simulator for the conventional cloud computing scenarios.[23] In conventional cloud computing, the clients request from the datacenter to create a list of VMs and run a list of tasks on the related VM. When the datacenter receives such a request, it tries to create the VMs and executes the related tasks. The tasks are considered as a bunch of operations that keep the VMs busy for hours. This flow of operation is similar to renting a VM from the datacenter operators such as Amazon EC2[24] and Microsoft Azure[25] for a certain time. However, edge computing scenarios differ from the conventional cloud computing scenarios in the sense that the tasks do not require too much time to be executed. The tasks offloaded by the mobile devices to the edge server is generally processed in sub-second intervals. Due to the small size of the tasks, creating a new VM for each request is not efficient.

GreenCloud is an extension of NS2 simulator.[21] The main motivation of GreenCloud simulator is to capture the energy usage of the computing and communication elements. Being an extension of NS2 provides GreenCloud to implement the full TCP/IP protocol reference model. Implementing full TCP/IP protocol reference model enables detailed modeling of the energy consumption on the network switches and links. However, this brings an overhead to the system regarding the memory usage and the simulation time that creates a disadvantage for GreenCloud.

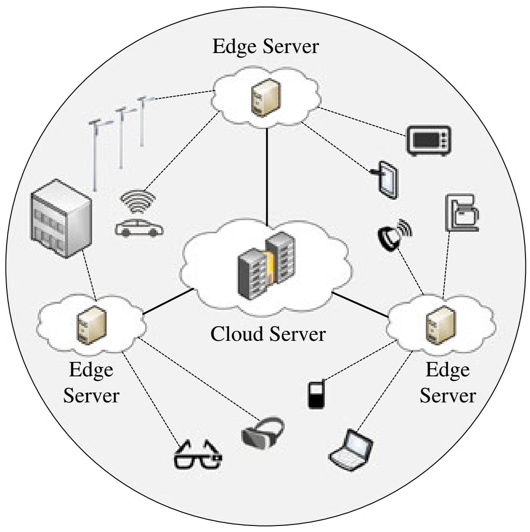

FIGURE 2 Typical edge computing architecture and end user devices

iCanCloud is an OMNeT++-based simulator platform.[22] The main aim of iCanCloud is to model and simulate the large environments with thousands of nodes. The performance and scalability are the primary objectives of iCanCloud, so that it supports executions of parallel simulations. Like the other simulators, it allows modeling of the computational aspects of the virtualized cloud environments. iCanCloud provides a graphical user interface to describe the simulation scenario.

Edge computing systems have different characteristics in terms of the users and the network topology compared to the cloud computing systems. The users of the edge computing architectures are mobile, but the cloud computing simulators do not consider mobility. The end-to-end transmission delay is generally considered as a static value in cloud computing simulators. However, the communication between the mobile users and the edge devices requires wireless interfaces that create a necessity for more detailed network modeling. In addition, the applications utilized in edge computing have also different characteristics when considering the IoT devices. In order to cover all the mentioned different requirements, new simulators focusing on the fog, edge, or IoT application scenarios are required. IOTSim,[15] SimIoT,[26] and iFogSim[16] are some of the examples of these simulators.

IOTSim is another simulator that is implemented by extending CloudSim.[15] It is proposed for simulating the edge computing environments where massive amount of data is sent to a big data processing system by the IoT application. It adds the storage and the big data processing layers into the CloudSim in order to support a big data processing system. In the storage layer, the network and storage delays are simulated for IoT applications. When it comes to the big data processing layer, it simulates MapReduce big data processing model to support the batch-oriented data processing paradigm.

SimIoT is developed by extending the SimIC[27] simulator. It basically adds an IoT layer to the SimIC in a way to allow a desired number of IoT devices to request for cloud resources. SimIoT has two critical modules, namely, the IoT device input system and the communication broker. The IoT device input system is responsible for gathering the sensor events and redirecting them to the communication broker. The communication broker converts the sensor data to the contextualized information and generates a request to be sent to the default SimIC cloud entities.

iFogSim runs on top of CloudSim.[16] It is developed for simulating IoT and fog environments by modeling components like the sensor, actuator, fog devices, and the cloud. iFogSim allows the definition of the location of devices getting service from the fog servers. However, this information is static and it is not updated by any mobility model. The network model in iFogSim is over simplistic where a fixed delay is assigned to links ignoring the effect of network load on the operation. As discussed above, for the realistic assessment of edge computing proposals, network resources and computational resources should be considered with satisfactory accuracy. To the best of our knowledge, an edge computing simulator covering aforementioned concerns has not yet been proposed. Therefore, we propose EdgeCloudSim open to the research community as a sophisticated tool based on CloudSim for the performance evaluation of the edge computing proposals.

3 | EdgeCloudSim ARCHITECTURE

EdgeCloudSim provides a modular architecture where each module addresses a specific aspect of edge computing with clearly defined interfaces to the other modules. To ease fast prototyping efforts, each module contains a default implementation that can be tuned via the simulation parameters. As explained in the previous sections, EdgeCloudSim relies on the capabilities of CloudSim for modeling the computational tasks. The VMs served by the hosts are basically the standard VMs provided by CloudSim with lower capacity and a more realistic CPU utilization model. In CloudSim, the bandwidth, RAM, CPU, and the storage resources are limited while creating a VM. When it comes to the tasks, there is no limitation for the number of tasks running on VMs. If there are many tasks to run, simply, the execution of tasks takes longer time. However, this modeling of execution time is not compatible with the edge computing approach, because we expect the edge servers to handle incoming tasks in a short time interval. Therefore, we implement a new CPU utilization model for the VMs where the number of maximum tasks running in parallel on the VMs is limited. The CPU utilization of a single task is static and it is defined in the configuration file, but a dynamic utilization model can be also implemented by rewriting the related CPU utilization class. A detailed block diagram of EdgeCloudSim modules is also shown in Figure 3.

3.1 | EdgeCloudSim modules

EdgeCloudSim uses CloudSim to provide the cloud computing capabilities such as allocating a VM on datacenters, executing a task on VMs, provisioning the cloud resources, and modeling the power consumption of the datacenters. In addition to the cloud computing capabilities, EdgeCloudSim provides other unique modules to handle the edge computing

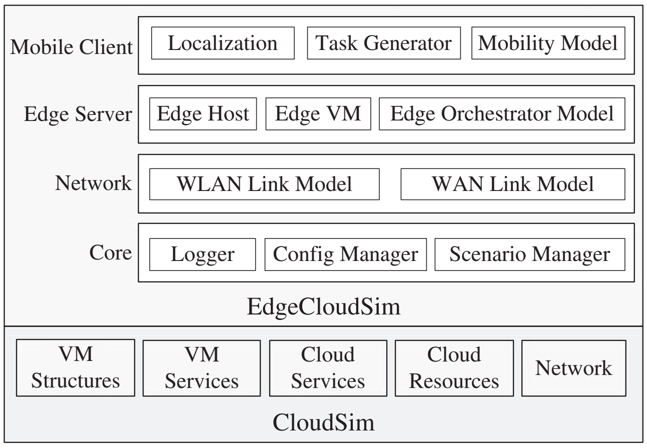

FIGURE 3 EdgeCloudSim block diagram. VM, virtual machine; WAN, wide area network; WLAN, wireless local area network

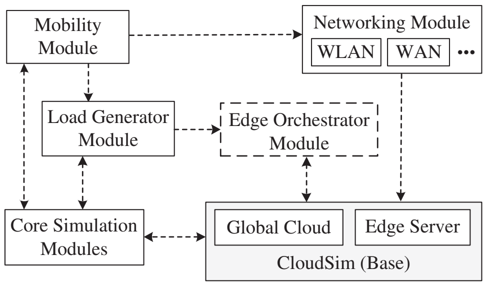

FIGURE 4 Relationship between EdgeCloudSim modules. WAN, wide area network; WLAN, wireless local area network

scenarios as depicted in Figure 4. In the current EdgeCloudSim version, there are five main modules available, namely, Core Simulation, Networking, Load Generator, Mobility, and Edge Orchestrator.

The core simulation module is mainly responsible for loading and running the edge computing scenarios from the configuration files. EdgeCloudSim makes it possible to load the properties of edge datacenters, the characteristics of applications, and the basic simulation settings dynamically. Therefore, users can run various simulation scenarios with different configuration without changing their source code. Another important feature of the core simulation model is the logging feature. Simulation logs are very important to evaluate results. EdgeCloudSim records results of the simulation in comma-separated value data format. The contents of the file and the format can be also modified according to the needs.

The edge orchestrator module is designed to fulfill the orchestration process that is a crucial issue on the edge computing systems. Mobile edge orchestrator has been already proposed by the ETSI. Similar to the mobile edge orchestrator, we propose an edge orchestrator that manages the resources in a way to increase the overall system performance. The edge orchestrator makes critical decisions such as creating new replicas, terminating the edge VMs, managing the computational resources of hosts, and offloading the tasks to the cloud or edge servers. It should collect information from the other entities to make the decision process more efficient. Therefore, it is closely coupled with the other modules. A generic edge computing architecture is given in Figure 5. There are two tiers in this architecture. The mobile devices and the edge services are located in the first tier, and the global cloud services are located in the second tier. In our design, the edge orchestrator is located between these two tiers and works in a centrally controlled manner.

The networking module particularly handles the transmission delay in the WLAN and WAN by considering both upload and download data. The mobile clients may use the cellular network or WLAN to access the cloud services. In CloudSim, it is possible to add link delays between the network components but these delays are static and fixed for all users. However, the network link quality may vary from time to time if we consider the mobile devices. In addition, if many mobile devices are using the same Wi-Fi access point, high number of devices would inevitably increase the link delay for either the local area network (LAN) or WLAN communication. In order to model LAN, metropolitan area network (MAN) and WAN communication more accurately, a networking module is provided in EdgeCloudSim. Before sending a task to a VM or downloading the result of the task from the VM, the transmission delay for the upload/download operation is calculated by the network link model. The default implementation of the networking module is based on

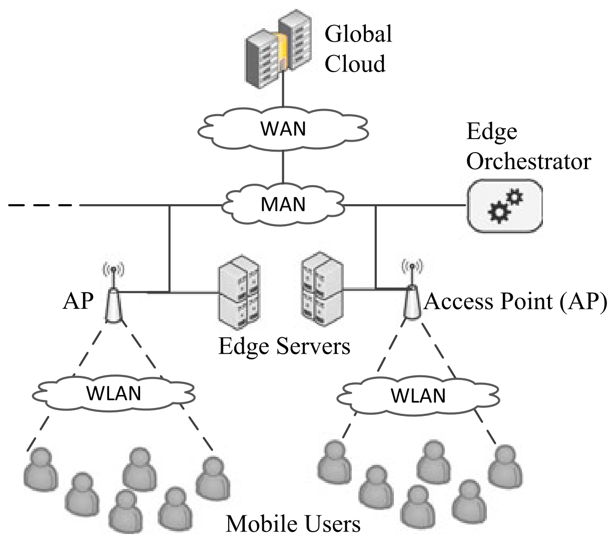

FIGURE 5 Generic edge computing architecture. MAN, metropolitan area network; WAN, wide area network; WLAN, wireless local area network

a single server queue model. The users of EdgeCloudSim can incorporate their own network behavior models into the networking module. Currently, we are working on a large scale WAN delay characterization crowdsourcing project and plan to incorporate a realistic WAN delay distribution model into the networking module.

The mobility module of EdgeCloudSim provides the mobility support that is one of the major lack in the cloud computing simulators. Since the current simulators focus on the conventional cloud computing principles, it is not considered in the framework. However, the devices used in the edge computing environments are mostly mobile. Ignoring the mobility in such systems can lead to incorrect results, because the coverage of the wireless networks is limited and the movement of the users causes problems such as congestion, latency, and packet loss. EdgeCloudSim considers the mobility, so the locations of the mobile devices are updated according to the mobility module. In our design, each mobile device has x and y coordinates that are updated according to the dynamically managed hash table. In the current version, the mobile devices move according to the nomadic mobility model, but it can be extended to a different model according to the simulation scenario to be planned. In addition to the mobile devices, we also define locations where dedicated Wi-Fi access points are utilized. As the default radio behavior, it is assumed that the coverage of the Wi-Fi access points is fixed and the link quality is the same for all of the covered clients regardless of their distance to the related access point.

The load generator module is responsible for generating tasks for the given application configuration. The mobility and load generator modules are the main components that provide input to other components. They are handled within the same (Mobile Client) layer as shown in Figure 3. Multiple application types can be defined in EdgeCloudSim since, in most of the cases, servers provide various number of services. For example, the mobile users can offload their tasks corresponding to the face recognition, health monitoring, and augmented reality applications. By default, the tasks are generated according to a Poisson arrival process in EdgeCloudSim. If other load generation models are required, the mobile device manager module should be modified. It should be noted that the data size and the length of the tasks should be decided with a proper distribution with respect to the selected task generation distribution. These variables are exponentially distributed random numbers by default because the tasks arrive according to a Poisson process.

3.2 | Ease of use and configurability

The simulators are mostly used to compare the performance of different models by considering many aspects. For example, the topology of the network, the number of entities used in the simulation, or the parameters of the algorithms can change. Handling all these variable options programmatically is a challenging issue. EdgeCloudSim uses three configuration files to decrease the programming effort and provide the configurability. The first file includes the key-value pairs as the simulation parameters such as the simulation time, warm-up period, and the number of mobile devices used in the scenario. The second file includes XML descriptions of the applications used in the simulation and their characteristics such as the task length, the input/output data size, the idle/active period of the applications, the interarrival time

```xml
<edge_devices>
    <datacenter arch="x86" os="Linux" vmm="Xen">
        <costPerBw>0.1</costPerBw>
        <costPerSec>3.0</costPerSec>
        <costPerMem>0.05</costPerMem>
        <costPerStorage>0.1</costPerStorage>
        <location>
            <x_pos>1000</x_pos>
            <y_pos>0</y_pos>
            <wlan_id>0</wlan_id>
            <attractiveness>2</attractiveness>
        </location>
        <hosts>
            <host>
                <core>16</core>
                <mips>40000</mips>
                <ram>32000</ram>
                <storage>500000</storage>
                <VMs>
                     <VM vmm="Xen">
                         <core>4</core>
                         <mips>10000</mips>
                         <ram>8000</ram>
                         <storage>125000</storage>
                     </VM>
                     <VM vmm="Xen">
                         <core>4</core>
                         <mips>10000</mips>
                         ...
```

FIGURE 6 Sample XML data for edge server configuration

of tasks generated by the related application, and the number of instructions to execute the emerging task. The characteristic of the applications is used by the load generator module while creating the tasks. The third file determines the edge server topology and the access points associated to the related location. The location of the edge access points and the specification of the edge devices are defined in this file in XML format. A part of a sample XML file is shown in Figure 6. EdgeCloudSim allows the users to run simulations with different configurations by accepting the related files as a command line option. By doing so, it makes the configuration and execution of the experiments easier.

3.3 | Extensibility

Extensibility is one of the most important features of the software tools. It provides a customizable environment in which the developers can change the behavior of components without changing the existing codebase. If the extensibility is low, the developers do not tend to use the related tool because they have to learn the details of existing architecture for customization and enhancement. EdgeCloudSim provides the default implementation of the modules mentioned earlier in this section. For example, it includes the nomadic mobility model as the mobility module and the simple queue model as the networking module. However, the users may want to implement these modules differently depending on their needs. To prevent modifying the existing codebase, EdgeCloudSim uses a factory pattern making easier to integrate the new modules. As shown in Figure 7, the scenario factory class knows the creation logic of the abstract modules and provides the concrete implementation of them. As a result, the users can utilize their own modules in EdgeCloudSim by taking the advantage of this pattern. The scenario factory is an abstract class and basically provides four different modules, which are the mobility, load generator, networking, and the edge orchestrator modules. This abstract class can be extended to support the modeling of custom modules.

3.4 | Complexity evaluation

EdgeCloudSim runs on top of CloudSim, which uses a discrete event simulation[20] engine. Therefore, the complexity of EdgeCloudSim depends on the number of events created in the simulation and the algorithms used in the modules. For example, if the designed edge orchestrator is too complex, the memory consumption is more likely to be high and the

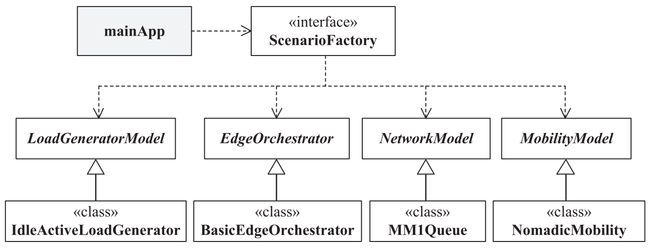

FIGURE 7 Factory pattern for simulation scenario

duration of the simulation is long. In our study, an experiment is set up to present how much memory does EdgeCloudsim use and how much time does it take to execute an ordinary simulation. A basic personal computer running on a Linux operating system with Intel core i7-5600U processor and 16 GB memory is used in the experiment. One hour long real life events are simulated under light and heavy load scenarios. The edge computing architecture used in the experiments is called as *two-tier with edge orchestrator* that consists of the mobile devices, edge servers, and cloud servers. The mobile devices operate three different applications, which correspond to the augmented reality, infotainment, and health monitoring applications. The modeling of the applications, mobility, network, and the two-tier with edge orchestrator architecture are explained in Section 4.

In the experiments, the time and memory complexities are analyzed for two different cases. In the first case, the memory consumption and the duration of the simulation processes are investigated with respect to the variable mobile devices. In the heavy load scenario, the augmented reality, infotainment, and health monitoring applications generate task offloading requests on every 2, 5, and 10 seconds in average, whereas for the light load scenario, the average interarrival time of tasks are 10, 25, and 50 seconds. Considerable number of events is generated during the simulations especially for the heavy load scenario. The results are shown in Figures 8A and 8B. It can be observed from the Figures that even a quite dense scenario with heavy load can be completed in a couple of minutes with half GB memory on a standard portable computer. In the second case, the time and memory complexities are investigated with respect to the variable number of edge servers. In this experiment, each edge server has one host that operates two VMs with 10 giga instruction per seconds (GIPS) CPU power. In the heavy load scenario, 500 mobile devices are used, whereas there are 200 mobile devices in the light load scenario. The memory consumption and the duration of the simulation process are shown in Figures 8C and 8D. It can be considered that, if the number of edge servers is high, the geographical area of the simulation scenario is also large. According to our evaluation, EdgeCloudSim can execute relatively large instances with 500 mobile devices and 50 edge servers operating 100 edge VMs in a reasonable time interval with the acceptable memory consumption.

4 | EdgeCloudSim IN OPERATION

In order to present EdgeCloudSim capabilities, three simulation setups are designed and the results are evaluated. In these setups, the effects of edge computing architecture, edge server capacity, and the mobility on the important performance metrics are investigated. We design a virtual environment that is similar to a university campus in our experiments. The students walk around and request services from the edge servers that are located at the buildings on the specific territories. Each building is also serving a wireless access point; hence, the mobile users are connected to the related access point and offload their tasks via this connection. Step-by-step explanation of how the EdgeCloudSim components are modeled to obtain an evaluation as close as possible to the real world scenarios is described in detail later in this section.

4.1 | Step 1: modeling the applications

In real life, the applications running on the mobile devices of the students can be variable. For example, it can be an augmented reality application on Google Glass,[28] an infotainment application,[29] or a health application using a foot-mounted inertial sensor.[30] In our simulations, the mobile devices utilize three different applications and the edge and cloud servers provide corresponding services. The load generator module of EdgeCloudSim is used to characterize the tasks generated by the aforementioned applications. The configuration of these applications is listed in Table 1.

Normally, the mobile devices do not generate service requests continuously. We use an idle/active task generation pattern to simulate the real life properly. According to this pattern, the users create tasks during the active period, and then,

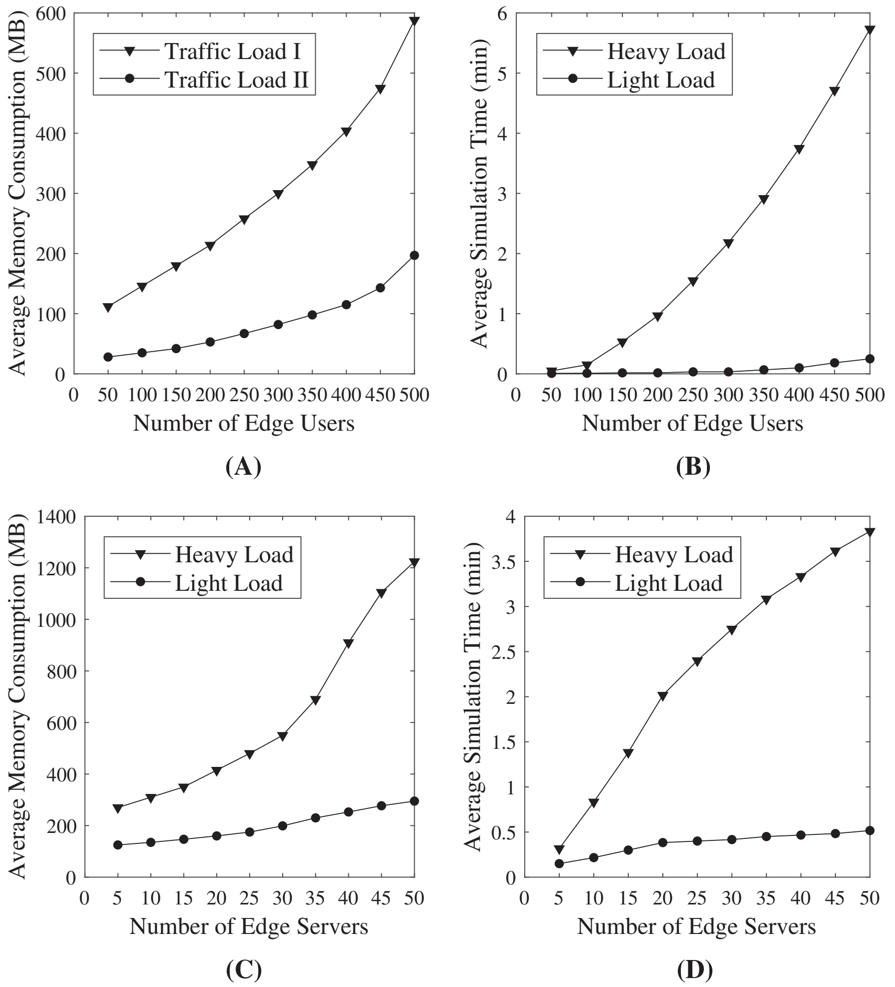

FIGURE 8 Complexity evaluation of EdgeCloudSim for heavy load and light load scenarios. A, Average memory consumption; B, Average simulation time; C, Average memory consumption; D, Average simulation time

TABLE 1 Applications used in the simulations

	AR	I	HM
Usage percentage (%)	50	30	20
Task interarrival time (s)	2	5	10
Idle period duration (s)	20	25	90
Active period duration (s)	40	45	15
Upload data size (kB)	1500	25	1250
Download data size (kB)	25	750	250
Task length (giga instruction)	20	7.5	2.5
Virtual machine utilization of tasks (%)	10	5	2

AR, augmented reality; HM, health monitoring; I, infotainent.

they wait during the idle period. It can be considered that there is a state machine for each client where the client can be either in the active or idle state. For example, the current wearable devices support both hearth rate and physical activity monitoring features. The gathered information on the wearable device can be sent to the nearby device to apply more sophisticated data processing. In this circumstances, the wearable device collects data for some time then it sends it to the edge services.

The tasks generated by the applications have random lengths in terms of the number of instructions and random input/output file sizes to upload/download. For example, the augmented reality application requires high CPU computation, small amount of data to download, and bigger amount of data to upload. Therefore, we use average of

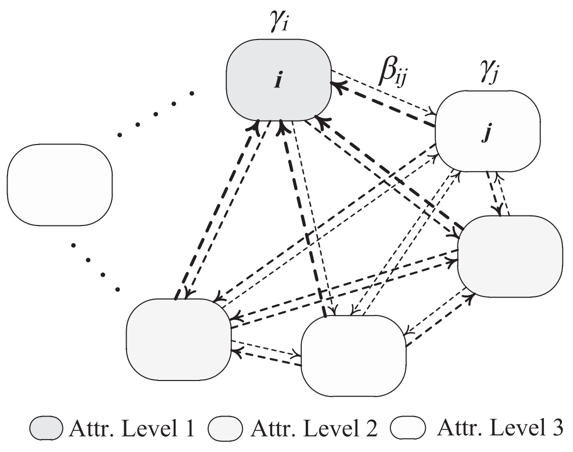

FIGURE 9 Mobility model used in the simulations

1500 kB upload-data size considering a typical jpeg image and 15 kB download-data size referring the recognized person's metadata. On the other hand, the infotainment application requires less CPU power compared to the augmented reality application. It has a request with small amount of data and the corresponding service returns bigger amount of data as a response. The task length directly affects the execution time of the task offloaded to the edge servers depending on the computational power of the VM running on the related server. In our simulations, the computational resources of cloud entities are assumed to be sufficiently large for all scenarios. Therefore, it can be assumed that each task offloaded to the cloud server is executed on a VM without blocking. However, congestion may occur in edge servers.

4.2 | Step 2: modeling the mobility

The mobility module of EdgeCloudSim is used to model mobility of the mobile users. In our experiments, the movement of the students is modeled based on a nomadic mobility model.[31] They wait in one place for a random amount of time and then move to another place. There are three location types with different attractiveness levels in our model. If the attractiveness level of the location is high, the student is likely to spend more time in that place. For example, the students spend more time in a library or a cafe than a kiosk. The mobility model is illustrated in Figure 9 where γ is the dwell time and β_{ij} refers the probability of moving from one location i to the other j. In our simulation, the attractiveness level of the location directly affects the dwell time that the student spends in the related place. The probability of moving to any location is the same for all locations.

4.3 | Step 3: modeling the network delay

The networking module of EdgeCloudSim is used to calculate network delay. In our experiments, each place in the proposed environment is covered by a dedicated Wi-Fi access point. When the mobile devices move to the coverage area of the access point, they join the related WLAN. Then, the mobile devices start sending tasks to the edge server. If a task is decided to be offloaded to the global cloud, the WAN connection provided by the Wi-Fi access point is used. EdgeCloudSim provides a default simple queue model as the networking module for WLAN and WAN models. To achieve a more realistic simulation environment, we use an empirical WLAN and WAN model by extending the default networking module. Another network model can be also used if a different access technology model. The experimental network model is developed by taking measurements from the real life deployments. For WLAN delay observation, an 802.11 family access point is used, and a fiber internet connection in Istanbul was utilized to observe WAN delays. The results of the empirical network delay analysis is shown in Figure 10.

4.4 | Step 4: modeling the edge computing architectures

In order to demonstrate the capabilities of EdgeCloudSim, we experimented with three different simulation setups. In the first setup, we compare three different architectures that are (i) single-tier, (ii) two-tier, and (iii) two-tier with edge orchestrator (EO). As shown in Figure 5, the generic edge computing architecture includes both edge servers and the

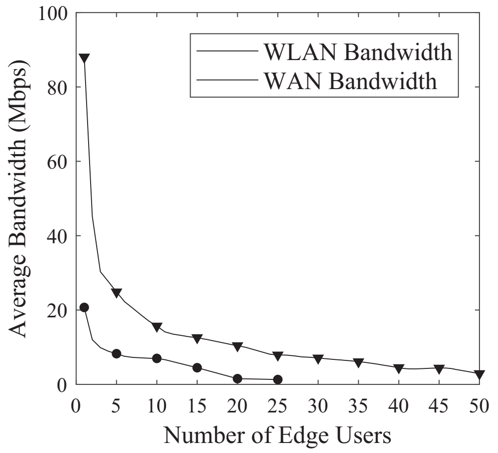

FIGURE 10 Measured delay values used in the empirical wide area network (WAN) and wireless local area network (WLAN) capacity models in the simulator

cloud servers. The single-tier architecture allows the mobile devices to utilize the edge server located in the same building with the student. When it comes to the two-tier architecture, the mobile devices can send their tasks to the global cloud by using the WAN connection provided by the connected access point. In real life scenarios, the decision of selection which service to utilize is decided by the edge orchestrator. In our simulation, the offloading ratio to cloud is decided as 10%, and hence, approximately one out of 10 tasks is offloaded to cloud. The two-tier with EO architecture has a considerable advantage, because, for the tasks that are executed on the first tier, only the two-tier with EO architecture can offload the tasks to other edge servers located in different buildings. In this simulation setup, the edge orchestrator uses the least-loaded algorithm while selecting an edge server to offload in the first tier. It is assumed that the edge servers and the edge orchestrator are connected to the campus network, and hence, they can exchange information in a fast manner. Other important parameters used in the first simulation setup are listed in Table 2.

In the second simulation setup, the effect of the edge server capacity on the results has been investigated. Deciding the edge servers' capacity is a significant problem that should be handled before the deployment of the edge system. In this setup, we evaluate three different deployment scenarios by the same parameters given in Table 2. We use 80, 40, and 20 GIPS edge servers for the first, second, and third, deployments respectively. Since the CPU speed of VMs are 10 GIPS, the number of VMs per edge server becomes 8, 4, and 2 for the related deployments.

In the third simulation setup, the impact of the dwell time of the students on the results is investigated. Since we use a nomadic mobility model, the location of the students is changed after a random amount of time passed. The dwell time of the students is calculated based on the model given in Figure 9. In this setup, we use three scenarios with different dwell time options for each location type. In the first scenario, which is called as *fast*, means waiting durations in the places with attractiveness level 1, 2, and 3 are 0.5, 1, and 2 minutes, respectively. In the second scenario, which is called as *normal*,

TABLE 2 Simulation parameters

Parameter	Value
Simulation time	33 minutes
Warm-up period	3 minutes
Number of repetitions	100
Number of places for type 1/2/3	2/4/8
Mean waiting time in place type 1/2/3	8/4/2 minutes
Provisioning algorithm on edge	Least loaded
Probability of cloud offloading	0.1
Number of VMs per edge server	8
CPU speed per edge/cloud VM	10/200 GIPS
LAN propagation delay	5 ms

Abbreviations: GIPS, giga instruction per seconds; LAN, local area network; VM, virtual machine.

means waiting durations in the related places are 2, 4, and 8 minutes. Finally, in the third scenario, which is called as *slow*, means waiting durations in the related places are 6, 12, and 24 minutes. The rest of the simulation parameters is the same as in Table 2.

5 | SIMULATION RESULTS AND DISCUSSIONS

As already mentioned, three simulation setups are investigated in order to present EdgeCloudSim capabilities. The first setup evaluates the performance of different edge computing architectures. Important performance metrics are shown in Figure 11. In Figure 11A, the average task failure values with respect to the number of the mobile devices are given. In Figure 11B, the average service time is shown. The devices in the single-tier architecture can only offload to the nearest edge server, and as a result, the devices experience the congestion at the attractive places with relatively higher number of users. Therefore, the number of failed tasks in the single-tier architecture is observed higher than the other architectures. Similarly, the average service time is worse than its competitor due to the congestion occurred on the hot spot locations. When it comes to the two-tier architectures, they present better results than the single-tier architecture because some of the tasks can be offloaded to the cloud servers. In this work, a basic probabilistic approach is used to decide offloading tasks to the cloud. Roughly, 10% of the tasks are sent to the cloud, so the two-tier architecture is slightly better than the single tier. If more sophisticated offloading mechanism is designed, the performance difference will be more than the presented scores. The two-tier with edge orchestrator architecture outperforms the others, since the edge orchestrator makes it possible to send the tasks to any edge server in the same LAN. Therefore, it can balance the load of the edge servers efficiently and avoids the congestion occurred in the attractive places where too many users are present.

In Figure 11C, the average WLAN delay values with respect to the number of mobile devices are shown. According to our experimental network model, the delay between the mobile devices and the edge servers becomes higher as the number of users connected to the same WLAN increases. Compared to the single-tier architecture, the two-tier architectures provide better WLAN delay performance, because they can redirect some of the requests to the global cloud. The LAN delay in the two-tier with an edge orchestrator architecture is growing faster than the two-tier architecture. It is misleading to evaluate this graph alone, since the results are also related to the task losses. The average task failure is very low in the two-tier with edge orchestrator architecture, thus the number of the mobile devices sending/receiving tasks simultaneously is getting higher. As a result, in this architecture, the WLAN delay increases faster as the number of the mobile devices increases.

The results given in Figure 11 are the average scores of all applications that generate tasks to be executed on the edge or cloud servers. EdgeCloudSim can provide the scores of each application separately as well. For example, the average processing time of different applications on the edge servers are shown in Figure 12. The processing time is the duration spent on the server while executing a task request. Since EdgeCloudSim runs on top of CloudSim, the processing time is calculated by the core CloudSim modules. Essentially, the processing time becomes higher if the number of the tasks running on the server increases, because the hosts used in our simulations operate a *space shared* VM scheduler.

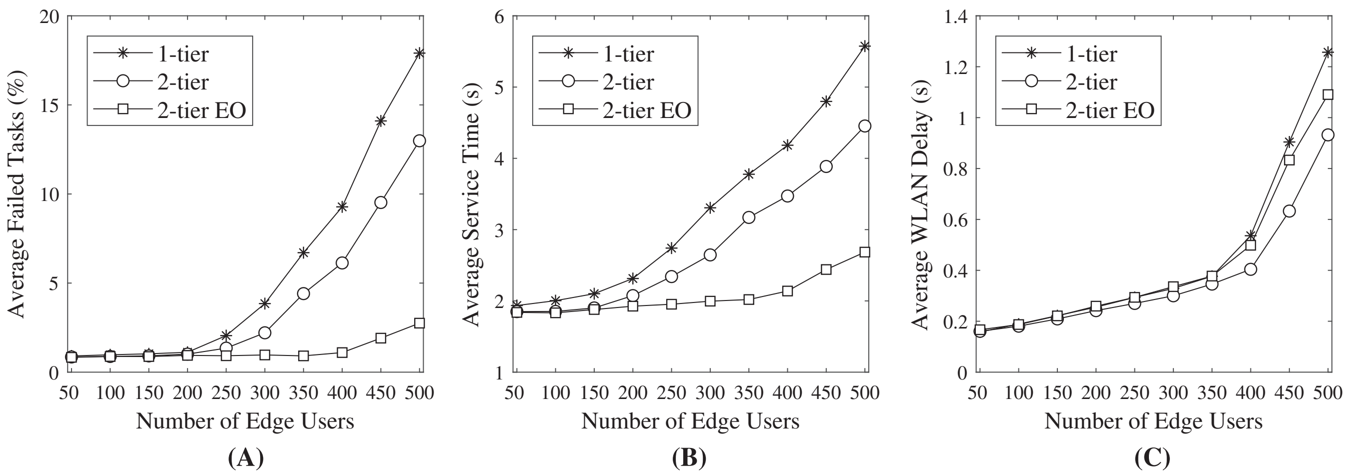

FIGURE 11 Comparative evaluation of the different edge computing architectures based on all application types. A, Average failed tasks; B, Average service time; C, Average virtual machine utilization. EO, edge orchestrator

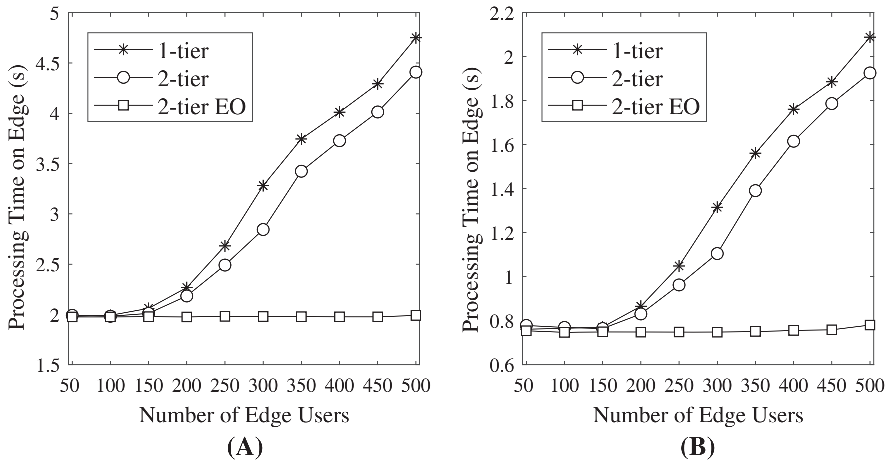

FIGURE 12 Average processing time for different application types. A, Average processing time for augmented reality application; B, Average processing time for infotainment application. EO, edge orchestrator

The average processing time of the tasks generated by the augmented reality and infotainment applications are shown in Figures 12A and 12B, respectively. Both Figures provide similar slopes for each architectures, but the values are different. This is an expected result, since the task length of the augmented reality application is higher than the infotainment application. The average processing time performance of the architectures are similar to the average service time performance in Figure 11B. The two-tier with an edge orchestrator architecture outperforms the others, since it can distribute tasks among the edge servers. The two-tier architecture provides slightly better performance than the single-tier architecture, since it can relay some tasks to the cloud servers.

In the second simulation setup, the effect of the edge server capacity on the results is investigated. In Figure 13, the performance of three deployments is compared. The deployments include different edge servers in terms of the CPU power. The most powerful edge server can operate 8 VMs, the least powerful one can operate 2 VMs, and the other one can operate 4 VMs. The average failed tasks due to the VM capacity and the average processing time values are shown in Figures 13A and 13B, respectively. The edge servers with 2 VMs start to experience congestion after 150 mobile devices, and the congestion starts after 350 devices for the edge servers with 4 VMs. The system can handle even 500 mobile devices, then the edge servers operates 8 VMs. These results are not surprising; in fact they can be estimated. The main motivation of investigating such a simulation setup is presenting how EdgeCloudSim can be used to evaluate and optimize the edge server capacity planning.

In the last simulation setup, the effect of the mobility on the simulation results is investigated. The nomadic mobility model is used in our simulations; hence, the speed of the mobile devices can be configured by modifying the dwell time parameter. In this setup, slow, normal, and fast scenarios are evaluated where the dwell time of the users for each place

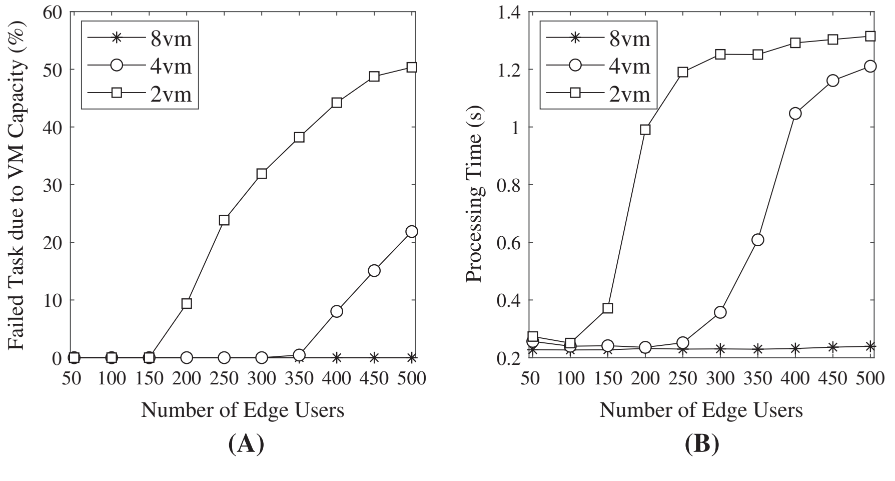

FIGURE 13 Performance evaluation of the system for different virtual machine (VM) sizes. A, Average failed task due to the VM capacity; B, Average processing time

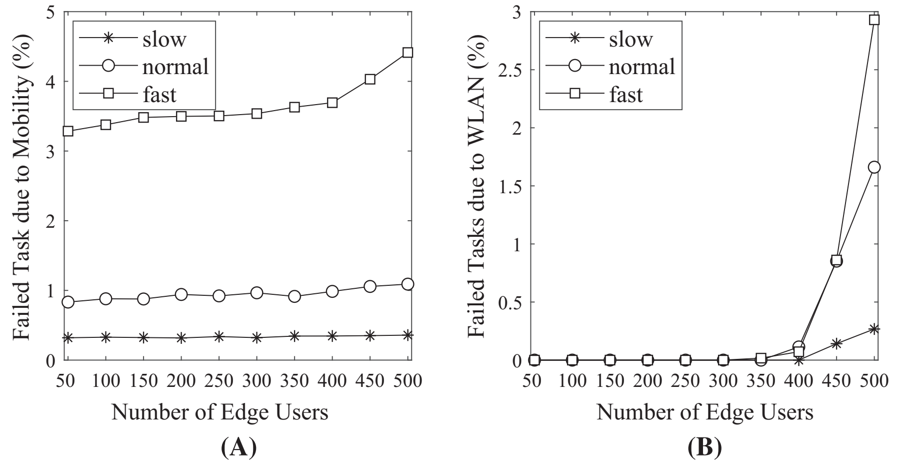

FIGURE 14 Performance evaluation of the system for different dwell time values. A, Average failed task due to the virtual machine (VM) mobility; B, Average failed task due to the VM wireless local area network failure

time is modified accordingly. The average failed tasks due to the mobility is given in Figure 14A. The task failure due to mobility occurs when the mobile device offloads a task and leaves the coverage area of the WLAN before getting the response. The task failure due to mobility occurs when the mobile device offloads a task and leaves the coverage area of the WLAN before getting the response. It can be clearly seen that the number of tasks that are failed due to the mobility increases in parallel to the residence time of the users. The average failed tasks due to the WLAN failure is shown in Figure 14B. If the users are too fast, the congestion occurs more likely in the hot spot places. As a result, the task failure ratio due to the WLAN congestion is proportional to the speed of the users.

6 | CONCLUSIONS AND FUTURE WORKS

The increasing trend toward novel edge computing paradigms requires the need for the evaluation of the proposed architectures and approaches The cost of experimenting on real edge/cloud environments or testbeds is a factor that pushes the researchers to evaluate their proposals through the use of simulators. EdgeCloudSim addresses this need for the edge computing domain that depends on both computational and networking aspects. EdgeCloudSim provides the mobility model, network link model, and edge server model to evaluate the various facets of edge computing. In addition to the simulational capabilities, EdgeCloudSim depicts an increased usability by providing a mechanism to get the configuration of devices and applications from the XML files instead of defining them programmatically. We provide this open edge computing simulator publicly available on GitHub to enable and motivate the research studies on the edge computing area.[17] As a future plan, we plan to add a hand-off mechanism on EdgeCloudSim as a future work. The hand-off mechanism would decrease the task failures due to mobility by using a VM migration method. Another future work we planned is adding the mist computing[32] feature to enhance computational capabilities of the mobile devices. In our current version, the mobile devices can offload the tasks to the edge or cloud servers, but executing tasks locally on the mobile device is not modeled. Finally, we want to add an energy consumption model for the mobile and edge devices and the cloud datacenters.

ACKNOWLEDGEMENTS

This work is supported by the Galatasaray University Research Foundation under grant 16.401.002, the Boğaziçi University Research Fund under grant 12663, and by the Republic of Turkey Ministry of Development under the TAM project with grant 2007K120610.

ORCID

Cagatay Sonmez http://orcid.org/0000-0003-1524-8589

REFERENCES

1. Dinh HT, Lee C, Niyato D, Wang P. A survey of mobile cloud computing: architecture, applications, and approaches. *Wirel Commun Mob Comput*. 2013;13(18):1587-1611.
2. Satyanarayanan M, Bahl P, Caceres R, Davies N. The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Comput*. 2009;8(4):14-23.
3. Fog computing and the internet of things extend the cloud to where the things are. 2015. https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf. Accessed May 01, 2018.
4. Multi-access edge computing. 2018. http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing. Accessed May 01, 2018.
5. Orsini G, Bade D, Lamersdorf W. CloudAware: empowering context-aware self-adaptation for mobile applications. *Trans Emerg Telecommun Technol*. 2017;29(4):e3210.
6. Pacheco J, Hariri S. Anomaly behavior analysis for IoT sensors. *Trans Emerg Telecommun Technol*. 2017;29(4):e3188.
7. García-Pérez CA, Merino P. Experimental evaluation of fog computing techniques to reduce latency in LTE networks. *Trans Emerg Telecommun Technol*. 2017;29(4):e3201.
8. Yaseen Q, Albalas F, Jararwah Y, Al-Ayyoub M. Leveraging fog computing and software defined systems for selective forwarding attacks detection in mobile wireless sensor networks. *Trans Emerg Telecommun Technol*. 2017;29(4):e3183.
9. D'Angelo G, Ferretti S, Ghini V. Modeling the internet of things: a simulation perspective. Paper presented at: 2017 International Conference on High Performance Computing and Simulation (HPCS); 2017; Genoa, Italy.
10. Kecskemeti G, Casale G, Jha DN, Lyon J, Ranjan R. Modelling and simulation challenges in internet of things. *IEEE Cloud Comput*. 2017;4(1):62-69.
11. Issariyakul T, Hossain E. *Introduction to Network Simulator NS2*. 1st ed. New York, NY: Springer; 2008.
12. Varga A, Hornig R. An overview of the OMNeT++ simulation environment. In: Simutools '08 Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshop; 2008; Marseille, France.
13. Riverbed modeler - (formerly OPNET). 2014. https://www.riverbed.com/gb/products/steelcentral/steelcentral-riverbed-modeler.html. Accessed May 01,2018.
14. 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.
15. 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.
16. Gupta G, Dastjerdi AV, Ghosh SK, Buyya R. iFogSim: a toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments. arXiv preprint arXiv:1606.02007; 2016.
17. EdgeCloudSim. 2018. https://github.com/CagataySonmez/EdgeCloudSim. Accessed May 01, 2018.
18. Taleb T, Samdanis K, Mada B, Flinck H, Dutta S, Sabella D. 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.
19. 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.
20. Byrne J, Svorobej S, Giannoutakis KM, et al. A review of cloud computing simulation platforms and related environments. Paper presented at: 7th International Conference on Cloud Computing and Services Science; 2017; Porto, Portugal.
21. Kliazovich D, Bouvry P, Audzevich Y, Khan SU. GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. Paper presented at: 2010 IEEE Global Telecommunications Conference GLOBECOM 2010; 2010; Miami, FL.
22. 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.
23. Makaratzis AT, Giannoutakis KM, Tzovaras D. Energy modeling in cloud simulation frameworks. *Future Gener Comput Syst*. 2017;79:715-725.
24. Amazon Elastic Compute Cloud (EC2). 2018. https://aws.amazon.com/ec2/. Accessed May 01, 2018.
25. Microsoft Azure. 2018. https://azure.microsoft.com. Accessed May 01, 2018.
26. Sotiriadis S, Bessis N, Asimakopoulou E, Mustafee N. Towards simulating the internet of things. Paper presented at: 2014 28th International Conference on Advanced Information Networking and Applications Workshops; 2014; Victoria, Canada.
27. 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.
28. Silva M, Freitas D, Neto E, Lins C, Teichrieb V, Teixeira JM. Glassist: using augmented reality on google glass as an aid to classroom management. Paper presented at: 2014 XVI Symposium on Virtual and Augmented Reality; 2014;Piata Salvador, Brazil.
29. Guo J, Song B, He Y, Yu FR, Sookhak M. A survey on compressed sensing in vehicular infotainment systems. *IEEE Commun Surv Tutor*. 2017;99:2662-2680.
30. Tunca C, Pehlivan N, Ak N, Arnrich B, Salur G, Ersoy C. Inertial sensor-based robust gait analysis in non-hospital settings for neurological disorders. *Sensors*. 2017;17(4):825.

31. Ribeiro A, Sofia R. A survey on mobility models for wireless networks. Technical Report. SITI-TR-11-01. 2011.

32. Uehara M. Mist Computing: Linking Cloudlet to Fogs. In: Lee R, ed. *Computational Science/Intelligence and Applied Informatics*. Cham, Switzerland: Springer; 2018:201-213.

How to cite this article: Sonmez C, Ozgovde A, Ersoy C. EdgeCloudSim: An environment for performance evaluation of edge computing systems. *Trans Emerging Tel Tech*. 2018;e3493. https://doi.org/10.1002/ett.3493