Simulation Modelling Practice and Theory 101 (2020) 102029

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

A comparative analysis of simulators for the Cloud to Fog continuum

David Perez Abreu[*], Karima Velasquez, Marilia Curado, Edmundo Monteiro

Centre for Informatics and Systems, Department of Informatics Engineering, University of Coimbra, Polo II, Pinhal de Marrocos, Coimbra 3030-290, Portugal

A R T I C L E I N F O

Keywords:
Fog
Edge
Cloud
Simulator
Experimental evaluation

A B S T R A C T

The Cloud to Fog continuum is a very dense and complex scenario. At the core level (Cloud) resources are vast, whilst they become scarce at the Edge (Fog). This complexity leads to the need of simulation tools in order to evaluate the performance of novel mechanisms that hardly can be tested in real scenarios. Thus, simulation represents a solution for early stage evaluation before moving to real-world (and more expensive and complex) testbeds. However, selecting the appropriate simulation tool can be complex in itself. This paper presents a conceptual review on six Cloud/Fog Simulation tools, describing their main characteristics and what they allow to experiment. A practical overview of the most representative Cloud/Fog simulators is presented, reporting about their resource consumption and execution time. The aim of this survey is to enlighten other researchers in the selection of the appropriate Cloud/Fog simulation tool for their goals and to know what they can expect from said tools.

1. Introduction

In past years, storage, computing, and network management had been pushed towards the Cloud. The proliferation of mobile and smart devices (e.g., smartphones, tablets, sensors, actuators), as well as the advent of newer services and applications, especially with the emergence of the Internet of Things (IoT), have introduced new challenges regarding how to deal with the substantial amount of data generated and consumed at the Edge of the service and communication infrastructure [1]. The technological developments have brought the tasks mentioned above (i.e., storage, computing, and management) closer to the end-user in order to provide lower latency, mobility support, and location awareness. To guarantee the continuum of the Cloud benefits to the Edge and fulfill the arisen requirements (e.g., low-latency, location awareness) the paradigm of Fog has been proposed in order to complement the use of the Cloud, and harbor these newer requirements [2].

The Fog paradigm is a model organized in layers that focuses on empowering ubiquitous access to a shared and scalable pool of computing resources. The main idea behind the model is to preserve all the benefits of the Cloud facilitating the deployment of distributed and latency-aware applications and services in computational nodes (i.e., Fog nodes) located between the Edge of the communication infrastructure (e.g., IoT network) and the Cloud [3].

Fog computing was envisioned to minimize the request-response time of applications besides providing computing resources and connectivity to centralized services (i.e., Cloud) to the devices located at the Edge of the communication infrastructure. To fulfill its purpose, the Fog paradigm has to guarantee the following characteristics: location awareness and geographical distribution, low-

[*] Corresponding author.

E-mail addresses: dabreu@dei.uc.pt (D. Perez Abreu), kcastro@dei.uc.pt (K. Velasquez), marilia@dei.uc.pt (M. Curado), edmundo@dei.uc.pt (E. Monteiro).

https://doi.org/10.1016/j.simpat.2019.102029
Received 15 July 2019; Received in revised form 28 October 2019; Accepted 22 November 2019
Available online 23 November 2019

latency, mobility support, heterogeneity data support, real-time interaction, federation and scalability of services/applications.

This inherently creates a more complex scenario to deploy application modules and services and to manage the resources in general. Novel mechanisms and procedures, specifically designed for the Cloud to Fog continuum are needed to exploit the characteristics of the Edge of the communication infrastructure, such as low-latency [4]. To evaluate the correctness and performance of these novel mechanisms and procedures, an evaluation must be carried out. Testing in real-world working infrastructures or scenarios is not practical, since one failure during the experiment could impact not only other tests, but also running services and applications. Thus, isolated environments are needed before launching novel solutions to the real-world. Creating a testbed for this kind of scenarios could carry high costs since there are many different devices and communication links involved, so this is not always a viable option. Simulation represents a solution to evaluate the proposed mechanisms and procedures during an early stage in their development. However, using simulation brings a new challenge in itself: selecting the right simulator.

There are several simulation tools for Cloud and Fog environments, with different characteristics [5–7]. They offer various functionalities in their reports, could include several characteristics such as fault injection, and the use of well-known input topology formats, among other features. This can become an issue for researchers, since determining the proper tool for their experiments can lead to an unnecessary delay in their work. Moreover, there is not a simulation tool that fulfills all the requirements for every type of experiments, so even if a simulation tool works for a particular research, it does not necessarily mean it will work for other, forcing researchers to try and find out which other tools they could use for a particular purpose.

This work analizes a selection of six simulators for Cloud/Fog environments, describing their main characteristics and offering a conceptual comparison among them, intending to enlighten the decision process for the selection of the proper simulation tool. The comparison includes features such as latest release date, language used, application model used, topology supported, and fault injection capabilities. Furthermore, three of these tools were appraised in a practical way, running actual simulations using different scenarios, applications, and placement policies in order to evaluate their performance regarding resource consumption (i.e., CPU and memory) and execution time.

The rest of the paper is organized as follows. Section 2 introduces a review of surveys on Cloud/Fog simulators. Section 3 presents the set of six Cloud/Fog simulators studied, including an individual description of their features. Then a comparative analysis is performed from two perspectives, conceptual and practical. Section 4 offers a functional comparison of the simulators from Section 3. A more practical evaluation on a subset of the Cloud/Fog simulators is also presented. Section 5 shows the environment description as well as some issues that arose during the implementation phase. Section 6 offers the results from the experiments on execution time, CPU, and memory consumption. Finally, Section 7 concludes the paper.

2. Related surveys

There have been several studies focused on Cloud/Fog simulators aimed at identifying their main features, pros and cons. Ahmed and Sabyasachi [8] study several Cloud simulators on the basis of their popularity, published papers, and features. The simulators analyzed are CloudSim [9], CloudAnalyst [10], GreenCloud [11], MDCSim [12], iCanCloud [13], NetworkCloudSim [14], EMUSIM [15], GroudSim [16], DCSim [17], MR-CloudSim [18], SmartSim [19], and SimIC [20]. The main limitation identified by the authors is the lack of mobility support. The analysis is restricted to a conceptual appraisal of simulators for Cloud; and many of the simulators studied can currently be considered deprecated since they have had no recent activity (i.e., over five years) [10,15,17,19], or show difficulties for downloading the tool (e.g., broken links, missing files) [11,16,20]. Sakellari and Loukas [21] present different solutions for Cloud environments, including mathematical modeling, prototyping, and simulation. Regarding simulation, they evaluate a set of simulators including CloudSim [9], TeachCloud [22], NetworkCloudSim [14], GreenCloud [11], MDCSim [12], and iCanCloud [13]. The appraisal is limited to a conceptual analysis of simulators only focused on Cloud, and once again, several tools under analysis could be considered as deprecated, such as iCanCloud and MDCSim.

Another comparative analysis is presented by Fakhfakh et al. [5]. They study CloudSim [9] and its extensions, including NetworkCloudSim [14], FederatedCloudSim [23], TeachCloud [22], FTCloudSim [24], WorkflowSim [25], ElasticSim [26], CloudAnalyst [10], and CloudSimSDN [27]. They also analyze other simulators like GreenCloud [11], CloudSched [28], MDCSim [12], iCanCloud [13], secCloudSim [29], and GroudSim [16]. The authors present a comparative table summarizing some characteristics such as platform, availability, energy and cost model, as well as programming language. The tools analyzed are only focused on Cloud and seem to be no longer supported by the community (i.e., iCanCloud, MDCSim, WorkflowSim, CloudAnalyst).

Byrne et al. [6] study thirty three tools to emulate/simulate Cloud environments, most of them (twenty-eight) being extensions of CloudSim [9]. A brief presentation of each tool is provided to later on compare them analytically by means of a summarizing table that lists the license, original release date and most recent update, documentation, availability, language, and platform. They select CloudSim as the *de facto* solution for Cloud simulations, identify limitations such as lack of support for distributed execution (i.e., parallel execution on distributed memory systems), and they also point out how very few simulators support other Cloud use cases such as Fog environments. No practical analysis is provided.

Suryateja [30] analyses seventeen Cloud simulators including CloudSim [9], CloudAnalyst [10], GreenCloud [11], iCanCloud [13], NetworkCloudSim [14], and many others that were also included in the previously mentioned surveys. The comparison criteria used are the platform, availability, programming language, cost, energy, communication modeling, Graphic User Interface (GUI), and simulation time. A summarizing table is presented listing the characteristics of each simulator, and the author reaches to the conclusion that for some particular purposes one simulator might be more appropriated than the others, but for more general purposes CloudSim is recommended based on its features and popularity. Svorobej et al. [7] offer an overview of Cloud/Fog simulators including iFogSim [31], FogNetSim++ [32], FogTorchπ [33], EdgeCloudSim [34], IOTSim [35], EmuFog [36], and

Fogbed [37]. The simulators are evaluated in a conceptual base according to their modeling of resources, applications, infrastructure, mobility, and scalability.

Qayyum et al. [32] compare their tool, FogNetSim++, with other Cloud/Fog simulators, such as EdgeCloudSim [34], MobIoTSim [38], SimpleIoTSimulator [39], IBM BLueMax [40], Google IoT Sim [41], iFogSim [31]/MyiFogSim [42], Cooja [43], FogTorch [44], RECAP simulator [45], EmuFog [36], Edge-Fog cloud [46], and MobileFog [47]. They make a conceptual comparison based on factors like language, platform, support of mobile nodes and scheduling algorithms, and the presence of an energy module. The discussion is narrowed down to a table where the characteristics are listed and limited to a conceptual appraisal. Furthermore, many of the simulators only have a repository and have not been formally introduced via a paper.

Some works do include a more practical approach to the evaluation of Cloud simulators. Four simulators were studied by Tian et al. [48], and compared in terms of performance, modeling elements, architecture, and scalability. The analyzed simulators are CloudSim, GreenCloud [11], iCanCloud [13], and CloudSched [28]. For the conceptual analysis, the typical characteristics are discussed (i.e., availability, platform, programming language); and for the practical analysis, the execution time and memory consumed in Megabyte (MB) are reported; some common simulation results (e.g., energy consumption, number of migrations) are also presented. The authors conclude that none of the simulators evaluated are suited for every job while also identifying challenging issues for Cloud simulation, such as modeling different Cloud layers and high extensibility to add new policies and algorithms. The work is also limited to Cloud simulators.

Sharkh et al. [49] divided their analysis into two categories: (i) simulators with a specific purpose or working on a limited scale; and (ii) simulators operating on a more global scale. Some of the factors used in the evaluation are scalability, main features, and environments. The authors proposed a set of tentative scenarios (e.g., workload planning, workflow modeling, resource allocation, service brokering) and recommended some simulators for each one. According to their findings, CloudSim [9] and GreenCloud [11] resulted in the most *general purpose* simulators, being recommended for most scenarios. From this study, many of the simulators studied (e.g., CloudAnalyst [10], MDCSim [12], TeachCloud [22], BigHouse [50]) can be considered as deprecated, since their latest update was over five years ago; others simulators from this study (e.g., GreenCloud [11], GroudSim [16]) showed problems when accessing the source code (i.e., broken links).

Alshammari et al. [51] compare GreenCloud [11] and Mininet [52] with a set of experiments tailored to appraise the accuracy of each tool in their particular areas (energy consumption and network, respectively). They come up with the conclusion that GreenCloud [11] was not able to accurately predict the energy consumption for a micro-datacenter, while Mininet [52] showed reasonable accuracy in modeling network performance.

Table 1 summarizes the evaluated surveys, indicating if the study is centered in Cloud environments or in the Cloud/Fog continuum; the focus of the paper (practical or conceptual appraisal); and some other issues found with the work.

From all the reviewed work, it is noticeable that many of the analyzed simulation tools are deprecated or show some inconvenient accessing the source code or executable file. Moreover, these works are mainly focused on the evaluation of Cloud simulation tools, with the exception of Byrne et al. [6], Svorobej et al. [7], and Qayyum et al. [32]. Although these three study the Cloud/Fog continuum, they are limited to a conceptual appraisal. There is not a study available (as far as authors know) that compares Cloud/Fog continuum simulation tools including a conceptual and practical assessment. This work presents a set of Cloud/Fog continuum simulation tools or Cloud simulation tools where the concept of Fog can be included in the experiments, combining a conceptual and practical evaluation. For this paper, the terms Fog and Edge will be used interchangeably.

3. Cloud/Fog simulators

To model the Cloud/Fog continuum, up until the lower levels that represent the IoT devices, the simulator must include some characteristics that would lead to more realistic results in such a complex environment. For instance, it must be possible to model features such as location awareness and low-latency, mobility support, as well as interoperability and scalability mechanisms. A proper Cloud/Fog simulator should include support for these features. This section introduces a group of six Fog simulators found in the literature.

Table 1
Summary of the related surveys.

Work	Environment	Focus	Other issues
Ahmed and Sabyasachi [8]	Cloud	Conceptual	Deprecated tools
Sakellari and Loukas [21]	Cloud	Conceptual	Deprecated tools
Fakhfakh et al. [5]	Cloud	Conceptual	Only CloudSim forks
Byrne et al. [6]	Cloud/Fog	Conceptual	A superficial analysis of the simulators
Suryateja [30]	Cloud	Conceptual	Unbalanced review/discussion of the simulators selected
Svorobej et al. [7]	Cloud/Fog	Conceptual	Lacks of a practical assessment
Qayyum et al. [32]	Cloud/Fog	Conceptual	Limited to a conceptual appraisal
Tian et al. [48]	Cloud	Conceptual/Practical	The assessment just considered the top tier of the Cloud to Fog continuum
Sharkh et al. [49]	Cloud	Conceptual/Practical	Deprecated tools
Alshammari et al. [51]	Cloud	Conceptual/Practical	Deprecated tools

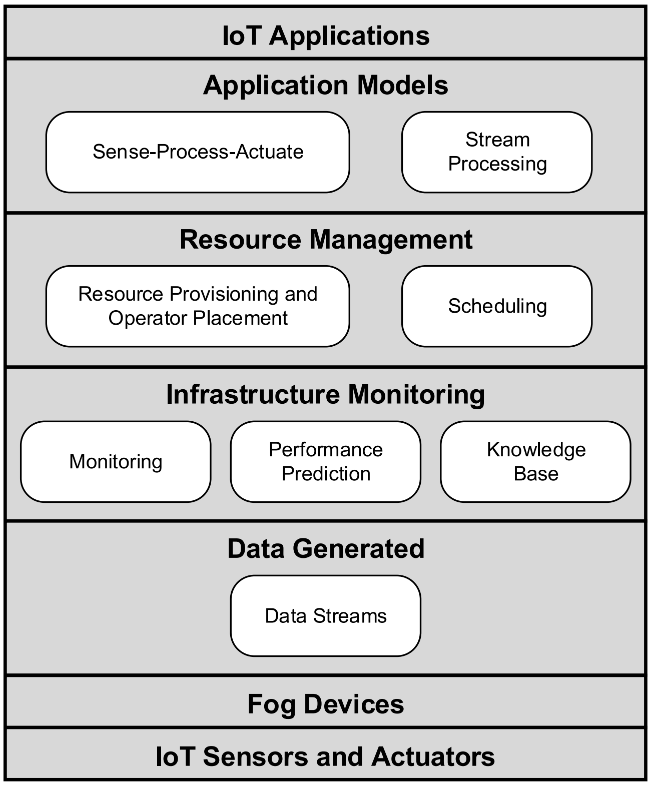


Fig. 1. iFogSim architecture (Adapted from [31]) .

The selection was based on the fulfillment of the Fog features previously mentioned, as well as regarding their acceptance (measured by the citation number, as seen in Section 4) and the support provided by the community (measured by availability of tutorials, documentation, examples, and/or discussion groups, as seen in Section 4). The acceptance and community support help in the selection by allowing to discard tools that have not been adopted by the community.

The simulators selected will be described individually in this section, and will be compared among each other regarding their features in Section 4. The descriptions presented in this section come from the information provided by their authors in the respective papers, but also from experience working with them. The description of each simulator includes the following features: general purpose, reported metrics, programming language used, license, supported topologies, application data model, mobility support, architecture, cost and energy models, federation and resource scalability support, type of simulation used, and research areas where it has been used.

3.1. iFogSim

iFogSim [31] is a toolkit that allows modeling IoT applications in Cloud/Fog environments. It allows the evaluation of the impact of resource management techniques on network congestion, energy consumption, latency, and cost. iFogSim is based on Java and is open-source [53]. iFogSim is an extension of the very well-known Cloud simulator, CloudSim [9].

Physical topologies can be defined using a GUI provided for this purpose (saved topologies are stored in JSON format), or programmatically using Java.

Applications are modeled following the Distributed Data Flow (DDF) model, where an application is built by a set of modules (processing elements) that communicate among each other. There is no mobility support in iFogSim, however, there is another branch based on iFogSim focused on mobility support and Virtual Machine (VM) migrations, called MyiFogSim [42].

The iFogSim architecture, shown in Fig. 1, is based on layers that are responsible for specific tasks. At the bottom, there is the layer that comprises the IoT devices (e.g., sensors and actuators) that interact with the real-world, representing the source or sink of data. These devices can be customized to simulate the effects of sending data (sensors) or performing actions (actuators), in order to model more sophisticated devices, such as smart cameras, environmental sensors, and mobile vehicles. iFogSim also defines *Fog devices* as any network element capable of hosting application modules. Fog devices are organized in a hierarchical tree topology, where the communication is only possible between a parent-child pair. The architecture offers three main services: (i) monitoring components, to keep track of resource usage, power consumption, and availability of devices (i.e., sensors, actuators, and Fog devices); (ii) resource management, to guarantee that Quality of Service (QoS) requirements are met, placement and scheduler components identify the best candidates for hosting an application module and allocate the resources for said module; and (iii) power monitoring, to monitor the energy consumption during the simulation. Power consumption receives a special treatment given that in Fog and IoT-based environments, this becomes a critical resource since the devices are usually battery-constrained.

The cost and energy models are inherited from CloudSim and adapted for iFogSim. This simulator supports VMs resource elasticity and migration. The type of simulation used is Discrete Event Simulation (DES). iFogSim is commonly used to assess allocation strategies, to measure their impact on energy consumption [54,55] and on latency [56–58], and also for evaluating architectures for Fog environments [59–61].

The DDF application model allows to recreate realistic composed applications. Nevertheless, by only offering tree-based topology support, it is not possible to model other types of topologies.

3.2. CloudSimSDN

CloudSimSDN [27] is a simulation framework for Software Defined Networking (SDN)-enabled Cloud environments. As with iFogSim, CloudSimSDN is built on top of the CloudSim toolkit; some additional components had to be added in order to support SDN controller behavior. The simulator also includes the notion of the Edge, by allowing the simulation of Edge switches. The framework allows the evaluation of resource management policies applicable to SDN-based Cloud datacenters, measuring performance metrics and energy consumption; it includes features such as adjustable bandwidth allocation or dynamic network configuration. CloudSimSDN allows to simulate all features of SDN that can be deployed in a Cloud datacenter [62]. This tool is based on Java, it is published under the GNU General Public License (GPL), and it is free to download [63].

Input topology can be provided via a JSON file, as program codes (written in Java), or via a GUI. Despite CloudSimSDN claims that it is possible to provide/build a topology using a GUI, the packages that support this functionality are not available in the main repository of the simulator.

During simulations, the resources (i.e., CPU power, memory, and storage size) defined by the user characterize the physical devices where VMs are placed. The entities that can be modeled include datacenters, physical hosts, links, VMs, VM schedulers, and workload schedulers. A physical node can be an SDN host or an SDN switch (i.e., Core, Aggregation, Edge). There is no mobility support included in CloudSimSDN; nevertheless, it is possible to use migration mechanisms provided by the CloudSim simulation core.

The architecture rests over the CloudSim DES core and has some layers on top to manage Cloud resources (i.e., datacenter, host switch, and link definitions), resource allocation and provisioning, the VM services, virtual topology, and workloads. On top of all this, there are layers to support user code and scenario description. The architecture is depicted in Fig. 2.

As with iFogSim, the cost and energy models are adapted from CloudSim. There is also support for VMs elasticity and migration, besides the possibility to use or define federation models. The type of simulation used is DES. CloudSimSDN is infrastructure-oriented; thus, it is commonly used in research works regarding datacenters and their performance [64–66]; mainly when the scenarios involve Virtual Networks (VN) and SDN techniques, such as mechanisms for Virtual Network Embedding (VNE) [67].

Having SDN support makes it a good solution for infrastructure-based studies, but it is not as fairly suited for works based on applications. In CloudSimSDN, applications are modeled via a set of tasks (i.e., compute processing and network transmission) that will be processed in a VM. Considering that measuring power consumption in switches is a key requirement of the simulator, it is necessary to have long packet transmission between VMs. CloudSimSDN assumes that VMs use their network bandwidth entirely during their lifetime, thus just one long packet transmission workload for each VM is given in the workload file. The aforementioned assumption impacts the granularity level of the application, restricting the communication process between the modules.

3.3. YAFS

Yet Another Fog Simulator (YAFS) [68] is a DES for Cloud/Fog networks. The main focus of YAFS is the performance evaluation of placement, scheduling, and routing strategies. Some of the metrics reported by YAFS include network utilization, network delay, response time, and waiting time; users can calculate other QoS metrics since raw resulting data is reported in a Comma-Separated Values (CVS)-based log. YAFS uses Python as programming language, is open-source, and can be downloaded for free under the MIT license [69].

Topologies can be defined using a JSON-format file, enabling the use of other tools (i.e., BRITE, CAIDA) to generate the desired scenario. Applications are modeled as a set of modules that run services, following the concept defined by iFogSim. Thus, a DDF is used to represent the applications where nodes are modules and edges define the data exchanged among them. It is possible to define a mobility pattern of the application model along the communication infrastructure.

The main architecture is defined by six classes: topology, core, application, selection, placement, and population. The architecture is shown in Fig. 3. When the simulation is running, the *selection, placement*, and *population* classes will create events dynamically (e.g., remove or add a node or link in the communication infrastructure). This enables a more realistic simulation by allowing the appearance of changes in the topology that could represent failures during the execution, thus it is possible to evaluate more complex policies able to handle these issues. Additionally, YAFS supports a set of graphics libraries that allow the visualization of the simulations performed.

There is the possibility to define nodes' attributes that will enable the characterization of cost and energy models. There is also support for scalability by means of clustering definition of Cloud and Fog nodes. YAFS uses a simulator engine based on discrete events. Since YAFS is a recent tool, there is less available research using it for experiments. However, it has been used in works focused on IoT applications deployed in the Fog to appraise the efficacy of placement policies [70,71].

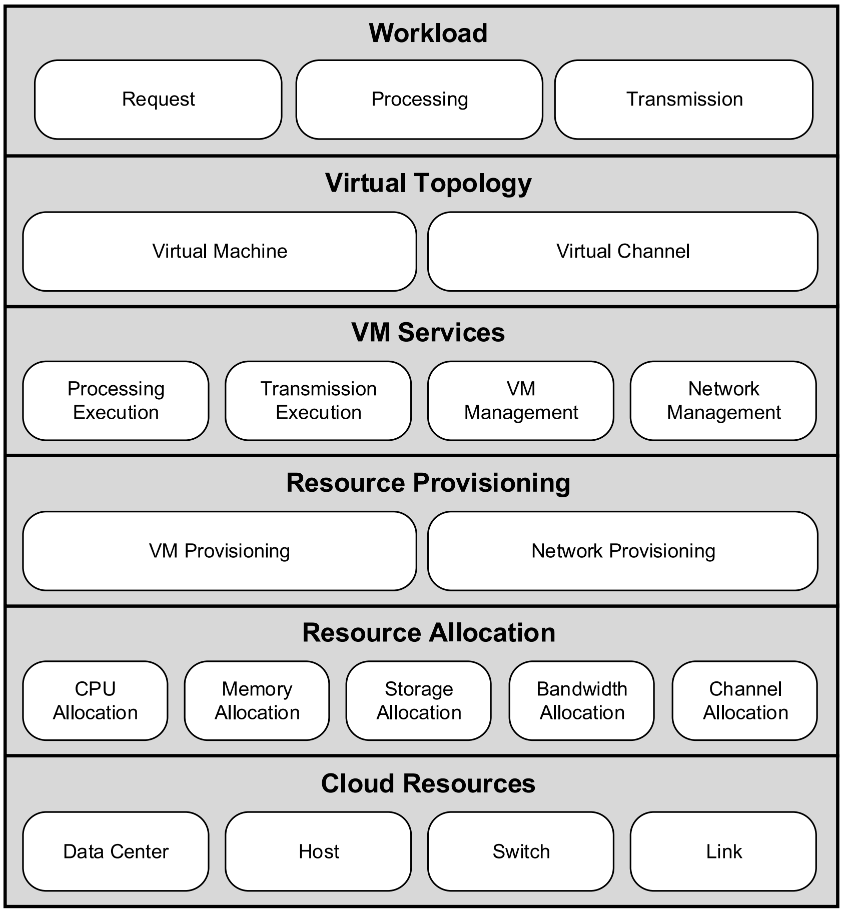


Fig. 2. CloudSimSDN architecture (Adapted from [27]) .

Graph animations are also provided by generating plots of the network on each event and generating a video with the combination of said plots. This eases the analysis of the results. Furthermore, by providing the results in a raw format, it is possible to customize the metrics for a particular need. On the other hand, being fairly new (released in 2019), this simulator has not been as used in research works as other more mature, such as iFogSim.

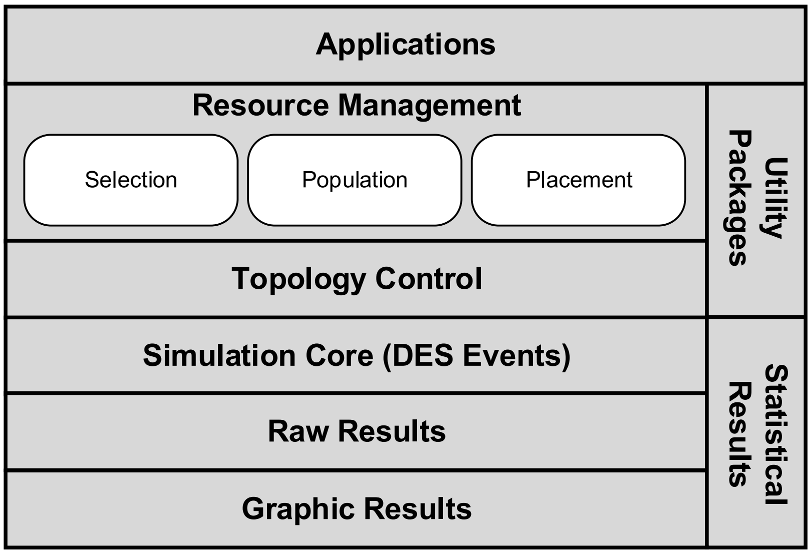


Fig. 3. YAFS architecture (Adapted from [68]) .

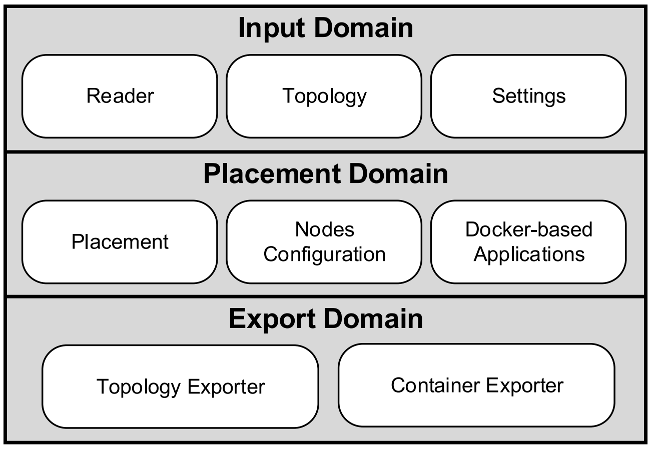


Fig. 4. EmuFog architecture (Adapted from [36]).

3.4. EmuFog

EmuFog [36] is an emulation framework for Fog environments that allows the simulation of Docker-based applications. EmuFog builds up from MaxiNet [72], which in turn is based on Mininet [52]. Respecting reported metrics, based on the fact that EmuFog uses MaxiNet, it is possible to keep track of events in each local node as a log regarding CPU and memory consumption; however, there is not a generic interface to deal with more global metrics, such as response time. The programming language used is Java. EmuFog is available under the MIT license, and it is open-source and free to download [73].

It is possible to use conventional topology generators such as BRITE [74] or also importing real-world datasets like from CAIDA [75]. The topology can also be customized to fulfill users' needs. Users can define the placement of Fog computing nodes on top of the specified topology according to different placement policies and also by specifying the Fog nodes' capabilities and expected workload. Mobility is not supported, neither for clients nor for Fog nodes.

The EmuFog architecture is illustrated in Fig. 4, and it is divided into three domains: (i) the input domain, with the logical description of the experimental setting, including topologies and initialization values; (ii) the placement domain which is involved with the execution of the placement mechanisms; and (iii) the export domain, focused on generating the output files needed to run the experiments.

The workflow begins with the topology generation (either external or customized) to a topology enhancement by applying a placement strategy taking into consideration the Fog nodes' characteristics and the Edge occupancy. This, together with the Docker-based application model, is fed to the MaxiNet engine for emulation. It is possible to use and tune cost models, but there is no energy model available. Regarding federation and elasticity, Emufog supports adaptive functions from the topology infrastructure; however, it does not define any mechanism to deal with computational nodes elasticity.

EmuFog does not follow any type of simulation, since it is an emulation tool. Consequently, EmuFog allows deploying real applications under a controllable and repeatable environment [76,77]. These features are appealing to measure the performance of complete functional applications, but it might not be the case for applications that are under development or specific mechanisms (e.g., placement or resilience strategies) that are not attached to a single application. Additionally, in this kind of emulation environments, more resources are required to complete the experiments. For example, to emulate a datacenter of 3200 hosts in MaxiNet, 12 physical machines were required, whereas a similar scenario could be simulated in a single physical machine [36].

3.5. FogTorchπ

FogTorchπ [33] is a tool based on a model [44] which aims to support QoS-aware deployment of IoT applications in Fog environments. The simulation tool uses Monte Carlo simulations to implement variations in communication links, which are used as inputs. Results are aggregated by computing two metrics for each obtained deployment: QoS assurance (percentage of runs that generated that deployment) and Fog resource consumption (aggregated average percentage of consumed RAM and storage). The tool uses Java as programming language, is published under the MIT license, and is free to download [78].

The Fog infrastructure consists of IoT devices, Fog nodes and Cloud datacenters, and the communication links that connect them. The model is abstract enough to allow the definition of arbitrary topologies. Applications are defined as a triplet composed by the software modules, the interactions between them, and users' requests. FogTorchπ does not support mobility in its simulations.

Fig. 5 shows the structure of the FogTorchπ architecture and operation. FogTorchπ uses latency and bandwidth (differentiating between upload and download per link) as QoS attributes. The Monte Carlo method is used to account for probabilities when assigning QoS profiles to communication links. For each run, a particular Fog infrastructure is provided as input, choosing a QoS profile for each communication link. Finally, the output is a set of eligible deployments.

The FogTorch base box, which is used by FogTorchπ, consists of the model by Brogi and Forti [44] and a set of algorithms that

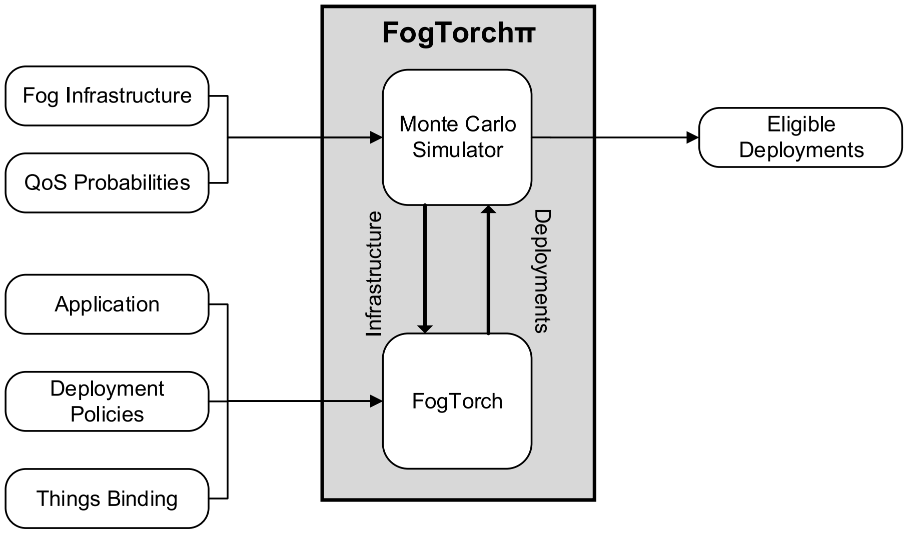


Fig. 5. FogTorchπ architecture (Adapted from [33]).

handle the deployment of applications in the Fog. Applications are viewed as a set of independently deployable components that work together and are bound to QoS constraints. It is possible to define costs and energy models and use them during the simulation process. Concerning scalability, FogTorchπ assumes that Cloud and Fog nodes have unbounded hardware capabilities which denote that real scalability and elasticity support is not contemplated in the simulator.

FogTorchπ uses DES and a Monte Carlo approach for simulation. This tool has previously been used in works related to offloading in Cloud/Fog environments [79,80].

For simple deployment evaluation, FogTorchπ could be a good option since it is focused on this task. Nevertheless, FogTorchπ is referred to by its authors as a prototype tool [44], thus it might not have enough maturity.

3.6. EdgeCloudSim

EdgeCloudSim is another fork from CloudSim [9] mainly focused on Edge computing [34]. This tool covers different aspects of modeling, including network modeling (link properties, network capacities), computational modeling (task execution, VM scheduling), and Fog specific modeling (mobility, offloading, orchestration). It is possible to model multi-layer scenarios that include several Edge servers being coordinated by upper-level Cloud solutions. Results are presented on a Comma Separated Values (CSV) format and include delay (for both Local Area Network -LAN- and Wide Area Network -WAN-) and average failed tasks due to mobility, utilization of VMs, service time, and cost. Being another fork from CloudSim, the programming language used is also Java. EdgeCloudSim is open-source and free to use under the GNU General Public License [81].

Regarding the topology, EdgeCloudSim allows to work with arbitrary topologies, modeling Wireless LAN (WLAN), LAN, and WAN. It is possible to define network link delays, which can vary during the simulation. The application data model is handled by a specific module called *Task Generator*, responsible for defining a set of tasks for a given configuration. Users can customize mobility by defining a model using the *Mobility Model* module inside the *Mobile Client* layer.

EdgeCloudSim provides a modular architecture, shown in Fig. 6. The mobile devices use applications, and the Edge and Cloud servers provide the corresponding services. A load generator is used to characterize the tasks generated by the applications executed. For configuration, three files are needed [82]: (i) simulation parameters in the form of key-value pairs (e.g., simulation time, mobile devices used), (ii) an Extensible Markup Language (XML) file containing the description of the applications and their characteristics (e.g., task length, input/output data size), and (iii) the Edge server topology and access point associated.

There is a possibility to use a cost model, but no energy model is available. There is also a lack of migration support, and it is only possible to group nodes that belong to the same tier, thus rendering impossible to group nodes from the entire Cloud/Fog spectrum.

EdgeCloudSim, as other tools based on CloudSim, uses DES. This tool has been used to evaluate orchestration in the Edge [83], VM allocation policies [84] and proposed Edge architectures [85,86], as well as fault-tolerance mechanisms [87].

EdgeCloudSim provides the possibility to define mobility, network link, and edge server models to measure different aspects of Cloud/Fog environments, making it suitable to evaluate the behavior of IoT and Edge applications/services. However, the simulator uses a single server queue model to calculate the network delay, restricting the behaviour of network access technologies to a simple model which impacts its accuracy.

4. Functional comparison of Cloud/Fog simulators

After the brief review of the simulation tools, this section presents a conceptual analysis of their features, based on the information

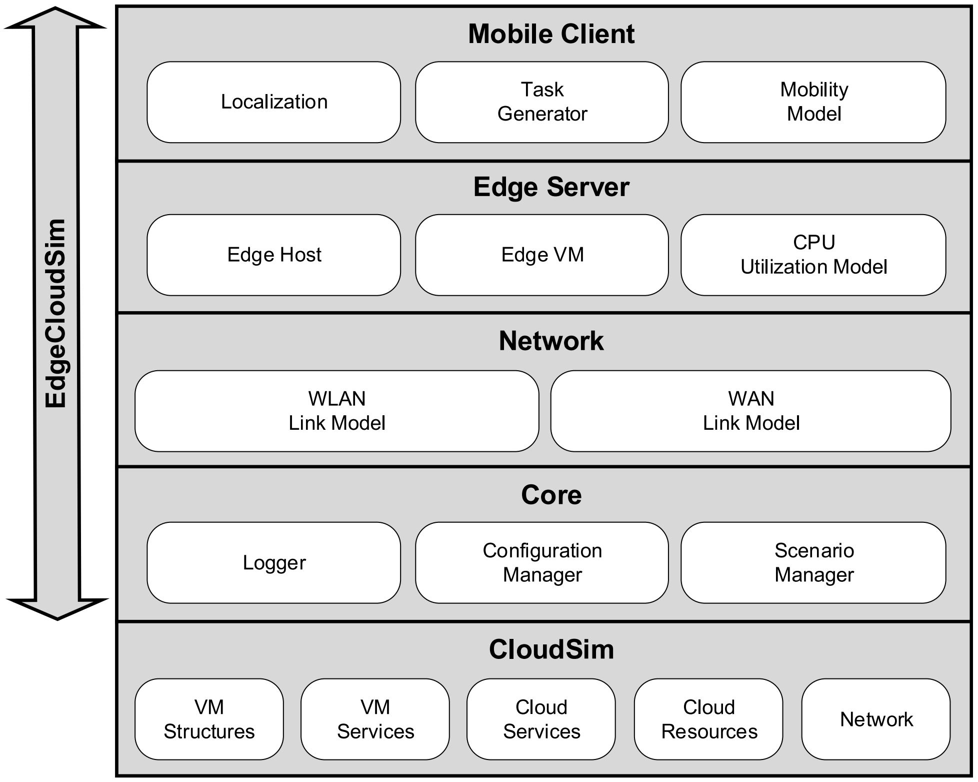


Fig. 6. EdgeCloudSim architecture (Adapted from [34]) .

obtained from the papers presenting them, the experience working with some of them (i.e., iFogSim, CloudSimSDN, EmuFog, YAFS), and in the documentation of each tool (when available). Table 2 summarizes the non-technical features used in the comparison, and Table 3 comprises the technical features.

In Table 2, *Latest release* refers to the year in which the latest release of the tool was published; *First release* corresponds to the year of the launch of the first version of the tool (as listed on its GitHub repository), in order to have an idea on its maturity; *Date of publication* indicates the year on which the paper that introduces the tool was published; *Community support* measures the backing of the creators of the tool and the research community in terms of availability of examples, tutorials/documentation, and/or existence of community groups. The *Community support* is measured in High (all three factors are present), Moderate (two factors present), and Low (only one-factor present); and *Citations* accounts for the number of hits obtained in Google Scholar when searching the name of each tool. *Citations* and *Latest Release* are accounted as to July 2019, when the research was performed.

From Table 2, it is noticeable that CloudSimSDN, YAFS, EmuFog, and EdgeCloudSim are the most recently updated projects, while FogTorchπ and iFogSim are not so far behind (two years since the latest update). The most mature tools (according to the Release Date and Date of Publication) are iFogSim, CloudSimSDN, and FogTorchπ; with YAFS being the most recent. This is also reflected in the *Citation* category, where EmuFog and YAFS have significantly fewer citations; this could be highly influenced by the lack of maturity of the tools and their recent creation.

Regarding the Community support, there is no official discussion group for any of the simulators. However, there are a couple of groups related to CloudSim [88,89], which is the base simulator from where iFogSim, CloudSimSDN, and EdgeCloudSim build up, and some threads in these groups are dedicated to these simulators. Respecting the examples, almost all the simulators include them in their source code, except for EmuFog. Finally, about the tutorials or documentation, iFogSim and FogTorchπ do not offer any documentation; for iFogSim just a couple of examples are provided with the source code. On the other hand, CloudSimSDN, YAFS,

Table 2
Non-technical Comparison of Cloud/Fog simulators regarding their characteristics.

Characteristics\Simulators	iFogSim	CloudSimSDN	YAFS	EmuFog	FogTorchπ	EdgeCloudsim
Latest Release	2017	2018	2019	2018	2017	2018
First Release	2016	2015	2018	2017	2016	2017
Date of publication	2017	2015	2019	2017	2017	2017
Community support	Low	Moderate	Moderate	Low	Low	Moderate
Citations	334	65	34	20	29	56

Table 3
Technical comparison of Cloud/Fog simulators regarding their characteristics.

Characteristics\Simulators	iFogSim	CloudSimSDN	YAFS	EmuFog	FogTorchπ	EdgeCloudSim
Language	Java	Java	Python	Java	Java	Java
Topologies	Tree	Arbitrary	Arbitrary	Arbitrary	Arbitrary	Arbitrary
Fault Injection	No	No	Yes	Yes	No	No
Application Model	Modular	Service-chain	Modular	Docker-based	Modular	Modular
Mobility	No	No	Yes	No	No	Yes
Cost and Energy Model	Yes	Yes	Yes	Partial	Yes	Partial
Federation and Scalability	Yes	Yes	Yes	Partial	No	Partial

EmuFog, and EdgeCloudSim include in their repositories a guide for installation and examples on how to use them; furthermore, YAFS also has a more detailed document covering the basic concepts related to the simulator [90].

As for the number of hits in Google Scholar, iFogSim shows the highest numbers (in the hundreds), which leads to thinking that it is the most used tool (more referenced). This could be because iFogSim has been available for longer (more mature), which will also explain the lower numbers of YAFS and EmuFog. On the other hand, FogTorchπ is in the bottom of the citations category, while being one of the most mature tools being under evaluation, which might lead to think it is not too popular among researchers.

The next evaluation is focused on the technical features. In Table 3, the row *Language* means the programming language used to code the experiments; the *Topologies* category refers to the types of network topologies that are supported by the tool; the presence of the feature *Fault Injection* indicates if the tool is able to simulate random failures in the topology or if dynamic topologies are supported; *Application Model* refers to the representation of an application in the context of the tool; *Mobility* indicates whether the simulator has support for this feature; the category *Cost and Energy Model* indicates if there is already a model (for cost and/or energy) implemented or the possibility for it to be added by the user via built-in features; and the row named *Federation and Scalability* shows if there is support to define federations and scale upwards or downwards the resources used by the VMs or building clusters (i.e., grouping or ungrouping computational nodes) in the Cloud/Fog environment. Since all evaluated tools include support for Fog environments, and all use DES (except EmuFog, which uses emulation), these two features will not be included in the table.

From Table 3 it is noteworthy that the most used programming language is Java, but newer tools (i.e., YAFS) use Python. About the topologies, the most useful is to have support for an arbitrary design, which will enable to recreate different simulation scenarios; iFogSim has a disadvantage by only allowing tree topologies. With iFogSim, the communication is only possible within the same branch of the tree, thus if a user wants to try, for instance, a customized placement policy, it would not be possible with iFogSim, since by placing application modules in different branches, the communication will not be carried out correctly (i.e., will not be included in the simulation's results).

Other essential features are the support for fault injection as well as adding or removing nodes and links arbitrarily. These features will allow to portrait more complex experiments and to test out a wider variety of mechanisms to handle resilience and fault tolerance in more realistic environments. From the selected tools, only YAFS and EmuFog offer fault injection support.

Regarding application modeling, different approaches are taken. A common way that is close to real environments is using a modular approach where the application is defined as a set of modules (or microservices) that constitute the whole. This method is used by iFogSim, YAFS, FogTorchπ, and EdgeCloudSim. On the other hand, CloudSimSDN uses a service-chain model for applications. CloudSimSDN is more oriented to infrastructure while the other tools are oriented to applications. Finally, EmuFog allows using real-life Docker-based applications in their emulation environment.

Mobility support is a key feature requirement for Cloud/Fog applications and services, considering they are usually attached to users that are moving between different access points at the Edge of the communication infrastructure. Out of the six simulators analyzed, only two offer native mobility support, YAFS and EdgeCloudSim. A fork from iFogSim, called MyiFogSim, supports mobility and VM migration. This is one of the aspects with more room for improvement in the Cloud/Fog simulation field, since the support is not only scarce but it is also basic.

The cost of deploying services and applications is a critical feature for end-users and stakeholders. For this reason, the possibility to define a model or behavior regarding monetary or energy consumption during the simulation process has been incorporated into a variety of simulators. From the simulators under study, Emufog and EdgeCloudSim are the only ones with partial support of this characteristic. Particularly, in EmuFog only experiments considering the monetary cost have been carried out; on the other hand, EdgeCloudSim authors' mentioned the support of energy consumption models for mobile and Edge devices, as well as the Cloud datacenters, as a needed feature for the simulator.

Fog environments are dynamic by nature. This feature requires adaptive functions to enable asking for more resources or release them on-demand, as well as to deal with variations on the service infrastructure (e.g., data bursts, communication failures). Simulators as iFogSim, CloudSimSDN, and YAFS support the dynamism and on-demand requirements of Fog services/applications via VM elasticity and migration, federation polices, and clustering of computational nodes. Emufog has scalability support regarding the communication and topology infrastructure; nevertheless, lacks on strategies to deal with on-demand requirements of services/applications inside computational nodes. Finally, EdgeCloudSim only supports Federation and Scalability between nodes of the same tier (only Cloud or only Fog), which means that it is not possible to achieve a proper orchestration along the Cloud to Fog continuum.

The reports obtained are also a technical feature under analysis. Table 4 lists the metrics reported by each simulator. The presence of a checkmark (✓) indicates that the simulator reports that metric, while a dash (–) says otherwise. It is noticeable that iFogSim,

Table 4
Metrics reported by the Cloud/Fog simulators.

Metrics\Simulators	iFogSim	CloudSimSDN	YAFS	EmuFog	FogTorchπ	EdgeCloudsim
CPU consumption	✓	✓	✓	✓	✓	✓
Memory consumption	✓	✓	✓	✓	✓	✓
Bandwidth consumption	✓	✓	✓	✓	✓	–
Energy consumption	✓	✓	✓	–	–	–
Deployment cost	✓	✓	✓	–	✓	–
Latency	✓	–	✓	–	✓	✓
Execution time	✓	✓	–	–	–	–
CPU time	✓	✓	✓	–	–	–
Network time	✓	✓	✓	–	–	–
Failed tasks	✓	✓	–	–	–	✓
Waiting time	–	–	✓	–	–	–
Link availability	–	✓	–	–	–	–
Node availability	–	–	✓	–	–	–

CloudSimSDN, and YAFS offer a more detailed report of the simulation, while EmuFog has the poorest. The metrics that are more often reported are those related to resource consumption (i.e., CPU, memory, bandwidth, and energy), while the metrics with the least support are those related with fault tolerance (i.e., failed tasks, waiting time, availability).

With these findings in mind, it was time to move forward to a more practical assessment of the tools. This preliminary analysis allowed to select the tools from this set that will be evaluated in a practical way, that is, prepare some basic experiments and measure the resource consumption of each tool to determine their performance under different scenarios. Both technical and non-technical features are considered in this process.

The first and foremost feature used for the selection was the support of the Fog concept and services/applications requirements in the experiments. Mainly, the following key features were considered [3]: (i) location awareness and low-latency, to achieve latency requirements it is crucial to have a notion regarding the logical location of infrastructure nodes to infer the communication cost between each other; (ii) interoperability and scalability, to allow the cooperation between different computing components as well as to support clusters of nodes or elastic computing; and (iii) mobility support, which means to have the possibility to provide continuous communication to Edge users even if they are moving. Secondly, for a fairer comparison, the focus on simulation tools instead of emulation, thus discarding EmuFog. Another important factor considered was the number of citations in combination with release date, to take into consideration more accepted and mature tools. Moreover, it was essential to have strong support fromthe community; and finally, the more metrics reported would lead to a more general purpose tool, that will fit more use-cases.

Based on the previous discussion and the outcomes of Tables 2–4, three tools were selected for the experimental evaluation: iFogSim, a Fog simulation tool with strong acceptance of the community (measured by its citations); CloudSimSDN, which not only includes the Fog features but also has a strong citation number; and YAFS, which although being relatively recent (low citation number), it includes the Fog characteristics and has a detailed documentation and metrics report.

FogTorchπ was discarded because of its low popularity (low number of citations while being available for a long time), and also because it is considered as a prototype tool by its authors [33]. EdgeCloudSim lacks some critical requirements in the Cloud to Fog continuum, such as, execution of tasks on Edge devices and task migration between Cloud and Fog tiers; additionally, EdgeCloudSim has limited support of application performance metrics which restricts the amount of information that can be gathered from experiments performed with this tool to resource consumption (e.g., CPU, memory), latency, and failed tasks. This issue drastically narrows its scope for experimental work. EmuFog was discarded since it is an emulation tool and not for simulation, and also for its even more limited metrics support.

The practical evaluation is focused on the performance of each tool, namely resource consumption and execution time, thus giving researchers a glance at what they can expect when performing their experiments. Any implementation issues found will also be documented. Since the focus of this paper is the performance of the simulators, and not of the applications and mechanisms used in the experiments, the values reported by each tool are not included in the analysis. The practical evaluation is presented in the following section.

5. Comparative evaluation of Cloud/Fog simulators

From the initial group of Fog simulators, iFogSim, CloudSimSDN, and YAFS were selected for a more *in-depth* analysis, given their acceptance of the community, the number of citations, and how recent their activity or updates were, but most importantly, because of their stronger support for critical Fog features, and richer metrics report. This appraisal has a practical approach, using the simulators in a set of experiments designed to assess their resource consumption (CPU, memory) and execution time. After describing the evaluation environment, some issues that arose during the implementation are explained. All the source code developed in this research, as well as the results gathered, are available via a GitLab repository [91].

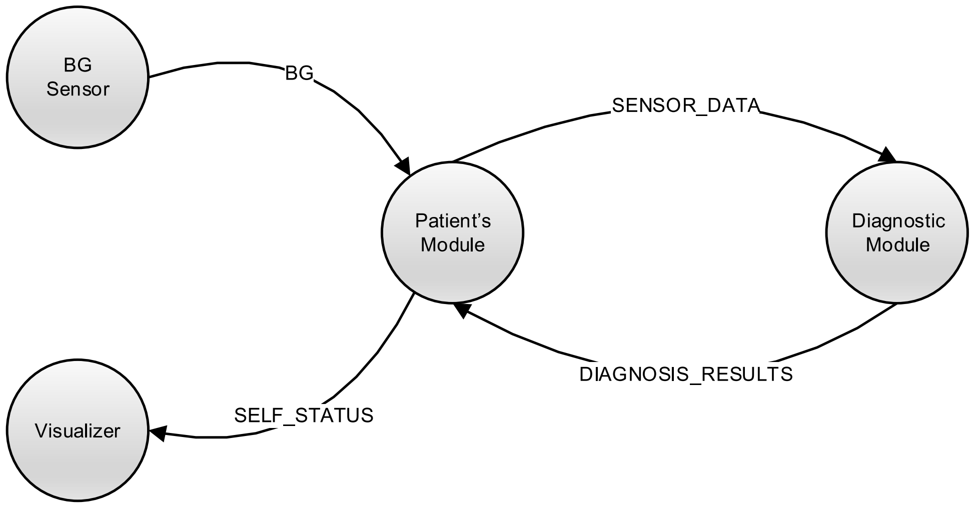


Fig. 7. Application #1 - eHealth .

5.1. *Simulation environment description*

To characterize the experiments conducted, as well as being able to recreate them, a hardware description is provided. Then, a depiction of the applications modeled is provided, and finally the infrastructure topology is explained.

The experiments were carried out on a PC with 8 GB 1333MHz DDR3 RAM and 3.40GHz Intel Core i7-3770HQ with 4 cores and 8 threads (2 threads per core) processor. The PC was running Ubuntu 18.04.2 LTS (Kernel $4.15.\ 0 - 52-$generic) operating system. Regarding the programming languages, the Java version used for iFogSim and CloudSimSDN was 12.0.1, while Python 2.7.15 was used for YAFS.

5.1.1. *Applications modeled*

For the actual experiments, three different types of applications were defined. The first application is a simple eHealth application, in Fig. 7, with a sensor attached to a patient module collecting data, that can be processed in the same module or sent to a diagnosis module for further processing. The diagnosis results are sent to the patient module for visualization.

The second application is a video Surveillance application (as described by Gupta et al. [31]), in Fig. 8, in which video is captured by the camera and sent to the motion detector module that can send a movement command to the camera via the PTZ (Pan-Tilt-Zoom) control module. It is also possible to send the video data to the object detector module which in turn can send the object location to the object tracker module to determine the PTZ command to follow the tracked object (using 10 ms intervals). Finally, it is also possible to send the video data about the detected object to the user interface.

Finally, the third application emulates a latency sensitive game (VRGame), shown in Fig. 9 (the EEG Tractor Beam Game as

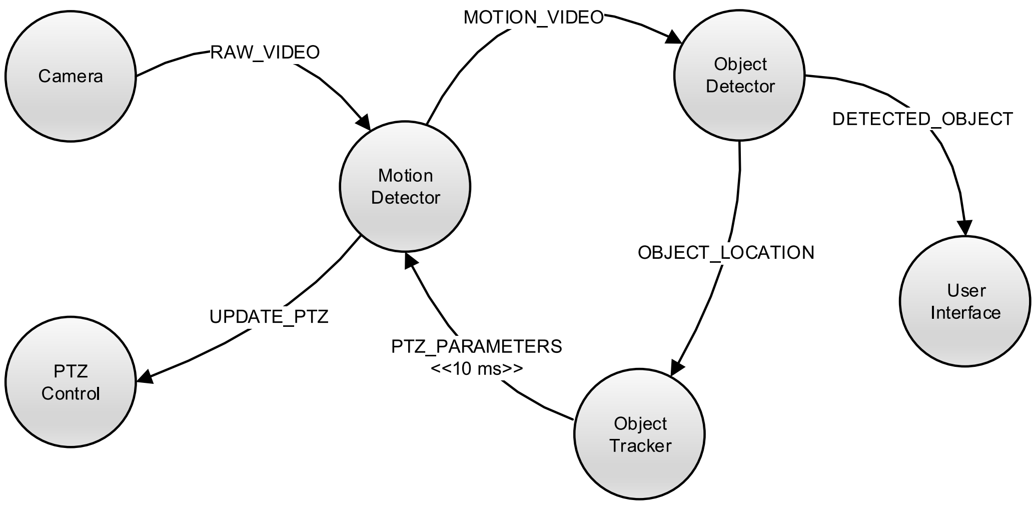


Fig. 8. Application #2 - Video Surveillance .

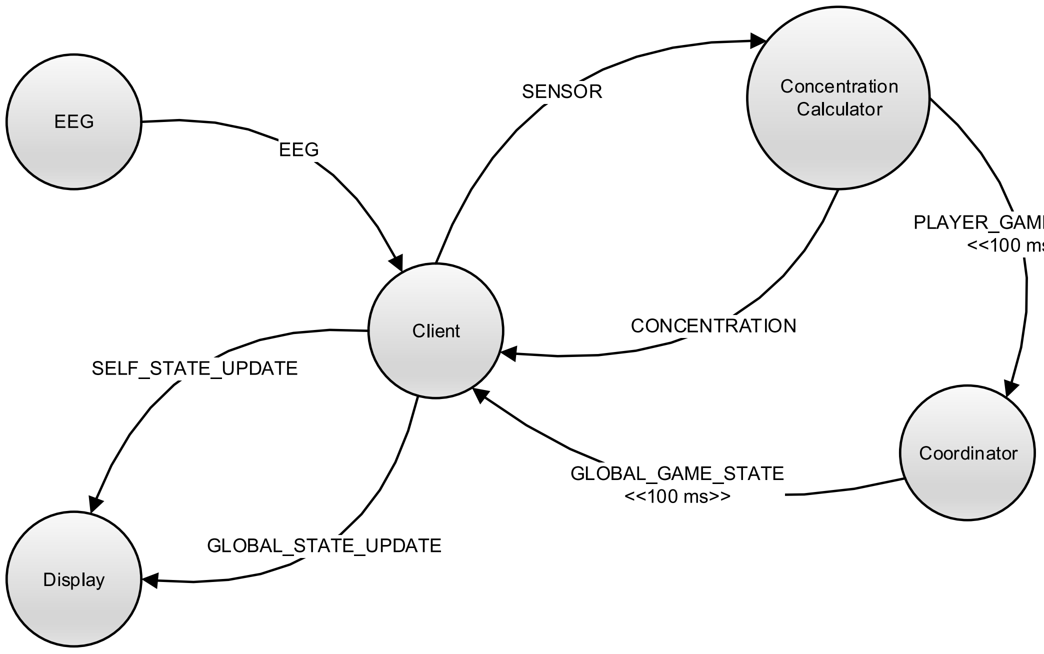


Fig. 9. Application #3 - VRGame .

described by Gupta et al. [31]). A helmet loaded with a sensor collects data in the electroencephalography (EEG) module that sends the EEG signals to the client module for processing. The client module sends the read value to the concentration calculation module to get the concentration level of the player, which is displayed by sending this value to the display module (i.e., inside an actuator). The final module, the coordinator, works at a global level to coordinate the game between multiple players. In Fig. 9, the messages marked between angular brackets parentheses (<<>>) are meant to be periodically sent, while the others are triggered by an input message.

The applications were selected in such a way to include a variety of complexity in terms of the number of modules and messages exchanged among those modules. Also, each application includes the use of sensors and actuators, which makes them relatable to IoT environments. These constitute typical applications to find in Cloud/Fog scenarios, which will regularly be used in simulations to evaluate Cloud/Fog related mechanisms, thus making them good examples to assess the simulation tools.

5.1.2. *Infrastructure topology*

The applications described in Section 5.1.1 were deployed in the topology shown in Fig. 10, with (from top to bottom) a Cloud layer with a centralized high-performance datacenter, a Fog Layer equipped with distributed micro-datacenters, and an IoT layer with basic processing capability devices. The resources available on the nodes in each layer are listed in Table 5. The links between the IoT and the Fog are heterogeneous regarding the bandwidth, to represent the different technologies that usually connect these two layers, while the links between the Fog and the Cloud are more homogeneous than the links closer to the Edge. This infrastructure represents a typical distribution of a Cloud/Fog/IoT environment, following a tree topology.

The decision of using this topology instead of a more diverse/complex one relies in the inherent limitation of iFogSim that only supports communication among the nodes belonging to the same branch within the tree, as described in Section 4. Thus, the tree topology was selected in order to include iFogSim in the experiments and fairly compare it with the other simulators (CloudSimSDN and YAFS), which support a more extensive range of topologies.

The experiments were performed on variants of the tree topology depicted in Fig. 10. The variants applied to the tree topology were intended to measure how the number of Fog and IoT devices impact the simulation environment. Thus, the number of Fog nodes was increased from 4 to 20 modes by intervals of 4 (i.e., 4, 8, 12, 16, and 20); and the same approach was used for the IoT nodes from 4 to 12 by intervals of 4 (i.e., 4, 8, 12). Additionally, 2 placement policies were used Cloud and Edge, where the first one favors the deployment of the application modules in the Cloud datacenter, while the second one tries to use the Fog micro-datacenters and only deploys the application modules on the Cloud when the micro-datacenters run-out of resources. Accordingly, each application was evaluated in all the possible scenarios previously described, giving a total of 30 scenarios (i.e., 15 nodes' configuration times 2 placement policies).

For both YAFS and iFogSim, the applications were implemented as a set of modules following the flow diagrams shown in Figs. 7 to 9, following the DDF model approach offered by the simulators. For CloudSimSDN, the modules were mapped to a set of services in a service-chain that was later deployed in the infrastructure according to the simulation model supported by this simulator. Furthermore, since CloudSimSDN does not implement the concepts of sensors and actuators, an abstraction of placing these devices inside a Virtual Machine was used; thus, these devices were modeled as a VM that executes their tasks.

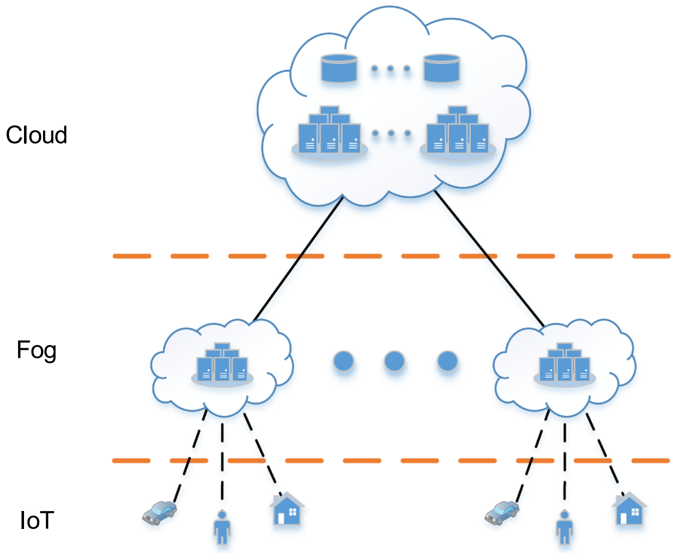
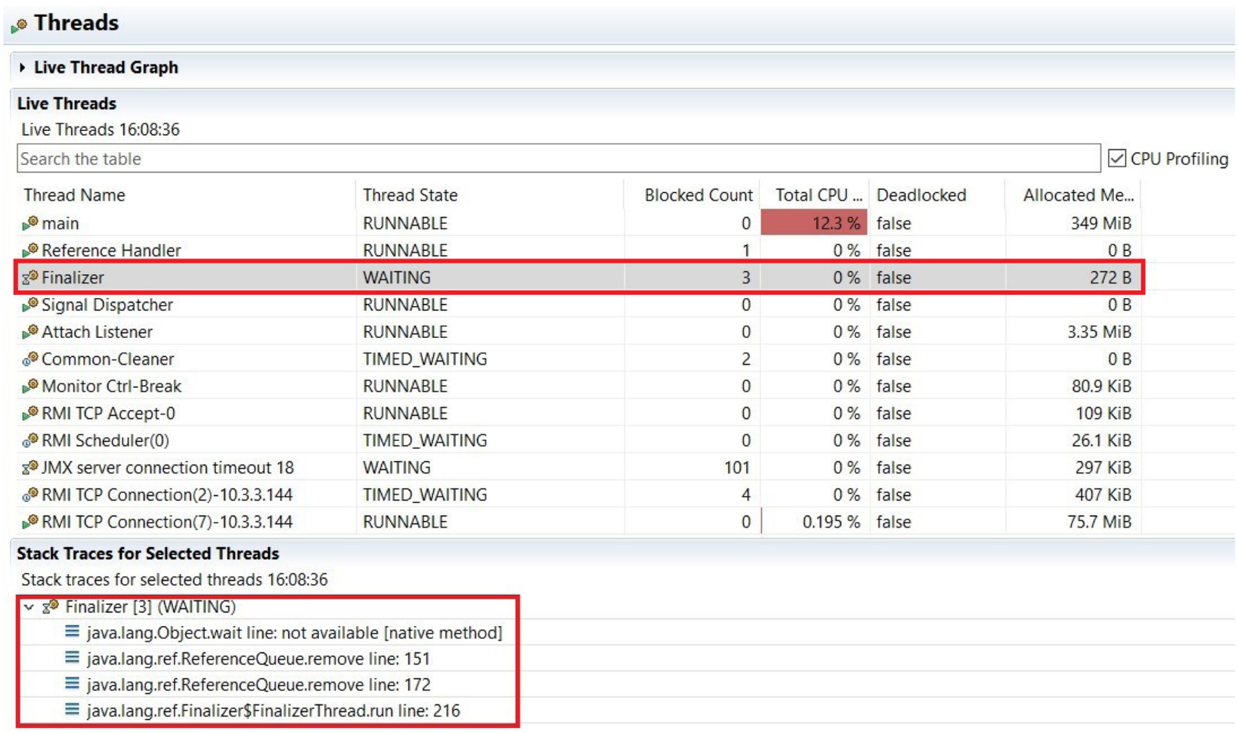


Fig. 10. Infrastructure topology .

Table 5
Nodes and links resources per layer (Cloud, Fog, IoT).

Resources\Layer	Cloud	Fog	IoT
CPU	44,800 MIPS	2800 MIPS	1000 MIPS
Memory	40 GB	4 GB	1 GB
Bandwidth	10 GB	1 GB	100 MB


Threads

Live Thread Graph

Live Threads

Live Threads 16:08:36

Search the table ☑ CPU Profiling

Thread Name	Thread State	Blocked Count	Total CPU ...	Deadlocked	Allocated Me...
main	RUNNABLE	0	12.3 %	false	349 MiB
Reference Handler	RUNNABLE	1	0 %	false	0 B
Finalizer	WAITING	3	0 %	false	272 B
Signal Dispatcher	RUNNABLE	0	0 %	false	0 B
Attach Listener	RUNNABLE	0	0 %	false	3.35 MiB
Common-Cleaner	TIMED_WAITING	2	0 %	false	0 B
Monitor Ctrl-Break	RUNNABLE	0	0 %	false	80.9 KiB
RMI TCP Accept-0	RUNNABLE	0	0 %	false	109 KiB
RMI Scheduler(0)	TIMED_WAITING	0	0 %	false	26.1 KiB
JMX server connection timeout 18	WAITING	101	0 %	false	297 KiB
RMI TCP Connection(2)-10.3.3.144	TIMED_WAITING	4	0 %	false	407 KiB
RMI TCP Connection(7)-10.3.3.144	RUNNABLE	0	0.195 %	false	75.7 MiB

Stack Traces for Selected Threads

Stack traces for selected threads 16:08:36

```
Finalizer [3] (WAITING)
    java.lang.Object.wait line: not available [native method]
    java.lang.ref.ReferenceQueue.remove line: 151
    java.lang.ref.ReferenceQueue.remove line: 172
    java.lang.ref.Finalizer$FinalizerThread.run line: 216
```

Fig. 11. Starvation condition in iFogSim running the VRGame simulation .

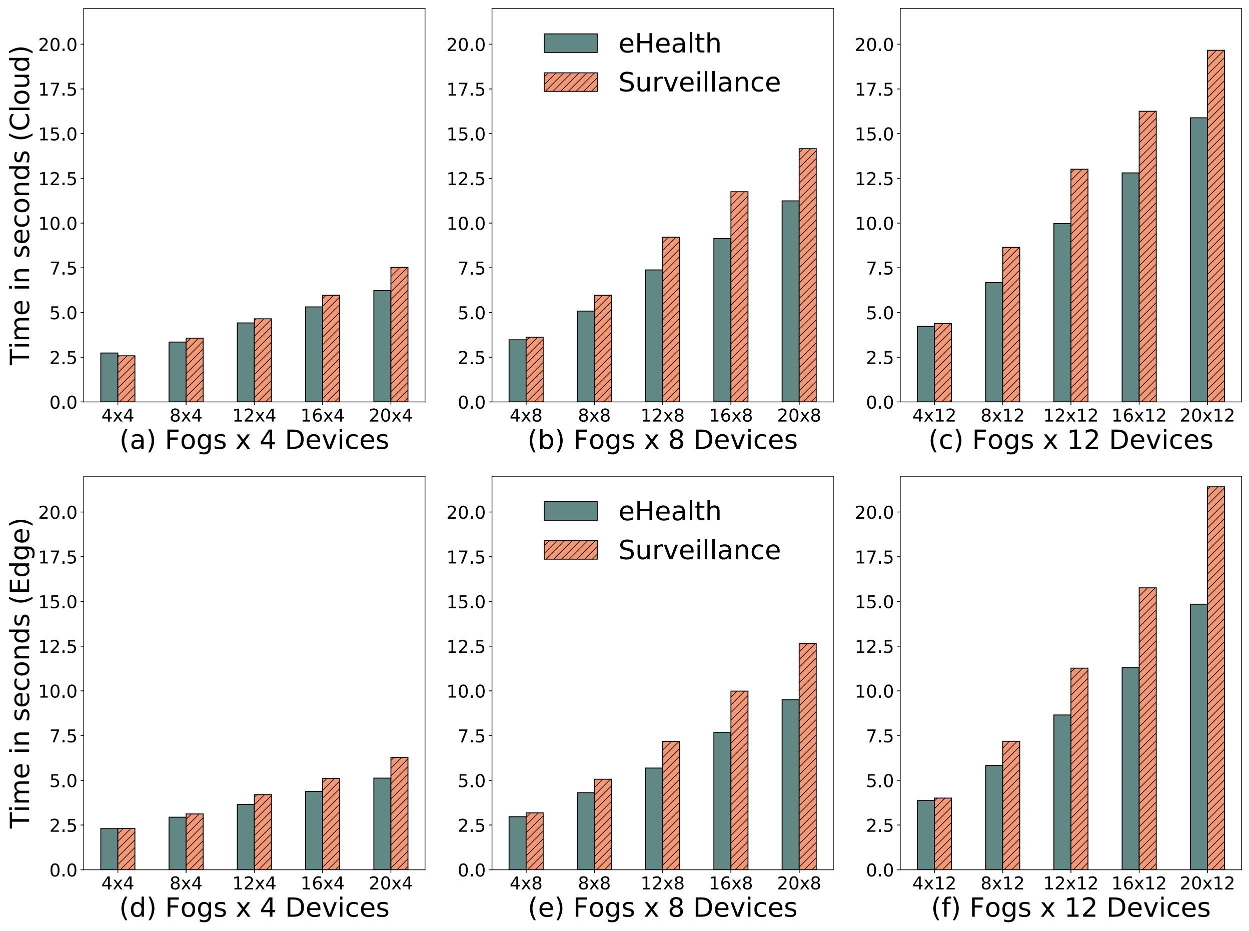


Fig. 12. Applications' execution time in iFogSim .

During the implementation of the experiments, some issues arose and are described in the following subsection.

5.2. Implementation issues

From the very beginning, an important issue that was considered lays on the fact that iFogSim and CloudSimSDN use different versions of CloudSim [9], which is the core for generating the discrete events during the simulation. The first one uses release 3.0.3 and the last one release 4.0 [92]. According to the release notes of these versions, the main changes from 3.0.3 to 4.0 version are the support for container virtualization and bug-fixes; being the last one critical in order to realize performance differences or problems during the experiments.

Another issue that had to be taken care of was the fact that CloudSimSDN uses a timeout for data flows: when the timeout is reached, all packets attached to this data flow are discarded, and no metric is reported. This variable (called *Time_Out*) is defined in a configuration class (i.e., Configuration.java) in the main branch of the simulator package. The timeout was initially set to 10 time units, which was insufficient for some of the experiments that were performed. So, it was decided to change the timeout to 550 time units to guarantee the proper completion of all the experiments and a fair comparison of the tools.

Furthermore, a waiting condition arose during the execution of some of the scenarios of the application VRGame with the iFogSim simulator, resulting in process starvation. This starvation issue was obtained in the following scenarios: 8*x*12, 12*x*8, 16*x*8, 16*x*12, 20*x*4, 20*x*8, and 20*x*12; *Fogs nodes x Devices* respectively. The experiments were repeated for Ubuntu 18.04.2 LTS and Windows 10 Pro-Version 1809 using Java versions 10.0.a and 12.0.1 to verify their behavior, getting the same results. In order to figure out the cause of the starvation, the tool Java Mission Control [93] was used to track the progress and behavior of the Java threads. It was noticeable that the starvation was caused by the thread *Finalizer* as depicted in Fig. 11. A further analysis of the behavior of the simulator under these conditions is out of the scope of this study; thus, the results from the mentioned experiments for iFogSim will not be included in the following section.

On a final implementation note, the evaluation was made in terms of resource consumption, namely memory, CPU, and execution time, in order to measure the performance of each simulator. The tool *psrecord* [94] was used to record the CPU and memory activity of the simulators' process. Timestamps were grabbed in the beginning and end of each experiment to measure the time. The

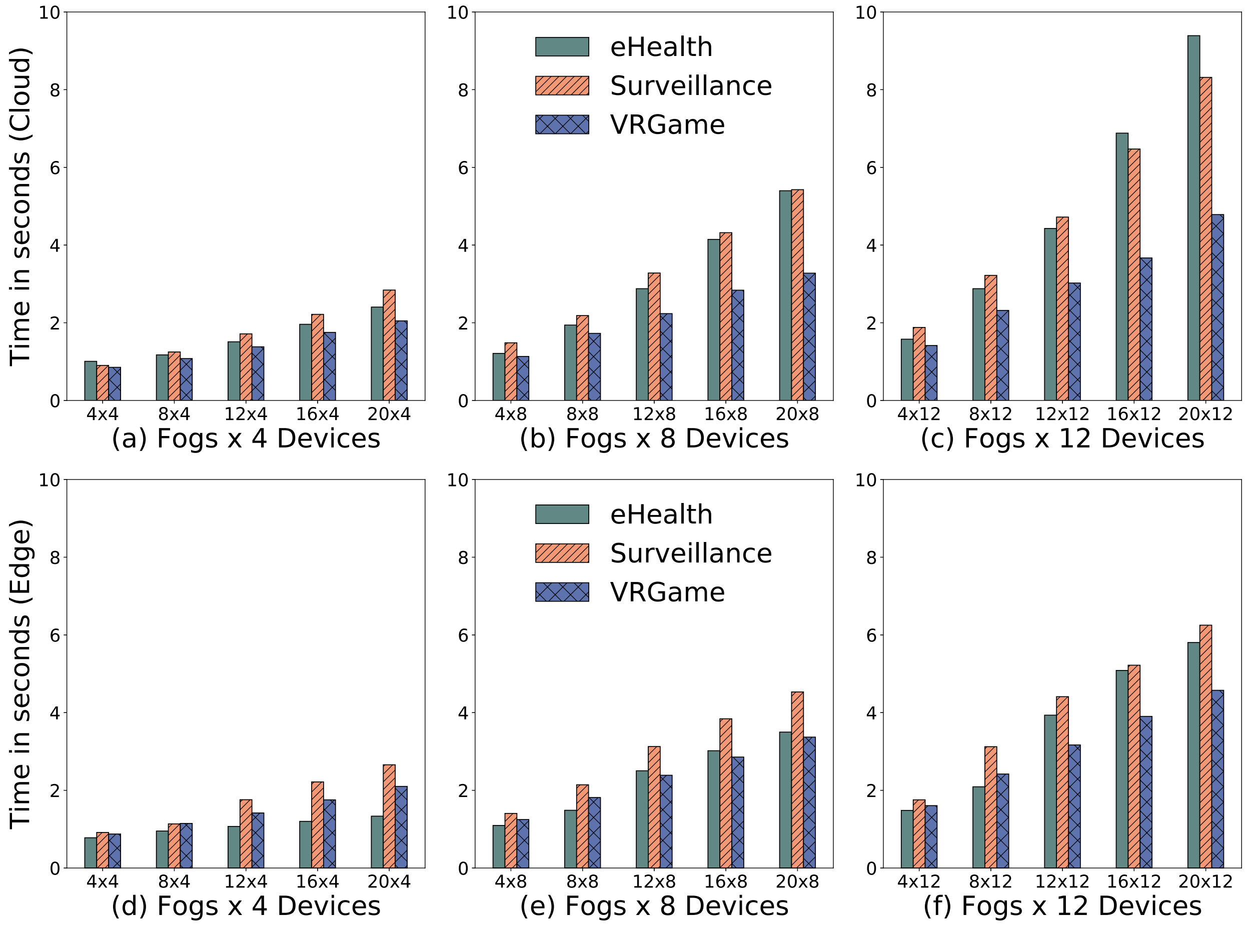


Fig. 13. Applications' execution time in CloudSimSDN .

experiments were repeated 30 times to minimize the statistical error. Also, after each run, a cleanup of the environment (remove Java and Python leftover processes from the operating system) was done to guarantee the equality of conditions between each run. The results obtained from the experiments are presented in the following section.

6. Experimental assessment of Cloud/Fog simulators

A performance analysis regarding execution time, CPU, and memory consumption of the simulation of three Cloud/Fog applications (i.e., eHealth, Surveillance, and VRGame) for iFogSim, CloudSimSDN, and YAFS is presented in this section. The entire results obtained with all the simulators in every experiment can be found in Tables A.1 through A.3, on Appendix A. The following aspects are taken into consideration for the discussion:

- Results are grouped by trend and organized for their relevance;
- The execution time is depicted in seconds;
- The CPU consumption is expressed in percentage use;
- The memory consumption is reported in MB.

Regarding the CPU consumption, the *psrecord* [94] tool aggregates the percentage of utilization of all CPU cores; thus it is possible to reach more than 100% of utilization. Additionally to the aforementioned aspects, the following elements are considered in the plots:

- In the bar plots, *AxB* represents: *A* the amount of Fog nodes, and *B* the number of devices per Fog node;
- In the box plots, *A_B* represents: *A* the policy used (*C* for Cloud and *E* for Edge), and *B* the number of Fog nodes;
- The Y-axis is normalized for all the subplots unless it is explicitly said otherwise.

In the case of iFogSim, as was explained in Section 5.2, the simulation for the VRGame application could not be completed in all

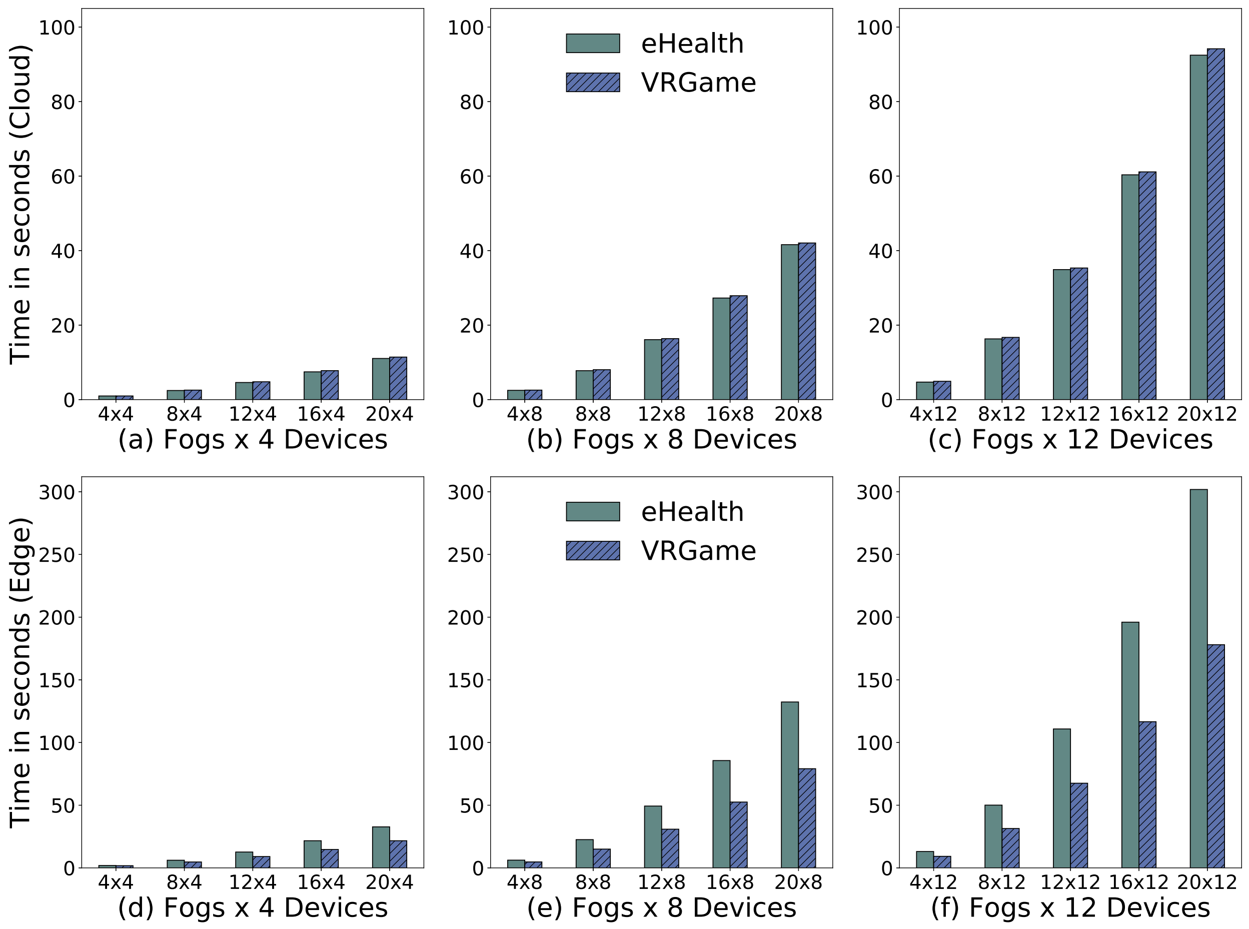


Fig. 14. Applications' execution time in YAFS .

the scenarios, given some waiting issues (i.e., process starvation) that presented themselves during the experiments, thus it was decided to exclude the plots related with this application for this simulator. Moreover, in the case of YAFS, some results are very far on the scale from one another, generating some visualization problems when they are placed in the same plot. Since the trend visible on the plots was the same, it was decided to remove the results corresponding to the Surveillance application in the case of YAFS, for the execution time measurement. For more details on these results, see Appendix A.

6.1. Execution time

The results of execution time in iFogSim are depicted in Fig. 12. It is possible to notice that as the scenario complexity increases (i.e., number of modules and the messages exchanged between them), so does the execution time, as it was expected. Another important thing to see is that the Cloud placement policy takes a slightly higher time to finish in comparison with the Edge policy. Placing more application modules in the Cloud generates more saturated network links due to message exchange, affecting the simulations' runtime [68]. The most demanding application was Surveillance since it has more modules and message exchanges than the others.

The execution time results for CloudSimSDN are shown in Fig. 13. As with iFogSim, the execution time increases with the complexity of the scenario. For this simulator, the most demanding application was the Surveillance application for most scenarios. The execution time for the eHealth application increased significantly with the complexity of the scenario, especially for the Cloud policy. Since the eHealth application has only one module outside the user's device, with the Cloud policy, this module is sent to the Cloud, generating more traffic and network congestion.

Fig. 14 depicts the execution time results for YAFS. In this case, the Y-axis is normalized per placement policy (i.e., Cloud and Edge). For the Cloud policy, the applications show similar behavior, with the VRGame taking a slightly higher time than the eHealth application; while for the Edge policy, it is the eHealth application the one that takes a significantly higher time that the VRGame. This could be related to the handling of DES in YAFS, that generates more DES processes when using the Edge policy [68].

In general, the most demanding application was the Surveillance application, which was expected since it has more modules and message exchanges. In all cases, YAFS took longer to complete the simulations.

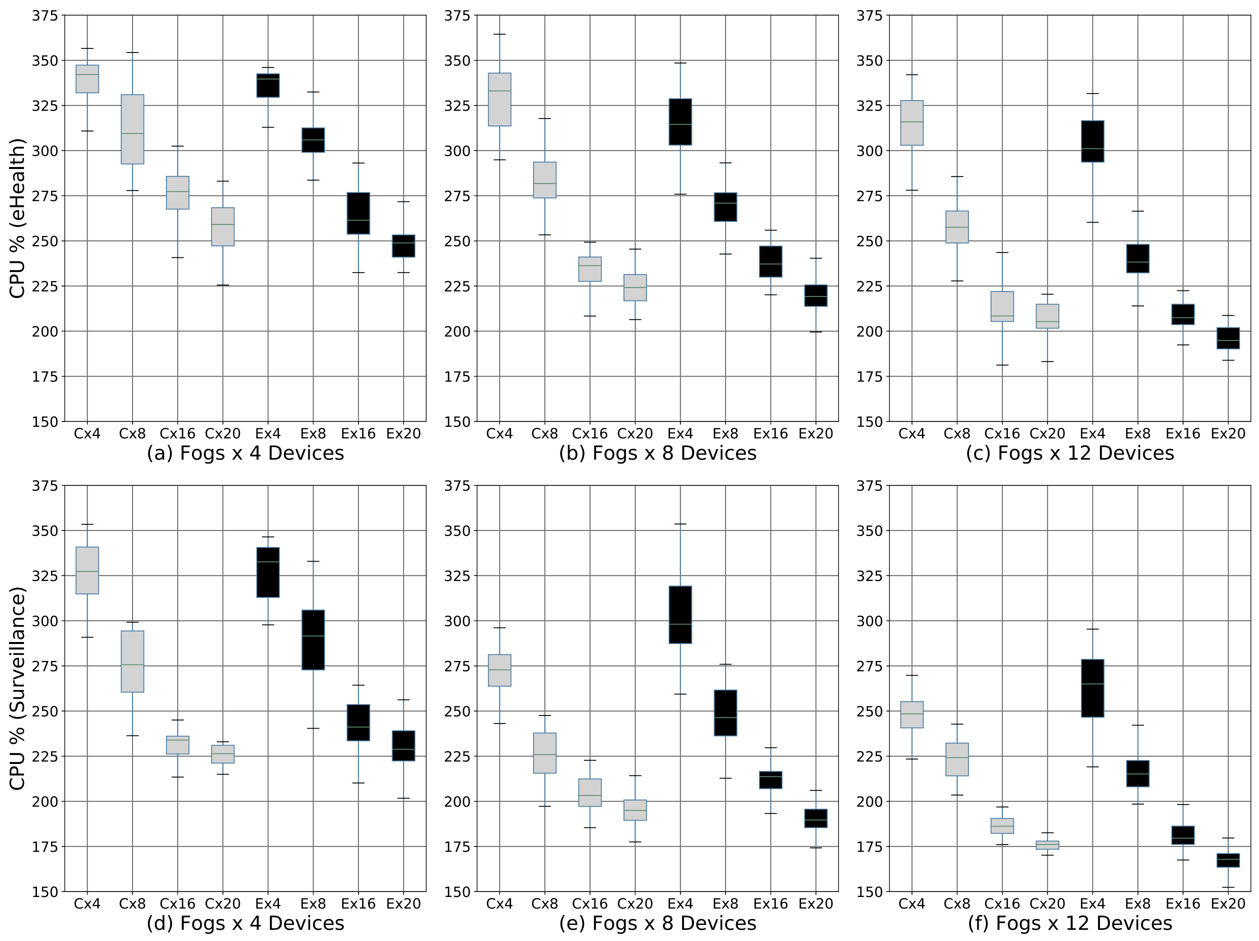


Fig. 15. Applications' CPU consumption in iFogSim .

6.2. CPU consumption

The results regarding CPU consumption are discussed next. Fig. 15 illustrates the results for iFogSim; similar results were obtained for CloudSimSDN. It is possible to see how the values decrease as the scenario gets more complex, which might seem contradictory. A multi-thread execution environment scheduled by the Java Virtual Machine (JVM) was observed (both iFogSim and CloudSimSDN use Java) with one thread assigned to the simulation and other threads associated to internal tasks for the JVM [95], such as reclaiming unused memory (i.e., garbage collection).

To verify this fact, Fig. 16 shows the histograms of CPU consumption for iFogSim during the simulation of the Surveillance application for different scenarios. It is noticeable that the CPU consumption remains around 100% in more complex scenarios and only exceeds this value at some points where the JVM's external tasks are performed, such as, at the beginning of the simulation when it is necessary to instantiate objects and other data structures. For shorter simulations (i.e., less complex scenarios), these peaks in processing affect more the average CPU consumption than for more extended simulation (more complex scenarios), generating the unexpected behavior in the CPU consumption. Although Fig. 16 refers to the simulations carried out in iFogSim, a similar behavior was observed for CloudSimSDN, both Java-based simulators.

Fig. 17 characterizes the CPU consumption with YAFS. In this case, a clear trend is present in the results, where the CPU consumption increases with the complexity of the scenarios, which is expected. Furthermore, results show that the Surveillance application is the most CPU demanding in YAFS, followed by the VRGame and eHealth. The Cloud placement policy consumes less CPU than the Edge policy. This could be due to the amount of DES processes used in YAFS when applying the Edge policy [68].

Overall, there is some interference in the CPU consumption from the Java internal processes with iFogSim and CloudSimSDN. YAFS presents a more predictable behaviour. The most CPU demanding application was the video Surveillance for YAFS and iFogSim, and the VRGame for CloudSimSDN.

6.3. Memory consumption

In relation to the results obtained for memory consumption, in the case of iFogSim (see Fig. 18), the values oscillate between 200

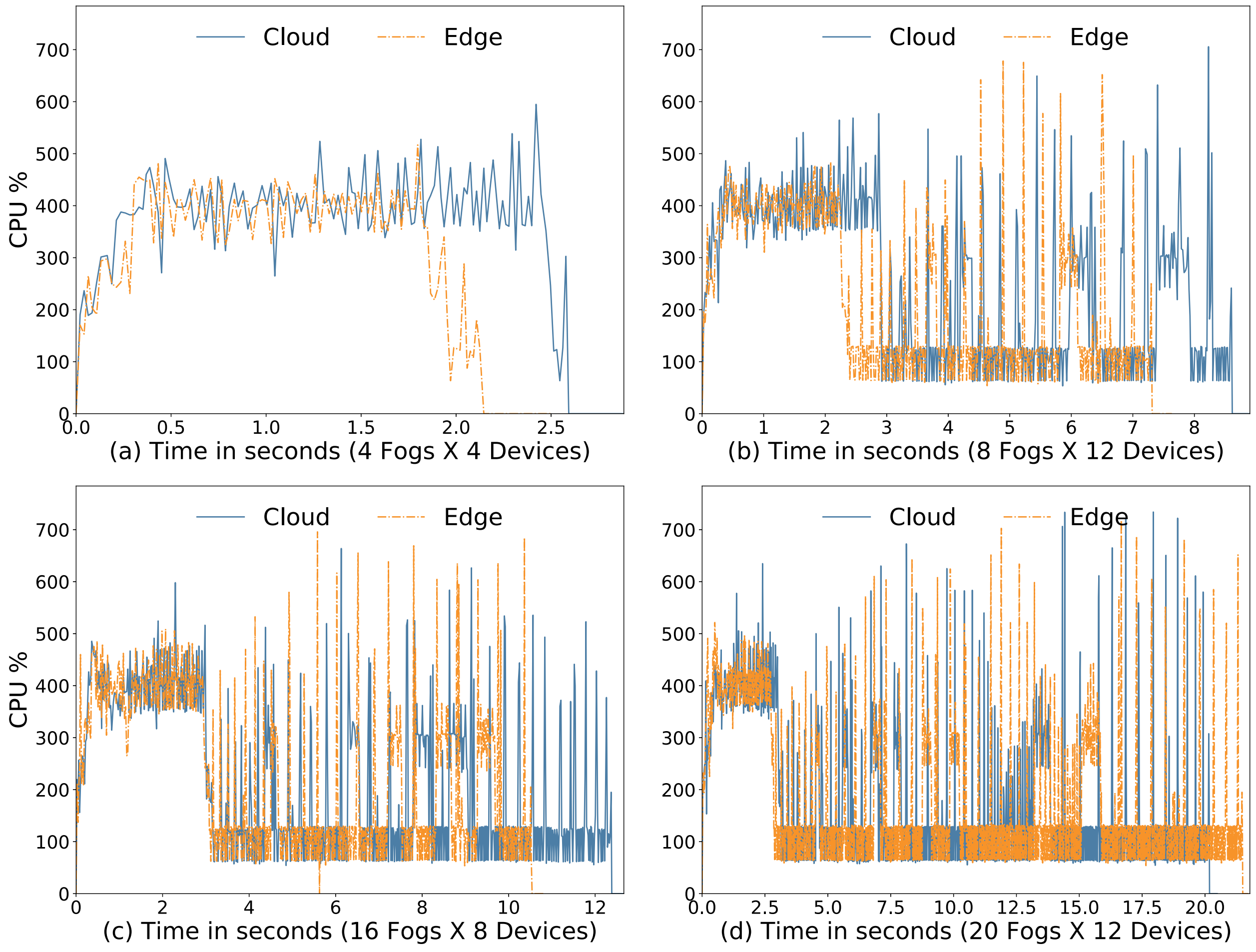


Fig. 16. CPU consumption in iFogSim during the simulation of the Surveillance application for different scenarios .

and 1000 MB; the Cloud placement strategy consumes more memory than the Edge one in the case of the eHealth application. For the other two applications (Surveillance and VRGame), memory consumption was fairly even. In general, memory consumption increases with the complexity of the scenarios, as expected. Moreover, the variance in the values obtained is high for iFogSim and CloudSimSDN; meanwhile, for YAFS the memory consumption was quite stable during the simulations. Notice that the Y-axis is not normalized on this case, to get a better visualization since the results were so far apart from one another.

For CloudSimSDN, the results oscillate between 90 and 200 for all the scenarios. The consumption for the Cloud policy was more prominent than for the Edge policy in the case of the eHealth application while remaining equitable for the other two applications. As the complexity of the scenarios increases the memory used slowly rises, with a high variance.

For YAFS, memory consumption goes from 100 to 350 MB. The memory used for the Cloud policy is higher than the one used for the Edge policy in the case of the eHealth application; while the trend is reversed for the other two applications (i.e., more memory is used for the Edge policy with the Surveillance application and the VRGame).

As expected, the memory consumption rises as the modelled application and scenario get more complex. iFogSim was the simulator with the highest memory consumption in average, while CloudSimSDN and YAFS had fairly similar results. Additionally, the values for memory consumption with YAFS show a very low variance.

A more detailed discussion of the results obtained is presented in the following subsection.

6.4. Discussion

The experiments performed allow to set down some conclusions regarding practical and technical issues for iFogSim, CloudSimSDN, and YAFS; and this is precisely the argumentation that is presented in this subsection. The discussion is grouped into two main categories: (i) an overall analysis of the results gathered in Section 6; and (ii) a review of strengths and weaknesses of the simulators regarding their usability and versatility.

Before moving forward with the grouped discussion, a special mention regarding some practical issues should be pointed out, and it is the starvation condition reached using iFogSim during some scenarios of the VRGame application. As discussed in Section 5.2, it

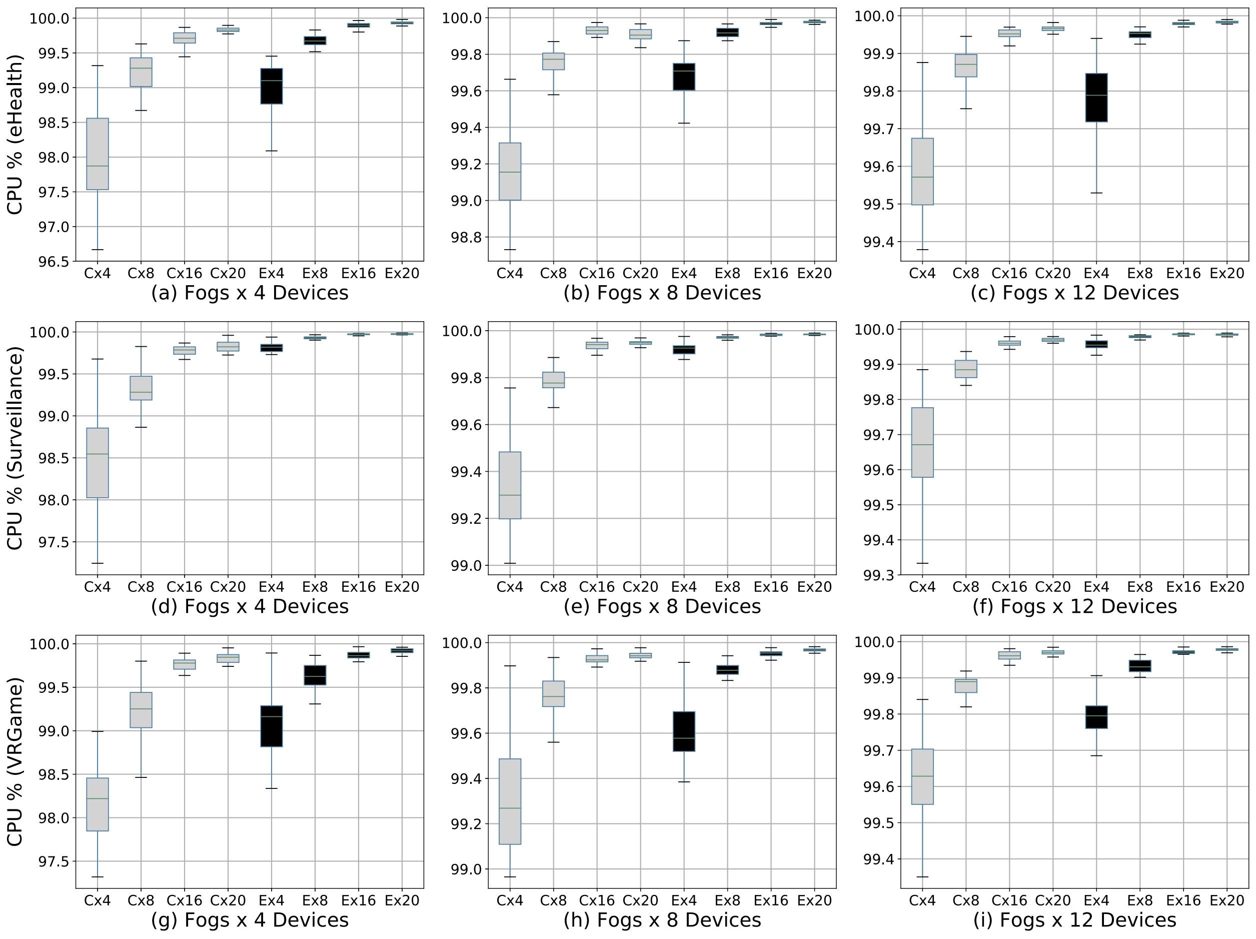


Fig. 17. Applications' CPU consumption in YAFS .

was impossible to complete the simulations for some scenarios, this means that the current version of iFogSim has some problems regarding scalability in the number of nodes that are under communication.

Concerning the execution time and CPU consumption among the experiments, it was clear that the Cloud policy for module placement increased the time required to complete the simulations for iFogSim and CloudSimSDN; however, it was the Edge policy approach which increased the simulation time for YAFS. The higher number of messages that must be transmitted in the Cloud policy placement mechanism generates a saturation in the network infrastructure that affects the runtime in iFogSim and CloudSimSDN; on the other hand, in the Edge approach there are more DES processes to control, given the number of modules per application, and this impacts YAFS' runtime. Despite that YAFS, on average, requires more time to complete the simulations, the details of the results gathered during the simulation could contrail the extra time. YAFS keeps all the events between modules in a raw file that could be used to compute different metrics and some useful correlations depending on the scenarios simulated.

An additional fact regarding the CPU usage lays on parallel execution. Even though a multi-thread behavior was reported by *psrecord* for iFogSim and CloudSimSDN, the applications and scenarios modeled were executed on a single thread in the three simulators. As described in Section 6, the multi-thread conditions reported on the Java-based simulators were generated for some extra tasks performed by the JVM, in order to fulfill particular tasks (see Fig. 16). Thus, the extra time to complete the simulations that YAFS takes is not related to the single thread execution; it is due to the necessity to process extra DES events besides the performance differences between Java and Python.

From the memory consumption analysis, it is possible to determine that iFogSim is the simulator that consumes more memory (up to 1000 MB on these experiments) while CloudSimSDN and YAFS were more similar in their consumption (around 200 MB in average). The results show that for all cases, the memory consumption increases with the complexity of the scenarios, being CloudSimSDN the simulator with lower memory consumption rate, followed closely by YAFS, and then iFogSim. Furthermore, YAFS was the simulator with less variance; thus better stability regarding memory disease in all the scenarios and application simulated. Finally, the results show that resource consumption (i.e., time, CPU, and memory) increased according to the complexity of the applications and scenarios modeled.

Moving forward to the strengths/weaknesses discussion of the simulators based on the practical assessment realized, regarding the metrics reports for the simulators, it is clear that iFogSim focused on reporting energy consumption and latency; and that

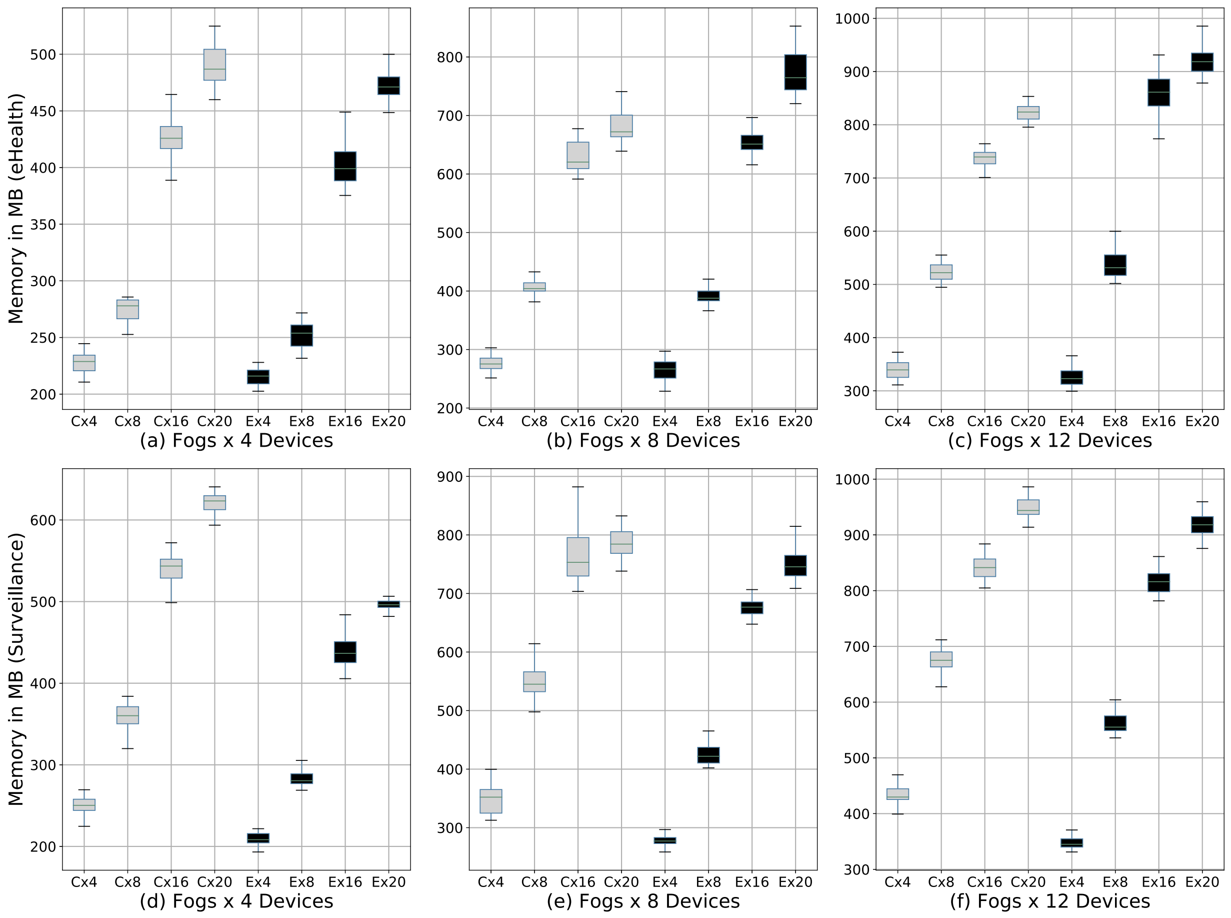


Fig. 18. Applications' memory consumption in iFogSim .

CloudSimSDN is slanted to report the service time regarding CPU and network, as well as, energy consumption. A different strategy is taken by YAFS, where all the events are stored in a raw format via a CSV file making possible to compute a widespread amount of metrics (i.e., network utilization, network delay, response time, waiting time, availability) according to the requirements of the model simulated.

Another essential requirement for Cloud/Fog/IoT simulators is the support offered to deal with topology infrastructure. iFogSim just supports tree infrastructures for communications, and this aspect limits the use of this simulator for complex and more diverse scenarios; on the other hand, CloudSimSDN and YAFS have support for diverse kinds of topologies. Particularly, YAFS also supports dynamic topologies where entities and network links can be created or removed during the simulation. Thus, these two last simulators could be more suitable to model and fulfill the demands for nowadays IoT applications in Cloud/Fog environments.

Respecting the learning curve required to use the simulators, it was harder to learn to use iFogSim than YAFS, given the number of classes and modules required to review in order to complete the modeling of the desired scenario and simulation. About the documentation, there is not much material available for iFogSim, while there is some basic documentation for CloudSimSDN and YAFS, including how to configure their environments, descriptions about inputs and outputs, as well as how to run some examples. Despite this, it is iFogSim the simulator with the highest amount of citations, a measure indicating that it is the most used Fog simulator in the research community.

It is also important to notice the number of bugs and problems on CloudSim [96], which boosted a new project to entirely refactor the simulator following good coding practices to get a more stable simulator. The new simulator is called CloudSimPlus, and it is impossible to use it as the base for iFogSim or CloudSimSDN, since the number of changes that were needed to apply, even though the theoretical approach remains the same.

All three evaluated simulators showed strengths and weaknesses, and are more fitted for a given type of simulation. CloudSimSDN is more suited for experiments aimed at evaluating the network infrastructure; particularly, the possibility to define the physical topology, the virtual networks, and the workloads separately, making it useful to validate Virtual Network Embedding (VNE) mechanisms, as well as, perform experiments related to SDN environments. iFogSim and YAFS are more inclined to experiments designed to assess the performance of an application in Cloud/Fog environments. iFogSim has better support regarding the simulation of algorithms and mechanisms that require measurements of resource consumption inside VMs; on the other hand, YAFS focuses on appraising the impact of population and placement of application modules in a communication infrastructure, which is enhanced by the possibility of including complex network theory and dynamic topologies in the simulations. The final choice of the proper Fog simulator to use will depend on the goals of the experiment, the environment to model, and the desired metrics to get.

7. Conclusions and future work

Simulation has proven to be an important tool to evaluate novel solutions in networking and communication environments. However, in order to properly assess these solutions, the simulator used must offer an environment similar enough to the one being recreated, to provide realistic results. In the case of Fog computing, there are several simulators, but each one offers different features that adapt to specific evaluations.

This work offers a conceptual review on six Cloud/Fog simulators: iFogSim, CloudSimSDN, YAFS, EmuFog, FogTorchπ, and EdgeCloudSim. From this set, EmuFog is the only emulation tool, being the other five simulation tools. iFogSim, CloudSimSDN, and EdgeCloudSim are all branches from the very well-known Cloud simulator CloudSim, and they inherit all their advantages and disadvantages, including the known bugs [96]. Most of the tools are Java-based, being YAFS the only one based on Python. iFogSim turned out to be the only one that restricts the topology supported to a tree, while the rest accept an arbitrary topology. The modular application model is the most adopted one (iFogSim, YAFS, FogTorchπ, and EdgeCloudSim). According to the citation count, iFogSim is the most used Cloud/Fog simulation tool, while EmuFog is the least used following the same criteria (citation count).

Furthermore, a practical assessment was carried out for three of these tools, namely: iFogSim, CloudSimSDN, and YAFS. This evaluation allowed to compare these tools regarding their execution time and resource consumption (CPU and memory). The evaluation comprises different application types and placement policies, to try to encompass a broader range of realistic scenarios for the Cloud to Fog to IoT continuum. YAFS was the most time-consuming simulator. On the other hand, YAFS was the simulator that consumed the lesser amount of resources, with iFogSim being the simulator that consumed more resources.

In general, CloudSimSDN is more oriented for assessing of infrastructure, making it suitable to validate Virtual Network Embedding mechanisms, as well as, to perform experiments related to SDN environments. iFogSim and YAFS are more fitted for application performance evaluation. Specifically, iFogSim offers better support regarding the simulation of algorithms and mechanisms that require measurement of resource consumption inside Virtual Machines; meanwhile, YAFS provides beneficial tools to evaluate the impact of population and placement of application modules in a communication infrastructure.

It is important to mention that during the practical assessment, some implementation issues were detected. For instance, a waiting condition arose with iFogSim during the simulation of the VRGame application, that prevented the simulation from finishing in more complex scenarios.

For future work, it is envisioned to add more simulators in the evaluation, such as FogNetSim++ [32], IOTSim [35], and Fogbed [37]. Also, it would be interesting to expand the practical evaluation to these simulators as well as to the simulators that were limited to conceptual analysis in this work. Finally, future evaluation of the same tools revised in this paper is expected, to corroborate if identified issues have been repaired.

Acknowledgments

David Perez Abreu and Karima Velasquez wish to acknowledge the Portuguese funding institution FCT - Foundation for Science and Technology for supporting their research under the Ph.D. grants ≪ SFRH/BD/117538/2016 ≫ and ≪ SFRH/BD/119392/2016 ≫ respectively.

The work presented in this paper was partially carried out in the scope of the projects: ≪ MobiWise: From mobile sensing to mobility advising ≫ (P2020 SAICTPAC/0011/2015), co-financed by COMPETE 2020, Portugal 2020 - Operational Program for Competitiveness and Internationalization (POCI), European Union's ERDF (European Regional Development Fund) and the Portuguese Foundation for Science and Technology (FCT); ≪ DenseNet - PTDC/EEI-SCR/6453/2014 ≫ via Foundation for Science and Technology (FCT); and ≪ Project 5G with Nr. 024539 (POCI-01-0247-FEDER-024539) ≫ European Regional Development Fund (FEDER), through the Regional Operational Programme of Lisbon (POR LISBOA 2020) and the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework.

Appendix A. Simulation results' details

Tables A.1–A.3 present the conglomerate results (i.e., the average of the 30 simulations) for execution time, CPU, and memory consumption respectively in each scenario. On the tables, *NaN* is shown in the case that the experiment could not be completed, and thus no value is reported. *#Fogs* refers to the number of Fog nodes used in the scenario (i.e., 4, 8, 12, 16, 20), while *#Devs* identifies the amount of devices per Fog node (i.e., 4, 8, 12), giving a total of fifteen scenarios combinations for each placement policy (i.e., Cloud and Edge).

Table A.1 lists the results regarding execution time (in seconds) for all the experiments on the three simulators. It is noticeable that as the scenarios get bigger and more complex, the execution time for the simulation increases, which is expected.

The results for CPU consumption in all the experiments for the three simulators are depicted in Table A.2. In this case, the increasing expected pattern as the scenario gets more complex is only shown for YAFS. For the Java-based simulators, a decreasing trend is shown due to the execution of additional Java Virtual Machine (JVM) functions, such as the garbage collector.

Table A.3 shows the results regarding memory consumption in all the experiments for all the simulators. It is possible to see how the increasing trend is present for all three simulators as the scenarios get bigger and more complicated.

Table A1
Simulations' execution time in iFogSim, CloudSimSDN, and YAFS per each application in seconds.

			eHealth			Surveillance			VRGame		
		#Fogs \#Devs	4	8	12	4	8	12	4	8	12
iFogSim	*Cloud*	*4*	2.74	3.48	4.23	2.58	3.62	4.38	3.87	7.09	9.99
		8	3.35	5.08	6.68	3.57	5.97	8.64	7.65	17.64	NaN
		12	4.42	7.38	9.97	4.65	9.21	13.01	12.58	NaN	NaN
		16	5.31	9.14	12.81	5.97	11.75	16.25	20.1	NaN	NaN
		20	6.22	11.24	15.89	7.52	14.16	19.66	NaN	NaN	NaN
	Edge	*4*	2.30	2.97	3.87	2.30	3.18	4.01	3.18	7.42	10.64
		8	2.94	4.31	5.83	3.12	5.06	7.18	5.09	18.21	NaN
		12	3.65	5.69	8.65	4.2	7.17	11.27	7.21	NaN	NaN
		16	4.38	7.69	11.3	5.11	9.99	15.76	9.82	NaN	NaN
		20	5.12	9.51	14.85	6.28	12.65	21.41	NaN	NaN	NaN
CloudSimSDN	*Cloud*	*4*	1.01	1.21	1.58	0.9	1.48	1.88	0.85	1.13	1.41
		8	1.17	1.94	2.88	1.25	2.19	3.22	1.08	1.73	2.32
		12	1.51	2.88	4.43	1.71	3.28	4.72	1.38	2.24	3.02
		16	1.96	4.15	6.88	2.22	4.32	6.47	1.75	2.84	3.67
		20	2.40	5.40	9.39	2.84	5.43	8.32	2.05	3.28	4.79
	Edge	*4*	0.78	1.10	1.48	0.92	1.41	1.75	0.88	1.25	1.61
		8	0.95	1.49	2.09	1.14	2.14	3.12	1.15	1.81	2.42
		12	1.07	2.50	3.93	1.76	3.13	4.41	1.42	2.39	3.17
		16	1.20	3.02	5.08	2.22	3.84	5.22	1.75	2.86	3.90
		20	1.33	3.50	5.80	2.66	4.53	6.25	2.10	3.37	4.57
YAFS	*Cloud*	*4*	0.98	2.49	4.71	1.13	2.76	5.13	0.98	2.56	4.92
		8	2.45	7.77	16.31	2.78	8.42	17.15	2.56	8.03	16.71
		12	4.60	16.10	34.91	5.13	16.88	36.12	4.79	16.37	35.34
		16	7.46	27.28	60.35	8.19	28.50	62.00	7.78	27.89	61.14
		20	11.05	41.60	92.46	11.96	43.03	95.20	11.42	42.05	94.18
	Edge	*4*	1.89	6.17	13.06	10.43	28.12	50.74	1.75	4.73	9.22
		8	6.11	22.54	50.07	33.81	101.38	192.84	4.71	14.99	31.41
		12	12.65	49.32	110.81	69.77	220.76	428.88	9.05	30.87	67.54
		16	21.66	85.61	195.99	119.24	384.88	751.21	14.69	52.51	116.59
		20	32.72	132.33	301.84	181.92	593.24	1181.62	21.65	79.11	178.02

Table A2
Simulations' CPU consumption in iFogSim, CloudSimSDN, and YAFS per each application in %.%

			eHealth			Surveillance			VRGame		
		#Fogs \#Devs	4	8	12	4	8	12	4	8	12
iFogSim	*Cloud*	*4*	339.24	328.62	314.34	324.93	272.76	247.36	247.65	197.63	197.88
		8	311.70	282.27	256.49	276.47	225.76	223.71	193.60	160.27	NaN
		12	305.51	259.69	224.12	238.88	223.08	200.29	171.60	NaN	NaN
		16	276.28	234.67	212.34	229.43	204.64	186.90	156.30	NaN	NaN
		20	257.63	223.77	207.11	225.27	195.34	175.54	NaN	NaN	NaN
	Edge	*4*	335.46	314.87	302.89	328.62	300.33	261.45	291.22	197.38	198.42
		8	308.70	268.28	239.50	289.24	245.45	215.74	238.95	160.83	NaN
		12	285.96	240.08	223.83	267.46	216.11	198.55	205.85	NaN	NaN
		16	263.88	238.43	208.09	241.90	210.99	181.20	195.56	NaN	NaN
		20	246.58	218.28	196.5	229.76	189.71	167.33	NaN	NaN	NaN
CloudSimSDN	*Cloud*	*4*	210.20	256.50	279.78	222.62	273.17	270.08	211.1	258.99	281.38
		8	237.99	278.45	265.88	259.82	259.09	235.39	255.51	304.58	286.55
		12	269.21	259.70	217.21	289.72	233.91	219.03	287.34	317.65	256.33
		16	266.73	250.90	198.99	271.61	221.69	189.52	308.37	288.45	239.70
		20	245.35	215.89	171.00	265.23	203.68	171.06	309.03	260.34	205.79
	Edge	*4*	189.58	228.52	256.18	220.11	266.77	259.89	210.41	265.98	283.75
		8	226.08	266.24	253.44	247.73	263.90	217.52	257.29	284.22	258.71
		12	239.46	245.67	211.27	263.89	238.34	211.32	277.57	276.37	250.58
		16	262.18	239.44	194.39	263.93	231.10	196.48	293.43	281.07	240.53
		20	269.22	226.46	193.10	251.11	223.39	187.31	283.47	258.25	232.30

(*continued on next page*)

Table A2 (*continued*)

			eHealth			Surveillance			VRGame		
		#Fogs \#Devs	4	8	12	4	8	12	4	8	12
YAFS	*Cloud*	*4*	97.38	99.18	99.59	98.44	99.31	99.67	98.12	99.31	99.63
		8	99.20	99.75	99.87	99.33	99.78	99.89	99.24	99.77	99.88
		12	99.56	99.86	99.93	99.66	99.88	99.94	99.65	99.87	99.93
		16	99.70	99.93	99.95	99.78	99.94	99.96	99.76	99.93	99.96
		20	99.83	99.91	99.96	99.82	99.95	99.97	99.83	99.94	99.97
	Edge	*4*	99.01	99.68	99.77	99.82	99.92	99.95	99.08	99.59	99.79
		8	99.68	99.92	99.95	99.93	99.97	99.98	99.62	99.87	99.93
		12	99.85	99.95	99.98	99.96	99.98	99.98	99.79	99.93	99.96
		16	99.90	99.97	99.97	99.97	99.98	99.99	99.87	99.95	99.97
		20	99.93	99.98	99.98	99.98	99.98	99.98	99.92	99.97	99.98

Table A3
Simulations' memory consumption in iFogSim, CloudSimSDN, and YAFS per each application in MB .

			eHealth			Surveillance			VRGame		
		#Fogs\#Devs	4	8	12	4	8	12	4	8	12
iFogSim	*Cloud*	*4*	227.83	276.44	339.42	250.98	349.24	436.16	282.62	431.59	559.62
		8	277.67	407.53	525.49	357.32	549.55	676.33	415.06	547.42	NaN
		12	350.01	538.70	671.48	449.97	685.01	787.41	499.74	NaN	NaN
		16	425.09	635.10	742.10	541.14	767.59	844.61	569.24	NaN	NaN
		20	491.97	691.24	831.32	620.09	802.18	950.50	NaN	NaN	NaN
	Edge	*4*	215.38	265.87	327.02	209.34	277.84	349.09	236.55	429.56	547.13
		8	253.03	391.40	538.27	283.82	425.68	562.47	341.96	545.76	NaN
		12	329.38	540.67	719.71	358.58	567.12	707.36	437.05	NaN	NaN
		16	402.17	653.05	859.66	439.66	676.06	818.82	515.70	NaN	NaN
		20	474.23	773.60	921.87	499.63	746.66	920.23	NaN	NaN	NaN
CloudSimSDN	*Cloud*	*4*	96.18	108.18	118.02	96.97	115.85	125.26	96.25	110.25	115.27
		8	100.82	126.33	139.79	106.02	133.10	143.98	104.66	122.89	130.87
		12	110.63	141.77	155.94	123.07	148.52	159.42	112.86	136.92	141.37
		16	126.15	167.21	176.93	134.37	164.48	168.01	123.10	151.36	150.14
		20	139.15	165.93	184.31	155.01	165.94	177.38	138.45	144.63	159.17
	Edge	*4*	91.10	98.99	114.41	95.78	106.9	116.94	96.66	107.00	118.33
		8	98.50	115.25	130.11	99.58	128.69	135.67	101.64	123.25	132.67
		12	99.60	139.80	152.47	116.80	136.39	151.14	107.9	136.77	145.93
		16	105.88	154.34	163.12	120.79	157.35	157.62	117.59	151.50	155.39
		20	111.09	152.15	166.10	142.80	155.81	159.84	132.97	145.14	160.73
YAFS	*Cloud*	*4*	118.30	121.13	123.81	119.44	122.49	125.24	120.24	125.56	130.46
		8	121.10	126.77	132.20	122.26	128.32	133.66	125.36	135.59	145.06
		12	123.79	132.64	140.89	125.18	134.06	142.34	130.41	145.46	160.09
		16	126.50	138.51	149.31	128.02	139.75	150.76	135.40	155.53	174.64
		20	129.43	143.99	157.69	130.90	145.50	159.15	140.41	165.39	189.27
	Edge	*4*	116.07	116.62	117.06	116.49	134.89	159.48	120.53	126.08	131.45
		8	116.63	117.55	118.51	117.12	154.26	203.23	126.11	136.68	147.33
		12	117.13	118.67	120.08	117.92	173.61	246.92	131.29	147.47	163.31
		16	117.47	119.84	121.54	118.75	193.13	290.49	136.61	158.23	178.93
		20	118.04	120.72	122.75	119.56	212.20	334.3	142.07	168.72	194.94

References

[1] M. Chiang, T. Zhang, Fog and iot: an overview of research opportunities, IEEE Internet Things J. 3 (6) (2016) 854–864, https://doi.org/10.1109/JIOT.2016.2584538.

[2] A.V. Dastjerdi, R. Buyya, Fog computing: helping the internet of things realize its potential, Computer 49 (8) (2016) 112–116, https://doi.org/10.1109/MC.2016.245.

[3] M. Iorga, L. Feldman, R. Barton, M.J. Martin, N.S. Goren, C. Mahmoudi, Fog computing conceptual model, Technical Report NIST Special Publication 500-325, National Institute of Standard and Technology, 2018, https://doi.org/10.6028/NIST.SP.500-325. Accessed: 2019-08-24

[4] K. Velasquez, D.P. Abreu, M.R.M. Assis, C. Senna, D.F. Aranha, L.F. Bittencourt, N. Laranjeiro, M. Curado, M. Vieira, E. Monteiro, E. Madeira, Fog orchestration for the internet of everything: state-of-the-art and research challenges, J. Internet Serv. Appl. 9 (1) (2018) 1–23, https://doi.org/10.1186/s13174-018-0086-3.

[5] F. Fakhfakh, H.H. Kacem, A.H. Kacem, Simulation tools for cloud computing: a survey and comparative study, 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), IEEE, Wuhan, China, 2017, pp. 221–226, https://doi.org/10.1109/ICIS.2017.7959997.

[6] J. Byrne, S. Svorobej, K. Giannoutakis, D. Tzovaras, P. Byrne, P.-O. Östberg, A. Gourinovitch, T. Lynn, A review of cloud computing simulation platforms and

related environments, Closer 2017: The 7th International Conference on Cloud Computing and Services Science, SCITEPRESS Digital Library, Porto, Portugal, 2017, pp. 679–691, https://doi.org/10.5220/0006373006790691.
[7] S. Svorobej, P. Endo, M. Bendechache, C. Filelis-Papadopoulos, K. Giannoutakis, G. Gravvanis, D. Tzovaras, J. Byrne, T. Lynn, Simulating fog and edge computing scenarios: an overview and research challenges, Future Internet 11 (3) (2019) 1–15, https://doi.org/10.3390/fi11030055.
[8] A. Ahmed, A.S. Sabyasachi, Cloud computing simulators: a detailed survey and future direction, 2014 IEEE International Advance Computing Conference (IACC), IEEE, Gurgaon, India, 2014, pp. 866–872, https://doi.org/10.1109/IAdCC.2014.6779436.
[9] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. Pract. Exp. 41 (1) (2011) 23–50, https://doi.org/10.1002/spe.995.
[10] B. Wickremasinghe, R.N. Calheiros, R. Buyya, Cloudanalyst: a cloudsim-based visual modeller for analysing cloud computing environments and applications, 2010 24th IEEE International Conference on Advanced Information Networking and Applications, IEEE, Perth, WA, Australia, 2010, pp. 446–452, https://doi.org/10.1109/AINA.2010.32.
[11] A.S. Kushwaha, B. Alam, G. Kaur, Observation of energy efficiency in green cloud simulator, 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence), IEEE, Noida, India, 2016, pp. 135–140, https://doi.org/10.1109/CONFLUENCE.2016.7508102.
[12] S. Lim, B. Sharma, G. Nam, E.K. Kim, C.R. Das, MDCSim: a multi-tier data center simulation, platform, 2009 IEEE International Conference on Cluster Computing and Workshops, IEEE, New Orleans, LA, USA, 2009, pp. 1–9, https://doi.org/10.1109/CLUSTR.2009.5289159.
[13] A. Núñez, J.L. Vázquez-Poletti, A.C. Caminero, G.G. Castañé, J. Carretero, I.M. Llorente, Icancloud: A Flexible and scalable cloud infrastructure simulator, J. Grid Comput. 10 (1) (2012) 185–209, https://doi.org/10.1007/s10723-012-9208-5.
[14] S.K. Garg, R. Buyya, Networkcloudsim: modelling parallel applications in cloud simulations, 2011 Fourth IEEE International Conference on Utility and Cloud Computing, IEEE, Victoria, NSW, Australia, 2011, pp. 105–113, https://doi.org/10.1109/UCC.2011.24.
[15] R.N. Calheiros, M.A. Netto, C.A. De Rose, R. Buyya, Emusim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications, Softw. Pract. Exp. 43 (5) (2013) 595–612, https://doi.org/10.1002/spe.2124.
[16] S. Ostermann, K. Plankensteiner, R. Prodan, T. Fahringer, Groudsim: an event-based simulation framework for computational grids and clouds, Euro-Par 2010 Parallel Processing Workshops, Springer, Heidelberg, Germany, 2011, pp. 305–313, https://doi.org/10.1007/978-3-642-21878-1_38.
[17] M. Tighe, G. Keller, M. Bauer, H. Lutfiyya, Dcsim: a data centre simulation tool for evaluating dynamic virtualized resource management, 2012 8th International Conference on Network and Service Management (Cnsm) and 2012 Workshop on Systems Virtualiztion Management (Svm), IEEE, Las Vegas, NV, USA, 2012, pp. 385–392.
[18] J. Jung, H. Kim, Mr-cloudsim: designing and implementing mapreduce computing model on cloudsim, 2012 International Conference on ICT Convergence (ICTC), IEEE, Jeju Island, South Korea, 2012, pp. 504–509, https://doi.org/10.1109/ICTC.2012.6387186.
[19] M. Shiraz, A. Gani, R.H. Khokhar, E. Ahmed, An extendable simulation framework for modeling application processing potentials of smart mobile devices for mobile cloud computing, 2012 10th International Conference on Frontiers of Information Technology, IEEE, Islamabad, India, 2012, pp. 331–336, https://doi.org/10.1109/FIT.2012.66.
[20] S. Sotiriadis, N. Bessis, N. Antonopoulos, A. Anjum, SimIC: designing a New Inter-cloud Simulation Platform for Integrating Large-Scale Resource Management, 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), IEEE, Barcelona, Spain, 2013, pp. 90–97, https://doi.org/10.1109/AINA.2013.123.
[21] G. Sakellari, G. Loukas, A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing, Simulation Modelling Practice and Theory 39 (1) (2013) 92–103, https://doi.org/10.1016/j.simpat.2013.04.002. S.I.Energy efficiency in grids and clouds
[22] Y. Jararweh, Z. Alshara, M. Jarrah, M. Kharbutli, M. Alsaleh, Teachcloud: a cloud computing educational toolkit, Int. J. Cloud Comput. 2 (2) (2013) 237–257, https://doi.org/10.1504/IJCC.2013.055269.
[23] A. Kohne, M. Spohr, L. Nagel, O. Spinczyk, Federatedcloudsim: a sla-aware federated cloud simulation framework, Proceedings of the 2Nd International Workshop on CrossCloud Systems, CCB '14, ACM, Bordeaux, France, 2014, pp. 3:1–3:5, https://doi.org/10.1145/2676662.2676674.
[24] A. Zhou, S. Wang, Q. Sun, H. Zou, F. Yang, Ftcloudsim: a simulation tool for cloud service reliability enhancement mechanisms, Proceedings Demo &#38; Poster Track of ACM/IFIP/USENIX International Middleware Conference, MiddlewareDPT '13, ACM, Beijing, China, 2013, pp. 2:1–2:2, https://doi.org/10.1145/2541614.2541616.
[25] W. Chen, E. Deelman, Workflowsim: a toolkit for simulating scientific workflows in distributed environments, 2012 IEEE 8th International Conference on E-Science, IEEE, Chicago, IL, USA, 2012, pp. 1–8, https://doi.org/10.1109/eScience.2012.6404430.
[26] Z. Cai, Q. Li, X. Li, Elasticsim: a toolkit for simulating workflows with cloud resource runtime auto-scaling and stochastic task execution times, J. Grid Comput. 15 (2) (2017) 257–272, https://doi.org/10.1007/s10723-016-9390-y.
[27] J. Son, A.V. Dastjerdi, R.N. Calheiros, X. Ji, Y. Yoon, R. Buyya, CloudSimSDN: modeling and simulation of software-defined cloud data centers, 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE, Shenzhen, China, 2015, pp. 475–484, https://doi.org/10.1109/CCGrid.2015.87.
[28] W. Tian, Y. Zhao, M. Xu, Y. Zhong, X. Sun, A toolkit for modeling and simulation of real-time virtual machine allocation in a cloud data center, IEEE Trans. Autom. Sci. Eng. 12 (1) (2015) 153–161, https://doi.org/10.1109/TASE.2013.2266338.
[29] U.U. Rehman, A. Ali, Z. Anwar, seccloudsim: secure cloud simulator, 2014 12th International Conference on Frontiers of Information Technology, IEEE, Islamabad, Pakistan, 2014, pp. 208–213, https://doi.org/10.1109/FIT.2014.47.
[30] P. Suryateja, A comparative analysis of cloud simulators, Int. J. Mod. Educ. Comput. Sci. 8 (4) (2016) 64–71, https://doi.org/10.5815/ijmecs.2016.04.08.
[31] H. Gupta, A. Vahid Dastjerdi, S.K. Ghosh, R. Buyya, Ifogsim: a toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments, J. Softw. 47 (9) (2017) 1275–1296, https://doi.org/10.1002/spe.2509.
[32] T. Qayyum, A.W. Malik, M.A. Khan Khattak, O. Khalid, S.U. Khan, Fognetsim + +: a toolkit for modeling and simulation of distributed fog environment, IEEE Access 6 (1) (2018) 63570–63583, https://doi.org/10.1109/ACCESS.2018.2877696.
[33] A. Brogi, S. Forti, A. Ibrahim, How to best deploy your fog applications, probably, 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), IEEE, Madrid, Spain, 2017, pp. 105–114, https://doi.org/10.1109/ICFEC.2017.8.
[34] C. Sonmez, A. Ozgovde, C. Ersoy, EdgeCloudSim: an environment for performance evaluation of Edge Computing systems, 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), IEEE, Valencia, Spain, 2017, pp. 39–44, https://doi.org/10.1109/FMEC.2017.7946405.
[35] X. Zeng, S.K. Garg, P. Strazdins, P.P. Jayaraman, D. Georgakopoulos, R. Ranjan, Iotsim: A simulator for analysing iot applications, Journal of Systems Architecture 72 (1) (2017) 93–107, https://doi.org/10.1016/j.sysarc.2016.06.008. Design Automation for Embedded Ubiquitous Computing Systems
[36] R. Mayer, L. Graser, H. Gupta, E. Saurez, U. Ramachandran, Emufog: extensible and scalable emulation of large-scale fog computing infrastructures, 2017 IEEE Fog World Congress (FWC), IEEE, Santa Clara, CA, USA, 2017, pp. 1–6, https://doi.org/10.1109/FWC.2017.8368525.
[37] A. Coutinho, F. Greve, C. Prazeres, J. Cardoso, Fogbed: a rapid-prototyping emulation environment for fog computing, 2018 IEEE International Conference on Communications (ICC), IEEE, Kansas City, MO, USA, 2018, pp. 1–7, https://doi.org/10.1109/ICC.2018.8423003.
[38] T. Pflanzner, A. Kertesz, B. Spinnewyn, S. Latr, Mobiotsim: towards a mobile iot device simulator, 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), IEEE, Vienna, Austria, 2016, pp. 21–27, https://doi.org/10.1109/W-FiCloud.2016.21.
[39] SimpleSoft, SimpleIoTSimulator, 2019, (https://www.smplsft.com/SimpleIoTSimulator.html). Accessed: 2019-10-18.
[40] IBM, IBM Bluemix Platform, 2019, (https://console.ng.bluemix.net). Accessed: 2019-10-18.
[41] Google, Google Cloud Platform, 2019, (https://cloud.google.com/solutions/iot). Accessed: 2019-10-18.
[42] M.M. Lopes, W.A. Higashino, M.A. Capretz, L.F. Bittencourt, Myifogsim: a simulator for virtual machine migration in fog computing, Companion Proceedings of the10th International Conference on Utility and Cloud Computing, UCC '17 Companion, ACM, Austin, Texas, USA, 2017, pp. 47–52, https://doi.org/10.1145/3147234.3148101.
[43] Contiki, Contiki Cooja, 2019, (http://www.contiki-os.org/start.html). Accessed: 2019-10-18.
[44] A. Brogi, S. Forti, Qos-aware deployment of iot applications through the fog, IEEE Internet Things J. 4 (5) (2017) 1185–1192, https://doi.org/10.1109/JIOT.2017.2701408.
[45] J. Byrne, S. Svorobej, A. Gourinovitch, D.M. Elango, P. Liston, P.J. Byrne, T. Lynn, RECAP simulator: simulation of cloud/edge/fog computing scenarios, 2017 Winter Simulation Conference (WSC), IEEE, Las Vegas, NV, USA, 2017, pp. 4568–4569, https://doi.org/10.1109/WSC.2017.8248208.
[46] N. Mohan, J. Kangasharju, Edge-fog cloud: a distributed cloud for internet of things computations, 2016 Cloudification of the Internet of Things (CIoT), IEEE, Paris, France, 2016, pp. 1–6, https://doi.org/10.1109/CIOT.2016.7872914.

[47] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, B. Koldehofe, Mobile fog: a programming model for large-scale applications on the internet of things, Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing, MCC '13, ACM, New York, NY, USA, 2013, pp. 15–20, https://doi.org/10.1145/2491266.2491270.

[48] W. Tian, M. Xu, A. Chen, G. Li, X. Wang, Y. Chen, Open-source simulators for cloud computing: Comparative study and challenging issues, Simulation Modelling Practice and Theory 58 (2) (2015) 239–254, https://doi.org/10.1016/j.simpat.2015.06.002. Special issue on Cloud Simulation

[49] M.A. Sharkh, A. Kanso, A. Shami, P. Öhlén, Building a cloud on earth: a study of cloud computing data center simulators, Comput. Netw. 108 (1) (2016) 78–96, https://doi.org/10.1016/j.comnet.2016.06.037.

[50] D. Meisner, J. Wu, T.F. Wenisch, Bighouse: a simulation infrastructure for data center systems, 2012 IEEE International Symposium on Performance Analysis of Systems Software, IEEE, New Brunswick, NJ, USA, 2012, pp. 35–45, https://doi.org/10.1109/ISPASS.2012.6189204.

[51] D. Alshammari, J. Singer, T. Storer, Performance evaluation of cloud computing simulation tools, 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), IEEE, Chengdu, China, 2018, pp. 522–526, https://doi.org/10.1109/ICCCBDA.2018.8386571.

[52] B. Lantz, B. Heller, N. McKeown, A network in a laptop: rapid prototyping for software-defined networks, Hotnets 2010, ACM, Monterey, California, 2010, pp. 1–6, https://doi.org/10.1145/1868447.1868466.

[53] H. Gupta, A. Vahid Dastjerdi, S.K. Ghosh, R. Buyya, iFogSim - GitHub Repository, 2019, (https://github.com/Cloudslab/iFogSim). Accessed: 2019-02-24.

[54] M.M. Mahmoud, J.J. Rodrigues, K. Saleem, J. Al-Muhtadi, N. Kumar, V. Korotaev, Towards energy-aware fog-enabled cloud of things for healthcare, Comput. Electr. Eng. 67 (1) (2018) 58–69, https://doi.org/10.1016/j.compeleceng.2018.02.047.

[55] T. Sigwele, P. Pillai, Y.-F. Hu, Saving energy in mobile devices using mobile device cloudlet in mobile edge computing for 5g, 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE, Exeter, UK, 2017, pp. 422–428, https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2017.69.

[56] R. Mahmud, K. Ramamohanarao, R. Buyya, Latency-aware application module management for fog computing environments, ACM Trans. Internet Technol. 19 (1) (2018) 9:1–9:21, https://doi.org/10.1145/3186592.

[57] T. Huang, W. Lin, Y. Li, L. He, S. Peng, A latency-aware multiple data replicas placement strategy for fog computing, J. Signal Process. Syst. (2019) 1–14, https://doi.org/10.1007/s11265-019-1444-5.

[58] A.R. Benamer, H. Teyeb, N. Ben Hadj-Alouane, Latency-aware placement heuristic in fog computing environment, On the Move to Meaningful Internet Systems. OTM 2018 Conferences, Springer International Publishing, Valletta, Malta, 2018, pp. 241–257, https://doi.org/10.1007/978-3-030-02671-4_14.

[59] S. Shukla, M.F. Hassan, L.T. Jung, A. Awang, M.K. Khan, A 3-tier architecture for network latency reduction in healthcare internet-of-things using fog computing and machine learning, Proceedings of the 2019 8th International Conference on Software and Computer Applications, ICSCA '19, ACM, Penang, Malaysia, 2019, pp. 522–528, https://doi.org/10.1145/3316615.3318222.

[60] P.G. Vinueza Naranjo, E. Baccarelli, M. Scarpiniti, Design and energy-efficient resource management of virtualized networked fog architectures for the real-time support of iot applications, J. Supercomput. 74 (6) (2018) 2470–2507, https://doi.org/10.1007/s11227-018-2274-0.

[61] E. Baccarelli, P.G.V. Naranjo, M. Scarpiniti, M. Shojafar, J.H. Abawajy, Fog of everything: energy-efficient networked computing architectures, research challenges, and a case study, IEEE Access 5 (1) (2017) 9882–9910, https://doi.org/10.1109/ACCESS.2017.2702013.

[62] J. Son, R. Buyya, A taxonomy of software-defined networking (sdn)-enabled cloud computing, ACM Comput. Surv. 51 (3) (2018) 59:1–59:36, https://doi.org/10.1145/3190617.

[63] J. Son, A.V. Dastjerdi, R.N. Calheiros, X. Ji, Y. Yoon, R. Buyya, CloudSimSDN - GitHub Repository, 2019, (https://github.com/Cloudslab/cloudsimsdn). Accessed: 2019-01-12.

[64] A. Al-Mansoori, J. Abawajy, M. Chowdhury, Sdn enabled bdsp in public cloud for resource optimization, Wireless Netw. (2018) 1–11, https://doi.org/10.1007/s11276-018-1887-9.

[65] J. Son, A.V. Dastjerdi, R.N. Calheiros, R. Buyya, Sla-aware and energy-efficient dynamic overbooking in sdn-based cloud data centers, IEEE Trans. Sustainable Comput. 2 (2) (2017) 76–89, https://doi.org/10.1109/TSUSC.2017.2702164.

[66] J. Son, R. Buyya, Latency-aware virtualized network function provisioning for distributed edge clouds, J. Syst. Softw. 152 (1) (2019) 24–31, https://doi.org/10.1016/j.jss.2019.02.030.

[67] A. Fischer, J.F. Botero, M.T. Beck, H. de Meer, X. Hesselbach, Virtual network embedding: a survey, IEEE Commun. Surv. Tutorials 15 (4) (2013) 1888–1906, https://doi.org/10.1109/SURV.2013.013013.00155.

[68] I. Lera, C. Guerrero, C. Juiz, YAFS: a simulator for IoT scenarios in fog computing, CoRR abs/1902.01091 (2019).

[69] I. Lera, C. Guerrero, C. Juiz, YAFS: yet Another Fog Simulator - GitHub Repository, 2019b, (https://github.com/acsicuib/YAFS). Accessed: 2019-03-13.

[70] I. Lera, C. Guerrero, C. Juiz, Comparing centrality indices for network usage optimization of data placement policies in fog devices, 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC), IEEE, Barcelona, Spain, 2018, pp. 115–122, https://doi.org/10.1109/FMEC.2018.8364053.

[71] I. Lera, C. Guerrero, C. Juiz, Availability-aware service placement policy in fog computing based on graph partitions, IEEE Internet Things J. 6 (2) (2019) 3641–3651, https://doi.org/10.1109/JIOT.2018.2889511.

[72] P. Wette, M. Draxler, A. Schwabe, F. Wallaschek, M. Zahraee, H. Karl, MaxiNet: distributed emulation of software-defined networks, Networking Conference, 2014 IFIP, IEEE, Trondheim, Norway, 2014, pp. 1–9, https://doi.org/10.1109/IFIPNetworking.2014.6857078.

[73] R. Mayer, L. Graser, H. Gupta, E. Saurez, U. Ramachandran, EmuFog - GitHub Repository, 2019, (https://github.com/emufog/emufog). Accessed: 2019-01-02.

[74] A. Medina, A. Lakhina, I. Matta, J. Byers, BRITE: an approach to universal topology generation, MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, IEEE, Cincinnati, OH, USA, 2001, pp. 346–353, https://doi.org/10.1109/MASCOT.2001.948886.

[75] CAIDA, Center for Applied Internet Data Analysis, 2019, (http://www.caida.org). Accessed: 2019-01-03.

[76] L. Grases, Design and Implementation of an Evaluation Testbed for Fog Computing Infrastructure and Applications, Institute of Parallel and Distributed Systems. University of Stuttgart, Germany, 2017 Master's thesis.

[77] C. Cicconetti, M. Conti, A. Passarella, An architectural framework for serverless edge computing: design and emulation tools, 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, 2018, pp. 48–55, https://doi.org/10.1109/CloudCom2018.2018.00024.

[78] A. Brogi, S. Forti, FogTorchPi - GitHub Repository, 2019, (https://github.com/di-unipi-socc/FogTorch). Accessed: 2019-02-09.

[79] V. De Maio, I. Brandic, First hop mobile offloading of dag computations, Proceedings of the 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid '18, IEEE Press, Washington, District of Columbia, 2018, pp. 83–92, https://doi.org/10.1109/CCGRID.2018.00023.

[80] V. De Maio, I. Brandic, Multi-objective mobile edge provisioning in small cell clouds, Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering, ICPE '19, ACM, Mumbai, India, 2019, pp. 127–138, https://doi.org/10.1145/3297663.3310301.

[81] C. Sonmez, A. Ozgovde, C. Ersoy, EdgeCloudSim - GitHub Repository, 2019, (http://bit.do/edgecs). Accessed: 2019-04-14.

[82] C. Sonmez, A. Ozgovde, C. Ersoy, Edgecloudsim: an environment for performance evaluation of edge computing systems, Trans. Emerg. Telecommun. Technol. 29 (11) (2018) 1–17, https://doi.org/10.1002/ett.3493.

[83] C. Sonmez, A. Ozgovde, C. Ersoy, Fuzzy workload orchestration for edge computing, IEEE Trans. Netw. Serv. Manage. 16 (2) (2019) 769–782, https://doi.org/10.1109/TNSM.2019.2901346.

[84] S.B. Ousmane, B.C.S. Mbacke, N. Ibrahima, A game theoretic approach for virtual machine allocation security in cloud computing, Proceedings of the 2Nd International Conference on Networking, Information Systems & Security, NISS19, ACM, Rabat, Morocco, 2019, pp. 47:1–47:6, https://doi.org/10.1145/3320326.3320379.

[85] C.S.M. Babou, D. Fall, S. Kashihara, I. Niang, Y. Kadobayashi, Home edge computing (hec): design of a new edge computing technology for achieving ultra-low latency, Edge Computing – EDGE 2018, Springer International Publishing, Seattle, WA, USA, 2018, pp. 3–17, https://doi.org/10.1007/978-3-319-94340-4_1.

[86] R.F. Hussain, M.A. Salehi, O. Semiari, Serverless edge computing for green oil and gas industry, CoRR abs/1905.04460 (2019).

[87] S. Lee, H. Kim, Aco-based optimal node selection method for qoe improvement in mec environment, 2019 International Conference on Electronics, Information, and Communication (ICEIC), IEEE, Auckland, New Zealand, 2019, pp. 1–4, https://doi.org/10.23919/ELINFOCOM.2019.8706396.

[88] ResearchGate, CloudSim - ResearchGate community group, 2019, (https://www.researchgate.net/topic/CloudSim). Accessed: 2019-02-10.

[89] G. Groups, CloudSim - Google community group, 2019, (https://groups.google.com/forum/#!forum/cloudsim). Accessed: 2019-05-10.

[90] I. Lera, C. Guerrero, YAFS Documentation - Release 3.0, 2019, (http://bit.do/yafsdoc). Accessed: 2019-03-11.

[91] A.D. Perez, V. Karima, Curado, Marilia, Monteiro, Edmundo, A survey of Cloud/Fog simulators - GitLab Repository, 2019, (https://git.dei.uc.pt/dabreu/

FogSimulatorsSurvey.git). Accessed: 2019-07-08.
[92] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, CloudSim - GitHub Repository, 2019, (https://github.com/Cloudslab/cloudsim). Accessed: 2019-06-14.
[93] ORACLE, Java Mission Control, 2019, (http://bit.do/javamissioncontrol). Accessed: 2019-03-20.
[94] GitHub, psrecord - A small tool to track process' activity, 2019, (https://github.com/astrofrog/psrecord). Accessed: 2019-02-10.
[95] ORACLE, The Java Virtual Machine Specification, 2019, (https://docs.oracle.com/javase/specs/jvms/se12/html/index.html). Accessed: 2019-04-19.
[96] M.C.S. Filho, R.L. Oliveira, C.C. Monteiro, P.R.M. Incio, M.M. Freire, CloudSim Plus: a cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness, 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), IEEE, Lisbon, Portugal, 2017, pp. 400–406, https://doi.org/10.23919/INM.2017.7987304.