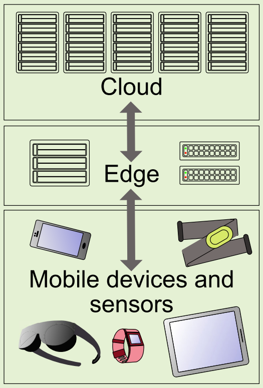
Demystifying Usability of Open-Source Computational Offloading Simulators: Performance Evaluation Campaign

Daria Alekseeva, *Student Member, IEEE*, Aleksandr Ometov, *Senior Member, IEEE*, and Elena Simona Lohan, *Senior Member, IEEE*



***Abstract*—Along with analysis and practical implementation, simulations play a key role in wireless networks and computational offloading research for several reasons. First, the simulations provide the ability to easily obtain the data for a complex system's model evaluation. Second, simulated data provide a controlled environment for experimentation, allowing models and algorithms to be tested for robustness and identifying potential limitations before deploying them in real-world applications. Choosing the most appropriate tool for simulation might be challenging and depends on several factors, such as the main purpose, complexity of data, researcher skills, community support, and available budget. As of the time of the present analysis, several system-level open-source tools for modeling computational offloading also cover the systems' communications side, such as CloudSim, CloudSim Plus, IoTSim-Edge, EdgeCloudSim, iFogSim2, PureEdgeSim, and YAFS. This work presents an evaluation of those based on the unique features and performance results of intensive workload- and delay-tolerant scenarios: XR with an extremely high data rate and workload; remote monitoring with a low data rate with moderate delays and workload requirements; and data streaming as a general human traffic with a relatively high bit rate but moderate workload. The work concludes that CloudSim provides a reliable environment for virtualization on the host resources, while YAFS shows minimal hardware usage, while IoTSim-Edge, PureEdgeSim, and EdgeCloudSim have fewer implemented features.**

***Index Terms*—Cloud computing, edge computing, fog computing, modeling, simulation.**

I. Introduction

The rapid growth of network services has forced the creation of new methods to offload the computations and data. The architecture of the Internet of Things (IoT)-driven scenarios, e.g., smart home and health monitoring, consists of the entities to sink and process data [1], [2]. The idea to utilize the resources of remote computers improved info-communication technologies, allowing one to store and process large amounts of data without wasting own machine's resources.

Historically, Mobile Cloud Computing (MCC) implies the computational offloading from mobile devices to cloud servers via the communication link [3]. This paradigm allows to use an enormous computing and storage capacity. Even though datacenters have high performance, they also consume a lot of power, which can cause global problems. The distance to the nearest datacenter greatly impacts the latency. As of February 2023, there are only 24 datacenters in Finland, where 18 are located in the capital region [4]. The connection time from northern Finland to the nearest Cloud server might be counted by several seconds. In contrast, some delay-sensitive applications would require orders of magnitude less [5].

Shifting the offloading to the edge of the network aims to assist in resolving the latency issue. An edge is a server near the gateways that can process data with lower communication latency than in the cloud because of its proximity to the user. Like edge, fog computing is another paradigm that processes

Manuscript received 4 August 2023; accepted 26 August 2023. Date of publication 13 September 2023; date of current version 14 December 2023. The work of Daria Alekseeva was supported in part by the Doctoral Training Network in ELectronics, Telecommunications and Automation (DELTA), in part by the Pekka Ahonen Fund, and in part by the Doctoral Grant of the Information Technology and Communications Science Faculty at Tampere University. The associate editor coordinating the review of this article and approving it for publication was Prof. Qiliang Li. *(Corresponding author: Daria Alekseeva.)*

The authors are with the Faculty of Information Technology and Communication Sciences, Tampere University, 33720 Tampere, Finland (e-mail: daria.alekseeva@tuni.fi).

Digital Object Identifier 10.1109/JSEN.2023.3310669

big tasks not far from the user with minimum delay [6]. Nowadays, there are more than six emerging paradigms for optimal processing data in remote servers, such as mobile edge, cloud computing, and fog computing, that have their own advantages according to the particular use case and its requirements [7]. The growing attention to the computing paradigms correlates with the growing number of emerging toolkits for computing paradigms simulations since actual implementations of testbeds appear to be close to impossible. There is a wide range of tools for validating, generating, transmitting, and offloading data, as shown in Fig. 1. The left pillar represents the nature of the environment, the central part of figure shows the tool's name, and the right part represents the research topics for which the tool was used. In 2022, MATLAB became the most popular tool among researchers applying it from wireless connections and IoT to offload scenarios due to its mathematical nature and significant base of toolboxes. Python-based Integrated Development Environment (IDE) gained the most popularity for modeling over the years because of its implementation simplicity and high availability of resources. Network Simulator 3 (ns-3) is a widely used network simulator that is keeping popularity over the years. Recently developed tools are gaining popularity in various research fields, e.g., satellite networks and connected vehicles.

The practical reason for simulations is the allowance to test the robustness of new models and algorithms without using real-world data, which might be challenging to obtain, and to identify potential issues and limitations before deploying them in real-world applications. Choosing the right tool for data simulation depends on several factors. The first and principal is the purpose of the simulation and what kind of data you need to obtain. Tools might have unique features suitable for specific scenarios only. Another factor is the user's skills and his/her preferences in the programming language, Operating System (OS) or Graphical User Interface (GUI). Last but not least is the amount of dedicated budget for purchasing.

The main contribution of this work is the evaluation of the existing open-source simulators used for modeling cloud, fog, and edge computing scenarios from the systems' communications side based on implemented and unique features and their performance. We analyzed the simulators used in the computational offloading, Mobile Edge Computing (MEC), and MCC. The work includes the evaluation of the following tools: CloudSim, CloudSim Plus, IoTSim-Edge, EdgeCloudSim, iFogSim2, PureEdgeSim, and YAFS.

Notably, not all tools shown in Fig. 1 under computing/offloading research topics were included in the comparison. MATLAB is a matrix programming language suited for analytic model validation by conducting extensive simulations. It has applicability in any engineering research and does not specialize in computing simulations; therefore, it was not included in this overview. OMNeT++ Discrete Event Simulator (OMNeT++) and ns-3 are also widely used network simulators among researchers. ns-3 is not suitable for the IoT simulation at the edge level since it does not support the scheduling and application composition features, while OMNeT++ does not support edge communication protocols. Therefore, they were not included either. GreenCloud is a packet-level simulation tool, which can measure the energy consumption of datacenter components. This simulator only focuses on the calculation of energy consumption to ensure energy-aware placement [8].

The rest of this article is organized as follows. Section II contains the introduction to the computing paradigms, the review of related works, the list of criteria, and the simulators' descriptions. Section III presents their performance comparison. This article ends with a discussion and recommendations.

II. Features Evaluation

This section introduces the fog–edge–cloud computing paradigms based on [7] and delves deeper into the literature review of works that compare tools. Then, it presents the developed set of metrics and the simulator's description.

A. Background on Computational Continuum

In 2011, the National Institute of Standards and Technology (NIST), USA, published an official paper providing a comprehensive definition of cloud computing. According to NIST, cloud computing is characterized as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [9]. Cloud computing is a powerful datacenter comprising several interconnected nodes through high-speed channels. The cloud architecture consists of two hierarchical levels: the end user and the datacenter (user cloud). A stable connection to the Internet is a key requirement for cloud service providers. Their primary goal is to allocate the appropriate node to complete the user's task computation while ensuring data security.

The cloud architecture's physical layer includes servers, network equipment, and storage devices. The lowest layer of the cloud consists of physical infrastructure drivers and cloud drivers, facilitating communication with hardware and other external clouds. The core of the cloud OS encompasses a Virtual Machine (VM) manager, network manager, storage manager, and other relevant components [10]. The top layer of the OS features various management tools, such as administrator tools, service managers, schedulers, and cloud interfaces. Clients connect to the server through this cloud interface.

The initial computing paradigm aimed to forward data to the cloud for analysis. In addition, cloud computing has enabled big data analysis, accessibility from multiple platforms, and fast computational speed due to its high computational power. However, the proliferation of wearables, sensors, and IoT devices has posed more stringent requirements in terms of mobility support, geodistribution, location

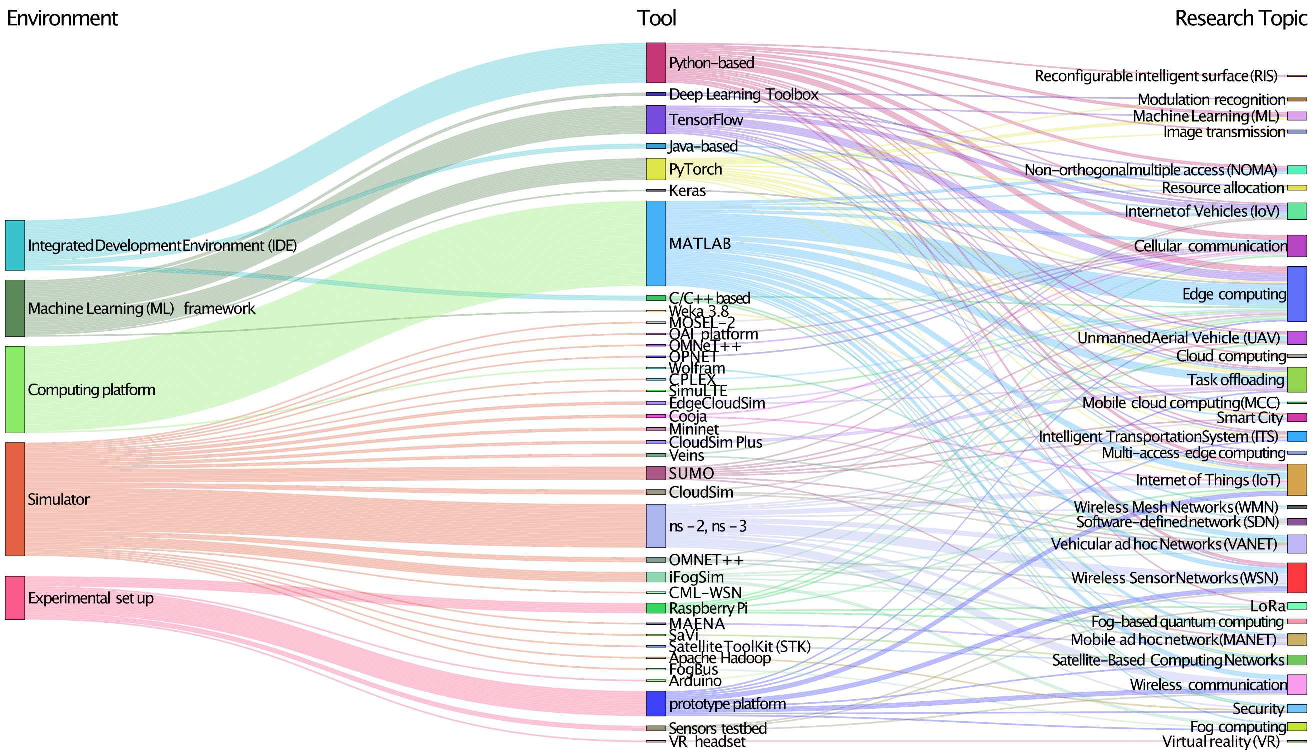


Fig. 1. Most used environments in the telecommunication, computer science, and engineering area as of 2023.

awareness, and latency. Consequently, new paradigms, such as edge and fog computing, have emerged, aiming to reduce the distance between end devices and the central server.

Fog computing is a distributed computing infrastructure that brings computational capabilities closer to the user while retaining cloud-like features. This approach allows for data storage and processing with lower latency, better location awareness, and higher Quality of Service (QoS) for real-time applications [11]. On the other hand, edge computing locates the datacenter at the actual network edge providing computing offloading, data storage, and data processing services [12]. These paradigms share similarities but disagree regarding processing location and types of used hardware [13]. Fog and edge computing aims to bring data processing nodes closer to the user, but the key difference lies in the role of the first node in the network. Edge devices are positioned as the first node, while fog nodes' proximity depends on the availability of servers. The hardware in edge computing may include lower end devices, whereas fog computing relies solely on server-based hardware. Edge computing provides computation on the end network servers, while fog computing processes data at the Local Area Network (LAN) level.

In summary, cloud computing was precisely defined by NIST in 2011 as a model for ubiquitous access to configurable computing resources. Since then, computing paradigms have evolved, leading to the emergence of fog and edge computing. These new paradigms focus on reducing latency and proximity between users and servers, see Fig. 2, presenting new opportunities and challenges for Information and Communication Technology (ICT), especially in the medical domain [7].

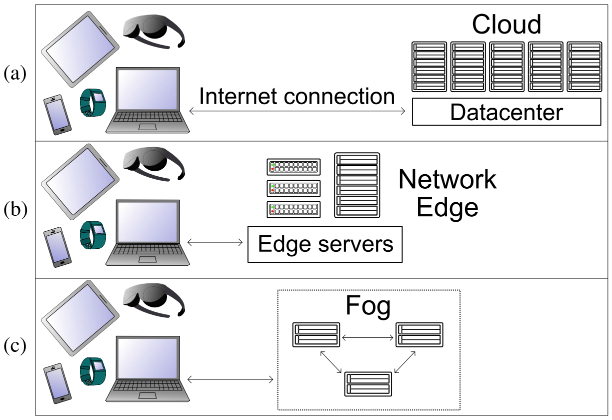


Fig. 2. Most common task offloading models. (a) Cloud computing. (b) Edge computing. (c) Fog computing.

B. Related Works

A comparative evaluation of simulators is necessary to demonstrate the superiority of a particular tool in a specific scenario becoming a subject of many works.

Aljabry and Al-Suhail [14] introduced a brief survey on network simulators for Vehicular ad hoc Network (VANET). The authors reviewed many popular simulators but provided no simulation results or performance evaluation. Kang et al. [15] presented a comprehensive survey on network simulators

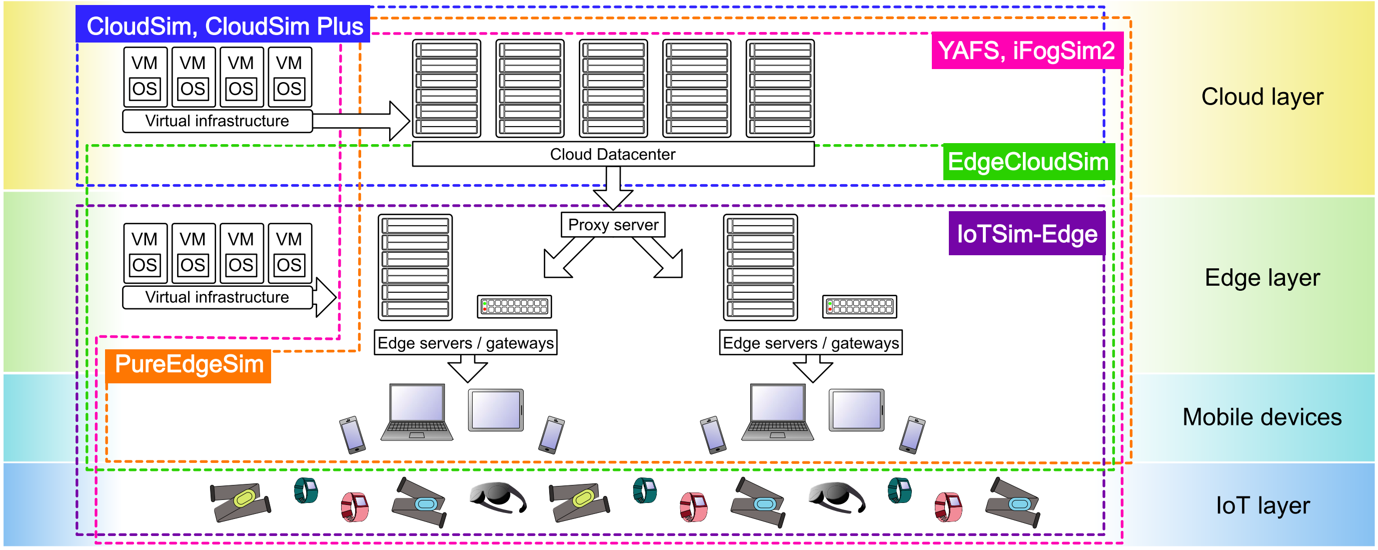


Fig. 3. Diversity of simulation tools for offloading scenarios and their applicability according to different abstraction levels.

for Airborne ad hoc Network (AANET) or Flying ad hoc Network (FANET) and Underwater Sensor Network (UWSN). Patel et al. [16] presented a comparative study on network simulators. The conclusion showed that OMNeT++ and Network Simulator 2 (ns-2) were the most appropriate network simulators for large-scale network simulators. The promising tool ns-3 was gaining popularity as an easy-to-use tool for simulating wireless networks. Anyway, the work did not provide any performance comparison between the simulators. The same comment applies to work by Toor and Jain [17], where they presented a survey on open-source network simulators, e.g., ns-2, ns-3, J-Sim, OMNeT++, OPNET, QualNet, and MATLAB. Bakni and Moreno [18] proposed an evaluation approach to describe the network simulator's behavior, capacity, and performance. Bakni et al. [19] applied the proposed approach for the Cisco Packet Trace network simulator and extended their work by applying the evaluation approach to several additional Wireless Sensor Network (WSN) modeling tools.

Sundani et al. [20] provided a comparison on WSN simulators based on key features, limitations, scalability, Central Processing Unit (CPU), and memory usage on more than ten simulators. Weingartner et al. [21] proposed a methodology for performance evaluation, which provides the comparison between network simulators. However, the information in the work is partly outdated already. Sarkar and Halim [22] provided the comparison based on the deployment mode, type, supported protocols, and network impairments. A strong contribution of this work was the presented comparison and the provided recommendations.

Simulators specialized in computational offloading have gained popularity recently and started to develop rapidly. Kunde and Mann [23] worked on theoretical and practical comparison of fog computing simulators, e.g., iFogSim, MyFogSim, and YAFS, showing the impact on simulation run time and other parameters. Fakhfakh et al. [24] analyzed the most popular simulators for cloud computing and compared them based on the supported modules (energy, mobility, and so on). Qu et al. [25] presented a new simulation platform for edge computing with distributed learning and blockchain models and introduced a comprehensive literature review on the existing cloud and edge simulators, focusing on the lack of the federated learning concept implementation. Unfortunately, this simulator is not available in open source.

C. Set of Criteria and Methodology

There is no standardized method to evaluate the simulator, but the proposed set of criteria exists in the literature [18]. Following the authors' example, we present our criteria for the simulator's evaluation. At any rate, the purpose of the simulation might differ from work to work. That is why we leave it to the readers to decide which tool is the best for them, based on the preferred programming language, OS, and the research aims.

1) Open Access: This criterion shows the code available in open access, which means that it is online and free of charge.

2) Programming Language: This characteristic highlights the programming language used for writing scripts and modules.

3) User Interface: Simulating tools can have a GUI to provide a user-friendly environment.

4) Documentation Availability: This criterion represents the availability of the related documentation: manuals, tutorials, videos, presentations, setup instructions, and so on.

5) Ease of Setup: This characteristic shows the experience of the tool's initial configuration.

6) Ease of Use: This characteristic shows the experience of the tool's usage. It could vary from easy to hard.

7) Scalability: This characteristic shows how scalable the tool is. The number of nodes in IoT scenarios could be more than 100, so it is critical to answer whether the tool allows scaling easily on such a number. We assume that the simulator is scalable if it allows to connect up to 100 end devices.

8) Supported Features: This characteristic represents the tool's key features in the implemented modules, e.g., mobility, orchestration, or networking.

Sections II-D and II-E introduce the investigated simulators. Since simulators work in the different abstraction levels, see Fig. 3, the tools were divided into two groups and introduced in separate subsections. The first subsection emphasizes the tools for simulating virtual cloud environments. They show the service availability, energy consumption, allocation of tasks, etc. The first group includes the following tools: CloudSim, EdgeCloudSim, IoTSim-Edge, and CloudSim Plus. The second subsection introduces the second group of tools that work with the network architecture. It describes the system from the user to processing server. They show task response time, which combines server processing time and delivery time and provides the choice of data processing location. This group includes iFogSim2, PureEdgeSim, and YAFS. Table I provides a comparable analysis of mentioned tools.

D. Simulators Focused on Computing Infrastructure and Application Services

1) CloudSim: The simulator was developed by the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Melbourne, VIC, Australia, in 2009. The primary idea of this project was to develop a toolkit for modeling and simulating recently emerged cloud computing infrastructure and their services [30]. CloudSim is an open-source simulator assisted with documentation and installation guides.

The simulation environment contains the following entities: a host, i.e., simulated hardware, VM, and datacenter; cloudlets, i.e., tasks, services from the user, and broker. The entity broker is responsible for negotiating between the user (cloudlet) and the cloud provider (datacenter) and allocating the resources there. The output results show the status of the cloudlet proceeding, start time and end time, processing time, and id of the datacenter and the VM where the cloudlet was sent. Cloudlet is defined by the following properties: length, file size, and output size. VM, the real cloud-based VMs, is defined by Million Instructions per Second (MIPS), image size (Mb), Random Access Memory (RAM) (Mb), bandwidth, and several CPU.

CloudSim supports virtualization, i.e., the creation of multiple VM on the physical server (host). Furthermore, it supports VM migration, which means that it allows simulation of the movements from one physical host to another. In real practice, this feature will enable us to adaptively allocate the workload on the servers for better system performance or to maintain the failed server without user interruption. Among the key features, CloudSim supports cloud datacenter network topology, which includes the wireless and physical interconnection of datacenter components, such as storage, computing entities, servers, and switches.

2) CloudSim Plus: CloudSim Plus is a new Java-based framework for modeling a cloud computing environment first released in 2016. Based on CloudSim but working as an independent project, CloudSim Plus improved its performance and simplified its usage by reducing the amount of duplicated code and restructuring the modules and packages [31]. The tool has a related project, CloudSim Plus, which is an open-source simulator with lots of available documentation on its webpage, white paper, and discussions on the Google forum. The online documentation is one of the most detailed explanations of the implemented features.

The CloudSim Plus project is similar to CloudSim but has improved structure, reducing the complexity of the scripts. The simulator consists of the physical layer, which includes servers, the logical layer, i.e., the datacenter network topology, and the virtualization layer, used for creating VM. The modules support the VM migration and vertical/horizontal VM scaling. It uses the same entities—hosts, datacenter, cloudlets, and broker, responsible for communication between the user and the datacenter.

3) EdgeCloudSim: It is an open-source tool for edge-specific modeling based on CloudSim and was developed in 2017. The main uniqueness of this project is that it considers both computational and networking resources compared to its predecessor CloudSim. The project's scripts can be run on Linux-based systems, including Mac OS via the preferable IDE for compilation [32].

The tool inherited CloudSim module for VM allocation in the datacenter and other computing features. Nevertheless, EdgeCloudSim includes unique features for edge computing architecture described further. The edge orchestrator module is responsible for making critical decisions on allocating or terminating the VM on the available resources and offloading tasks to the cloud or edge server to increase the overall system performance. The networking module is responsible for LAN or Wireless Local Area Network (WLAN) communication delay for both directions Uplink (UL) and Downlink (DL). EdgeCloudSim supports mobility, so the devices' locations and, consequently, the delay are updated according to this module.

Running the simulation requires defining the application, edge device, and configuration parameters. To avoid the overloaded script, the mentioned parameters are stored in three separate files named accordingly. The application file contains information about application data size, battery level, usage distribution on the devices, active and idle duration, and so on. The configuration file sets the cloud and simulation parameters, such as the number of mobile devices, orchestration policy, and architecture. It is possible to set the computing capabilities to mobile devices or use them as sensors without a processing unit. The edge device file includes edge datacenter parameters. It defines the number and location of the edge server, computing and storage capabilities, and the number of VMs deployed on it. The number of edge servers is scaled—the tool allows linking more than ten servers and deploying multiple VMs on each of them. Still, the specific of such structure needs to define each server separately and, thus, not user-friendly.

EdgeCloudSim benefits in investigating the Quality of Experience (QoE) for the edge-based computing scenarios.

TABLE I
Main Simulators Characteristics

	iFogSim2	*PureEdgeSim*	*CloudSim*	*CloudSim Plus*	*IoTSim-Edge*	*EdgeCloudSim*	*YAFS*
Open access	✓	✓	✓	✓	✓	✓	✓
Prog. Lang.	Java	Java	Java	Java	Java	Java	Python
GUI	✓			✓	✓		
Documentation availability	Tutorials are in the open access	Available documentation and papers	Official website data	Community support and documentation	Poor documentation	Tutorials and community support	Up-to-date documentation and guidelines
Ease of setup	*Easy*: Follows official guidelines	*Easy*: Maven build	*Easy*: Follows official guidelines	*Moderate*: Poor guidelines, Maven build and compile	*Difficult*: Manual dependencies, no error handling	*Easy*: Shell script setup	*Moderate*: Manual packages setup
Ease of use	*Moderate*: Each parameter needs to be defined, no description	*Easy*: Need for specific configurations; no comments	*Moderate*: Script contains VM and Cloudlet parameters, documentation	*Easy*: Straightforward in presence of dependencies & files	*Moderate*: Overloaded configuration file; possible to add new classes	*Easy*: Separated configuration files	*Easy*: Intelligible script and straightforward structure
Supported features:							
Scale	medium	large	large	large	large	large	large
Mobility	✓	✓			✓	✓	✓
Orchestration	✓	✓				✓	✓
Networking	✓	✓	✓	✓	✓	✓	✓
Virtualization			✓	✓		✓	
Containerization			✓			✓	
Energy	✓	✓	✓	✓	✓		
Microservices					✓		✓
Tracking events						✓	✓

OS – Operating System RAM – Random Access Memory SSD – Solid State Drive UL, DL – Uplink, Downlink N/A – not available

The supported modules simulate scalable scenarios and provide information on package delivery in the dedicated server.

4) IoTSim-Edge: This tool was developed in 2019 and is based on CloudSim project. It is focused on IoT-driven offloading scenarios such as planning the capacity of Road-Side Unit (RSU) for the intelligent transportation system or sensor deployment in the smart building scenario [8].

The archive file is available in GitHub and complemented with the user manual. Unfortunately, the file was not updated for a long time, and no recent file is available. The project built under Maven contains the pom.xml, but the old versions and missing dependencies could fail the Maven build process.

The IotSim-Edge simulator consists of the following entities Edgelet, i.e., a generated task from the IoT sensor; MicroElements (MEL), i.e., an abstract component of the application that represents the services in the form of microservice; edge devices, i.e., a laptop, smartphone, or other devices that host MEL; edge datacenter, i.e., the core edge infrastructure; and EdgeBroker, which allocate users' requests with accordance to their requirements. After setting the required parameters (MIPS, RAM, battery capacity, location, and so on), the simulator allocates MEL for edgelets and outputs their execution time. The simulator supports such features as energy consumption, mobility, and networking.

E. Simulators Focused on Network Architecture and Resource Allocation

1) YAFS: It is a SimPy-based highly configurable simulator designed on a complex network theory for analysis of fog-driven computing scenarios developed in 2019. It allows the creation of a scalable, dynamic network and simulates the request in it [33].

The installation guide and project documentation are available in the open source on the official webpage. Installation is roughly simple. If it fails to find the matching "YAFS," the distribution must upload manually to the Python home file. Up-to-date documentation, user guides, referred papers, and solutions to solve Python errors are available online.

Execution requires defining the network and application. Network topology is modeled as a graph that contains computing capacity and bandwidth information for each node and data rate and propagation information for each link. The communication time, i.e., latency, is calculated as the ratio between message size and set bandwidth plus the propagation speed. The service time is the time that is needed to process the packet.

The network architecture is presented as an orgraph with the forest of trees topology. A node represents a cloud server connected to the proxy server linked to a set of gateways. Each gateway could be linked to the defined set of mobile devices, including sensors and actuators. There is no separate entity for them; sensors and actuators are mobile devices with no computing capacity that generate or consume data and connect to the main mobile device. The application is set as a group of modules that can generate or process a packet. Modules could be deployed on the cloud server or in the group of edge servers, depending on the orchestration policy.

One of the advantages of YAFS is a highly configurable and flexible architecture that allows running a scalable network in terms of the number of nodes and modules. It supports virtualization, microservices, and other features easily defined as logical relationships by customized configurations.

2) iFogSim2: It is a tool to simulate the fog computing environment developed upon the CloudSim and measures the impact of resource management techniques in network congestion, latency, cost, and energy consumption [34]. It was updated in 2021 and inherited features from the old version (iFogSim). The rising number of IoT large-scale scenarios

TABLE II
QUALITATIVE APPLICATION REQUIREMENTS COMPARISON

Application parameters	XR*	Monitoring	Streaming	Ref.	Qualitative legend ranging			
					◔	◑	◕	●
Bitrate	●	◔	◑	[26], [27], [28]	$< 1Mbps$	$1 - 100Mbps$	$0.1 - 1Gbps$	$> 1Gbps$
Payload	●	◑	◑	[26], [28]	$< 0.4kB$	$0.4 - 1kB$	$1 - 1.5kB$	$> 1.5kB$
Latency	◔	◑	◕	[26], [27], [28], [29]	$< 50ms$	$50 - 100ms$	$0.1 - 0.15s$	$> 0.15s$

* XR is an umbrella term for all real-and-virtual combined environments and human-machine interactions, including AR, VR, and MR [28].

turns iFogSim2 into a high-potential tool for simulating the fog-driven use cases and deploying them with minimum costs. It is an open-source simulator available from GitHub. There are guidelines available on the GitHub page. Also, the CloudSim tutorial page contains several guidances of iFogSim2 Project Structure for beginners as it is one of CloudSim's related projects.

The simulation entities consist of FogDevice, representing the actual fog computing resource; fog broker, responsible for the task distribution; and sensor class, i.e., cameras and temperature sensors. The output file shows the execution time of the proposed topology and energy consumption on each node (device and server). The topology allows the creation of nodes in different layers.

The iFogSim2 shortcomings are partly solved in the extension MobFogSim, whose primary purpose is to simulate the mobility in the IoT gateways and cloud datacenters. In turn, MobFogSim limits the scope of creating clusters in edge/fog computing environments and lacks documentation.

3) PureEdgeSim: It is an open-source simulator for studying dynamic and highly heterogeneous networks based on CloudSim Plus. It was developed in 2018 for deploying edge, fog, and cloud scenarios. It allows the connection of thousand of devices and supports their mobility with the location manager.

Many research papers are available in open source as well as the official GitHub pages containing the setup instructions and step-by-step video with examples. The main simulation file takes the parameters of cloud and edge servers and mobile devices stored in the .hml files. Simulation configuration consists of simulation time, device count, orchestration algorithms, and other parameters that could be changed according to the scenario. It allows for collecting the energy consumption, task execution time, and task success rate.

III. SIMULATORS PERFORMANCE COMPARISON

The performance evaluation is a numerical characteristic for comparison of the scripts. The script performance highly depends on the programming language, used libraries, project architecture, and other parameters. Even though all projects are evaluated together in this work, it is important to mention that the evaluated tools differ in the initial aim and their structure, and it is better to take in mind the difference between the following two groups that were already introduced earlier—tools that are major for virtualization or task allocation in the servers/datacenter and those that are focused on packet's arriving time and describe the network architecture. The first group includes CloudSim, IoTSim-Edge, and EdgeCloudSim, while the second group includes PureEdgeSim and YAFS. CloudSim Plus and iFogSim2 was not included in the performance evaluation due to compilation problem.

A. Test Scenarios

As one of the motivations, emerging resource-hungry applications should meet rigorous communication requirements set by the standardization bodies. In contrast to traditional light use cases, e.g., remote sensing or monitoring, 3GPP TR 26.928 "Extended Reality (XR) in 5G" considers the media delivery bitrate >1 Gb/s to provide a sufficient media quality and low latency, mentioning the need for potential standardization [28], see Table II. The end-to-end latency for the XR environment, referred to as immersive motion-to-photon, including rendering and decoding, states around 20 ms or less [29] for a smooth user experience. Nonetheless, the online video stream, 3GPP TR 26.925 "Typical traffic characteristics of media services on 3GPP networks," refers to the 150-ms max packet delay budget that the user will barely notice. Today, the recommended bitrate for Full HD video lies between 3 and 12 Mb/s, while for the 4K UHD, 5–25 Mb/s [27], which is smaller than the XR.

Following the standardized recommendations, we further focus on the three scenarios taken as examples of intensive workload and delay-tolerant scenarios to make the test scenarios closer to real-life implementation. They also include the standards of different modalities, static or moving, based if the user is moving or not while delivering/accepting the service.

XR scenario corresponding to remote Augmented Reality (AR)-assisted telesurgery is a used example of extremely high data rate with intensive workload and static modality, detailed in Table III, based on the standardization summary [35]. AR applications process the video data generated by the laparoscope, or 3-D ultrasound probe, equipped with a small camera, and process it on a server (private server, edge, or cloud), according to 3GPP. The laparoscope and robotic medical instruments (trocars, graspers, and scissors) are inserted through tiny incisions in the patient's body. Since all organs function during the operation, the video is transmitted to the console monitor with ultrasmall delays to prevent healthy tissues' perforation. The telesurgery scenario assumes that the patient and the operating doctor are physically located on different continents operating remotely. In that case, the IEEE standard defines the performance requirements

TABLE III
QoS Metrics Composed With [26], [36], and [37]

Use case	XR Scenario	Monitoring	Streaming
Latency	< 10 ms	< 100 ms	< 150 ms
Availability	> 99.9999 %	> 99.9999 %	> 99.99 %
Packet size	1500 B	< 1000 B	500 B
Bit rate	4 Gbps	0.5 Mbps	9 Mbps
User speed	Stationary	< 500 km/h	
Battery	Unlimited	Energy-constrained	
Scale	1 device	> 1000 devices	500 devices

TABLE IV
Simulation Parameters for Scenarios in Table II

	Parameter	XR Scenario	Monitoring Scenario	Streaming Scenario
General	Duration	4 hours	1 month	2 hours
	Task rate	10 fps	1000 sampl.ps	60 fps
	Bit rate	4 Gbps	500 kbps	9 Mbps
	Application	33000 MIPS	4 MIPS	5000 MIPS
Group 1	Host:			
	Amount	10 machines	10 machines	10 machines
	RAM	16, 000 MB	16, 000 MB	16, 000 MB
	Storage	100, 000 MB	100, 000 MB	100, 000 MB
	CPU	4 cores	4 cores	4 cores
	Processor's speed	1000 MIPS	1000 MIPS	1000 MIPS
	Bandwidth	10, 000 Mbps	10, 000 Mbps	10, 000 Mbps
	VM:			
	RAM	512 MB	512 MB	512 MB
	Storage	10, 000 MB	10, 000 MB	10, 000 MB
	CPU	1 core	1 core	1 core
	Bandwidth	1000 Mbps	1000 Mbps	1000 Mbps
	Processor's speed	1000 MIPS	1000 MIPS	1000 MIPS
	Cloudlet:			
	Length	2000 MIPS	10 MIPS	100 MIPS
	UL data size	1500 kB	1 kB	500 kB
	DL data size	500 kB	1 kB	500 kB
Group 2	Cloud server:			
	RAM	16, 000 MB	16, 000 MB	16, 000 MB
	Processor's speed	100, 000 MIPS	100, 000 MIPS	100, 000 MIPS
	Bandwidth	12 Gbps	12 Gbps	12 Gbps
	Proxy server:			
	RAM	20 MB	20 MB	20 MB
	Processor's speed	10, 000 MIPS	10, 000 MIPS	10, 000 MIPS
	Bandwidth	100 Mbps	5 Mbps	10 Mbps
	Edge server:			
	RAM	20 MB	20 MB	20 MB
	Processor's speed	10, 000 MIPS	10, 000 MIPS	10, 000 MIPS
	Bandwidth	100 Mbps	5 Mbps	10 Mbps
	Mobile Device:			
	RAM	0.02 MB	0.02 MB	0.02 MB
	Processor's speed	100 MIPS	100 MIPS	100 MIPS
	Bandwidth	100 Mbps	5 Mbps	10 Mbps
	Devices number	1	1000	100

for multicast video traffic for medical applications via Public Land Mobile Network (PLMN) [36]. The 3-D ultrasound probe augments the main anatomical image with the 3-D volume data, producing a data stream above 1 Gb/s. The AR image from a 3-D ultrasound probe requires a precise robotic instrument's location in the patient's body. The images are exchanged in a total of 240 images/s over cellular communication.

The monitoring scenario numerically corresponds to cardiac telemetry, i.e., a low data rate with moderate delays and workload requirements, and moving modality, detailed in Table III. A wireless wearable telemetry device includes body sensors, e.g., ECG, respiratory rate, and SpO$_2$, which provides 24/7 monitoring of the patient's health. Due to the on-body way of wearing, the device must be small and energy-efficient. Cardiac telemetry devices require to keep devices alive for at least a month without recharging. From the performance side, it requires a highly reliable, always-on connection with the hospital to process the patient analytics and raise the alarm in an emergency. The number of devices varies on the patient location: up to 1000 wearables/km^2 in the hospital area or about ten devices per km^2 in suburban areas [26].

The streaming scenario is an example of general human traffic with a relatively fast bit rate but moderate payload, as detailed in Table III. The application aims to provide access to the user's favorite online show in Full HD to watch from a mobile device. User location could be static or moving with the user, e.g., if the user is sitting in the train, the speed could reach up to 500 km/h. The packet size is 500 B, and the end-to-end delay requirement is set as 150 ms [37]. The number of devices varies up to 500 active devices in city areas.

To sum up, all three scenarios differ regarding payload and required bit rate. The intense XR scenario, i.e., AR-assisted surgery, requires transmissions of the uncompressed captured images 256 × 256 × 256 voxels 24 bits 10 frames/s that is 4 Gb/s. Monitoring scenario, i.e., cardiac monitoring, measures data with a frequency of up to 1000 samples/s, thus requiring up to 500 kbps bit rate. Streaming scenario, i.e., online video streaming, transmits compressed images from/to the user device. Full HD video resolution 1920 × 1080 24 bits 60 frames/s and H.264 gives up to 9 Mb/s. The recommended performance metrics of the use cases are presented in Table III.

B. Overall System Model

The system contains 80 sensors/actuators implemented in the IoT device for the AR telesurgery scenario and 1000 sensors as wearables devices for cardiac monitoring. The devices are wirelessly connected to the gateways, which can offload data to the edge or cloud datacenter over a 5G network. The datacenter allocates resources for VM or MEL to process cloudlet or edgelet, which refers to the application workload.

Assume that the channel bandwidth for mobile devices over cellular network is 1 MHz where the system bandwidth of 20 MHz and 20 users are simultaneously connected [38]. The transformation of channel bandwidth (Hz) to bit rate (bps) depends on many factors, including the chosen technology, coding rate, and modulation. Let us say that the average 20 MHz is approximately 100 Mb/s; consequently, 1 MHz is 5 Mb/s. The proxy server is connected to the datacenter via optical fiber, which can reach 1 Gb/s.

The processor speed, usually measured in MIPS, represents the number of million instructions (MIs) required in one cycle, where by "instruction" means a specific hardware operation to process the task. Specification of the widely used ARM platform in the embedded IoT states that ARM Cortex-M4 has 100-MHz processor frequency and 128-kB RAM; ARM Cortex-M3 has 72 MHz and 20 kB RAM [39]. Cortex-M3 and M4 cores achieve 1.25 MIPS/MHz [40], which is approximately 125 MIPS at 100 MHz [41]. Compared to the datacenter capacity, the server processor, e.g., Intel Xeon [42], has 38.5-MB cache and 2.50-GHz processor frequency with up to 28 cores built to process up to 100 000 MIPS [43].

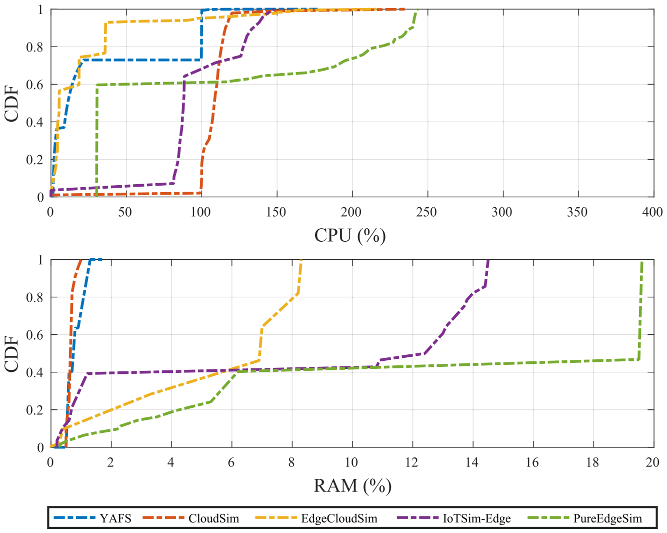


Fig. 4. CPU and RAM performance for XR scenario.

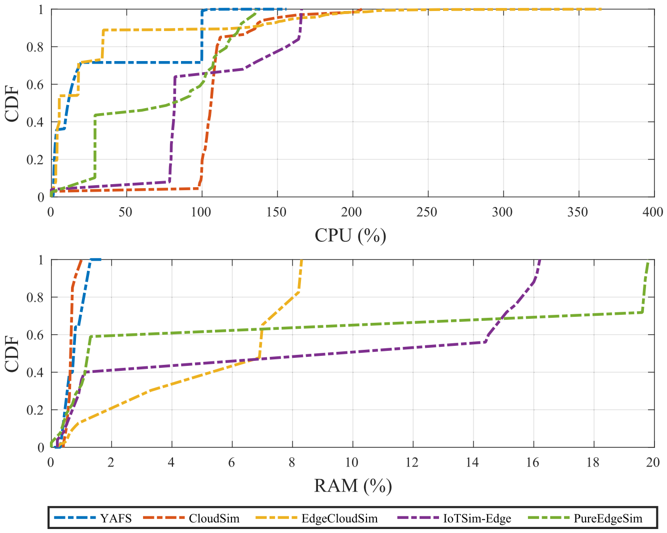


Fig. 5. CPU and RAM performance for monitoring scenario.

The reference simulation for Group 1 models the allocation of the received tasks (cloudlets) and VMs in the datacenter that was generated with the required size and task frequency. Other simulation parameters are presented in Table IV [44].

All simulations were conducted on Ubuntu OS deployed via the VirtualBox with dedicated four CPUs and 12 GB of memory. Simulation scripts were launched in the terminal via a bash script. The bash script runs the process after a slight delay (5 s), fixating the start and end time of the running and measuring the CPU utilization and memory every 1 s via -ts Linux command. The exception was made for IoTSim-Edge—due to the Maven compilation issues launched via the cmd line, the script was run in Eclipse. The results were collected the same way—measuring the CPU utilization and memory every 1 s via -ts Linux command by process name. CloudSim Plus and iFogSim2 did not participate in the performance evaluation due to compilation failure.

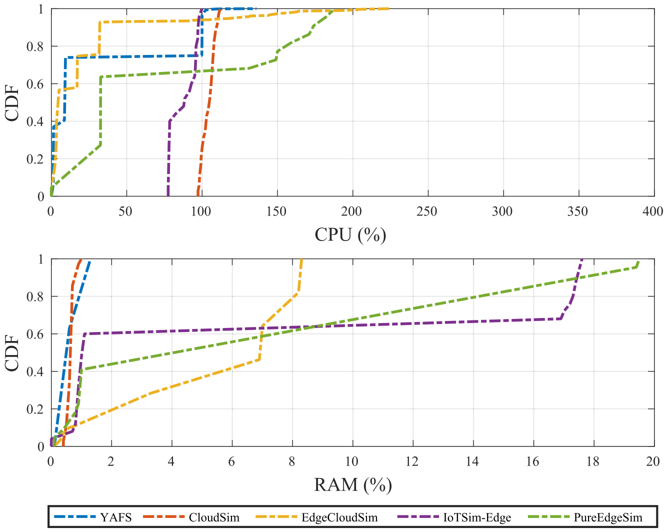


Fig. 6. CPU and RAM performance for streaming scenario.

C. Evaluation Results

The main aim of this section was to understand how much hardware resources are utilized while executing the script, but not explain the behavior of their performance, as those are implementation-specific and could not be affected. Assuming that simulators received the same scenario according to their capabilities, we compare each of them regarding CPU, RAM usage, and the simulation execution time.

1) Maximum CPU and RAM Values: Table V presents the maximum CPU and RAM values of each simulator. The built-in Python simulator YAFS showed the best (minimum) usage in CPU and memory for XR scenario (see Fig. 4). It is explained by the nature of languages and their different methods [45]. However, it showed the worst results for monitoring and streaming scenarios in Figs. 5 and 6. The last two scenarios had relatively smaller workloads but applied devices on a bigger scale. The simulator's peculiarity is that it creates the link for each sensor/device. Thus, a growing number of devices will add to their network graph complexity and reflect on the hardware usage. The same happened with the EdgeCloudSim; even though it works in the virtual abstraction, it still requires specifying the links for the sensor nodes (see Fig. 3), thus the growth of the hardware load. CloudSim showed the opposite, and its CPU and RAM usage stayed the biggest during XR scenario, as the biggest simulated workload was applied.

IoTSim-Edge showed the best (minimum) CPU usage and worst (maximum) RAM usage from Group 1 for all scenarios. Figs. 4–6 show that the IoTSim-Edge slope rises sharply during processor and softly during memory load's peaks. The longest raise corresponds to the highest CPU or RAM. CloudSim was the best in terms of RAM usage from Group 1. PureEdgeSim showed the worst RAM usage from Group 2 for all scenarios and the worst CPU usage for XR and streaming scenarios. YAFS was the best in terms of RAM usage from Group 2.

TABLE V
HW Resource Usage Results

Sc.	Gr.	Simulator	max. CPU (%)	$\sigma^{CPU}(\%)$	CPU load (%)	max. RAM (Mb)	$\sigma^{RAM}(\%)$	RAM load (%)	Time (s)
XR	Gr. 1	*CloudSim*	236	18.4258	43.2	120	0.1108	3.4056	33.7
		EdgeCloudSim	216	33.8019	16.2	996	2.6062	56.7	30.049
		IoTSim-Edge	148	29.2764	10.9	1739	6.3260	11.3	21
	Gr. 2	*YAFS*	177	41.3611	1636.0	204	0.3114	559.5	0.747
		PureEdgeSim	243	91.1760	26.0	2351	7.7460	41.3	39
Monitoring	Gr. 1	*CloudSim*	206	28.1260	19.9	120	0.1143	2.3	35.489
		EdgeCloudSim	365	51.9856	41.2	996	2.6624	151.4	90.46
		IoTSim-Edge	166	43.2875	7.0	1943	7.2618	11.1	18
	Gr. 2	*YAFS*	156	42.0586	10704.0	204	0.3091	526.4	2914.81
		PureEdgeSim	136	43.3098	7.7	2375	9.3983	15.9	18
Streaming	Gr. 1	*CloudSim*	112	4.3031	15.9	120	0.1165	1.2	39.575
		EdgeCloudSim	224	37.1019	28.9	996	2.5953	95.3	54.227
		IoTSim-Edge	100	8.8019	9.4	2111	8.2061	8.7	21
	Gr. 2	*YAFS*	136	41.9370	216.5	156	0.3385	71.1	4026.46
		PureEdgeSim	188	66.6102	7.6	2339	9.3886	12.3	16

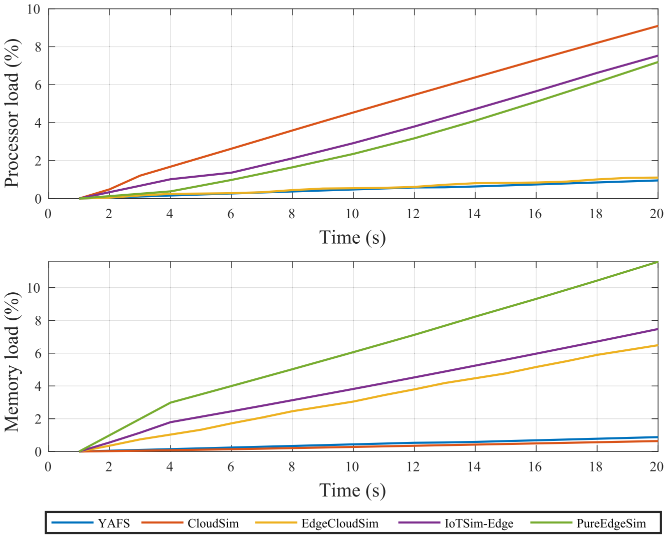


Fig. 7. CPU and RAM cumm. load for XR scenario.

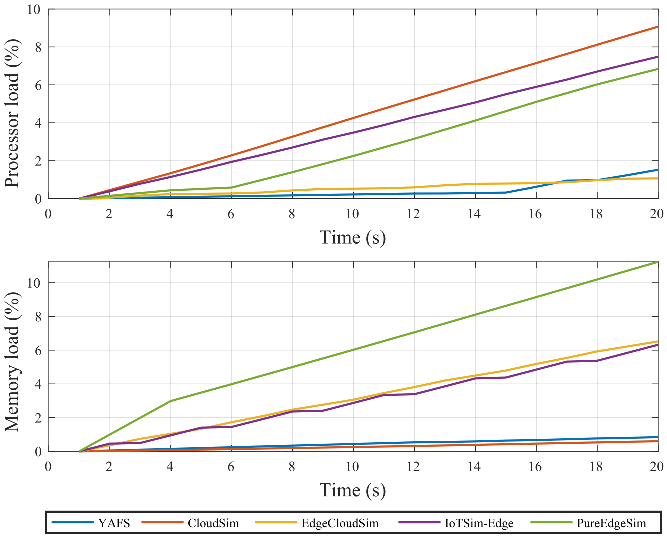


Fig. 9. CPU and RAM cumm. load for streaming scenario.

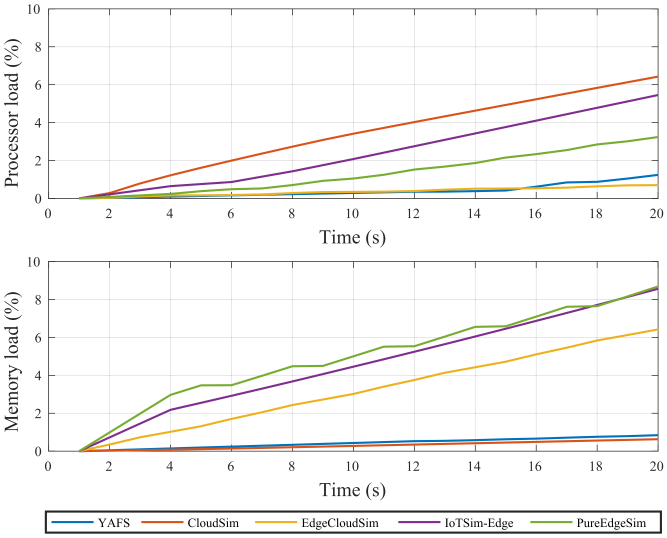


Fig. 8. CPU and RAM cumm. load for monitoring scenario.

2) Load Over Time: Absolute values are sometimes not enough to estimate the processor and memory load. Dedicating many resources could increase the processing speed but overload the processor in the case of extended use. We integrated data over time to evaluate the tool's performance, where the higher values represent the most load.

Fig. 7 presents the cumulative normalized processor and memory load for XR scenario, Fig. 8 presents the cumulative normalized processor and memory load for monitoring scenario, and Fig. 9 presents the cumulative normalized processor and memory load for streaming scenario. YAFS and CloudSim showed that the RAM curves have a slow elevation over time, but the CloudSim CPU curve increased rapidly. EdgeCloudSim showed a slow peak of the CPU curve over time but a sharp curve of RAM usage. Summing up, the results conclude that the different ways of code implementation influence the CPU and RAM usage on the hardware.

3) Simulation Run Time: The results about tool's run-time behavior could be derived from Table V. IoTSim-Edge made the fastest run from Group 1 in all scenarios. Python-based YAFS from Group 2 worked very fast during the XR scenario and was overloaded with the increased graph complexity during monitoring and streaming scenarios, where the best result was showed by PureEdgeSim.

TABLE VI
SUMMARY OF SIMULATORS ADVANTAGES AND DISADVANTAGES

	Use cases	**Advantages**	**Limitations**
iFogSim2	• Resource management • Health Monitoring • Crowd-sensed Data Collection • Audio Translation Service	• Simulates a complex topology from sensor/actuator through devices to servers; • Allows to specify the latencies between entities; • Shows the amount of consumed energy by each node.	• No choice of wireless connectivity (manual configuration); • No edge communication protocols in the simulation tool; • Minimum allowed sensor-generating data is 2 ms; • Set communication latency as constant value; • Allows to connect up to 15 sensors/actuators to the edge.
PureEdgeSim	• Reinforcement and Deep Learning Scenarios on Edge computing-driven scenarios • SDN • Task Scheduling and Security	• Allows to specify the device's battery parameters; • Allows to specify the communication technology for mobile device (e.g., cellular, WiFi, Ethernet, etc.); • Supports the Mist Computing by setting the computing capabilities to the end device; • Allows for ranging of nodes for coverage variation; • Orchestration architecture and algorithms support.	• Difficult to customize the mobility module due to complexity; • Limits in forming node clusters; • Limits in augmenting microservice management techniques; • The execution time, waiting time for each packets, as well as energy consumption for device, edge server and cloud server calculates as the average value; • Uniform user distribution only for square area.
CloudSim	• Large-scale Cloud Computing datacenter scenarios • VM performance in the Cloud Data Centers • Allocation of resources on VMs • Simulation of Federated Clouds	• Supports virtualization and VM migration; • Supports modeling for containerized cloud environments, referring to a new type of service CaaS; • Supports the interconnection between Data Center components; • Trustable and validated tool with community support.	• Does not suit for PaaS or SaaS real-time applications analyzes; • Does not suit for security algorithms or platform implementation.
CloudSim+	• VM lifecycle management • Allocation and scheduling of VM • Simulation of Federated Clouds	• Supports user-defined policies for resource allocation; • Supports virtualization and VM migration; • Has an implemented event listener to provide event notification for simulations.	• Does not suit for security algorithms or platform implementation.
IoTSim-Edge	• Healthcare scenarios • Smart Buildings • Intelligent RSU Deployment • IoT-driven scenarios	• Works with the heterogeneous environment; • Supports diverse communication technologies and IoT data protocols; • Allows to check the mobility, add velocity and location coordinates of the device; • Allows to specify the battery capacity for the IoT devices.	• Omits state-of-the-art communication protocols (i.e., BLE, 5G, 6G, etc.); • Does not send processed data back to the actuators, suits only for scenarios that need only the raw data collection from sensors; • The documentation does not specify the units for parameters.
EdgeCloudSim	• Healthcare scenarios • AR scenarios • VM allocation	• Has an implemented network and battery modules; • Allows to run parallel simulations and obtain data from them; • Uses active/idle task generation pattern to act as a real device.	• Does not support customized mobility, cluster formation, and microservers; • Assumes the link quality between the device and the gateway nodes remains the same despite their distance; • Network model is represented as constant delays.
YAFS	• Resource allocation in Fog computing scenarios • Crowd-sensed Data Collection • VR games scenarios	• Dynamic topology and dynamic allocation support; • Allows to generate the events using customized distributions; • Provides fully customized links and nodes attributes; • Intelligible and fully user customized configuration; • Selective data transmission between the modules.	• Does not support communication technology diversity; • The documentation lacks the parameters definitions and their units; • Network and propagation models are defined with constants.

IV. CONCLUSION AND DISCUSSION

Developing a good simulation tool for modeling networks is considered as a valuable contribution to academic work. On one hand, most of the existing reliable network simulators do not consider cloud entities, such as datacenter, host, VM, or broker. On the other hand, computational simulators do not count network delays and mobility highlighting the need for Edge- and Fog-specific simulators forced by emerging computing paradigms.

CloudSim was one of the first open-source simulators, which implemented avant-garde technologies. It has been a validated and trusted tool for many years. Fortunately for humanity and unfortunately for the developers, technologies change fast, but this tool lacks upcoming modules. Notwithstanding the mentioned limitations, CloudSim inspired other independent projects that used it as a base project for more improved simulators, i.e., CloudSim Plus. In turn, CloudSim Plus inherited many entities and features CloudSim, but the reengineered project improved the performance and deployed new modern features, i.e., event listener, VM migration, and parallel computing. Developers advanced the scripts made them more human readable to improve the usability.

EdgeCloudSim is an easy-to-set-up and easy-in-use tool, which simulates the task execution in various edge scenarios (edge cloud and mobile edge) and supports orchestration and networking models. A significant disadvantage of this tool is lack of energy model and task migration. IoTSim-Edge suits primarily if the research evaluates IoT devices and their interaction with edge devices. Despite its more user-friendly appearance, it lacks essential features such as virtualization and orchestration.

PureEdgeSim and iFogSim2 are focused on offloading to the edge. iFogSim2 allows to simulate complex IoT scenarios considering the delays between sensors and IoT device. It allows to connect the servers into N-tier hierarchy and allows to specifies the delays between them. The tool allows to model latency-sensitive applications and, thus, suits to the AR-assisted surgery simulations. In contrast, PureEdgeSim comprises a range of orchestration architectures, including mist computing, i.e., offloading on mobile devices, mist edge, mist cloud, and edge cloud, which is applicable to telemetry scenarios in the medical domain. The implemented energy model shows the energy consumption and task failures due to battery run out.

YAFS is a perspective project, which provides full support for the user in the open source. It plans to implement geolocalization and other features to the existing ones and solve the Python compatibility issues; hence, some limitations might be irrelevant in the near future. Project YAFS works on implantation modern features for fog computing architecture, and it allows to model VR scenarios; thus, it is suitable for VR-assisted remote medical scenarios as well.

Summarizing the above, there are many tools available in the open source designed for modeling cloud, fog, and edge computing scenarios. Each of them has its limitations and advantages that could suit the research specific aim. Table VI summarizes the simulation tools investigated in this work. Choosing the best simulator for the research questions could take some time, so this article aims to help in minimizing this time overviewing the most popular tools for modeling the offloading scenarios for medical applications.

References

[1] Y. Cheng, H. Zhao, and W. Xia, "Energy-aware offloading and power optimization in full-duplex mobile Edge computing-enabled cellular IoT networks," *IEEE Sensors J.*, vol. 22, no. 24, pp. 24607–24618, Dec. 2022.

[2] R. Yadav et al., "Smart healthcare: RL-based task offloading scheme for Edge-enable sensor networks," *IEEE Sensors J.*, vol. 21, no. 22, pp. 24910–24918, Nov. 2021.

[3] J. Liu, S. Guo, Q. Wang, C. Pan, and L. Yang, "Optimal multi-user offloading with resources allocation in mobile Edge Cloud computing," *Comput. Netw.*, vol. 221, Feb. 2023, Art. no. 109522.

[4] *Finland Data Centers*. Accessed: Dec. 22, 2022. [Online]. Available: https://www.datacentermap.com/finland/

[5] C. Yi, J. Cai, and Z. Su, "A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications," *IEEE Trans. Mobile Comput.*, vol. 19, no. 1, pp. 29–43, Jan. 2020.

[6] N. Mäkitalo et al., "Action-oriented programming model: Collective executions and interactions in the Fog," *J. Syst. Softw.*, vol. 157, Nov. 2019, Art. no. 110391.

[7] D. Alekseeva, A. Ometov, O. Arponen, and E. S. Lohan, "The future of computing paradigms for medical and emergency applications," *Comput. Sci. Rev.*, vol. 45, Aug. 2022, Art. no. 100494.

[8] D. N. Jha et al., "IoTSim-Edge: A simulation framework for modeling the behaviour of IoT and Edge computing environments," 2019, *arXiv:1910.03026*.

[9] P. Mel et al., "The NIST definition of Cloud computing," Comput. Secur. Division, Inf. Technol. Lab., Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NIST Special Publication 800-145, 2011. [Online]. Available: http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf

[10] Y. Lin, L. Shao, Z. Zhu, Q. Wang, and R. K. Sabhikhi, "Wireless network Cloud: Architecture and system requirements," *IBM J. Res. Develop.*, vol. 54, no. 1, pp. 4:1–4:12, Jan. 2010.

[11] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed., MCC Workshop Mobile Cloud Comput.*, Aug. 2012, pp. 13–16.

[12] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in Industrial Internet of Things: Architecture, advances and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2462–2488, 4th Quart., 2020.

[13] V. Prokhorenko and M. A. Babar, "Architectural resilience in Cloud, Fog and Edge systems: A survey," *IEEE Access*, vol. 8, pp. 28078–28095, 2020.

[14] I. A. Aljabry and G. A. Al-Suhail, "A survey on network simulators for vehicular ad-hoc networks (VANETS)," *Int. J. Comput. Appl.*, vol. 174, no. 11, pp. 1–9, Jan. 2021.

[15] S. Kang, M. Aldwairi, and K.-I. Kim, "A survey on network simulators in three-dimensional wireless ad hoc and sensor networks," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 9, Sep. 2016, Art. no. 155014771666474.

[16] R. L. Patel, M. J. Pathak, and A. J. Nayak, "Survey on network simulators," *Int. J. Comput. Appl.*, vol. 182, no. 21, pp. 23–30, Oct. 2018.

[17] A. S. Toor and A. K. Jain, "A survey on wireless network simulators," *Bull. Electr. Eng. Informat.*, vol. 6, no. 1, pp. 62–69, Mar. 2017.

[18] M. Bakni and M. Moreno, "An approach to evaluate network simulators: An experience with packet tracer," *Revista Venezolana de Computación*, vol. 5, pp. 29–36, Jan. 2018.

[19] M. Bakni, L. M. M. Chacón, Y. Cardinale, G. Terrasson, and O. Curea, "WSN simulators evaluation: An approach focusing on energy awareness," 2020, *arXiv:2002.06246*.

[20] H. Sundani, H. Li, V. Devabhaktuni, M. Alam, and P. Bhattacharya, "Wireless sensor network simulators a survey and comparisons," *Int. J. Comput. Netw.*, vol. 2, no. 5, pp. 249–265, 2011.

[21] E. Weingartner, H. vom Lehn, and K. Wehrle, "A performance comparison of recent network simulators," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2009, pp. 1–5.

[22] N. I. Sarkar and S. A. Halim, "A review of simulation of telecommunication networks: Simulators, classification, comparison, methodologies, and recommendations," *J. Sel. Areas Telecom. (JSAT)*, vol. 2, no. 3, pp. 10–17, 2011.

[23] C. Kunde and Z. Á. Mann, "Comparison of simulators for Fog computing," in *Proc. 35th Annu. ACM Symp. Appl. Comput.*, Mar. 2020, pp. 1792–1795.

[24] F. Fakhfakh, H. H. Kacem, and A. H. Kacem, "Simulation tools for Cloud computing: A survey and comparative study," in *Proc. IEEE/ACIS 16th Int. Conf. Comput. Inf. Sci. (ICIS)*, May 2017, pp. 221–226.

[25] G. Qu, N. Cui, H. Wu, R. Li, and Y. Ding, "ChainFL: A simulation platform for joint Federated Learning and blockchain in Edge/Cloud computing environments," *IEEE Trans. Ind. Informat.*, vol. 18, no. 5, pp. 3572–3581, May 2022.

[26] *Study on Communication Services for Critical Medical Applications*, Standard 3GPP TR 22.826 V17.2.0, Rel. 17, Mar. 2021.

[27] *Typical Traffic Characteristics of Media Services on 3GPP Networks*, Standard 3GPP TR 26.925 V17.1.0, Rel. 17, Mar. 2022.

[28] *Extended Reality (XR) in 5G*, Standard 3GPP TR 26.928 V18.0.0, Rel. 18, Mar. 2023.

[29] *Virtual Reality (VR) Media Services over 3GPP*, Standard 3GPP TR 26.918 V17.0.0, Rel. 17, Apr. 2022.

[30] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of Cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exp.*, vol. 41, no. 1, pp. 23–50, Aug. 2011.

[31] M. C. S. Filho, R. L. Oliveira, C. C. Monteiro, P. R. M. Inacio, and M. M. Freire, "CloudSim plus: A Cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, May 2017, pp. 400–406.

[32] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of Edge computing systems," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 11, Nov. 2018, Art. no. e3493.

[33] I. Lera, C. Guerrero, and C. Juiz, "YAFS: A simulator for IoT scenarios in Fog computing," *IEEE Access*, vol. 7, pp. 91745–91758, 2019.

[34] R. Mahmud, S. Pallewatta, M. Goudarzi, and R. Buyya, "IFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in Edge and Fog computing environments," *J. Syst. Softw.*, vol. 190, Aug. 2022, Art. no. 111351.

[35] D. Alekseeva, A. Ometov, and E. S. Lohan, "Towards the advanced data processing for medical applications using task offloading strategy," in *Proc. 18th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2022, pp. 51–56.

[36] *Service Requirements for Cyber-Physical Control Applications in Vertical Domains*, Standard 3GPP TS 22.104 V18.0.0, Rel. 18, Mar. 2021.

[37] A. L. H. Chow, H. Yang, C. H. Xia, M. Kim, Z. Liu, and H. Lei, "EMS: Encoded multipath streaming for real-time live streaming applications," in *Proc. 17th IEEE Int. Conf. Netw. Protocols*, Oct. 2009, pp. 233–243.

[38] W. B. Qaim, A. Ometov, C. Campolo, A. Molinaro, E. S. Lohan, and J. Nurmi, "Understanding the performance of task offloading for wearables in a two-tier Edge architecture," in *Proc. 13th Int. Congr. Ultra Modern Telecommun. Control Syst. Workshops (ICUMT)*, Oct. 2021, pp. 1–9.

[39] *Cortex-M3*. Accessed: Jul. 24, 2023. [Online]. Available: https://developer.arm.com/Processors/Cortex-M3

[40] *Arm Cortex-M4*. Accessed: Jul. 24, 2023. [Online]. Available: https://www.st.com/content/st_com/en/arm-32-bit-microcontrollers/arm-cortex-m4.html

[41] *Instructions Per Secondl*. Accessed: Jul. 24, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Instructions_per_second

[42] *Intel Xeon Platinum 8180 Processor*. Accessed: Jul. 24, 2023. [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/120496/intel-xeon-platinum-8180-processor-38-5m-cache-2-50-ghz.html

[43] *Export Compliance Metrics for Intel Microprocessors*. Accessed: Jul. 24, 2023. [Online]. Available: https://www.intel.com/content/www/us/en/support/articles/000005755/processors.html

[44] E. Barbierato, M. Gribaudo, M. Iacono, and A. Jakóbik, "Exploiting CloudSim in a multiformalism modeling approach for Cloud based systems," *Simul. Model. Pract. Theory*, vol. 93, pp. 133–147, May 2019.

[45] S. A. Abdulkareem and A. J. Abboud, "Evaluating Python, C++, Javascript and Java programming languages based on software complexity calculator (Halstead Metrics)," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 1076, no. 1, Feb. 2021, Art. no. 012046.

Daria Alekseeva (Student Member, IEEE) received the B.Sc. and M.Sc. degrees from the Saint Petersburg State University of Telecommunications (SUT), Saint Petersburg, Russia, in 2017 and 2019, respectively. She is currently pursuing the Ph.D. degree with Tampere University (TAU), Tampere, Finland. Her research interests include wireless communications, network security, computing paradigms, and neural network technologies.

Aleksandr Ometov (Senior Member, IEEE) received the M.Sc. and D.Sc. (Tech.) degrees from the Tampere University of Technology (TUT), Tampere, Finland, in 2016 and 2018, respectively. His research interests include wireless communications, information security, computing paradigms, and wearable applications.

Elena Simona Lohan (Senior Member, IEEE) received the M.Sc. degree from the Polytechnic University of Bucharest, Bucharest, Romania, in 1997, the D.E.A. degree (French equivalent of master) from École Polytechnique, Paris, France, in 1998, and the Ph.D. degree from the Tampere University of Technology (TUT), Tampere, Finland, in 2003. She is currently a Professor with the Electrical Engineering Unit, Tampere University (TAU), Tampere. Her current research interests include wireless location techniques, wearable computing, and privacy-aware positioning solutions.