Simulation Modelling Practice and Theory xxx (xxxx) xxxx

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

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

A survey and taxonomy of simulation environments modelling fog computing

Andras Markus*, Attila Kertesz

Department of Software Engineering, University of Szeged, Hungary

ARTICLE INFO

Keywords:
Fog computing
Cloud computing
Internet of things
Simulation
Taxonomy
Survey

ABSTRACT

In the past ten years, the latest advances in Information and Communication Technology had a significant impact on distributed systems by giving birth to paradigms such as Cloud Computing, Fog Computing and the Internet of Things (IoT). The environments they created are closely coupled in most cases: IoT sensors and devices generate data that have to be stored, processed and analysed by cloud or fog services, depending on the actual application needs. These IoT-Fog-Cloud systems are very complex, and the use of simulations in their design, development and operational processes is inevitable. Nowadays, there are many simulator solutions available to model and analyse these systems depending our research needs, but in many cases it is hard to grasp their differences, and implementing certain scenarios in different tools is time consuming. The goal of this work is to help researchers and practitioners in this regard by proposing a survey and taxonomy of the available simulators modelling clouds, IoT and specifically fogs, which is the latest, currently still forming paradigm. The main contributions of this study are our novel viewpoints for classification including software quality, which is performed by analysing the source code of the considered simulators. We also propose comparison tables for three groups of simulators that reveal their differences and the way they model the elements of these systems. Finally, we discuss the relevant findings of our classifications, and highlight open issues that need further research.

1. Introduction

In the past decade we experienced how rapidly distributed computing infrastructures evolve. The use of cloud technologies was almost unavoidable in 2010 for successful service providers of the future. Miniaturisation and improvements in battery lifetimes have led to small computational devices that can interact and communicate among themselves and with the environment through the Internet giving birth to the Internet of Things (IoT) paradigm. As the number of things grows, the vast amount of data they produce requires the assistance of cloud services for storage, processing and analysis. Such IoT-Cloud systems can be utilised in many application areas ranging from local smart homes to mid-range smart cities, or wider smart regions. To cope with the possibly huge number of communicating entities, data management operations are better placed close to their origins, resulting in better exploiting the edge devices of the network. In the latest distributed computing paradigm called Fog Computing [1] groups of such edge nodes forms a fog, where data processing and analysis can be performed with reduced service latency and improved service quality compared to a remote cloud utilisation.

As a result, cloud and fog technologies can be used together to aid data management needs of IoT environments, but their

* Corresponding author.
E-mail addresses: markusa@inf.u-szeged.hu (A. Markus), keratt@inf.u-szeged.hu (A. Kertesz).

https://doi.org/10.1016/j.simpat.2019.102042
Received 29 June 2019; Received in revised form 12 December 2019; Accepted 13 December 2019
1569-190X/

application give birth to complex systems that still needs a significant amount of research. It is obvious that significant investments, design and implementation tasks are required to create such IoT-Fog-Cloud systems in reality, therefore it is inevitable to use simulations in the design, development and operational phases of such establishments. This rationale has led many scientist to create simulators to investigate and analyse certain properties and processes of similar complex systems. There are already existing survey papers highlighting the basic capabilities of simulation tools in clouds, IoT and fogs, also comparing them by certain views, e.g. by Ragman et al. [2] or by Puliafito et al. [3]. Nevertheless, we believe that modelling Fog Computing in such simulators is far from complete, and there is a need to gather and compare how key properties of fogs are represented in these works to trigger further research in this field.

The main contributions of this survey are: (i) a detailed introduction and analysis of simulators modelling cloud, IoT and fog systems; (ii) the identification and definition of viewpoints (including software quality metrics) and a taxonomy for classifying them; and (iii) a throughout categorisation of these simulators with comparison tables revealing their properties related to fog modelling, and the identification of open issues still remaining.

The remainder of this article is as follows: Section 2 introduces related works proposing surveys in similar research fields and states our methodology for literature review. Section 3 is composed of five subsections, in which we give short introductions of the analysed simulators, define the taxonomy elements used for categorising them, provide classifications with comparison tables for three groups of simulators, and discuss the relevant findings of our classifications. Finally, Section 4 states future recommendations, and Section 5 concludes our work.

2. Related work and literature review

The methodology for finding suitable works for our investigations was twofold. First, we looked for recently published surveys targeting Fog Computing, and narrowed their scope to Fog modelling and simulation. Hence, we filtered the group of their cited papers, and kept the ones using simulators. In this way, we could build on their results, as well as go beyond their findings by further analysing these works. Second, to extend the group of considered solutions, we performed literature search with the following engines: Google Scholar, ResearchGate, Scopus and Dimensions. We performed detailed searches for publications in the last five years (2015-–2019) with the following keywords (for all fields): *Fog Computing + Simulation*. Table 1 shows the results of our literature search per publication year. We went through the search results and gathered only those works that contained solutions having open-source simulator implementations or containing novel approaches in the field of Cloud and Fog Computing. ResearchGate does not provide detailed information on search results, thus we omitted it in the comparison table. We also considered relevant solutions referenced in the processed articles, but we skipped those tools or simulators, which were published more than 10 years ago.

There already a couple of survey papers addressing different aspects of Cloud Computing and Fog Computing supporting IoT systems. These works helped us to identify taxonomy categories for comparing our considered papers, i.e. solutions modelling Cloud and Fog Computing in simulated environments. Next, we summarise the most relevant surveys we found, then compare them with our research aims.

Rahman et al. [2] looked for available solutions for simulating Cloud Computing, and modelling data centres and their networks. The authors investigated capabilities of cloud simulators with the following categories: (i) graphical user interface, (ii) application model, (iii) communication model, (iv) energy model, (v) virtual machine (VM) support, (vi) SLA support, (vii) cost model. They investigated 15 simulators in detail, and also listed the applied programming language, the utilised platform, and the code availability of the simulators. Probably this study is the closest to our work in terms of comparison methodology, but we focused more on the quality of the simulator software, and broadened the research scope to IoT and fog areas.

Yousefpour at al. [4] presented a survey on Fog Computing and made comparisons with the following main points of views: computing paradigms such as cloud, fog, mobile, mobile-cloud, edge, and frameworks and programming models for real applications, software and tools for simulated applications.

Svorobej at al. [5] examined different fog and edge computing scenarios, and investigated Fog and Edge computing with various aspects, such as (i) Application Level Modelling, (ii) Infrastructure and Network Level Modelling, (iii) Mobility and (iv) Resource Management and (v) Scalability, and made a comparison of seven available fog tools.

Hong and Varghese [6] summarised resource management approaches in Fog and Edge Computing focusing on their utilised infrastructure and algorithms, while Ghanbari at al. [7] investigated different resource allocation strategies for IoT systems. This second work classified the overviewed works to eight categories: QoS-aware, context-aware, SLA-based, efficiency-aware, cost-aware, power-aware, utilisation-aware, and load balancing-aware resource allocation. We found this work particularly interesting, hence it

Table 1
Literature search results in 5th June, 2019.

Year	Google Scholar	Scopus	Dimensions
2019	3320	914	2940
2018	6140	1560	5447
2017	4460	707	3389
2016	2870	310	2100
2015	2160	206	1650
Sum	18 950	3697	15 526

Table 2
Comparison of related surveys according to their main contributions.

Survey	Year	Aim
Yousefpour at al. [4]	2018	paradigms and research topics for fogs
Hong and Varghese [6]	2018	resource management in the fog and edge
Rahman et al. [2]	2019	simulating cloud infrastructure
Svorobej at al. [5]	2019	fog and edge simulation challenges
Ghanbari at al. [7]	2019	resource allocation methods for IoT
Puliafito et al. [3]	2019	fog characteristics for IoT
Our survey	2019	fog models and software quality of simulators

opened the investigations to the field of IoT.

Puliafito et al. [3] investigated the benefits of applying Fog Computing techniques to support the needs of IoT services and devices. In their survey they described the characteristics of fogs, and introduced six IoT application groups exploiting fog capabilities. They also gathered fog hardware and software platforms supporting the needs of these IoT applications.

Table 2 summarises the relevant surveys representing the current state of the art close to our research aims. Though cloud solutions are dominating, some started to open investigations towards fog, edge and IoT fields. To the best of our knowledge, there is no detailed study available targeting the fog modelling capabilities and the usability of simulation tools. In our work we still do not neglect cloud and IoT solutions, hence they are tightly connected to fogs in most cases. We also investigate the software quality of the available simulators, which has a direct effect on the learning curve of utilisation and experiment reliability.

Referring back to our literature search methodology, we checked all references of these six surveys and found 39 papers fitting our aims. Then we analysed these works in detail focusing on simulator availability and fog modelling, further reducing the number of papers to 13. In parallel, based on the throughout literature searches we performed, we found 30 papers containing simulators that could potentially be used for modelling Fog Computing environments having considerable user communities. Most of these started to be developed as cloud simulators, and were later extended to model fogs as well. As a result, we arrived to 43 solutions to serve as a base for our research investigations. In the next section we give a short introduction to the main goals and properties of these works, then compare them in a detailed taxonomy.

3. Modelling fog computing in simulators

In this section we present an introduction and detailed comparison of the currently available simulators in the fields on cloud, fog and IoT. We considered the following properties as taxonomy categories for the investigation: (i) the topology and layers of a simulator, (ii) the type of a simulator (i.e. a generic, event-driven or a specialised, network simulator), (iii) the date of the latest modification (to help filtering outdated solutions), (iv) the applied cost model in a simulator (i.e. pre-defined, static or real provider pricing model), (v) geographic location management (mostly for fog placement optimisation), (vi) utilised sensor model, (vii) configurable network settings or protocols, (vii) VM management functionality, (viii) power usage or energy consumption calculations, and finally (ix) support for hierarchical organisation model (mostly in case of IoT and fog simulators).

3.1. Introduction of general cloud simulators

As we mentioned before, the life of most simulators we consider started as a cloud or network simulator. Some stayed at this level, and others were extended to support the analysis of fog, edge or IoT management. This means that we cannot avoid the investigation of cloud simulators.

The one of the most referred and widely used simulators is the CloudSim simulation toolkit [8], which is a popular solution for simulating cloud environments. In CloudSim, users define tasks by creating so-called cloudlets, which are processed by virtual machines running on cloud resources. This open-source, Java-based solution is available on GitHub [9], and it is built on SimJava [10]. CloudSim is a discrete event simulator, and its architecture has five layers (i.e. network, cloud resources, cloud services, VM services, user interface structures). Virtual machines in CloudSim has three states, and users can configure only static usage costs for resources (i.e. memory, bandwidth, CPU and storage usage). CloudSim also contains a power model, but it is only restricted to CPU energy consumption. Communication between components in the network is modelled with the bandwidth parameter and a delay value.

There are many extensions of CloudSim, focusing on mostly one aspect of Cloud Computing or providing implementation for a missing or insufficient feature of the original software. Unfortunately, it seems that the developers found no need to collect all of the extensions into a single repository, and there are also many referred studies without publishing or citing available source code. Next we introduce the ones we found relevant for our research.

CloudAnalyst [11] is one of the oldest extensions of CloudSim and contains important additions by implementing (i) application users, (ii) internet connections, (iii) simulations defined by time periods, (iv) service brokers. CloudAnalyst is available on cloudbus.org [12].

CDOSim [13] is an extension to simulate cloud deployment options for migration support. The simulator can measure the cost of a deployment and its response time. The novelty of this work to introduce MIPIPS, which is fine-grained version of MIPS for low level instruction definitions.

TeachCloud [14] is an other extension for educational purposes. The main features added in this version are: (i) cloud workload generator, (ii) MapReduce framework, and (iii) new cloud network models. It is available on GitHub [15].

EMUSIM [16] has the purpose to bring emulation and simulation into one tool to predict service behaviour, when the resource pool is changing. First, an emulation phase can be used to extract information on application behaviour, then in the second phase this information is fed into a simulation model to arrive to more accurate methods for application simulation. Its source is available on cloudbus.org [17].

CEPSim [18] has the aim to simulate complex event processing (CEP) and stream processing systems in cloud environments. CEP is usually related to Big Data management, in this version CEP systems are represented by Directed Acyclic Graphs. Users can define how to process input streams with special queries that can run continuously in the simulation with user-defined runtime. This work is also available on GitHub [19].

CloudEval [20] is a CloudSim extension to support virtual machine consolidation with different migration algorithms. The authors have created a domain specific language (DSL) for algorithm implementation for investigating specific VM migration strategies. The proposed DSL is based on the Groovy programming language, and it contains the followings: descriptions for (i) data centre and the VM scheduling strategies, (ii) evaluation metrics, and finally (iii) data collection and performance metrics.

CloudSimSDN simulator [21] is an extension to simulate Software Defined Networking (SDN) methods, which makes network elements (e.g. switches) dynamically programmable. Its new components (e.g. Link, Switch and Host) enable networking with virtual links in the Physical Topology. There is also a Virtual Topology layer in the top level of the CloudSimSDN architecture. The configuration of a simulation is based on JSON files which contain specifications for the nodes, policies and bandwidth values, and CSV files which contain the workload on the defined topology. This tool is available on Github [22].

The ContainerCloudSim [23] package was created by responding to the growing trend of container technologies. It is an extension that enables running containers on top of VMs in a simulation. In this solution the Container entity is hosted by a VM, and the container task has a very similar working as a Cloudlet.

NetworkCloudSim [24] extends CloudSim with enhanced network options for modelling the interconnection of data centres. NetworkCloudlet is a novel component in this version, which can be used to simulate different task types (namely communication and computation tasks). Other extended components are the NetworkDataCenter and the NetworkHost (which can be manager by the former), these classes make it possible to simulate network traffic (using bandwidth and latency parameters) through switches. It also uses the ContainerCloudSim package.

DynamicCloudSim [25] was created to simulate instability and dynamic changes in VM provisioning, which may occur in cases the host machine serves more than one virtual machines at the same time. The newly introduced features are the followings: (i) running different type of task (i.e. input-output, CPU, bandwidth-bound) on VMs, (ii) managing instability caused by hardware and network issues, (iii) managing heterogeneity caused by different type of hosts, (iv) introducing VM performance changes and noise, and finally (v) defining failures during task execution. The source code of this tool is available on GitHub [26].

Finally, CloudSim Plus [27] is a redesigned and refactored version of CloudSim, which aims to provide easier usage and more reliable simulation. It is also available on GitHub [28].

Concerning solutions outside the CloudSim world, the DISSECT-CF cloud simulator [29] is an event-driven simulator written in Java, available on GitHub [30]. It can be used to model cloud Infrastructure-as-a-Service systems, having an energy consumption model both for physical and virtual machines with min, max and idle power states. Concerning resource management, there are four possible states of a physical machine, while virtual machines can be in 11 different states including migration. To model network communication between nodes, users can configure latency, as well as input, output and disk bandwidth. For further information, a recent study on its comparison to CloudSim is available in [31].

DCSim [32] is an event-driven simulator (using the Java programming language). It was developed for simulating virtualized resource management. The main ability of DCSim is to provide VM sharing mechanism for VMs belonging to a single, multi-tiered application. Its architecture is composed of: DataCentre, Host, VMAllocation, and there are unique managers for networking, virtual machines and power consumption. During simulation the following values are measured: (i) dynamic VM allocation (migration), (ii) SLA violation, (iii) host operation hours and utilisation, (iv) power consumption and (v) simulation and algorithm running time. DCSim is available on GitHub [33].

GroudSim [34] aims to simulate Grid and Cloud systems in a scalable way. This Java-based software is a discrete-event simulator with the main abilities to provide cost calculation, and simulations using background-load of the resources. The architecture is composed of Entity, Job and Cost elements, and it contains a cost model using time units of CPU usage, or data usage (in GB).

GreenCloud [35] is an extension of NS-2 [36] packet-level network simulator and based on TCL scripts and C++. The main purpose of the simulator is to present energy-aware cloud datacentre analysis including energy-usage measurements of the system components (i.e. servers, switches, links). Its energy consumption model includes PUE, and DC infrastructure efficiency (both are represented in Watt). The architecture components can be servers and switches, and the tool is able to simulate TCP/IP protocols for networking.

ICanCloud [37] is an OMNET++ based simulator [38] (programmed in C++), which aims to simulate cloud infrastructures. Its novelty is that it introduces a hypervisor component for managing cloud brokering policies. Its experiments are focused on the trade-offs between cost and performance of a given application, which can be run with Amazon VM instances types only. The architecture has the following elements: cloud system, hypervisor, application repository, VM repository, hardware models. It also contains a cost model, which follows the pay-as-you-go manner. The virtual machine handling is quite simple, it can execute the jobs, but neglects cloud-specific network simulation. It uses the Inet framework (included in OMNET++) for TCP/UDP protocols. It is available at

[39].

Finally, SPECI [40] was developed with the aim to simulate cloud-scale data centres focusing on their performance and behaviour, which it is based on a discrete-event simulator called DES Java.

3.2. Introduction of IoT simulators

Some of these cloud simulators followed the new trend represented by the Internet of Things. Such things are sensors and small devices that can be connected to the Internet, and used to monitor environments or gather special-purpose data. They are rarely utilised ”alone”, cloud services are generally needed to store and process their data. Once such data management is performed at the cloud network edge or close to the users in purpose, we arrive to the latest trend of Fog Computing. This reasoning led us to introduce the category of IoT simulators in our research. The revised cloud simulators had the aim to model entities of the Internet of Things world, which in some cases led to the introduction of fog features. Nevertheless, we can also find solutions focusing on IoT system operation, somewhat neglecting or hiding cloud and fog capabilities. In our view, IoT simulators belong to a separate category, hence IoT devices and sensors can be modelled as independent entities of data sources having special properties (e.g. data generation frequency, behaviour models, limited connectivity). In this section we compare works focusing on IoT management.

SysML4IoT [41] is an extension of SysML and it is designed for model-driven development for IoT systems. In SysML4IoT an IoT system has two main component groups: devices (Tag, Sensor, Actuator), and services (Human, Digital Artifact), and this approach follows the publish/subscribe pattern to specify sensor behaviour. With this tool a high level model can be designed of an IoT system, that can later be transformed to a source code, hence an implementation of a system.

CrowdSenSim [42] is a simulator designed for mobile crowd-sensing. The main features are the followings: (i) users can read in the layout of cities, (ii) can manage user mobility and (iii) perform energy measurements for sensing tasks. The modules or layers are the followings: User Mobility, City Layout and CrowdSensing module. For managing mobility, latitude, longitude and altitude of a device is considered, and it may also use real sensors (e.g. accelerometer, temperature and pressure). This simulator contains a power model using idle, transmission and reception modes, and the cost of the sensing is also measured. It is available at [42].

The DPWSim [43] simulation toolkit helps to develop and test IoT application with web services using the DPWS standard without any physical devices. DPWS works with publish/subscribe mechanism on the architecture based on spaces, devices, operations and events. The main abilities of the DPWSim are the following: (i) Platform Independence due to JVM, (ii) virtual device management for DPWS and it supports SOAP and HTTP protocols. DPWSim is available on GitHub [44].

SimIoT [45] is an extension of SimIC [46] with IoT layer, and the authors demonstrate its utilisation with an IoT-based health care application use case. This extension provides an extra layer for communication between the entities. Its architecture has three main layers: User level (device-sensor), SimIoT level (communication broker) and SimIC level (cloud entities). The virtual machines are used to execute tasks, and there is a bandwidth constraint for governing the network load.

There are a few studies mentioning the use of a special software to simulate or manage IoT systems. Angelakis et al. [47] used the Mixed Integer Linear Program in Matlab to model resource allocation problems in heterogeneous IoT networks. Simulink [48] is also a Matlab-based solution that provides secure model based design for IoT system analysis. Thomas and Irvine [49] presented an LTE approach for IoT sensor networks. They performed experiments with the OMNET++ network simulator to investigate how many sensor nodes can be transmitted per resource block.

The SmartSim tool [50] was designed for energy-metering in IoT smart homes. This simulation tool was developed in Python, and it is able to generate smart device energy traces based on energy models (with frequency, duration, time and activity). It is available on GitHub [51].

There is a semi-simulated solution for modelling IoT, fog, and cloud systems called MobIoTSim [52], which is an Android-based software for IoT device simulation and management. It was designed with the aim to avoid expensive device and sensor purchases for developing and evaluating IoT applications. By executing simulations, the mimicked devices can connect to real cloud providers (e.g. IBM Bluemix), and communicate over the network using MQTT and HTTP protocols. MobIoTSim has the following layers: sensor, device, application and cloud. This simulation tool available on GitHub [53].

IOTSim [54] is a CloudSim extension that supports Internet of Things and Big Data simulation. IoT has appeared in the CloudSim model as a three layer architecture: perception, network and application layer, with the following details: CloudSim Simulation Layer, Storage Layer, Big Data Processing Layer and User Code Layer. The aim of this work was to simulate a MapReduce approach for Big Data processing, hence the novelty of this work is the MapReduce function implemented by the MapCloudlet and ReduceCloudlet entities. Unfortunately, no detailed information can be found on sensors or actuators, and there were no available source code cited.

Finally, DISSECT-CF-IoT [55] is an extension of the DISSECT-CF cloud simulator. The novelty of this extension is the sensor model, which contains detailed configuration options (e.g. measurement delay, data generation frequency and generated file size parameters), detailed network configurations for IoT devices, and different approaches for multi-cloud management strategies for IoT applications. It also contains extensible pricing models of four real providers (Amazon, Azure, IBM Bluemix and Oracle), both for IoT and cloud side costs. To enable large-scale simulations, it uses different XML description files for easy configuration management. It is available on GitHub [56].

3.3. Introduction of fog simulators

Finally, we arrive to the latest category of fog simulators responding to the latest trend called Fog Computing. In this category we focus on key properties required to model a Fog Computing environment. In their development we can also identify the close relation to clouds: most of them are extensions of cloud or IoT simulators. Nevertheless they follow different architectural models: some has centralised, some more decentralised, peer-to-peer communication schemes. As a result, understanding the model elements, functions and implementation details can be very hard and time-consuming; that is one of the issues our survey aims to relax.

Brogi et al. [57] presented a novel cost model for deploying applications on a fog infrastructure. The approach is based on a simulation prototype called FogTorchΠ (available on GitHub [58]), which determines a deployment on fog according to actual resource consumption and cost needs. To calculate these metrics, the simulator uses Monte Carlo methods. The main capabilities of this simulator are the monthly cost calculation extended with subscription/data transfer cost of IoT devices (using bandwidth and latency parameters), and it takes into account geolocation information. The cost model differentiates two methods: (i) to choose one pre-defined virtual machine with the given costs, or (ii) to build the necessary units of the CPU cores, RAM and HDD for a virtual machine. The connections of fog nodes and clouds have different types, and both vertical and horizontal hierarchies are possible. The realisation of an IoT-Fog-Cloud architecture follows the 3-tier model sensor-fog-cloud, but it lacks some important features of cloud/fog simulation (e.g. virtual machine simulation and management). This tool focuses on providing a fog searching algorithm for the given parameters. FogDirSim [59] is a closely related simulation tool from the same authors, which provides compatibility with CISCO FogDirector across REST services to secure managing IoT applications on fog architectures.

Abbas et at. [60] presented a work using the OPNET network simulator [61] and aims to present the fog security service for end-to-end security between the fog layer and IoT devices. They represent the difference between the cloud-fog-device layer architecture and the fog-device architecture with a decentralised approach. Authors describe an encryption solution for Fog layer and its devices in a mixed real-life system and simulation possibilities. First, they run a scenario with real devices (e.g. iPhone, Samsung Galaxy) to get some benchmark values, then they used the OPNET network simulator parameterized with the measured values to get information about different traffic loads.

Tychalas and Karatza proposed PDES [62] for Fog Computing, the Parallel Discrete Event Simulation implemented in C programming language. They defined a system with the following parameters: a cloud of 128 VMs, a cluster of 32 raspberries, a cluster of 64 PCs and 64 smartphones. If a task arrives to the system, it can be stored in a resource queue. The tasks are independent, and they prefer the shortest queue with the lowest load for placing a task. There are three thresholds in the system for task allocation (representing the cloud, raspberry and smartphone categories). It seems that there is no possibility for detailed and dynamic configuration of simulated system components, the provided solution focuses on analysing task scheduling algorithms to decrease the response time.

The FogNetSim++ [63] is built on the OMNeT++ discrete event simulator, which is focuses on network simulation, and available on GitHub [64]. This simulator was designed to provide configuration options to handle fog networks (e.g. fog node scheduling algorithms and devices hand-over). The implementation programming language is C++. The main capabilities of this simulator are the handling of communication protocols such as MQTT or CoAP, and the different mobility models such as LinearMobility or TractorMobility.

Edge-Fog [65] is a Python-based simulator, in which a fog layer is represented by networking devices (e.g. routers and switches). The authors use this tool to present their LPCF algorithm (i.e. least processing cost first), which orders tasks to available nodes by minimizing processing time and network costs. This approach is decentralised, thus the edge layer has device-to-device connection. It is available on GitHub [66].

The Yet Another Fog Simulator (YAFS) [67] is proposed to simulate application deployment on a fog infrastructure. The main capabilities of the simulator are the following: (i) dynamic application module allocation, (ii) network failures and (iii) user mobility. YAFS is a Python based discrete event simulator, and provides JSON-based scenario definition. The simulation results contain information such as the network utilisation, response time, network delay. Its source is available on GitHub [68].

EdgeNetworkCloudSim [69] is an extension of CloudSim that supports the edge computing paradigm focusing on consolidation and orchestration on the edge (mostly mobile) devices. The EdgeNetworkCloudSim is based on NetworkCloudSim, and the extensions are the following: EdgeService models the service chain, EdgeDatacenterBroker communicates with a user, and EdgeVms helps to define a service app. This work is also available on GitHub [70].

The PureEdgeSim [71] tool was proposed to design and model cloud, fog and edge applications focusing on high scalability of devices and heterogeneous systems. PureEdgeSim is based on CloudSim Plus, and it uses XML descriptions for the simulation, where the users can configure the data and parameters of geographical location, energy model and virtual machine settings.

One the most referred fog simulators is iFogSim [72], which is also based on CloudSim. iFogSim can be used to simulate real systems and follows the sensing, processing and actuating model, therefore the components are separated to these three categories. The main physical components are the following: (i) fog devices (including cloud resources, fog resources, smart devices) with possibility to configure CPU, RAM, MIPS, uplink- and downlink bandwidth, busy and idle power values, (ii) actuators with geographic location and reference to the gateway connection, (iii) sensors, which generates data in form of a tuple representing information. The main logical components aim to model a distributed application: the (i) AppModule is a processing element of iFogSim, and the (ii) AppEdge realises the logical dataflow between the virtual machines. The main management components are following: the (i) Module Mapping searches for a fog device to serve a virtual machine, if no such device is found, the request is sent to an upper tier object, and the (ii) Controller launches the application on a fog device. For simulating fog systems, first we have to define the physical components, then the logical components, finally the controller entity. Although numerous articles and online source codes are available for the usage of this simulator, but based on our experiments suggest that there is a lack of source code comments for many methods, classes and variables. As a result, application modelling with this tool requires a relatively long learning curve, and its operations take valuable time to understand. It is available on GitHub [73]. After its appearance, it has been used for many research works and various experiments. For example, in [74] a smart city network architecture is presented for Fog Computing called FOCAN as a case study, and in [75] the authors purposed to combine Fog and Internet of Everything into the so-called Fog of Everything paradigm.

There are also more advanced extensions of iFogSim. MyiFogSim [76] (available on GitHub [77]) was designed to manage virtual machine migration for mobile users. The main capability of this simulator is the modelling of the user mobility and its connection with the VM migration policy. The evaluation contains a comparison to a simulation without VM migration. Another extension is the iFogSimWithDataPlacement [78] tool (available on GitHub [79]), which proposed a way for data management investigation on how data are stored in a fog system. It also considers latency, network utilisation and energy consumption for data placement. The authors presented a parallel Floyd-Warshall algorithm to define the shortest distance between the nodes for the optimal data placement.

EdgeCloudSim [80] is another CloudSim extension that is available on GitHub [81]. The main capabilities of this simulator are the network modelling extension for WLAN, WAN and the device mobility. They aimed to respond to the disadvantage of iFogSim's simple network model that ignores network load and does not provide content mobility.

StormOnEdge/SpanEdge [82] is another decentralised tool related to edge computing, aiming to model data stream-processing. Developers can build-up a geographically distributed network by installing the parts of a stream processing application near to the data source for latency and bandwidth reduction. It is available on GitHub [83].

We can also find simulation solutions the field of Fog Computing exploiting the use of Matlab, Docker or other real service APIs. Zhang et al. [84] proposed to investigate simulated resource allocation in a 3-tier fog network by using by MatLab framework. They aim to provide management functionalities for data service operators, data service subscribers and fog nodes.

DockerSim [85] aims to support the analysis of container-based SaaS systems in simulated environments. It is based on the OMNET++ and iCanCloud network simulators to model container behaviour, network, protocol and OS process scheduling behaviour. It is available on GitHub [86].

EmuFog [87] is an extensible emulation framework for fog infrastructures, and also a useful tool for emulating real applications. After designing a network topology, EmuFog enables to connect fog nodes in the topology, and to run Docker applications on it. It is available on GitHub [88].

DISSECT-CF-Fog is a direct extension of DISSECT-CF-IoT, available on GitHub [89], written in Java programming language. The purpose of this simulator extension is to model fog devices and nodes, and by building on the core DISSECT-CF simulator and on DISSECT-CF-IoT extension functions, it is able to model IoT-Fog-Cloud systems. The main benefit of this fog extension is offers the possibility of detailed configuration settings through XML configuration files, and it requires only minimal programming knowledge for define additional scenarios. DISSECT-CF-Fog contains an own simulation time unit for the general and time-independent simulations. The network operations, such as bandwidth, latency simulations and file transfers between the physical components are supported by its core simulator. To create a physical topology, any horizontal and vertical connections are allowed, but for now the vertical communication supports only upward direction. The IoT layer provides the management of smart devices and those sensors with the abilities of simulating sensor measurement time, data generation frequency with size configuration. Its application management layer handles the VMs and pairs the compute tasks (generated based on the received amount of data) to the VMs. The cost module is responsible for evaluating pricing methods, capable of calculating both dynamic cloud and IoT side costs based on real provider schemes. Finally, the logical data-flow are defined by the physical topology by default for simplicity.

Table 3
Comparison of the examined cloud simulators with implementation-related properties.

Simulator	Core simulator	Last modified or published	Type
CloudSim	SimJava	2019	event-driven
CDOSim	CloudSim	2012	event-driven
NetworkCloudSim	CloudSim	2011	event-driven
TeachCloud	CloudSim	2015	event-driven
CloudAnalyst	CloudSim	2009	event-driven
EMUSIM	CloudSim	2012	event-driven
CEPSim	CloudSim	2016	event-driven
CloudEval	CloudSim	2016	event-driven
CloudSimSDN	CloudSim	2019	event-driven
ContainerCloudSim	CloudSim	2019	event-driven
DynamicCloudSim	CloudSim	2017	event-driven
CloudSim Plus	CloudSim	2019	event-driven
DISSECT-CF	-	2018	event-driven
SPECI	DES Java	2009	event-driven
DCSim	-	2014	event-driven
GroudSim	-	2011	event-driven
GreenCloud	NS-2	2011	network
ICanCloud	OMNET+	2015	network

Table 4
Comparison of the examined cloud simulators with cloud modelling properties.

Simulator	Architecture	Cost model	Network model	VM management	Energy model
CloudSim	Network, Cloud Resources, Cloud Services, VM services and User Interface Structure	Static cost for physical resources: memory, storage, bandwidth, and CPU usage	Bandwidth and the delay value between the entities	3 states of VMs and it executes Cloudlets	Power model is based on max power and a constant (static power) values
CDOSim	Similar to parameters of CloudSim				
NetworkCloudSim	NetworkDataCenter, Switch and NetworkHost		Simulate network traffic (bandwidth + latency) through switches	3 states of VMs and it executes Cloudlets	Power model is based on max power and a constant (static power) values
TeachCloud	Similar to parameters of CloudSim				
CloudAnalyst	Similar to parameters of CloudSim				
EMUSIM	Similar to parameters of CloudSim				
CEPSim	Similar to parameters of CloudSim				
CloudEval	Similar to parameters of CloudSim				
CloudSimSDN	Physical Topology (Link, Switch and Host and Cloud Data Center) and Virtual Topology	Static cost for physical resources: memory, storage, bandwidth, and CPU usage	Virtual link with bandwidth values	3 states of VMs and it executes Cloudlets	Power model is based on max power and a constant (static power) values
ContainerCloudSim	Similar to parameters of CloudSim				
DynamicCloudSim	Similar to parameters of CloudSim			Dynamic VM with task's length, I/O and bandwidth coefficient	Power model is based on max power and a constant (static power) values
CloudSim Plus	Similar to parameters of CloudSim				
DISSECT-CF	Event System, Unified resource sharing, Energy Modelling, Infrastructure Simulation and Infrastructure Management	N/A	Latency, input bandwidth, output bandwidth and disc bandwidth between nodes	11 states of Virtual Machine and it executes compute tasks	Energy model for PM and VM including min-, max- and idle power
SPECI	N/A				
DCSim	DataCentre, Host and VMAllocation	N/A	Bandwidth manager	simulated VM manager, and VM allocation	Power model includes idle- and max power
GroudSim	Entities (CloudSite,GridSite), Job, Cost	Cost calculation for job execution	Background load on resources	N/A	N/A
GreenCloud	Servers, Switches and Links	N/A	TCP/IP	N/A	PUE and DC infrastructure efficiency
ICanCloud	Cloud System, hypervisor, application repository, VMs repository hardware models	N/A	N/A	N/A	N/A

Table 5
Comparison of the examined cloud simulators with software metrics.

Simulator	Language	Lines of Code	Comments (%)	Duplication (%)	Files	Bugs	Vulnerabilities	Code Smells
CloudSim	Java, XML	20 752	33.9	21.4	225	33	136	1.2k
CDOSim	Java				N/A			
NetworkCloudSim	Java							
TeachCloud	Java, XML, JSON	33 905	49.1	N/A	395		N/A	
CloudAnalyst	Java, HTML,XML	44 861	12.1	N/A	248		N/A	
EMUSIM	Java, XML, Python	1665	3.33	N/A	21		N/A	
CEPSim	Java, Scala	5875	20.5	N/A	82		N/A	
CloudEval	Java, Groovy				N/A			
CloudSimSDN	Java, XML	12 585	16.8	15.7	120	23	109	1.2k
ContainerCloudSim	Java				N/A			
DynamicCloudSim	Java, XML	16 778	31.5	14.2	173	65	428	754
CloudSim Plus	Java, XML	32 425	34.8	5.6	441	36	2	1.3k
DISSECT-CF	Java, XML	5152	42.6	0.3	62	9	17	173
SPECI	Java				N/A			
DCSim	Java, Markdown	9006	13.1	N/A	143		N/A	
GroudSim	Java				N/A			
GreenCloud	C+, TCL script				N/A			
ICanCloud	C+, HTML, XML, JS	2 901 003	4.1	N/A	12 463		N/A	

3.4. Detailed taxonomy for fog modelling in simulators

To compare the works we introduced in the previous subsections we define 10 taxonomy categories. These categories appear in the comparison tables we placed next to the introduction of the considered simulators: Tables 3–5 are used to summarise cloud simulators; Tables 6–8 are used for IoT simulator comparison; and finally Tables 9–11 depicts and compares properties of fog simulators. In the followings we define our taxonomy elements, then provide a discussion for the mapped survey papers:

- Simulation type:
  Simulation in general has a long history of research, and many classification schemes are available. Roth [90] stated that there are three major types of simulation models: discrete, continuous, and combined. For dynamic systems discrete event simulation models are the most widespread. In order to categorise the overviewed solutions, we define the following types of simulators: (i) network simulators aim to simulate the network connections and data transfers between the nodes. They are useful for modelling low-level interactions in systems, but their disadvantage is that it is hard to create higher-level abstract components (i.e. cloud or fog resources, IoT sensors) to build up complex systems (from their building blocks, i.e. network entities). Generally in these cases the simulation time may increase significantly. The other type category is the general, (ii) event-driven simulators, which use the discrete event simulation model, where the inner working of the systems can be modelled by specific time moments (i.e. events). These events are usually mapped to system states, and the state transitions to certain component operations. The disadvantage of these simulators is the lack of built-in network operations and properties, however one may choose the level of abstraction more easily, which is an advantage – as we have seen in DISSECT-CF or NetworkCloudSim to take into account the network traffic during the generation of virtual machines, communication or data forwarding. In Fig. 1 we can see the ratio of the defined simulator types for the considered works. More than the half of the investigated simulators (with 62.8%) of falls in the event-based category to simulate complex systems composed of cloud, IoT or fog elements.
- Implementation:
  The next category aims to reflect the development details of the considered simulators. Many of them have a single developer or a small group of contributors, only some have bigger developer communities (CloudSim or OMNET + +).
  We have seen in the previous subsections that some solutions become very popular, and influenced other researchers to extend the core tools with additional functionalities. Fig. 2 presents a graph highlighting the connections among the overviewed simulators based on the realised extensions. The bottom circle represents the core or base simulators, their extensions within the same system

Table 6
Comparison of the examined IoT simulators with implementation-related properties.

Simulator	Core (Simulator)	Last modified/ published	Type
SysML4IoT	–	2016	–
CrowdSenSim v1.0.0	–	2017	event-driven
DPWSim	–	2014	service-messaging events
SimIoT	SimIC	2014	N/A
SmartSim	–	2016	N/A
IOTSim	CloudSim	2016	event-driven
MobIoTSim	Android	2019	semi-simulated
DISSECT-CF-IoT	DISSECT-CF	2019	event-driven

Table 7
Comparison of the examined IoT simulators with IoT modelling properties.

Simulator	Architecture	Cost model	Geolocation	Sensor model	Network model	VM management	Energy model
SysML4IoT	N/A						
CrowdSenSim v1.0.0	User Mobility, City Layout and CrowdSensing module	Only energy cost of the sensing	Latitude, longitude and altitude	Real sensors	N/A		power in idle, transmission and reception mode
DPWSim	Spaces, Devices, operations and Events	N/A	N/A	N/A	SOAP and HTTP protocols	N/A	N/A
SimIoT	User level (device-sensor), SimIoT level(communication broker) SimIC level (Cloud Entities)	N/A	N/A	N/A	Bandwidth constraint	VM for execute task	N/A
SmartSim	N/A						
IOTSim	CloudSim Simulation, Storage, Big Data Processing, User Code Layer	Similar to parameters of CloudSim					
MobIoTSim	Sensor, Device, Application and Cloud layer	N/A	N/A	N/A	MQTT	N/A	N/A
DISSECT-CF-IoT	Sensor, Smart Device, and Cloud	Dynamic IoT and cloud side costs	N/A	Delay, frequency and file size	Latency, input bandwidth, output bandwidth and disc bandwidth between nodes for devices and nodes	11 states of Virtual Machine and it executes compute tasks	Energy model for PM and VM including min-, max- and idle power

Table 8
Comparison of the examined IoT simulators with software metrics.

Simulator	Language	Lines of Code	Comments (%)	Duplication (%)	Files	Bugs	Vulnerabilities	Code Smells
SysML4IoT				N/A				
CrowdSenSim v1.0.0	HTML, JS, C+, Python	44 437	2.8	N/A	346		N/A	
DPWSim	Java,HTML	55 346	51.2		551			
SimIoT	Java	N/A			N/A			
SmartSim	Python	946	12,7		13			
IOTSim	Java	N/A			N/A			
MobIoTSim	Java,XML	5490	4,6		75			
DISSECT-CF-IoT	Java,XML	7160	37.1	0.3	91	23	90	306

Table 9
Comparison of the examined Fog simulators with implementation-related properties.

Simulator	Core (Simulator)	Last modified/ published	Type
FogTorchΠ	–	2018	N/A
FogDirSim	–	2018	N/A
OPNET	–	2019	network
PDES	–	2018	event-driven
FogNetSim +	OMNET +	2018	network
Edge-Fog	–	2017	N/A
YAFS	–	2019	event-driven
EdgeNetworkCloudSim	NetworkCloudSim	2017	event-driven
PureEdgeSim	CloudSim Plus	2019	event-driven
iFogSim	CloudSim	2017	event-driven
MyiFogSim	iFogSim	2017	event-driven
iFogSimWithDataPlacement	iFogSim	2018	event-driven
EdgeCloudSim	CloudSim	2019	event-driven
StormOnEdge/SpanEdge	–	2016	N/A
Zhang et al. - Matlab	–	2017	N/A
DockerSim	iCanCloud	2017	network
EmuFog	–	2019	emulator
DISSECT-CF-Fog	DISSECT-CF-IoT	2019	event-driven

categories are placed on top of them, while the arrows lead to extensions for solutions modelling other systems as well. This graph shows that many variations of a base simulator exist, and we also know that a concrete simulator has many development versions that may have different features. This fact makes it very hard for researchers to choose the right version for their investigations, and for developers to create an improved solution of different versions of the same simulator. Our taxonomy aims to reveal some implementation details later, in this regard.

- Publication date:
  In the last decades we have seen the evolution of distributed systems: cloud systems got matured around 2010, then various things started to appear to form IoT systems to generate data for clouds, finally fog nodes were created to improve application execution quality of cloud services. Our investigations cover the past 10 years, and we believe that the publication and development dates can help us to place the considered simulators in this evolution timeframe. The publication date of a solution determines the technological novelties and capabilities its model is based on. It is true that a well maintained and updated simulator can follow the latest trends, but its software can ware out in a few years due to the corrections and extensions. The comparison tables reveal that CloudSim and DISSECT-CF were born quite early, and they are still under development to respond to the ongoing technology changes. On the other hand, old tools without improvements cannot fulfil recent researcher needs.
- Cost model:
  To predict costs of certain operations in a complex system is an important and useful feature for researchers. With the advent of commercial solutions in these distributed systems, it became inevitable to provide simulated cost calculations for users planning to enter this market. Though commercial solutions for Fog Computing are still in their infancy, we can find various cost models for clouds and IoT in the considered simulators.
- Geolocation:
  One of the goals of Fog Computing is to reduce data transfer and service response times, which require the use of geographical information of system elements (e.g. sensors, nodes or users).
  In most cases simulators having this property use two different representations: storing the (i) X and Y coordinates of an element in a system or the (ii) longitude and latitude values for more specific distance calculations. We applied this taxonomy category only for IoT and fog simulators.
  Mobility is a closely related property to geolocation, solutions offering this feature enable dynamic changes in the location of certain system elements.
- Sensor model:

Table 10
Comparison of the examined Fog simulators with fog modelling properties.

Simulator	Architecture	Cost model	Geolocation	Sensor model	Network model	VM management	Energy measurement
FogTorchΠ	Cloud Data Centre, Fog Node and Thing	N/A	Coordinates	Coordinates	Latency, bandwidth	N/A	N/A
FogDirSim	N/A						
OPNET	Device, Fog and Cloud	N/A	Benchmarked values	N/A	Benchmarked values	N/A	N/A
PDES	N/A						
FogNetSim++	Devices, Fog nodes, Broker node and Base station	Pay-as-you-go, subscription (monthly, etc.), pay-for-resources and hybrid model	Regions	Sensor node is data generator and user node generate or receive data, wire and wireless nodes	Execution delay, packet error rate, handovers, latency	N/A	Energy model for devices and fog nodes based the task computed
Edge-Fog	Edge and Fog and Data Store layer	N/A					
YAFS	Cloud, Fog, Sensor and Actuator	Cost of execution in cloud	Coordinates	Instructions, bytes	Bandwidth and link propagation	N/A	Power in watt
EdgeNetworkCloudSim	EdgeService, EdgeDatacenterBroker~ and EdgeVms	Similar to parameters of CloudSim			Simulate network traffic (bandwidth + latency) through switch	Similar to parameters of CloudSim	
PureEdgeSim	Cloud, Fog and Edge	N/A	Coordinates	N/A	Network usage (LAN and WAN bandwidth), latency latency requirement allocated bandwidth for each task	Task file size, tasks CPU utilisation 3 state of VMs	Energy consumption, battery capacity, idle and max consumption
iFogSim	Sensors, Actuators, Fog devices and Data Centers	Static cost for physical resources: memory, storage, bandwidth and CPU usage	Latitude, longitude	Output size, latency, CPU usage length, network usage length	Uplink bandwidth, downlink bandwidth, and uplink latency	3 state of VMs executes Tuples	Power model is based on max power and a constant static power) values
MyiFogSim	Mobile sensors, Mobile actuators, Mobile~ devices, and Data centre	Static cost for physical resources: memory, storage, bandwidth and CPU usage	Coordinates	Similar to parameters of iFogSim			
iFogSimWithDataPlacement	New components: DataPlacement, Infrastructure Partition, WorkLoad Repartition	Similar to parameters of iFogSim					
EdgeCloudSim	CloudSim + Network, Edge Server and Mobile Client	Static cost for physical resources: memory, storage, bandwidth and CPU usage	Coordinates	N/A	Transmission delay, WLAN/WAN, upload/download data	Similar to parameters of CloudSim	
StormOnEdge/SpanEdge	Edge Data Centre, ~ Central Data Centre with master-worker connection	N/A	N/A	Size in Bytes, message Count	Latency	N/A	N/A
Matlab (Zhang et al.) DockerSim EmuFog	N/A						
DISSECT-CF-Fog	Sensor, Smart Device, Fog Node and Cloud	Dynamic IoT and cloud side costs	Coordinates	Delay, frequency and file size	Latency, input bandwidth, output bandwidth and disc bandwidth between nodes for devices and nodes	11 states of Virtual Machine and it executes compute tasks	Energy model for PM and VM including min-, max- and idle power

Table 11
Comparison of the examined Fog simulators with software metrics.

Simulator	Language	Lines of Code	Comments (%)	Duplication (%)	Files	Bugs	Vulnerabilities	Code Smells
FogTorchΠ	Java, XML	2748	15.9	8.3	39	21	31	308
FogDirSim	Python, YAML	5641	1.4	N/A	84		N/A	
OPNET		N/A			N/A			
PDES	C		N/A					
FogNetSim++	C++	20 199	5.7		59			
Edge-Fog	Python	887	17.2		66			
YAFS	Python, JS, HTML, JSON	31 597	22.0		208			
EdgeNetworkCloudSim	Java, HTML	113 654	27.5		571			
PureEdgeSim	Java, XML	3308	12.2	4.3	30	18	101	301
iFogSim	Java, XML	27 754	25.3	24.3	290	124	248	1.5k
MyiFogSim	Java, XML	32 723	23.2	23.5	328	174	275	2k
iFogSimWithDataPlacement	Java, Protocol Buffers	212 780	7.6	N/A	2313		N/A	
EdgeCloudSim	Java, XML	6232	14.3	29.7	54	14	22	496
StormOnEdge/SpanEdge	Java, XML	1417	10.3	34.1	17	9	11	232
Matlab (Zhang et al.)		N/A		N/A	N/A		N/A	
DockerSim	C++, INI	48 118	22.7		336			
EmuFog	Java	2570	77.6		52			
DISSECT-CF-Fog	Java, XML	9870	33.3	2.0	118	31	192	482

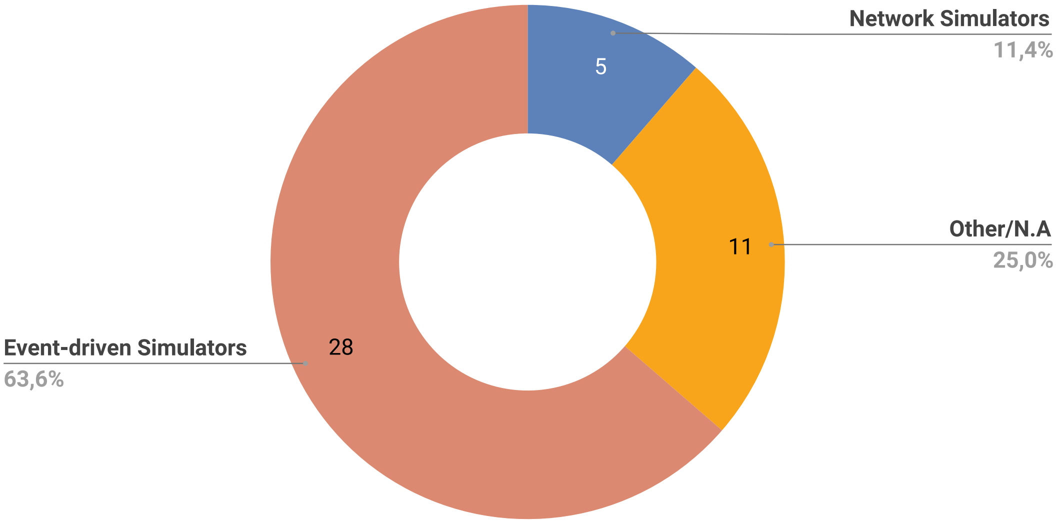

Fig. 1. The ratio of the simulator types.

Sensors are key elements of IoT systems, and in many cases they appear in fog environments as well. Therefore it is important to know, how sensors (or devices or things) are represented in certain simulators. In general, they should provide interfaces to discover, connect and monitor, they have certain data generation frequency, possibly data storing or queuing properties. Their model may also reflect behavioural information, such as actuation, failures or latencies.

- Network model:
  One of the crucial points of a simulator for distributed systems is how it handles the network operations, especially the representation of bandwidth and latency. The corresponding configuration settings, and the fine- or coarse-grained networking functions usually have a significant impact on the simulation time and accuracy of a simulator. Fine-grained models can support low-level protocol representations (e.g. HTTP or MQTT communication) to provide realistic simulations.
- VM management:
  Representing virtual machines and their management functions are some of the basic properties of modelling clouds and fogs. They determine the configuration means of VMs in the simulated environments, that can highly affect the usability of the simulator. The corresponding features of this category are: migration, task execution, network load and background workload. Some fog solutions neglect this category to focus more on physical fog node management (e.g. FogTorchΠ or YAFS).
- Energy model:
  Physical elements of real-life distributed systems consume energy and affect carbon emissions. In order to develop solutions reducing these values, simulators should provide energy metering and power usage predictions. In general, three options are used for energy models: idle, min and max consumption.

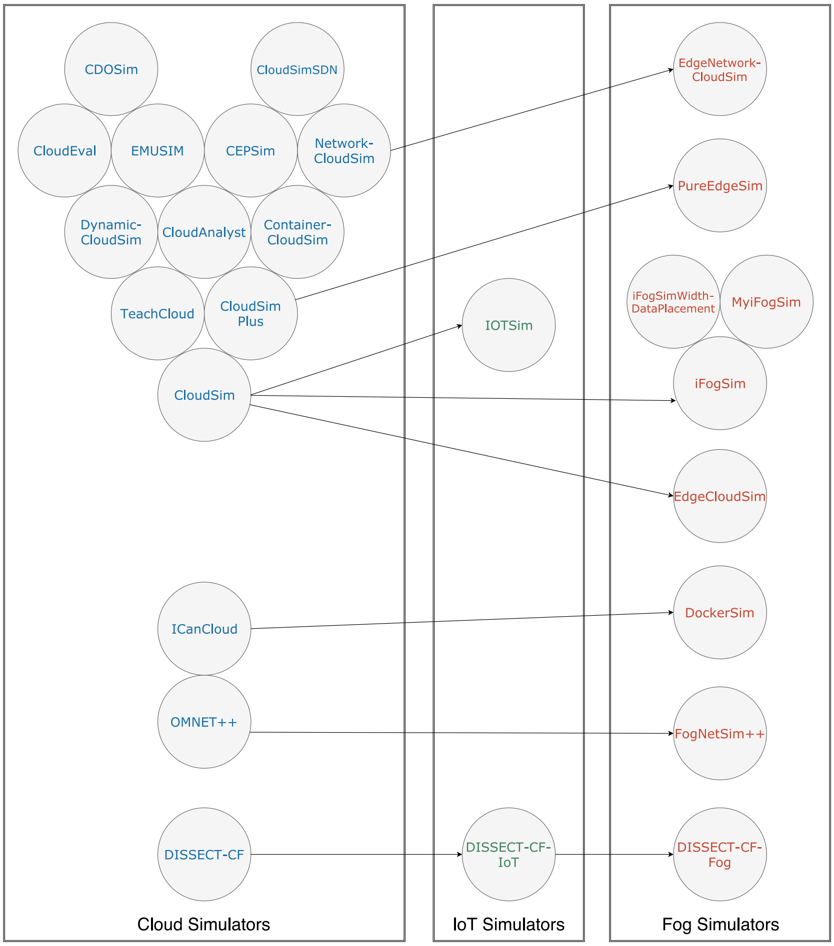

Fig. 2. Visualized relationships between the examined Cloud, IoT and Fog simulators.

- Source code metrics:
  As a final taxonomy element we chose source code metrics to represent and compare the software quality of the considered simulators. In general, bad software quality makes it hard to read, understand and reuse the source code for extensions, and it also negatively affects simulation time and accuracy. In this work we used the static code analyser called SonarQube [91] to calculate certain quality metrics. This tool is able to measure various quality features of the source code of a software. In cases we could not apply SonarQube, we used an even simpler tool called CLOC [92] that can handle almost any type of programming language, but provides less information. Certainly, we could analyse only simulators with published, open-source code. We considered the following metrics for our investigation:
  - Language: the applied programming language to implement the simulator. We also highlighted additional languages where applicable, e.g. the ones used for serialisation.
  - Lines of code: this metric represents the number of lines in the source code. Note that blank lines are not counted.
  - Comments: this metric counts a ratio (in %) of comment lines to the total lines of code (i.e. blank lines plus lines of code).
  - Duplication: it denotes the ratio of the code duplications to the whole source code.
  - Files: this metric tells us the number of files consist the software.
  - Bugs: it shows the number of bugs in the software, where bugs represent wrong language constructions (e.g. missing operand casting), according to the definition of SonarQube.
  - Vulnerabilities: this metric tells us the number of vulnerabilities in the software, which are possible security issues (e.g. public member notation in a class) in the SonarQube terminology.
  - Code smells: it counts the number of code smells, which are code blocks where modification and understanding could be time-consuming (e.g. empty statements), based on the definition of SonarQube.

    In general, when a tool has acceptable quality metrics, it also has a positive effect on the application of the simulator. Concerning the evaluation of the metrics we can state that higher values for code duplication complicate the readability of the source code, and the bigger ratio of comment lines helps researchers understanding and reusing the code. Needless to say, the researchers should prefer less number of bugs, vulnerabilities and code smells, when deciding to use a simulator for an

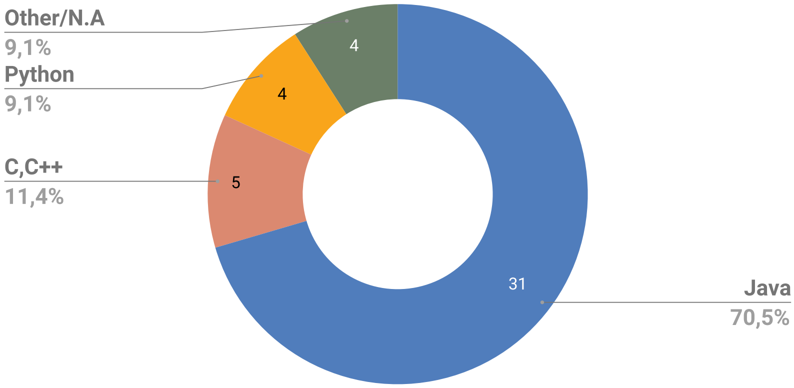

Fig. 3. The ratio of the programming languages used to implement simulators.

investigation. Concerning the implementation of the reviewed works, Fig. 3 shows that almost 70% of the investigated simulators are written in Java, which is a platform independent programming language.

3.5. Discussion

In this subsection we provide further discussions on the comparison of the analysed simulators. In the previous subsections we provided short introductions of the main properties of the overviewed works, defined the taxonomy elements used for categorising them, and provided classifications with comparison tables for three groups of simulators, namely: cloud, IoT and fog simulators. Though the tables provide detailed information on the properties of the tools, we summarise the most valuable findings of these comparisons.

For cloud solutions, Table 3 shows that most of the cloud simulators apply the generic, event-driven model to simulate entities of cloud systems. We can also see that 12 out of 18 simulators are based on CloudSim. Table 4 depicts that only limited (mostly static) cost calculations can be performed on the majority of these tools, and mostly the bandwidth parameter is the only way to model network communications. VM management functions are less supported in the network simulator category, and they are also surprisingly simple for the cloud-specific ones, except for DISSECT-CF (offering 12 functions). The energy model is also quite simple for most solutions (using static, physical CPU power consumption values), GreenCloud and DISSECT-CF have additional possibilities. Table 5 reflects software quality. Java is the dominating programming language, and the source code length varies between 5 to 30 thousand in general. iCanCloud is exceptionally long, which is due to the underlying OMNET+ + framework size. Code duplications and the number of files seem to be proportional to the code length, the number of bugs and vulnerabilities are the highest in the CloudSim family. The percentage of comments are surprisingly low in most cases, only TeachCloud and DISSECT-CF are close to 50%.

For solutions with IoT support, Table 6 shows that IoT system modelling is still not mature enough compared to Cloud Computing simulation solutions. Simulators in this group appeared around 2014, showing a wider variety of approaches (fully simulated, semi-simulated and real environments), and there seems to be no converging among them. Table 7 depicts that IOTSim and DISSECT-CF-IoT have the most detailed models reflecting almost all taxonomy elements. Important to note that geolocation support only appears in CrowdSenSim, meanwhile only DISSECT-CF-IoT supports dynamically measured IoT-side cost calculation besides cloud side-costs. The third comparison table in this group, Table 8 details the measured source code metrics of IoT solutions. We can see that only DISSECT-CF-IoT as an extension maintains the quality of its base simulator (having similar good values for comments and code duplication), while other extensions often worsen quality compared to their core solutions. Finally, we can see a huge variation in the lines of code metric, CrowdSenSim and DPWSim have tens of thousands lines, while SmartSim has less then a thousand, and DISSECT-CF-IoT and MobIoTSim stands in the middle.

Considering fog modelling, according to Table 9 we can state that models of Fog Computing are evolving more rapidly than the ones for the IoT based on the latest publications, which is approved by the fact that we found twice as much solutions for the fog group. There are many independently developed tools (the half of the considered studies), and six works extend in a direct or indirect way some CloudSim solution. It also seems that former (purely) network simulation solutions are not considered any more to be the base of new extensions for fogs. Table 10 shows that all defined taxonomy categories are covered by at least one simulator, but the most wanting category is VM management, hence only seven out of 18 tools are able to consider the utilisation of network operations of a virtual machine. We can see a great improvement over IoT simulators for geolocation support and network models. Only the iFogSim simulator (and its variants) and the DISSECT-CF-Fog satisfy all of our categories. Unfortunately, many simulators (FogTorchΠ, OPNET or SpanEdge) aim to model one specific capability of Fog Computing, which makes them less productive and usable for general simulations. Table 11 depicts that the iFogSim simulator has relatively bad software quality: about one forth of its code is duplicated, and it has more than 25 thousand lines of code having more than 1500 code smells, which is the worst of all simulators. On the contrary, DISSECT-CF-Fog and PureEdgeSim have the best ratio of duplication, and EmuFog has the best ratio of comment lines.

4. Future research challenges

In the previous section we have seen the state-of-the-art in cloud, IoT and fog simulation. Modelling Cloud Computing solutions is the earliest field, the CloudSim tool and its family dominates investigations in this area. Though new cloud management algorithms are still developed and validated with them, they are rarely updated or maintained. The IoT field brought up new simulation methods: they partly effected the former cloud simulators by triggering extensions to model cloud support functionalities for IoT data management, and novel simulators also appeared to focus on the device and sensor handling capabilities of IoT systems. These solutions are not really converging, and the non-standard approaches in this area make it hard to come up with comprehensive solutions.

Fog Computing modelling is the latest direction, and it tries to build on previous cloud and IoT system simulators as Fig. 2 suggests. This are is under active research, and fog modelling is sill in its infancy, mainly due to the still forming real-world fog applications. By taking a look at the results of this survey, we can state that some properties of fog modelling do appear in all three groups of simulators, hence we need to be aware of the latest improvements in all areas to design better solutions for fogs. Nevertheless, in general such extensions have a negative effect of the software quality, which also affects the usability of a simulator. We managed to show such effects with the evaluation of our proposed metrics.

Focusing on the currently available fog simulators, we can summarise that though usable solutions can be found for analysing specific fog capabilities or use cases, but general, complex fog environments are still hard to be addressed with a single tool. It is also unlikely that near future solutions would target to come up with a general simulator, rather extensions will appear to cover more fog management related properties. One direction, which has already been started is the modelling of container-based fog node behaviour, another one is the location-aware management of fog nodes. Table 10 is used to reveal these trends. Cost modelling and energy-aware management are also missing features in many simulators, hence they still need extensive research. The sensor models are quite simple in most tools, therefore sensor and device behaviour analysis and modelling should also be a target feature of future research.

The further extension of current simulators will likely be the case for the future of fog modelling, therefore a higher attention is needed to maintain software quality. Table 11 already revealed warning signs (see metrics duplication, bugs and code smells) in this matter, as a result software reengineering methods are highly encouraged to be used in the future to arrive to reliable and maintainable extensions.

5. Conclusion

In the past decade we have seen how the latest technological advances shaped distributed systems and led to the emerging of clouds, IoT systems and finally fogs. The complex networks and environments they established are closely coupled: the data generated by IoT devices have to be stored, processed and analysed by cloud or fog services to ensure reliability and sustainability.

We also know that for efficiently designing, developing and operating these complex systems one cannot avoid the use of simulations. We can find numerous simulators for these purposes, but in many cases it is hard to reveal their differences, and implementing our use cases with different solutions is time consuming.

The aim of our research was to respond to these problems, therefore we proposed a survey and taxonomy of the available simulators modelling clouds, IoT and fogs. The novelty of our work lies in the applied viewpoints to perform the classifications. In out taxonomy we separated the considered simulators to three groups, and presented comparison tables based on the taxonomy to reveal their differences and to highlight how they model the elements of IoT-Fog-Cloud systems. We can conclude that a comprehensive model for Fog Computing is still wanting, and complex fog environments are hard to be addressed with a single simulator. Our main recommendation for further research is to continue model extensions of simulators to better grasp fog capabilities with the warning to maintain software quality. This way can lead to easier understanding and shorter learning curves for designing and performing experimentation in the field. Based on our results we believe that this study managed to reveal the approaches to be combined and improved to provide better solutions in the future.

Acknowledgement

This research was supported by the Hungarian Scientific Research Fund under the grant number OTKA FK 131793, and by the European Union, co-financed by the European Social Fund (EFOP-3.6.1-16-2016-00008 and EFOP-3.6.3-VEKOP-16-2017-00002).

References

[1] R. Mahmud, R. Kotagiri, R. Buyya, Fog computing: a taxonomy, survey and future directions, Internet Everything Algorithms Methodol.Technol. Perspect. (2018) 103–130.

[2] U. Rahman, K. Bilal, A. Erbad, O. Khalid, S.U. Khan, Nutshell - simulation toolkit for modeling data center networks and cloud computing, IEEE Access PP (2019). 1–1

[3] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, O. Rana, Fog computing for the internet of things: a survey, ACM Trans. Internet Technol. 19 (2019) 18:1–18:41.

[4] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J.P. Jue, All one needs to know about fog computing and related edge computing paradigms: a complete survey, CoRR abs/1808.05283 (2018).

[5] S. Svorobej, P.T. Endo, M. Bendechache, C. Filelis-Papadopoulos, K.M. Giannoutakis, G.A. Gravvanis, D. Tzovaras, J. Byrne, T. Lynn, Simulating fog and edge computing scenarios: an overview and research challenges, Future Internet 11 (2019).

[6] C. Hong, B. Varghese, Resource management in fog/edge computing: A survey, CoRR (2018). Abs/1810.00305

[7] Z. Ghanbari, N. Navimipour, M. Hosseinzadeh, A. Darwesh, Resource allocation mechanisms and approaches on the internet of things, Cluster Comput. (2019).
[8] R.N. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. Pract. Experience 41 (2011) 23–50.
[9] Cloudsim simulator online: https://github.com/cloudslab/cloudsim, accessed in May, 2019.
[10] F. Howell, R. Mcnab, Simjava: a discrete event simulation library for java, Simul. Ser. 30 (1998).
[11] B. Wickremasinghe, Cloudanalyst: a cloudsim-based tool for modelling and analysis of large scale cloud computing environments, Distrib. Comput. Project (2009).
[12] CloudAnalyst simulator online: http://www.cloudbus.org/cloudsim/CloudAnalyst.zip, accessed in June, 2019.
[13] F. Fittkau, S. Frey, W. Hasselbring, CDOSim: simulating cloud deployment options for software migration support, in: 2012 IEEE 6th International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), pp. 37–46.
[14] Y. Jararweh, Z. Alshara, M. Jarrah, M. Kharbutli, M. Alsaleh, Teachcloud: a cloud computing educational toolkit, Int. J. Cloud Comput. 2 (2013) 237–257.
[15] TeachCloud simulator online: https://github.com/CloudComputingSim/TeachCloud, accessed in June, 2019.
[16] 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. Experience 43 (2012) 595–612.
[17] EMUSIM simulator online: http://www.cloudbus.org/cloudsim/emusim, accessed in June, 2019.
[18] W.A. Higashino, M.A. Capretz, L.F. Bittencourt, CEPSim: Modelling and simulation of complex event processing systems in cloud environments, Future Generation Computer Systems 65 (2016) 122–139. Special Issue on Big Data in the Cloud
[19] CEPSim simulator online: https://github.com/virsox/cepsim, accessed in June, 2019.
[20] Z. Wang, Z. Zhou, S. Jia, H. Liu, D. Li, J. Cheng, Cloudeval: a simulation environment for evaluating the dynamic cloud VM consolidation, Proceedings of the 9th EAI International Conference on Simulation Tools and Techniques, SIMUTOOLS'16, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 2016, pp. 37–45.
[21] J. Son, A.V. Dastjerdi, R.N. Calheiros, X. Ji, Y. Yoon, R. Buyya, CloudsimSDN: modeling and simulation of software-defined cloud data centers, in: 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing,pp. 475–484.
[22] CloudSimSDN simulator online: https://github.com/Cloudslab/cloudsimsdn, accessed in June, 2019.
[23] S.F. Piraghaj, A.V. Dastjerdi, R.N. Calheiros, R. Buyya, Containercloudsim: an environment for modeling and simulation of containers in cloud data centers, Softw. Pract. Experience 47 (2017) 505–521.
[24] S.K. Garg, R. Buyya, Networkcloudsim: modelling parallel applications in cloud simulations, in: 2011 Fourth IEEE International Conference on Utility and Cloud Computing, pp. 105–113.
[25] M. Bux, U. Leser, Dynamiccloudsim: simulating heterogeneity in computational clouds, Future Gener. Comput. Syst. 46 (2015) 85–99.
[26] DynamicCloudSim simulator online: https://github.com/marcbux/dynamiccloudsim, accessed in June, 2019.
[27] M.C.S. Filho, R.L. Oliveira, C.C. Monteiro, P.R.M. Inácio, 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) (2017) 400–406.
[28] CloudSim Plus simulator online: https://github.com/manoelcampos/cloudsim-plus, accessed in June, 2019.
[29] G. Kecskemeti, DISSECT-CF: a simulator to foster energy-aware scheduling in infrastructure clouds, Simul. Modell. Pract. Theory 58P2 (2015) 188–218.
[30] DISSECT-CF simulator online: https://github.com/kecskemeti/dissect-cf, accessed in May, 2019.
[31] Z. Mann, Cloud simulators in the implementation and evaluation of virtual machine placement algorithms, Softw. Pract. Experience (2017).
[32] M. Tighe, G. Keller, M. Bauer, H. Lutfiyya, DCSIm: a data centre simulation tool for evaluating dynamic virtualized resource management, Proceedings of the 2012 8th International Conference on Network and Service Management, CNSM 2012, (2012), pp. 385–392.
[33] DCSim simulator online: https://github.com/digs-uwo/dcsim, accessed in June, 2019.
[34] 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 Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 305–313.
[35] D. Kliazovich, P. Bouvry, Y. Audzevich, S.U. Khan, Greencloud: a packet-level simulator of energy-aware cloud computing data centers, in: 2010 IEEE Global Telecommunications Conference GLOBECOM 2010, pp. 1–5.
[36] NS-2 simulator online: https://www.isi.edu/nsnam/ns, accessed in June, 2019.
[37] A. Nez, J.L. Vzquez-Poletti, A. Caminero, G.G. Casta, J. Carretero, I.M. Llorente, ICAncloud: a flexible and scalable cloud infrastructure simulator, J. Grid Comput. 10 (2012) 185–209.
[38] OMNeT+ + simulator online: https://omnetpp.org, accessed in June, 2019.
[39] ICanCloud simulator online: https://www.arcos.inf.uc3m.es/old/icancloud, accessed in June, 2019.
[40] I. Sriram, SPECI, A simulation tool exploring cloud-scale data centres, CoRR abs/0910.4568 (2009).
[41] B. Costa, P.F. Pires, F.C. Delicato, Modeling iot applications with sysML4iot, in: 2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 157–164.
[42] C. Fiandrino, A. Capponi, G. Cacciatore, D. Kliazovich, U. Sorger, P. Bouvry, B. Kantarci, F. Granelli, S. Giordano, Crowdsensim: a simulation platform for mobile crowdsensing in realistic urban environments, IEEE Access 5 (2017) 3490–3503.
[43] S. Han, G.M. Lee, N. Crespi, N. Luong, H. Kyoungwoo, M. Brut, P. Gatellier, DPWSIm: a simulation toolkit for iot applications using devices profile for web services, 2014 IEEE World Forum on Internet of Things, WF-IoT 2014, (2014).
[44] DPWSim simulator online: https://github.com/sonhan/dpwsim, accessed in June, 2019.
[45] S. Sotiriadis, N. Bessis, E. Asimakopoulou, N. Mustafee, Towards simulating the internet of things, in: 2014 28th International Conference on Advanced Information Networking and Applications Workshops, pp. 444–448.
[46] S. Sotiriadis, N. Bessis, N. Antonopoulos, A. Anjum, SimIC: designing a new inter-cloud simulation platform for integrating large-scale resource management, Proceedings - International Conference on Advanced Information Networking and Applications, AINA, (2013), pp. 90–97.
[47] V. Angelakis, I. Avgouleas, N. Pappas, D. Yuan, Flexible allocation of heterogeneous resources to services on an iot device, IEEE Internet Things J. 2015 (2015).
[48] Z. Li, K. Liu, Y. Su, Y. Ma, Adaptive resource allocation algorithm for internet of things with bandwidth constraint, Trans. Tianjin Univ. 18 (2012).
[49] D. Thomas, J. Irvine, Connection and resource allocation of iot sensors to cellular technology-LTE, in: 2015 11th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME), pp. 365–368.
[50] D. Chen, D. Irwin, P. Shenoy, Smartsim: a device-accurate smart home simulator for energy analytics, in: 2016 IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 686–692.
[51] SmartSim simulator online: https://github.com/sustainablecomputinglab/smartsim, accessed in June, 2019.
[52] A. Kertesz, T. Pflanzner, T. Gyimothy, A mobile iot device simulator for iot-fog-cloud systems, J. Grid Comput. (2018).
[53] MobIoTSim simulator online: https://github.com/sed-szeged/MobIoTSim, accessed in May, 2019.
[54] X. Zeng, S. Garg, P. Strazdins, P.P. Jayaraman, D. Georgakopoulos, R. Ranjan, IOTSIm: a cloud based simulator for analysing iot applications, J. Syst. Archit. (2016).
[55] A. Markus, J.D. Dombi, Multi-cloud management strategies for simulating iot applications, Acta Cybern. 24 (2019) 83–103.
[56] DISSECT-CF-IoT simulator online: https://github.com/andrasmarkus/dissect-cf/tree/pricing, accessed in May, 2019.
[57] A. Brogi., S. Forti., A. Ibrahim., Deploying fog applications: how much does it cost, by the way? Proceedings of the 8th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER, INSTICC, SciTePress, 2018, pp. 68–77.
[58] FogtorchII simulator online: https://github.com/di-unipi-socc/FogTorchPI, accessed in May, 2019a.
[59] FogDirSim simulator online: https://github.com/di-unipi-socc/FogDirSim, accessed in June, 2019b.
[60] N. Abbas, M. Asim, N. Tariq, T. Baker, S. Abbas, A mechanism for securing iot-enabled applications at the fog layer, J. Sens. Actuator Netw. 8 (2019) 16.

[61] OPNET simulator online: http://opnetprojects.com/opnet-network-simulator, accessed in June, 2019.
[62] D. Tychalas, H. Karatza, Simulation and performance evaluation of a fog system, in: 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC), pp. 26–33.
[63] T. Qayyum, A. Malik, M.A. Khan, O. Khalid, S.U. Khan, Fognetsim++: A toolkit for modeling and simulation of distributed fog environment, IEEE Access PP (2018). 1–1
[64] Fognetsim++ simulator online: https://github.com/rtqayyum/fognetsimpp, accessed in june, 2019.
[65] N. Mohan, J. Kangasharju, Edge-fog cloud: a distributed cloud for internet of things computations, in: 2016 Cloudification of the Internet of Things (CIoT), pp. 1–6.
[66] EdgeFog simulator online: https://github.com/nitindermohan/EdgeFogSimulator, accessed in June, 2019.
[67] I. Lera, C. Guerrero, C. Juiz, YAFS: a simulator for iot scenarios in fog computing, ArXiv: abs/1902.01091(2019).
[68] YAFS simulator online: https://github.com/acsicuib/YAFS, accessed in June, 2019.
[69] M. Seufert, B.K. Kwam, F. Wamser, P. Tran-Gia, Edgenetworkcloudsim: placement of service chains in edge clouds using networkcloudsim, in: 2017 IEEE Conference on Network Softwarization (NetSoft), pp. 1–6.
[70] EdgeNetworkCloudSim simulator online: https://github.com/lsinfo3/EdgeNetworkCloudSim, accessed in June, 2019.
[71] PureEdgeSim simulator online: https://github.com/CharafeddineMechalikh/PureEdgeSim, accessed in June, 2019.
[72] H. Gupta, A.V. Dastjerdi, S. Ghosh, R. Buyya, Ifogsim: a toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments, Softw. Pract. Experience (2016).
[73] IFogSim simulator online: https://github.com/Cloudslab/iFogSim, accessed in May, 2019.
[74] P.V. Naranjo, Z. Pooranian, M. Shojafar, M. Conti, R. Buyya, FOCAN: A fog-supported smart city network architecture for management of applications in the internet of everything environments, J. Parallel Distrib. Comput. (2018).
[75] E. Baccarelli, P.V. Naranjo, M. Shojafar, M. Scarpiniti, J. Abawajy, Fog of everything: energy-efficient networked computing architectures, research challenges, and a case study, IEEE Access PP (2017) 2169–3536.
[76] 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, New York, NY, USA, 2017, pp. 47–52.
[77] MyiFogSim simulator online: https://github.com/marciocomp/myifogsim, accessed in June, 2019.
[78] M.I. Naas, J. Boukhobza, P. Raipin Parvedy, L. Lemarchand, An extension to ifogsim to enable the design of data placement strategies, in: 2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC), pp. 1–8.
[79] IFogSimWithDataplacement simulator online: https://github.com/medislam/iFogSimWithDataPlacement, accessed in June, 2019.
[80] 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), (2017), pp. 39–44.
[81] EdgeCloudSim simulator online: https://github.com/CagataySonmez/EdgeCloudSim, accessed in June, 2019.
[82] H.P. Sajjad, K. Danniswara, A. Al-Shishtawy, V. Vlassov, Spanedge: towards unifying stream processing over central and near-the-edge data centers, in: 2016 IEEE/ACM Symposium on Edge Computing (SEC), pp. 168–178.
[83] StormEdge/SpanEdge simulator online: https://github.com/Telolets/StormOnEdge, accessed in June, 2019.
[84] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F.R. Yu, Z. Han, Computing resource allocation in three-tier iot fog networks: a joint optimization approach combining stackelberg game and matching, IEEE Internet Things J. 4 (2017) 1204–1215.
[85] Z. Nikdel, B. Gao, S.W. Neville, Dockersim: full-stack simulation of container-based software-as-a-service (saas) cloud deployments and environments, in: 2017 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), pp. 1–6.
[86] DockerSim simulator online: https://github.com/Znikdel/DockerSim, accessed in June, 2019.
[87] 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) (2017) 1–6.
[88] EmuFog emulator online: https://github.com/emufog/emufog, accessed in June, 2019.
[89] DISSECT-CF-Fog simulator online: https://github.com/andrasmarkus/dissect-cf/tree/fog-extension, accessed in September, 2019.
[90] P.F. Roth, Discrete, continuous, and combined simulation, Proceedings of the 20th Conference on Winter Simulation, (1988), pp. 56–60.
[91] SonarQube online: https://www.sonarqube.org, accessed in May, 2019.
[92] CLOC online: http://cloc.sourceforge.net, accessed in May, 2019.